Neural Networks for Signal Processing IV

PROCEEDINGS OF THE 1994 IEEE WORKSHOP

Edited by

John Vlontzos
Jenq-Neng Hwang
Elizabeth Wilson

Neural Networks for Signal Processing IV

PROCEEDINGS OF THE 1994 IEEE WORKSHOP

94 TH0688-2

This document has been approved for public release and sale; its distribution is unlimited.

19941209 078

NEURAL NETWORKS FOR SIGNAL PROCESSING IV

PROCEEDINGS OF THE 1994 IEEE WORKSHOP

Fourth in a Series of Workshops
Organized by the IEEE Signal Processing Society
Neural Networks Technical Committee

Edited by

John Vlontzos
Jenq-Neng Hwang
Elizabeth Wilson

Published under the sponsorship of the
IEEE Signal Processing Society
in cooperation with the IEEE Neural Networks Council
with support from Intracom S.A. Greece
and co-sponsored by ARPA

or
&I
ed
Codes
and/or
ecial
A-1

The Institute of Electrical and Electronics Engineers, Inc.
New York, NY

DTIC QUALITY INSPECTED 1

Robust Estimation for Radial Basis Functions
A.G. Bors and I. Pitas 105

Network Architectures

The Use of Recurrent Neural Networks for Classification
T.L. Burrows and M. Niranjan 117

Network Structures for Nonlinear Digital Filters
J.N. Lin and R. Unbehauen 126

Locally Excitatory Globally Inhibitory Oscillator Networks:Theory and Application to Pattern Segmentation
D. Wang and D. Terman 136

A Unifying View of Some Training Algorithms for Multilayer Perceptrons with FIR Filter Synapses
A. Back, E.A. Wan, S. Lawrence, and A. C. Tsoi 146

Spectral Feature Extraction Using Poisson Moments
S. Çelebi and J. Principe 155

Application of the Fuzzy Min-Max Neural Network Classifier to Problems with Continuous and Discrete Attributes
A. Likas, K. Blekas, and A. Stafylopatis 163

Time Signal Filtering by Relative Neighborhood Graph Localized Linear Approximation
J.A. Sørensen 171

Classification Using Hierarchical Mixtures of Experts
S.R. Waterhouse and A.J. Robinson 177

A Hybrid Neural Network Architechture for Automatic Object Recognition
T. Fechner and R. Tanger 187

Time Series Prediction Using Genetically Trained Wavelet Networks
A. Procházka and V. Sys 195

A Network Of Physiological Neurons With Differentiated Excitatory And Inhibitory Units Possessing Pattern Recognition Capacity
E. Ventouras, M. Kitsonas, S. Hadjiagapis, N. Uzunoglu, C. Papageorgiou, A. Rabavilas, and C. Stefanis 204

Learning with Imperfect Perception

A Learning Algorithm for Multi-Layer Perceptrons with Hard-Limiting Threshold Units
R.M. Goodman and Z. Zeng 219

The Selection of Neural Models of Non-Linear Dynamical Systems by Statistical Tests
D. Urbani, P. Roussel-Ragot, L. Personnaz, and G. Dreyfus 229

Pruning Recurrent Neural Networks for Improved Generalization Performance
C.W. Omlin and C.L. Giles 690

Speech Processing

Recurrent Network Automata for Speech Recognition: A Summary of Recent Work
R. Gemello, D. Albesano, F. Mana, and R. Cancelliere 241

Acoustic Echo Cancellation for Hands-Free Telephony Using Neural Networks
A.N. Birkett and R. A. Goubran 249

Minimum Error Training for Speech Recognition
E. McDermott and S. Katagiri 259

Connectionist Model Combination for Large Vocabulary Speech Recognition
M.M. Hochberg, G.D. Cook, S.J. Renals, and A.J. Robinson 269

Neural Tree Network/Vector Quantization Probability Estimators for Speaker Recognition
K. Farrell, S. Kosonocky, andR. Mammone 279

Parallel Training of MLP Probability Estimators for Speech Recognition: A Gender-Based Approach
N. Mirghafori, N. Morgan, and H. Bourlard 289

LVQ as a Feature Transformation for HMMs
K. Torkkola 299

Autoassociator-Based Modular Architecture for Speaker Independent Phoneme Recognition
L. Lastrucci, G. Bellesi, M. Gori, and G. Soda 309

Non-linear Speech Analysis Using Recurrent Radial Basis Function Networks
P.A. Moakes and S.W. Beet 319

Word Recognition Using a Neural Network and a Phonetically Based DTW
Y. Matsuura, H. Miyazawa, and T.E. Skinner 329

A Monolithic Speech Recognizer Based on Fully Recurrent Neural Networks
K. Kasper, H. Reininger, D. Wolf, and H. Wüst 335

Fuzzification of Formant Trajectories for Classification of CV Utterances Using Neural Network Models
B. Yegnanarayana, C. C. Sekhar, and S.R. Prakash 345

Minimum Error Classification of Keyword-Sequences
T. Komori and S. Katagiri 352

Hybrid Training Method for Tied Mixture Density Hidden Markov Models Using Learning Vector Quantization and Viterbi Estimation
M. Kurimo 362

Image Processing

Moving Object Classification in a Domestic Environment Using Quadratic Neural Networks
G. Lim, M. Alder, C.J.S. deSilva, and Y. Attikiouzel 375

Application of the HLVQ Neural Network to Hand-Written Digit Recognition
B. Solaiman and Y. Autret 384

Ensemble Methods for Automatic Masking of Clouds in AVIRIS Imagery
C.M. Bachmann, E.E. Clothiaux, J.W. Moore, K. J. Andreano, and D. Q. Luong 394

Saddle-Node Dynamics for Edge Detection
Y.F. Wong 404

Application of SVD Networks to Multi-Object Motion-Shape Analysis
S.Y. Kung, J.S. Taur, and M.Y. Chiu 413

Neural Networks for Robust Image Feature Classification: A Comparative Study
S.V.R. Madiraju and C.C. Liu 423

Medical Imaging with Neural Networks
C.S. Pattichis and A.G. Constantinides 431

High Resolution Image Reconstruction Using Mean Field Annealing
T. Numnonda and M. Andrews 441

Hardware Neural Network Implementation of Tracking System
G.G. Lendaris, R.M. Pap, R.E. Saeks, C.R. Thomas, and R.M. Akita 451

Fast Image Analysis Using Kohonen Maps
D. Willett, C. Busch, and F. Seibert 461

Analysis of Satellite Imagery Using a Neural Network Based Terrain Classifier
M.P. Perrone and M.J. Larkin 700

Medical Applications

Neural Networks and Higher Order Spectra for Breast Cancer Detection
T. Stathaki and A.G. Constantinides 473

Medical Diagnosis and Artificial Neural Networks: A Medical Expert System Applied to Pulmonary Diseases
G.-P.K. Economou, C. Spriopoulos, N.M. Economopoulos, N. Charokopos, D. Lymberopoulos, M. Spiliopoulou, E. Haralambopulu, and C.E.Goutis 482

Modeling of Glaucoma Induced Changes in the Retina and Neural Net Assisted Diagnosis
S. von Spreckelsen, P. Grumstup, J. Johnsen, and L.K. Hansen 490

Toward Improving Excercise ECG for Detecting Ischemic Heart Disease with Recurrent and Feedforward Neural Nets
G. Dorffner, E. Leitgeb, and H. Koller 499

Towards Semen Quality Assessment Using Neural Networks
C. Linneberg, P. Salamon, C. Svarer, L.K. Hansen, and J. Meyrowitsch 509

Use of Neural Networks in Detection of Ischemic Episodes from ECG Leads
N. Maglaveras, T. Stamkopoulos, C. Pappas, and M. Strintzis 518

EEG Signal Analysis Using a Multi-Layer Perceptron with Linear Preprocessing
S.A. Mylonas and R.A. Comley 671

Adaptive Processing And Communication

A Neural Network Trained with the Extended Kalman Algorithm Used for the Equalization of a Binary Communication Channel
M. Birgmeier 527

Neural-Net Based Receiver Structures for Single- and Multi-Amplitude Signals in Interference Channels
D.P. Bouras, P.T. Mathiopoulos, and D. Makrakis 535

A Hybrid Digital Computer-Hopfield Neural Network CDMA Detector for Real-Time Multi-User Demodulation
G.I. Kechriotis and E.S. Manolakos 545

A Hopfield Network Based Adaptation Algorithm for Phased Antenna Arrays
M. Alberti 555

Blind Deconvolution of Signals Using a Complex Recurrent Network
A.D. Back and A.C. Tsoi 565

Improving the Resolution of a Sensor Array Pattern by Neural Networks
C. Bracco, S. Marcos, and M. Benidir 573

Other Applications

Sensitivity Analysis on Neural Networks for Meteorological Variable Forecasting
J. Castellanos, A. Pazos, J. Rios, and J.L. Zafra 587

Continuous-Time Nonlinear Signal Processing: A Neural Network Based Approach for Gray Box Identification
R. Rico-Martínez, J.S. Anderson, and I.G. Kevrekidis 596

A Quantitative Study of Evoked Potential Estimation Using a Feedforward Neural Network
A. Dumitras, A.T.Murgan, and V. Lazarescu 606

Neural Estimation of Kinetic Rate Constants from Dynamic Pet-Scans
T. Fog, L.H. Nielsen, LK. Hansen, S. Holm, I. Law, C. Svarer, and O. Paulson 616

Auditory Stream Segregation Based on Oscillatory Correlation
D. Wang 624

Application of Neural Networks for Sensor Performance Improvement
S. Poopalasingam, C.R. Reeves, and N.C. Steele 633

NeuroDevice - Neural Network Device Modelling Interface for VLSI Design
P. Ojala, J. Saarinen, and K. Kaski 641

Encoding Pyramids by Labeling RAAM
S. Lonardi, A. Sperduti, and A. Starita 651

Reconstructed Dynamics and Chaotic Signal Modeling
J.M. Kuo and J.C. Principe *661*

A Neural Network Scheme for Earthquake Prediction Based on the Seismic Electric Signals
S. Lakkos, A. Hadjiprocopis, R. Comley, and P. Smith *681*

Neural-Network Based Classification of Laser-Doppler Flowmetry Signals
N.G. Panagiotidis, A. Delopoulos and S.D. Kollias *709*

Author Index *721*

Preface

This book contains papers presented at the Fourth IEEE Workshop on Neural Networks for Signal Processing (NNSP'94) at the Porto Hydra Resort Hotel, Ermioni, Greece on September 6 - 8, 1994.

The Workshop, sponsored by the Neural Network Technical Committee of the IEEE Signal Processing Society, in cooperation with the IEEE Neural Network Council and with co-sponsorship from ARPA and Intracom S.A. Greece, is designed to serve as a regular forum for researchers from universities and industry who are interested in interdisciplinary research on neural networks for signal processing applications. In the present scope, the workshop encompasses up-to-date research results in several key areas, including learning algorithms, network architectures, speech processing, image processing, adaptive signal processing, medical signal processing, and other applications. The Conference Proceedings is crafted to be an archival reference in the rapidly growing field of Neural Networks for Signal Processing.

Our deep appreciation is extended to Dr. Leon Bottou of Neuristique Inc., Paris, France; Dr. Dan Hammerstrom of Adaptive Solution Inc., Beaverton, Oregon; Dr Kazuo Asakawa of Fujitsu Laboratories Ltd., Kawasaki, Japan; and Professor Andreas S. Weigend of University of Colorado, Boulder, Colorado; for their insightful plenary talks and invited presentations. Thanks to Dr. Gary Kuhn of Siemens Corporate Research, Princeton, New Jersey, for organizing a wonderful evening panel discussion on "Neural Networks for Industrial Applications". Our sincere thanks go to all the authors for their timely contributions, to all the members of the Program Committee for the outstanding and high-quality program, and to Dr. Demetris Kalivas, the Finance and Registration Chair. We are very pleased to acknowledge Professor Yu-Hen Hu, the Neural Networks Technical Committee Chair of IEEE SP Society, Professor Sun-Yuan Kung for catalyzing ARPA and other sponsorships, and Dr. Barbara Yoon for her continued enthusiasm and support in this emerging cross-disciplinary field. Our special gratitudes go to Ms. Elizabeth Lustig of University of Washington and Mrs. Myra Sourlou of Intracom S. A. whose organization assistances to the Workshop have been invaluable.

John Vlontzos, Intracom
Jenq-Neng Hwang, University of Washington
Elizabeth Wilson, Raytheon Company

Learning Algorithms

A NOVEL UNSUPERVISED COMPETITIVE LEARNING RULE WITH LEARNING RATE ADAPTATION FOR NOISE CANCELLING AND SIGNAL SEPARATION

Marc M. Van Hulle
Laboratorium voor Neuro- en Psychofysiologie
K.U.Leuven
Campus Gasthuisberg
Herestraat
B-3000 Leuven, BELGIUM
Tel.: + 32 16 34 59 61
Fax: + 32 16 34 59 93
E-mail: marc@neuro.kuleuven.ac.be

Abstract— A new ANN-based approach to adaptive noise cancelling and separating slow-varying signals is introduced. The network's weights are continuously modified using a fast unsupervised competitive learning rule, called Fast Boundary Adaptation Rule or FBAR, performing adaptive scalar quantization of the input signal. The rule maximizes information-theoretic entropy and yields a non-parametric model of the input probability density function. Contrary to classic unsupervised competitive learning, our system adapts its own learning rate, and hence does not require a "cooling scheme." Furthermore, contrary to most of the other noise cancelling approaches, our system does not require *a priori* knowledge or an explicit model of the joint noise and signal characteristics.

INTRODUCTION

Signal separation and noise cancelling are widely researched topics in signal processing since their application increases the performance of *e.g.* pattern recognition in speech and image processing. Signals received by microphones and antennas typically comprise unknown mixtures of several signal sources. Sensors often are multisensitive: the signal provided by a sensor can be an unknown superposition of signals emitted in its neighborhood. In addition, sensors are noisy and their characteristics may change over time. A major field of applications are the so-called smart sensors. In these sensors, an integrated microcomputer is used for performing dynamic correction of changes in sensor characteristics and in environmental conditions [1].

Signal separation was introduced in the Artificial Neural Network (ANN) field by Hérault and co-workers [2,3]. They proposed a fully connected re-

0-7803-2026-3/94 $4.00 © 1994 IEEE

cursive ANN in which the weights are adapted so as to model the mixture process. Their method performs a blind separation of sources by assuming that they are statistically independent; it is also assumed that the transformation matrix describing the linear mixture is invertable. The separation relies on the computation of higher-order statistical moments in order to achieve signal independence. Hence, their method is not suited for separating slow-varying signals such as signal drift. Furthermore, it requires a separate filtering stage to obtain zero mean estimates of the reconstructed signals.

Noise cancelling is another prime application of signal processing research. The signals received by an actual system are often corrupted by additive noise. In most cases, the noise is non-stationary and of an arbitrary probability density function (*p.d.f.*) type. Furthermore, since the source signal is non-stationary, the signal-to-noise ratio changes momentarily. Kalman filtering [4] is a well known procedure for noise cancelling, however, the dynamics of the source signal must be linear Gaussian and *a priori* known. Several attempts have been undertaken in order to overcome these limitations (*e.g.* [5]) but they often lead to complicated solutions. Widrow and his co-workers were among the first to introduce adaptive filtering into the ANN field. They developed an adaptive filter system for noise cancelling [6] that is very similar to a single ADALINE-unit but without the threshold. The system requires a separate noise channel containing noise that is correlated with the noise added to the signal; the filter weights are adapted so as to minimize the power of the reconstructed signal. Recently, other ANN-based approaches have emerged and used *e.g.* for speech enhancement purposes: a multilayer perceptron is trained using samples of noisy speech at its input and clean speech at its output [7,8]. In another proposal, not the speech sample itself is input but a set of parameters obtained from (classical) statistical speech and noise models [9]: the network is trained to perform nonlinear spectral estimation by representing the shape of the distribution of speech and noise spectral parameters. The estimated spectral magnitude of the clean speech signal is then combined with the phase of the noisy speech to produce a clean signal estimate by means of overlap-and-add resynthesis –a computationally heavy procedure.

In this article, we propose an ANN-based system for performing signal separation and noise cancelling of slow-varying signals. The weights are modified "on line" using a fast, unsupervised competitive learning rule maximizing information-theoretic entropy. The rule, called Fast Boundary Adaptation Rule or FBAR [10,11], performs adaptive scalar quantization and yields a non-parametric model of the input *p.d.f.* by its N quantization levels. Contrary to the aforementioned approaches, our system operates in an unsupervised mode and hence, does not require *a priori* knowledge or an explicit model of the (joint) noise and signal characteristics. Furthermore, contrary to other ANN-based approaches, our system does not require a training mode or a "cooling scheme." Instead it uses two FBAR-based ANNs: one adaptive and another non-adaptive. The latter is used as a reference for adapting the former. Both ANNs are identical and differ only in their learning rates. Finally, since our method does not rely on filtering, it is ideally suited for separating slow-varying signals such as sensor drift and $1/f$ noise sources.

FAST BOUNDARY ADAPTATION RULE

An N-point scalar quantizer can be considered a function which maps a scalar-valued input signal x into one of N quantization levels $y_1, y_2, ..., y_N$. The quantizer is specified by the values of these quantization levels and the N disjoint quantization intervals $R_1, R_2, ..., R_N$. An adaptive scalar quantizer is intended to capitalize on the structure underlying the input signal distribution $p(x)$ with a minimal overall distortion due to quantization. Many distortion measures have been proposed in literature [12] but the most commonly used are the mean squared error (MSE) distortion and the mean absolute error (MAE) distortion:

$$MAE \equiv \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} \mid x - y_i \mid p(x) \;\; dx, \tag{1}$$

with x_{i-1} and x_i the boundary points of interval R_i, and with $x_0 = -\infty$ and $x_N = \infty$ for an unbounded $p.d.f.$ $p(x)$. In case of high-resolution quantization, N is very large and the quantization interval lengths are small so that $p(x)$ is roughly constant over the individual intervals. Hence, $p(x) \approx p_i$ in interval R_i, and $p_i = p(R_i)/\Delta_i$, with $p(R_i)$ the probability of $x \in R_i$ and with $\Delta_i = x_i - x_{i-1}$ the size of interval $R_i \equiv [x_{i-1}, x_i)$; if x_0 is infinite, then $R_1 \equiv (x_0, x_1)$. Suppose that with probability nearly one, x takes on values in a finite interval $[a, b)$, hence, eq. (1) can be approximated as:

$$MAE \approx \sum_{i=1}^{N} \frac{p(R_i)}{\Delta_i} \int_{x_{i-1}}^{x_i} \mid x - y_i \mid \;\; dx, \tag{2}$$

with $x_0 \equiv a$ and $x_N \equiv b$. Under the high-resolution assumption, the centroid of each interval can be approximated by its midpoint $y_i \approx \frac{x_{i-1}+x_i}{2}$ and substituted for in eq. (2). The necessary condition for minimizing MAE in the high-resolution case is then obtained by taking the derivatives of the substituted equation with respect to the x_i's and setting them equal to zero:

$$p(R_j) = p(R_{j+1}), \quad j = 1, ..., N-1. \tag{3}$$

The latter implies a maximization of the information-theoretic entropy:

$$I = -\sum_{j=1}^{N} p(R_j) \;\; log_2 \, p(R_j), \tag{4}$$

irrespective of the type of input $p.d.f.$

The necessary condition eq. (3) is realized by our Boundary Adaptation Rule (BAR) as follows. Assume that at time step t, $x \in R_j$. We then modify R_j by increasing x_{j-1} and decreasing x_j, or in the general case:

$$\Delta x_j = \eta(Act_{R_{j+1}} - Act_{R_j}), \quad j = 1, ..., N-1, \tag{5}$$

with η the learning rate, a positive scalar, and with Act_{R_j} the code membership function of interval R_j:

$$Act_{R_j}(x) = \begin{cases} 1 & \text{if } x \in R_j \\ 0 & \text{if } x \notin R_j \end{cases}$$

defined with respect to the boundary points at the previous time step. The proof of convergence towards equiprobable quantization intervals is given in [11]. At convergence, the expected noise intensity of the boundary points equals $\eta^2 \frac{2}{N}$. The speed of convergence, in case the $p.d.f.$ is bounded and the boundary points are initialized outside its range, is on the order of $\mathcal{O}(\frac{\eta}{N(N-1)})$.

The fastest rule, called Fast BAR or FBAR, is found by updating all boundary points each time an input is presented:

$$\Delta x_j = \eta(\sum_{k=j+1}^{N} \frac{Act_{R_k}}{N-j} - \sum_{k=1}^{j} \frac{Act_{R_k}}{j}) \quad j = 1, ..., N-1. \tag{6}$$

At convergence, the expected noise intensity of the boundary points equals $\eta^2 \frac{1}{(N-j)j}$, and thus for $j = N/2$ with N even, $\eta^2 \frac{4}{N^2}$; the convergence speed is now on the order of $\mathcal{O}(\frac{\eta}{N-1})$. Hence, contrary to BAR, average boundary point dynamics and noise intensities are approximatively independent of N if η increases proportionally with N. Or, for the same N and η, FBAR is N times faster than BAR. Previously, the quantization performance of FBAR was assessed and compared with that of five popular unsupervised competitive learning rules and that of the standard Lloyd I algorithm [10].

NOISE CANCELLING AND SIGNAL SEPARATION

In this article we will limit ourselves to the suppression of zero-mean additive noise, in which case the input signal can be written as:

$$x[t] = s[t] + d[t] \tag{7}$$

where $s[t]$ denotes the signal and $d[t]$ the noise. The aim is to reconstruct $s[t]$ from noisy observations $x[t]$ by separating its estimate $\widehat{s[t]}$ from the estimated noise contribution $\widehat{d[t]}$. This way, noise cancelling is considered here to be a limiting case of signal separation. This will be done by estimating the possibly non-stationary $p.d.f.$ of $d[t]$ with FBAR, and using this information for subtracting $\widehat{d[t]}$ from $x[t]$. Hence, noise cancelling will be performed in the signal magnitude domain, based on $p.d.f.$ estimation instead of spectral magnitude estimation as in the classical case. The basic assumption is that the $p.d.f.$ of s varies on a slower time-scale than that of d; the validity of this assumption will be assessed in this section.

Before further elaborating on our application, we first show the performance of FBAR in estimating non-stationary $p.d.f.$s in general. Consider the speech example in Fig. 1 (top left); the signal originates from TIMIT, a popular speech database. The signal is quantized with $N = 32$ intervals (5 bit quantizer) with $\eta = 0.02$. We observe that the boundary point traces shown in Fig. 1 (top right) seem to keep track of the speech signal by performing only a single update of the boundary points per time step. The corresponding codebook utilization $\{p(Act_{R_i}) \mid 1 \leq i \leq k\}$ is shown in the bottom part of Fig. 1: it shows that, notwithstanding the signal is highly non-stationary, the necessary condition eq. (3) is satisfied on average.

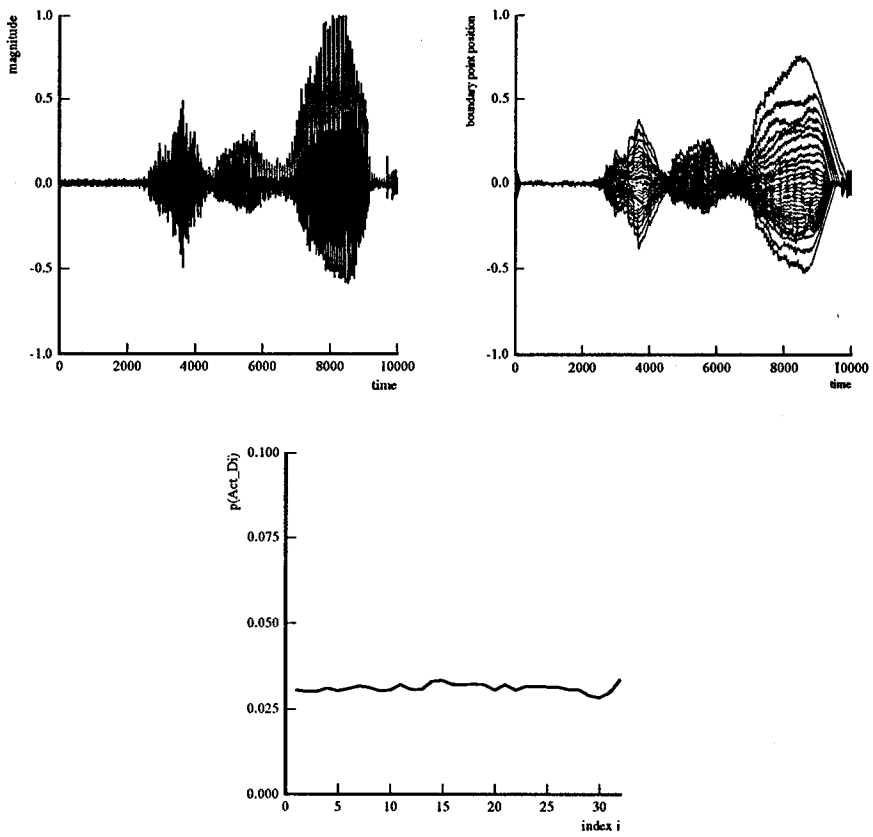


Figure 1: *Top left :* Example speech signal comprising a silence, a consonant and two vowels (*i.e.* the word */she/*). Time units are expressed in 0.625 milliseconds. *Top right :* Temporal evolution of boundary points using FBAR with $\eta = 0.02$. *Bottom :* Codebook utilization.

The previous $p.d.f.$-estimation property of FBAR can now be used in several ways, the simplest being the subtraction of the estimated $p.d.f.$ median from x. Indeed, since FBAR provides us with an estimate of the $p.d.f.$ using equiprobable quantization intervals, the trace of boundary point $x_{\frac{N}{2}}$, with N even, represents the trace of the $p.d.f.$ median. Now since it is assumed that the $p.d.f.$ of $s[t]$ varies slower than that of $d[t]$, the estimated $p.d.f.$ approximates that of $d[t]$ and hence, the trace of its median provides us with the desired estimate $\widehat{s[t]}$. We will now verify this assumption with a synthetic signal comprising Gaussian white noise, with zero mean and standard deviation 0.1, added to a sampled sine wave of magnitude 0.5 and frequency $\frac{f_s}{10{,}000}$ Hz. The noise cancelling performance is assessed by calculating the MSE distortion between $\widehat{s[t]}$ and $s[t]$ on the last 10,000 samples, and plot-

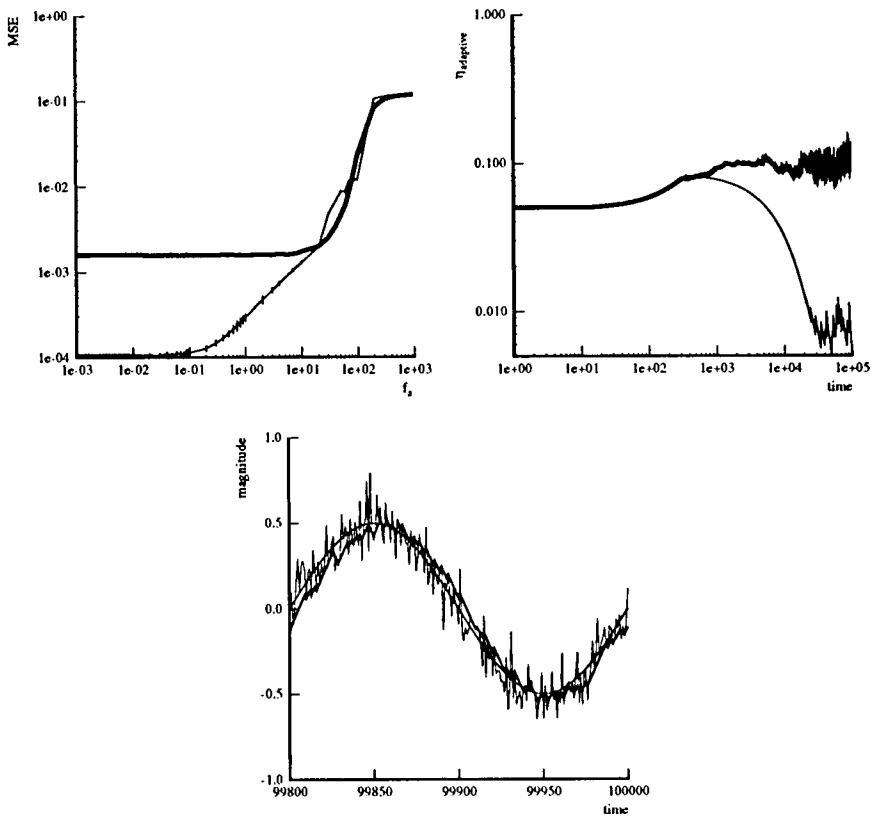


Figure 2: Noise cancelling performance of FBAR on Gaussian white noise added to a sine wave. *Top left :* MSE distortion plotted as a function of f_s, for $\eta = 0.1$ (thick line) and adaptive (thin line). Vertical bars represent standard deviations. *Top right :* Evolution of $\eta_{adaptive}$ as a function of time for $f_s = 0$ (thin line) and 100 (thick line). *Bottom :* Example of noise cancelling in case of $f_s = 50$.

ting the calculated MSE as a function of f_s. The result is shown in the top left portion of Fig. 2 (thick line) for $\eta = 0.1$ and $N = 8$. We observe that for $f_s = 0.001$ to 10, the MSE is almost the same and that starting from $f_s = 10$ it rises sharply. In the former case, the error is due to the switching nature of FBAR; in the latter case, the sharp increasing error is due to the fact that the basic assumption, the negligible influence of the signal's $p.d.f.$, fails and FBAR looses track of the exact evolution of s (though the result may still be proportional to the clean signal). Note that the MSE between s and $\hat{s}$ and between d and $\hat{d}$ are identical, as is easily verified. Hence the results in Fig. 2 (top left) also show the performance in estimating d. Now since Gaussian white noise was used for d, we conclude that FBAR is not a filtering procedure, and hence due to this feature, we conjecture that it is ideally suited for separating slow-varying signals.

LEARNING RATE ADAPTATION IN UNSUPERVISED COMPETITIVE LEARNING

One obvious way to circumvent the aforementioned shortcomings, is to make η adaptive: for low f_s, η should decease so as to produce a low quantization error; for high f_s, η should increase in order to keep track of the signal as much as possible. At first sight, this seems unfeasible in an unsupervised learning setting. However we can use some valuable *a priori* information: FBAR is aimed at producing an equiprobable codebook utilization. Hence, any divergence from this can be detected and used for adjusting η. The codebook utilization at time t can be estimated by the codebook utilization in the last T time steps using a moving average estimate:

$$\widehat{CU} \equiv \{p(Act_{R_i}, t, T) = \sum_{\tau=1}^{T} \frac{Act_{R_i}[\tau]}{TN} \mid 1 \leq i \leq N\}. \tag{8}$$

The problem is then reduced to interpreting a given divergence in equiprobable codebook utilization in terms of a change in η. A robust, yet simple solution is to use two FBAR quantizers running in parallel on the same input signal x: one with a fixed learning rate called $\eta_{reference}$, and another with a variable learning rate, called $\eta_{adaptive}$. The first one is used as a reference against which the learning rate of the second is adapted. As a measure of divergence in codebook utilization, the MAE between the present codebook utilization and an equiprobable one is taken. The MAE is determined for both the reference and the adapted quantizers: $MAE_q(CU)$, with $q \in \{reference, adapted\}$. The procedure for modifying $\eta_{adaptive}$ is as follows:

```
if (η_adaptive > η_reference)
    if (MAE_adaptive(CU) > MAE_reference(CU))
       decrease η_adaptive;
    else
       increase η_adaptive;
else
    if (MAE_adaptive(CU) > MAE_reference(CU))
       increase η_adaptive;
    else
       decrease η_adaptive;
```

The actual update is done with a leaky integrator equation:

$$\eta_{adaptive}[t] = \alpha \eta_{adaptive}[t-1] + (1-\alpha)\eta_{base} \tag{9}$$

with α a constant and η_{base} equal to 0.001 in case $\eta_{adaptive}$ is to decrease, and 1.0 in the opposite case. In order to obtain robust estimates, $\eta_{adaptive}$ is adapted only if the abovementioned conditions on MAE also hold for the previous two time steps.

As an example, consider again the previous synthetic signal. We take $\eta_{reference} = 0.3$, $\alpha = 0.9999$ and $T = 256$. Every simulation was repeated 20 times. The average result is shown in the top left part of Fig. 2 (thin line). We observe that the residual MSE is dramatically decreased for low f_s. The evolution of $\eta_{adaptive}$ for a typical run is shown in the top right part of Fig. 2 for $f_s = 0$ (thin line) and 100 (thick line). We see that $\eta_{adaptive}$ for $f_s = 0$ at first increases (the starting value = 0.05) and then gradually decreases to a low value. The first increase is interesting since it is used for rapidly positioning the boundary points within about 1,000 time steps. The evolution in case of $f_s = 100$ shows a fast increase in $\eta_{adaptive}$. The maximum range in average $\eta_{adaptive}$ thus achieved equals more than 20 times the lowest average. The bottom part of Fig. 2 shows a sample of the original signal x (thin line), the sine wave s (thin line) and its estimate $\hat{s}$ (thick line). In the case of $f_s = 0$, the MSE value equals $1.1\ ^{-4}$ and the signal-to-noise ratio 19.6 dB on average. This can be further improved by increasing T, *e.g.* for $T = 1024$ we obtain 29.4 dB, but then the MSE performance for higher f_s values decreases: since the adaptive quantizer then considers both the d and s signals as belonging to the same *p.d.f.*, $\eta_{reference}$ "cools down" to a low value for all cases. On the other hand, in case signal separation is not the issue, it also signifies that this property can be used as an automatic "cooling scheme" for adjusting the learning rate over time.

Finally, we have re-applied the previous scheme on the signal shown in Fig. 1 (top left). Note that the role of noise and signal are now reversed. The MSE of the reconstructed speech signal equals $1.8\ 10^{-3}$ for $f_s = 0$, $2.8\ 10^{-3}$ for $f_s = 1$, $5.9\ 10^{-3}$ for $f_s = 10$, and $9.0\ 10^{-2}$ for $f_s = 100$. The clean speech signal variance equals $1.9\ 10^{-2}$.

DISCUSSION

In this contribution, a fast unsupervised competitive learning rule was introduced for cancelling additive noise and seperating slow-varying signals. The rule called FBAR performs scalar quantization and yields a non-parametric model of the input *p.d.f.* by maximizing the information-theoretic entropy of the quantizer's codebook. We believe that entropy maximization offers four important advantages: 1) By maximizing entropy, the network's weights estimate medians rather than means and the former are well-known to be less sensitive to input signal outliers. 2) The maximum entropy principle often serves as a criterion to select *a priori* probability distributions when little or nothing is known: for a given amount of data, the distribution which best describes our knowledge is the one that maximizes information-theoretic entropy, subject to the given evidence as constraints. 3) Entropy maximization has been succesfully applied to obtain an optimal mapping of continuous onto discrete random variables [13]. 4) Since entropy maximization corresponds to an equiprobable quantization, the desired (optimal) result is known in advance. As a result of the latter, we were able to increase the rule's performance by using two identical configurations, one with a fixed and another with a variable learning rate. The first was used as a reference for adapting the second. This way, an explicit "cooling scheme" was not needed.

ACKNOWLEDGEMENT

The author is a postdoctoral researcher of the Belgian National Fund for Scientific Research.

REFERENCES

[1] D. Estève, F. Baillieu and G. Delapierre, "Integrated Silicon-based sensors: basic research activities in France," Sensors and Actuators A, vol. 33, pp. 1-4, 1992.

[2] J. Hérault, C. Jutten and B. Ans, "Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé," Proc. Xéme Colloque GRETSI, Nice, France, 1985, pp. 1017-1022.

[3] C. Jutten and J. Hérault, "Analog implementation of a permanent unsupervised learning algorithm," Neurocomputing, Editors: F. Fogelman and J. Hérault, NATO ASI Series, 1990, vol. F 68, pp. 145-152.

[4] R.E. Kalman, "A new approach to linear filtering and prediction problems," Trans. ASME, J. Basic Eng., vol. 82, pp. 35-45, 1960.

[5] M. Ohta and E. Uchino, "A design for a general digital filter for state estimation of an arbitrary stochastic sound system," J. Acoust. Soc. Am., vol. 80, pp. 804-812, 1986.

[6] B. Widrow and S.D. Stearns, Adaptive signal processing, Englewood Cliffs, NJ: Prentice-Hall, 1985.

[7] M. Trompf, R. Richter, H. Eckhardt and H. Hackbarth, "Combination of Distortion-Robust Feature Extraction and Neural Noise Reduction For ASR," Proc. EUROSPEECH 93, Berlin, Germany, 1993, pp. 21-23.

[8] S. Tamura, "An analysis of a Noise Reduction Neural Network," Proc. Int. Conf. Acoust. Speech Signal Processing, 1989, pp. 2001-2004.

[9] F. Xie and D. Van Compernolle, "A family of MLP based nonlinear spectral estimators for noise reduction," submitted.

[10] M.M. Van Hulle and D. Martinez, "On an unsupervised learning rule for scalar quantization following the maximum entropy principle," Neural Computation, vol. 5, pp. 939-953, 1993.

[11] M. M. Van Hulle and D. Martinez, "On a novel unsupervised competitive learning algorithm for scalar quantization," IEEE Trans. on Neural Networks, vol. 5 (3), in press.

[12] J. Makhoul, S. Roucos and H. Gish, "Vector quantization in speech coding," Proc. of the IEEE, vol. 73, pp. 1551-1588, 1985.

[13] I. Grabec, "Self-Organization of Neurons Described by the Maximum-Entropy Principle," Biol. Cybern., vol. 63, pp. 403-409, 1990.

A STATISTICAL INFERENCE BASED GROWTH CRITERION FOR THE RBF NETWORK

Visakan Kadirkamanathan
Department of Automatic Control & Systems Engineering
University of Sheffield
P.O.Box 600, Mappin Street, Sheffield S1 4DU, UK
visakan@acse.sheffield.ac.uk

Abstract. In this paper, a growth criterion is derived using statistical inference for model sufficiency. This criterion is developed for recursive estimation or sequential learning with neural networks. A growing Gaussian Radial Basis Function (GaRBF) network trained by the extended Kalman Filter (EKF) algorithm on-line, named *Incremental Network* is developed. Incremental Network is similar to the resource allocating network (RAN). The criterion for growth is based on the network prediction error and the expected uncertainty in the network output. The criterion is computed within the EKF estimation and hence no additional computations are required. This is in contrast to the need for search in the RAN formulation. The Incremental network performance on a function interpolation problem is shown to be superior in convergence speed and approximation accuracy than the RAN networks and a fixed size RBF network.

INTRODUCTION

Feedforward artificial neural networks (ANNs) are a class of models that may be used to model some unknown system or process having an unambiguous input – output mapping. The network size, often measured by the number of hidden units in a single hidden layer network, reflects the capacity of the network to approximate an arbitrary function. The problem is therefore to estimate the network parameters and its size.

The need for determining the optimal architecture or network size is due to the conflicting feature of the modelling task. Firstly, a sufficiently complex or large model is needed to ensure that the network is capable of providing an adequate approximation to the underlying process generating the observations. Secondly, an unnecessarily large model will suffer from 'over-fitting' where the network reproduces the observations but will perform poorly to unseen data.

0-7803-2026-3/94 $4.00 © 1994 IEEE

Finding the suitable network size for a given problem invariably involves a search over all possible sizes, a computationally exhaustive process considering the training times involved for each of the networks. In general, a trial and error approach is adopted in finding the suitable network size and the search is terminated as soon as a satisfactory performance is achieved.

Theoretical tools such as *Decision theory* have been used to determine an approximate rule of thumb in choosing the network size for learning a given number of observations [Baum & Haussler, 1989]. Recently, Bayesian statistics has provided a general framework or procedure, namely *Bayesian model comparison* for determining the most probable network among those investigated [MacKay, 1992]. For all the above procedures, the data must be available *en-bloc* and rely on the arbitrary selection of appropriate sizes for investigation. A form of limited search over different network sizes combined with model selection based on *Minimum Description Length* (MDL) was also developed [Smyth, 1991]. The task of finding the optimal network size is even more difficult in a recursive or sequential estimation problem.

These observations have led to investigations into dynamic architecture networks, where, instead of searching over different size networks, a network is constructed as part of the training procedure. The two approaches of network construction are

- Choose a large network and prune it by deleting units. eg. Skeletonization [Mozer & Smolensky, 1989], optimal brain damage [LeCun, Denker & Solla, 1990], weight elimination [Weigend, Rumelhart & Huberman, 1991].

- Choose a small network and grow it by adding units. eg. Cascade correlation [Fahlman & Lebierre, 1990], resource allocating network [Platt, 1991].

The growth criteria in most of these approaches are based on heuristics, such as the increase or decrease in the approximation error by addition or deletion of units. In optimal brain damage [LeCun et al.,1990], however, the deletion of the units are based on the Hessian of the error surface with respect to the network parameters. Except the RAN, all the other networks require all data to be available together.

In this paper, we provide a criterion for growth based on the statistical inference of model sufficiency. The notion of model sufficiency is that "the model with given size is deemed to be sufficient if the prediction error on the data is within a certain level of confidence exhibited by the network". For example, if the uncertainty in the network parameters are high, the network may exhibit large prediction errors which are within the expectations of the network and hence no new units will be added. The estimation is carried out using the *extended Kalman filter* (EKF) which also provides an estimate for the network uncertainty.

THE GROWTH CRITERION

Consider an interpolation problem where for the input – output observation set $\{(\mathbf{x}_n, y_n)|n = 1, \ldots, N\}$, $\mathbf{x}_n \in \Re^M$ and $y_n \in \Re$,

$$y_n = f_*(\mathbf{x}_n) + \eta \tag{1}$$

where η is a zero mean Gaussian noise with variance σ^2_{ns} and $f_*(.)$ is the underlying function. Let the network chosen to approximate the underlying function provide an input – output mapping described by $f(\mathbf{x}; \mathbf{p})$, where $\mathbf{p}$ is the vector consisting of network parameters being adapted.

For a trained network where the parameter $\mathbf{p}$ has been estimated along with its *error covariance matrix*, the network output uncertainty can be determined. The measure of uncertainty is the variance of the network output, given by,

$$\sigma^2_y(\mathbf{x}) = Var[f(\mathbf{x}; \mathbf{p})] \tag{2}$$

Under Gaussian assumptions, the network output can be described by the Gaussian probability distribution with mean $f(\mathbf{x}; \mathbf{p})$ and variance $\sigma^2_y(\mathbf{x})$.

If the network used to interpolate the underlying function is of sufficient size, under Gaussian assumptions, we would expect the interpolation error or the *prediction error* to lie within a bound determined by the network uncertainty and noise variance for a certain percent of the data, with a certain level of confidence. The null hypothesis for the statistical inference of model sufficiency is stated as follows:

$$\mathcal{H}_0 : \frac{|e|}{\sqrt{Var[f(\mathbf{x}; \mathbf{p}) + \eta]}} = \frac{|e|}{\sqrt{\sigma^2_y(\mathbf{x}) + \sigma^2_{ns}}} < z_\alpha \qquad \text{for } \alpha\% \text{ of data} \tag{3}$$

where z_α is the value of the Z-statistic at $\alpha\%$ level of significance and e is the prediction error for the observation $(\mathbf{x}, y)$, given by,

$$e = y - f(\mathbf{x}; \mathbf{p}) \tag{4}$$

If the condition for $\mathcal{H}_0$ is violated, the null hypothesis that the model is sufficient is rejected. The rejection implies that the network complexity must be increased to match the complexity of the underlying function, and hence a new unit or basis function is added to the model. The addition of a new unit increases the complexity of the network so that its capacity to be a sufficient model is increased. Note that a criteria for pruning the network has also been suggested from statistical inference, based on the estimated parameter uncertainties [Buntine & Weigend, 1992].

RECURSIVE ESTIMATION

Recursive estimation (sequential or on-line learning) with ANNs requires a posterior estimate of the underlying function to be obtained from its prior estimate and the current or new input – output observation. For a network of fixed number of parameters being adapted, this becomes a problem of estimating the parameters recursively. Recursive parameter estimation fits into the Bayesian statistical framework naturally.

Consider the Gaussian radial basis function (GaRBF) network whose input – output mapping is given by,

$$f(\mathbf{x};\mathbf{p}) = \sum_{k=1}^{K} w_k b_k(\mathbf{x};\mathbf{u}_k) = \mathbf{b}^T\mathbf{w} \tag{5}$$

where $\mathbf{w} = [w_1, \ldots, w_K]^T$ are the linear coefficients, $\mathbf{b} = [\ldots, b_k(\mathbf{x};\mathbf{u}_k), \ldots]^T$ are the K basis functions constructed at the hidden layer and $\mathbf{u}_k = [u_{k1}, \ldots, u_{kM}]^T$ is the kth RBF unit centre. The basis functions are parametrised through $\mathbf{u}_k$. In general, the output of an ANN can be written as a linear combination of a set of basis functions as given in (5). The basis functions in the GaRBF network have the form,

$$b_k(\mathbf{x};\mathbf{u}_k) = \exp\left\{-\frac{1}{r_k^2}||\mathbf{x}-\mathbf{u}_k||^2\right\} \tag{6}$$

If we choose to adapt only the linear coefficients $\mathbf{w}$, under Gaussian assumptions, the prior and posterior probability distributions for $\mathbf{w}$ can be described by Gaussian distribution. Let the prior be a distribution with mean $\mathbf{w}_{n-1}$ and covariance matrix $\mathbf{P}_{n-1}^{-1}$, the posterior with mean $\mathbf{w}_n$ and covariance matrix $\mathbf{P}_n^{-1}$ and the likelihood distribution for the observation y be with mean $\mathbf{b}_n^T\mathbf{w}_{n-1}$ and variance $R_n = \sigma_{ns}^2$, where $\mathbf{b}_n = [\ldots, b_k(\mathbf{x}_n;\mathbf{u}_k)\ldots]^T$. Applying Bayes' theorem, expressions for $\mathbf{w}_n$ and $\mathbf{P}_n^{-1}$ can be obtained. This is in fact the Kalman filter algorithm [Candy, 1986], where,

$$\begin{aligned}
e_n &= y_n - f(\mathbf{x}_n;\mathbf{p}_{n-1}) = y_n - \mathbf{b}_n^T\mathbf{w}_{n-1} && (7)\\
\mathbf{w}_n &= \mathbf{w}_{n-1} + e_n\mathbf{k}_n && (8)\\
R_y &= \left[R_n + \mathbf{b}_n^T\mathbf{P}_{n-1}\mathbf{b}_n\right] && (9)\\
\mathbf{k}_n &= R_y^{-1}\mathbf{P}_{n-1}\mathbf{b}_n && (10)\\
\mathbf{P}_n &= \left[\mathbf{I} - \mathbf{k}_n\mathbf{b}_n^T\right]\mathbf{P}_{n-1} + Q_0\mathbf{I} && (11)
\end{aligned}$$

where Q_0 is a scalar that allows a small random variation to the parameters being adapted. This random walk model allows the parameters to continue adapting to new observations. The subscripts (n) and $(n-1)$ denote the posterior and prior estimates respectively. The vector $\mathbf{k}_n$ is the Kalman gain.

The matrix $\mathbf{P}$ represents the uncertainty in the estimated parameters while R_y reflects the uncertainty in the expected output for the given input observation, *ie.*,

$$R_y = Var[f(\mathbf{x};\mathbf{p}) + \eta] \tag{12}$$

The growth criterion proposed in the last section has to be modified for the recursive or sequential estimation problem, where there is access to only the current observation. The model is deemed sufficient "if the prediction error on the new observation is within that expected by the network with a certain level of confidence", so that sufficiency is determined from the current observation alone. Substituting for the output uncertainty from (12), the null hypothesis of model sufficiency is now:

$$\mathcal{H}_0 : \frac{|e_n|}{\sqrt{R_y}} < z_\alpha \tag{13}$$

The terms e_n and R_y are computed as part of the Kalman filter estimation algorithm and hence there is no computational overhead in testing for model sufficiency.

The estimation and growth criterion of the model with nonlinearly appearing coefficients, such as $\mathbf{u}_k$ in the GaRBF network, can be extended where estimation is carried out by the extended Kalman filter (EKF) algorithm and model sufficiency tested similarly. The only difference in the computation is the replacing of $\mathbf{w}$ and $\mathbf{b}_n$ by $\mathbf{p}$ and the vector $\mathbf{d}_n$ respectively, where $\mathbf{d}_n = \nabla_p f(\mathbf{x}_n; \mathbf{p})$ is the gradient of $f(.)$ with respect to $\mathbf{p}$ evaluated with $\mathbf{p}_{n-1}$. This is equivalent to approximating the probability distributions to be Gaussian around the estimates.

THE INCREMENTAL NETWORK

The growth criterion developed above is independent of the model structure, even though it was demonstrated on the GaRBF network, and hence is applicable to any type of model. Now, the question of what type of basis function to be added, if the model is deemed insufficient, has to be addressed. In a block estimation environment there will be no restriction on the type of basis function. However, in a recursive estimation problem, the addition must be a localised basis function, which while ensuring localisation of the current observation is also nearly orthogonal to the existing basis functions. It was shown that GaRBF functions, specifically the basis functions allocated by the RAN, was observed to exhibit these properties [Kadirkamanathan 1991; Kadirkamanathan & Niranjan 1993].

The network based on the statistical inference growth criterion and the RAN basis function allocation, is named *Incremental Network* (IncNet). The IncNet incorporates the advantages of using EKF for near optimal estimation and the growth criterion that detects model insufficiency. It is essentially a Gaussian RBF network whose coefficients ($\mathbf{w}$), unit means ($\mathbf{u}_k$) and variances (r_k) are estimated. When the model sufficiency null hypothesis is rejected for the nth observation, the new $(K+1)$th basis function allocated is a Gaussian RBF, whose parameters are assigned as follows (similar to RAN):

$$w_{K+1} = e_n \qquad \mathbf{u}_{K+1} = \mathbf{x}_n \qquad r_{K+1} = r_0 \tag{14}$$

with r_0 being an appropriate constant. The EKF estimation algorithm has to be modified to accommodate the increase in parameters. The parameter vector $\mathbf{p}_n$ simply tags the new parameters to the existing ones,

$$\mathbf{p}_n = \left[\mathbf{p}_{n-1}^T, w_{K+1}, \mathbf{u}_{K+1}^T, r_{K+1}\right]^T \tag{15}$$

and the parameter error covariance matrix becomes,

$$\mathbf{P}_n = \begin{pmatrix} \mathbf{P}_{n-1} & 0 \\ 0 & P_0\mathbf{I} \end{pmatrix} \tag{16}$$

where P_0 is an estimate of the uncertainty in the initial values assigned to the parameters and $\mathbf{I}$ is an identity matrix of dimension $(M+2) \times (M+2)$, where M is the dimensionality of the input space.

Its operation can be summarised as follows:

- Given $(\mathbf{x}_1, y_1)$, Choose the first basis function with parameters $w_1 = y_1$, $\mathbf{u}_1 = \mathbf{x}_1$, $r_1 = r_0$ and $\mathbf{P}_0 = P_0\mathbf{I}$ with dimensionality $(M+2) \times (M+2)$.
- For each data $(\mathbf{x}_n, y_n)$, $n = 2, \ldots, N$, use EKF estimation to determine $R_y = \mathbf{d}_n^T \mathbf{P}_{n-1} \mathbf{d}_n + R_n$. Also determine the predicted error $e_n = y_n - f(\mathbf{x}_n; \mathbf{p}_{n-1})$.
  - If $|e_n| < z_\alpha \sqrt{R_y}$, continue with EKF estimation to determine $\mathbf{p}_n$ and $\mathbf{P}_n$.
  - If $|e_n| > z_\alpha \sqrt{R_y}$, add a new hidden unit with $w_{K+1} = e_n$, $\mathbf{u}_{K+1} = \mathbf{x}_n$, $r_{K+1} = r_0$ and the dimensionality of $\mathbf{P}_{n-1}$ is increased by $M+2$ rows and columns with diagonal elements of P_0 and 0 elsewhere. The EKF estimation is then applied to determine the posterior $\mathbf{p}_n$.

The difference between the IncNet and the RAN is mainly in the growth criterion. The RAN decides to add a new unit based on the novelty of the current pattern [Platt, 1991]. The novelty is determined by the two criteria: $|e_n| > e_{\min}$ and $||\mathbf{x}_n - \mathbf{u}_0|| > \epsilon_n$, where $e_{\min}$ is a measure of the desired accuracy, $\mathbf{u}_0$ the nearest Gaussian centre to $\mathbf{x}_n$ and ϵ_n is decreased with time allowing the network to form finer and finer approximation. Finding the nearest Gaussian centre involves a search which adds to the computational overhead.

IncNet on the other hand, adds a new unit if the prediction error is not within the statistical expected bounds. As a result, new units are added only if the existing parameters have been estimated with high confidence and the network complexity is not sufficient to make the errors sufficiently small. Unlike in the RAN, the IncNet growth criterion on the prediction error bound begins at a high value and decreases with training and increases after the addition of new basis functions. A new unit will also be added if the observations do not contribute to the basis functions keeping $\mathbf{b}_n$ sufficiently small, ensuring a novelty detection similar to the RAN's second criterion.

The RAN also uses LMS algorithm to adapt the parameters when a new unit is not added. The convergence of RAN can be increased by the use EKF, this extended network referred to as RAN-EKF. [Kadirkamanathan & Niranjan 1993]. Note that we can choose to adapt either only the linear coefficients $\mathbf{w}$ or the parameters $\mathbf{p} = [\mathbf{w}, \ldots, \mathbf{u}_k, \ldots]$ for the IncNet depending upon the application.

EXPERIMENTAL RESULTS

The performance of the incremental network (IncNet) was compared to the different forms of the resource allocating network, namely the RAN [Platt, 1991] and RAN-EKF [Kadirkamanathan 1991]. The problem chosen for the investigation is the Hermite function interpolation problem given in [MacKay, 1992], where the underlying function generating the observations are

$$f_*(x) = 1.1(1 - x + 2x^2) \exp\left\{-\tfrac{1}{2}x^2\right\} \tag{17}$$

The training data comprises of 40 observations generated randomly in the interval [-4,+4]. During training, the samples were presented one by one

and repeated for 20 cycles. The test data contains 200 observations sampled uniformly in the same interval. The accuracy of the approximation achieved by the networks were measured by the root mean square error (RMSE) over the test data. Noise was added only to the training data. For details of the network parameters and training algorithms for RAN and RAN-EKF, refer to [Kadirkamanathan & Niranjan, 1993]; The IncNet parameters for the EKF algorithm are the same as for RAN-EKF with $z_\alpha = 2$, $r_0 = 1.0$.

In the first experiment, the RAN, RAN-EKF and the IncNet were trained on a noisy data and their on-line performance are shown in Figure 1. The results clearly show the fast convergence achieved by the IncNet while RAN-EKF converged faster than the RAN. This is to be expected since RAN uses the LMS adaptation which is computationally simpler in comparison to the RAN-EKF which uses the computationally complex EKF algorithm. Note also that the growth pattern for the RANs differed significantly from that of IncNet.

In order to investigate the robustness of the incremental network, varying levels of Gaussian noise were added to the training data, where the noise variance was increased from 0.0001 to 0.1. Since RAN-EKF performs better than the RAN, the RAN was not used in this second set of experiments. A fixed size Gaussian radial basis function (GaRBF) network of 16 hidden units was used in these tests to evaluate the advantages of using growing networks. This GaRBF network was trained using the EKF algorithm and is denoted by RBF-EKF.

Figure 2(a) shows the network sizes achieved by the networks with increasing noise variance while Figure 2(b) shows the network approximation error in finding the underlying function. The results clearly demonstrate the superior performance of the IncNet over the RAN-EKF which in turn performed better than the fixed size RBF-EKF. The IncNet formed fewer units while achieving the best approximation amongst the networks. It should be noted that with high level of noise, the IncNet and RAN-EKF parameters (R_n and $e_{\min}$ respectively) should be modified accordingly, but was not done in the experiments – hence the relatively poor results at high noise. The results however demonstrate that to some degree, an overestimation of the noise level is unlikely to affect the final network size for IncNet and RAN-EKF.

CONCLUSIONS

A growth criterion has been developed from statistical inference of model sufficiency. Its application to the recursive estimation or on-line modelling problem led to the development of the *Incremental Network*. This Gaussian radial basis function network is similar to the RAN in allocation of new units and to its extension RAN-EKF in using the extended Kalman filter algorithm. The growth criterion however is different for IncNet.

Performance on a function interpolation problem demonstrated the ability of the Incremental Network to form compact network with good approximation. It is evident from the experimental results that the Incremental Network was able to use its basis functions effectively, whereas the RAN failed to use

the statistical information and hence assigned more basis functions than was needed. The approximation error was also higher for RAN. The comparison with RBF-EKF of 16 units also show that the growing networks provide a better approximation while determining the appropriate complexity.

Acknowledgements

The author is grateful for the support of the Engineering and Physical Sciences Research Council (EPSRC) of UK under the grant GR/J46661.

References

[1] Baum E., & Haussler D., "What size net gives valid generalization", *Neural Computation*, Vol.1, No.1, 1989.

[2] Buntine W.L., & Weigend A.S., "Bayesian backpropagation", *Complex Systems*, Vol.2 pp 321-355, 1992.

[3] Candy J.V., "Signal processing: The model based approach", McGraw-Hill, New York, 1986.

[4] Fahlman S.E., & Lebiere C., "The cascade-correlation architecture", In Touretzky D.S. (ed.), *Neural Information Processing Systems 2*, Morgan Kaufmann, CA: San Mateo, 1990.

[5] LeCun Y., Denker S. & Solla S.A., "Optimal brain damage", In Touretzky D.S. (ed.), *Neural Information Processing Systems 2*, Morgan Kaufmann, CA: San Mateo, 1990.

[6] MacKay D.J.C., "Bayesian interpolation", *Neural Computation*, Vol. 4, No.3, pp. 415-447, 1992.

[7] Mozer M.C., & Smolensky.P, "Skeletonization: A technique for trimming the fat from a network via relevance assignment", In Touretzky D.S. (ed.), *Neural Information Processing Systems 1*, Morgan Kaufmann, CA: San Mateo, 1989.

[8] Kadirkamanathan V., *Sequential learning in artificial neural networks*, PhD Thesis, Cambridge University Engineering Department, 1991.

[9] Kadirkamanathan V. & Niranjan M., "A function estimation approach to sequential learning with neural networks", *Neural Computation*, Vol. 5, pp 954-975, 1993.

[10] Platt J., "Resource allocating network for function interpolation", *Neural Computation*, Vol. 3, No. 2, pp.213-225, 1991.

[11] Smyth P., "On stochastic complexity and admissible models for neural network classifiers", In Lippmann R., Moody J., Touretzky D.S. (eds.), *Neural Information Processing Systems 3*, Morgan Kaufmann, CA: San Mateo, 1991.

[12] Weigend A.S., Rumelhart D.E. & Huberman B., "Generalization by weight elimination with application to forecasting", In Lippmann R.P., Moody J. & Touretzky D.S. (eds.), *Neural Information Processing Systems 3*, Morgan Kaufmann, CA: San Mateo, 1991.

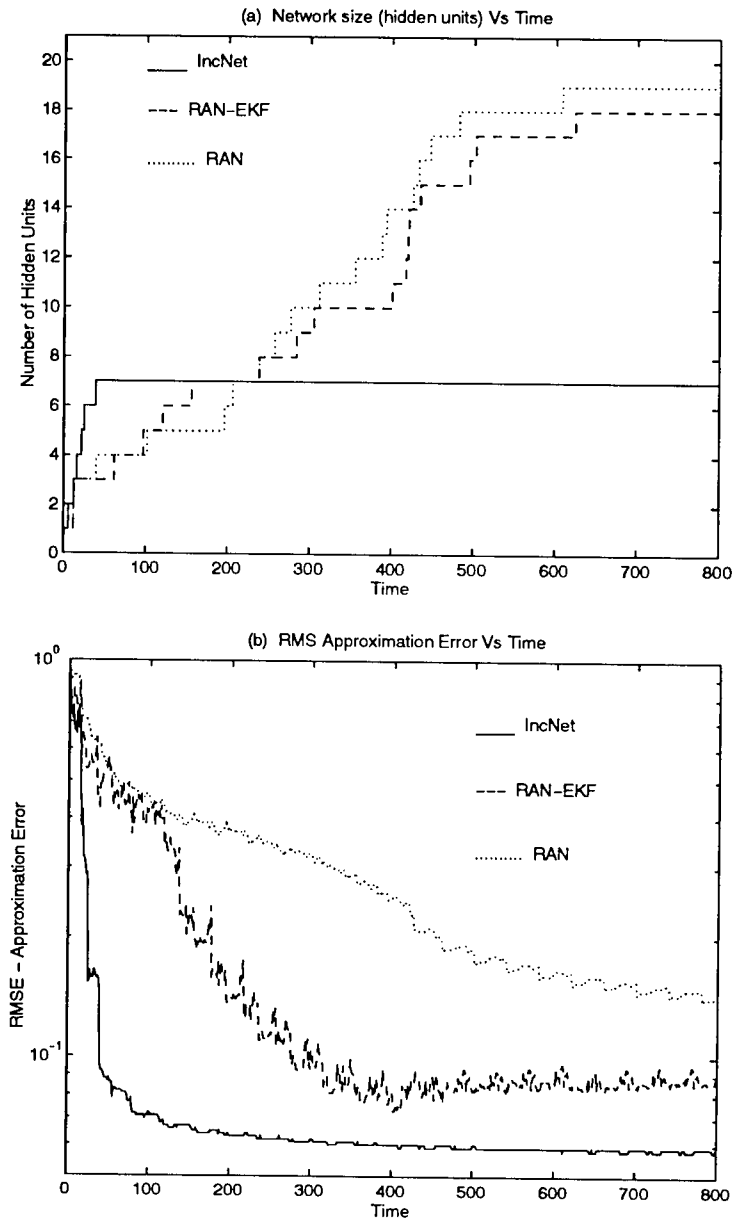


Figure 1: On-line performance of INet, RAN-EKF, RAN: (a) Growth Pattern (b) Approximation Error.

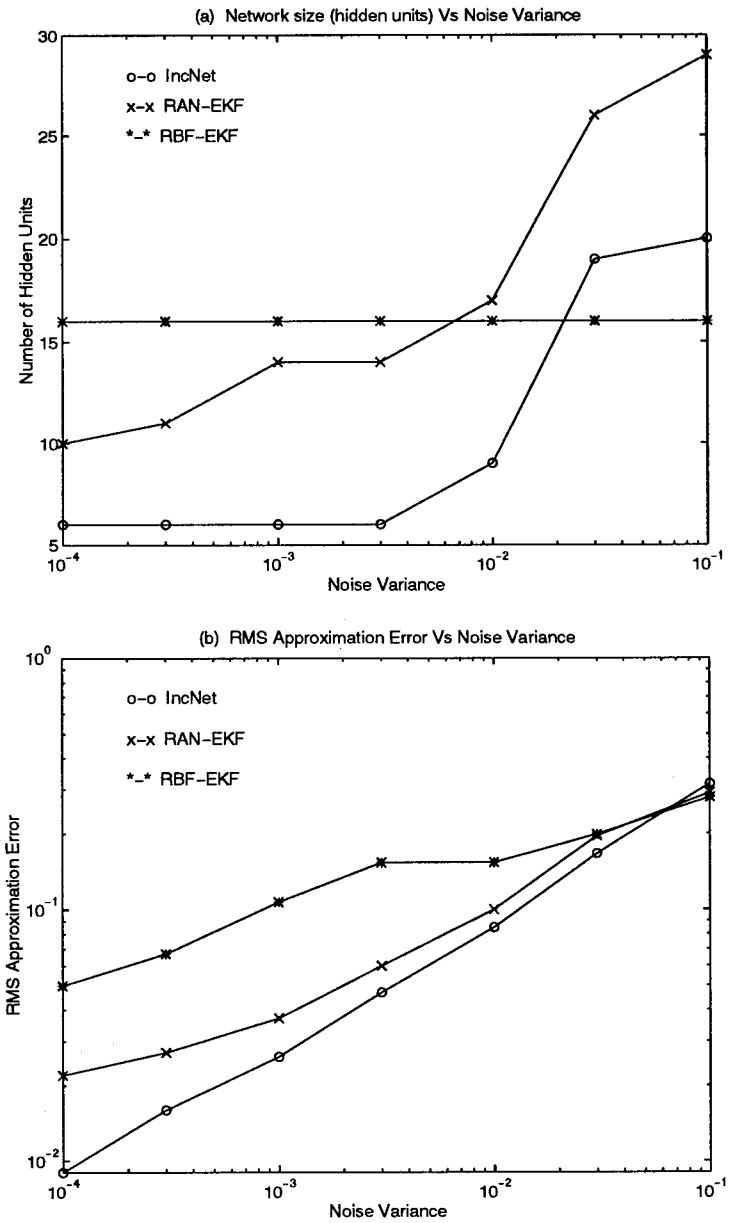


Figure 2: The effect of noise on the network sizes and approximation accuracy for INet, RAN-EKF, RBF-EKF: (a) Growth Pattern (b) Approximation Error.

NEURAL NETWORK INVERSION TECHNIQUES FOR EM TRAINING AND TESTING OF INCOMPLETE DATA

Jenq-Neng Hwang, Chien-Jen Wang

Information Processing Laboratory
Dept. of Electrical Engr., FT-10
University of Washington
Seattle, WA 98195, USA
Tel: (206) 685-1603, Fax: (206) 543-3842
e-mail: hwang@ee.washington.edu

Abstract

The expectation-maximization (EM) algorithm is a successful statistical approach for maximum likelihood estimation of incomplete-data problems. The performance of an EM algorithm highly depends on assumptions made about the probability density function (commonly, the multivariate Gaussian) of the multivariate data. When the EM algorithm is used for classification applications, it is commonly done by replacing the missing values based on the estimated probability density function of the same class for getting the maximum likelihood labeling without jointly considering the discrimination among classes. In this paper, we propose an EM procedure based on a neural network inversion technique for improving the training accuracy using incomplete data sets and the classification accuracy in testing new incomplete data. Our approach relaxes the assumption made about the probability density function, and more importantly, the missing value replacements take into account of the discrimination among classes.

0-7803-2026-3/94 $4.00 © 1994 IEEE

1 Introduction

Real-world regression and classification tasks may involve high dimensional data sets with arbitrary patterns of missing elements. Take for examples, in remote sensing applications, there has been a large set of satellite brightness measurement data (e.g., SSMI remote sensing measurements) with their corresponding experimentally measured geophysical parameters (ground-truth data) being available [1]. Unfortunately, a lot of these ground-truth data have one or several missing elements and thus make them infeasible for use as valid training data in regressing the nonlinear function of the scattering environment. Another example of such an incomplete data set is the "heart-disease" data set from the UCI machine learning database where 920 records in total are available for 5 categories of heart diseases with 14 attributes each. There are only 299 of the records are complete, the others have one or several missing attribute values (11% of all values are missing).

The *expectation-maximization* (EM) algorithm is a very general iterative algorithm for *maximum likelihood* (ML) estimation in incomplete-data problem [2, 9]. Given the observed elements $\mathrm{x}^{(o)}$ of an incomplete datum $\mathrm{x} = [\mathrm{x}^{(o)}, \mathrm{x}^{(m)}]$, the EM algorithm acquires the missing elements $\mathrm{x}^{(m)}$ by

1. estimating the distribution parameters θ (e.g., the mean vector and the covariance matrix) of the presumed multivariate probability density function $P(\mathrm{x}|\theta)$ based on the set of complete multivariate data $\{\mathrm{x}\}$ (and/or the available observed elements of the incomplete data); then

2. conditioned on the estimated distribution parameters $\hat{\theta}$, replacing the missing elements with conditional expectation values $\hat{\mathrm{x}}^{(m)} = E[\mathrm{x}^{(m)}| \mathrm{x}^{(o)}, \hat{\theta}]$; further

3. combining the originally complete and the newly completed data to reestimate the distribution parameters θ; and then

4. replacing the missing elements with new conditional expectation values based on the newly reestimated distribution parameters,

and so forth, iterating until convergence.

The performance of an EM algorithm highly depends on assumptions made about $P(\mathrm{x}|\theta)$ (commonly, the multivariate Gaussian) of the multivariate data. To relax this critical performance sensitivity to the presumed distribution, multivariate mixture Gaussian formulations have also been proposed as a more general model-free estimation of density function for EM applications [7, 4].

When the EM algorithm is used for classification applications, where the training/testing data to be trained/classified contain missing elements, it is commonly done by replacing the missing elements based on the estimated probability density function $P(\mathrm{x}|\theta_k)$ of the same class, say k-th, to

get the maximum likelihood labeling without jointly considering the discrimination among classes. More specifically, in the training phase, the missing elements of an incomplete datum with the known class are replaced based on the reestimated distribution parameters $\theta_{\mathbf{k}}$ of the corresponding class. These two processes iterate until convergence and the distribution functions $\{P(\mathrm{x}|\theta_k),\ k = 1, 2, ..., K\}$ can thus be used directly in a maximum likelihood (ML) classification framework. If a deterministic classification framework is preferred, then the converged and missing-element replaced incomplete data as well as the originally complete data can be used to train a deterministic classifier (e.g., a neural network) for future classification. The above procedure (either for a probabilistic or deterministic classifier) is called in our paper as *EM training* with incomplete data. After a classifier is built and ready for testing, the new inputs might contain missing elements. In this case, we no longer reestimate the distribution parameters during the process of replacing the missing elements. We simply test the hypothesis of the incomplete data being created from one of the distribution functions (i.e., one of the classes) and replace the missing elements based on the corresponding distribution functions. This procedure is entitled as *EM testing* with incomplete data in our paper. Note that in the above discussions, the EM training/testing procedures seldom consider the interactions of data among different classes in either reestimating the distribution parameters or replacing the missing elements.

In this paper, we propose the EM training and the EM testing procedures based on a neural network inversion technique for improving the training accuracy using incomplete data and the classification accuracy in testing new incomplete data. A similar approach based on exhaustive search of missing values instead of our proposed systematic network inversion method has been proposed [12]. Our approach alleviates the great sensitivity of the classifier performance to the assumption made about the probability density function, and more importantly, the missing value replacements take into account of the data interaction (discrimination) among different classes. Section 2 will give a brief review of the neural network inversion technique and its relationship to the *maximum a posterior* (MAP) estimation of the missing elements. The application of the proposed EM training and EM testing procedures to the classification of IRIS data is presented in Section 3. Finally, in Section 5, the concluding remarks are given.

2 Network Inversion of an MLP

The forward system dynamics in the retrieving phase of an L-layer feedforward multilayer perceptron (MLP) can be described by the following iterative

equations (for $1 \leq i \leq N_{l+1}, \quad 0 \leq l \leq L-1$):

$$\begin{aligned} u_i(l+1) &= \sum_{j=1}^{N_l} w_{ij}(l+1)a_j(l) + \theta_i(l+1) = \sum_{j=0}^{N_l} w_{ij}(l+1)a_j(l) \\ a_i(l+1) &= f(u_i(l+1)) \end{aligned} \tag{1}$$

where $a_j(l)$ ($u_j(l)$) denotes the activation value (net input) of the jth neuron at the lth layer; $\theta_j(l)$ (or $w_{i0}(l)$) denotes the bias of the jth neuron at the lth layer; $w_{ij}(l)$ denotes the weight value linked between the ith neuron at the lth layer and the jth neuron at the $(l-1)^{th}$ layer; and f is the nonlinear activation function (usually sigmoid). The inputs x are denoted as $\{a_j(0), \forall j\}$, and the outputs y are denoted as $\{a_i(L), \forall i\}$.

2.1 Back-Propagation Network Learning

The learning phase of an MLP uses the back propagation learning rule, an iterative gradient descent algorithm designed to minimize the mean squared error E between the the desired target vector $\{t_i\}$ and the actual output vector $\{a_i(L)\}$ [11]:

$$w_{ij}(l) \Longleftarrow w_{ij}(l) - \eta \frac{\partial E}{\partial w_{ij}(l)} \tag{2}$$

where

$$E = E(\{w_{ij}(l)\}, \{a_j(0)\}) = \frac{1}{2} \sum_{i=1}^{N_L} (t_i - a_i(L))^2 = \frac{1}{2} \sum_{i=1}^{N_L} (t_i - y_i)^2. \tag{3}$$

2.2 Network Inversion of an MLP

The inversion of a network will generate the input vector $\mathbf{x} = \{a_j(0)\}$ that can produce a desired output vector. By taking advantage of the *duality* between the weights and the input activation values in minimizing the mean squared error E (see Equation (3)), the iterative gradient descent algorithm can again be applied to obtain the desired input vector x [8, 5].

$$a_j(0) \Longleftarrow a_j(0) - \eta \frac{\partial E}{\partial a_j(0)} = x_j - \eta \frac{\partial E}{\partial x_j}, \quad \forall j \tag{4}$$

The idea is similar to the back-propagation algorithm, where the error signals are propagated back to tell the weights the manner in which to change in order to decrease the output error. The inversion algorithm back-propagates the error signals to the input layer to update the activation values of input units so that the output error is decreased. In order to avoid the input activation values, $\{a_j(0)\}$, from growing without limits, a small modification of

the updating rule was usually made;

$$u_j(0) \Longleftarrow u_j(0) - \eta \frac{\partial E}{\partial u_j(0)}, \quad \forall j \tag{5}$$

where $u_j(0) = f^{-1}(a_j(0))$ is a "pseudo" net input created to allow flexible gradient descent search without limiting the dynamic ranges (e.g., usually we assume $0 \leq a_j(0) \leq 1$).

2.3 Maximum A Posterior (MAP) Classifier Training

In a classification application, it is normally assumed that the input vector, $\mathrm{x} \in \mathbb{R}^n$, belongs to one of K classes, C_k, $1 \leq k \leq K$. The main objective of a classification task is to decide to which of the K classes the vector x belongs. The decision can be made based on some forms of deterministic discriminant function, e.g., the Euclidean distance measure. A more general decision rule is based on the probabilistic decision, such as the *maximum a posteriori* (MAP) approach which guarantees the minimum classification error. In a MAP approach, for each of the classes one requires to estimate the posterior probability, $P(C_k|\mathrm{x})$, which is usually computed via the Bayes' rule:

$$P(C_k|\mathrm{x}) = \frac{P(\mathrm{x}|C_k)P(C_k)}{P(\mathrm{x})} \propto P(\mathrm{x}|C_k)P(C_k) \tag{6}$$

where $P(\mathrm{x}|C_k)$ is the conditional distribution (also known as likelihood) and $P(C_k)$ is the *a priori* probability of the class C_k.

Since $P(C_k)$ is relatively easier to compute, most conventional pattern recognition literatures have been focusing on the research of estimating the likelihood $P(\mathrm{x}|C_k)$. On the other hand, when an MLP is used for this classification task, there is usually an input layer of n neurons corresponding to the n-dimensional input vector x, one or two layers of "appropriately chosen" hidden neurons, and one output layer of K neurons with each one representative of one of the K different classes (e.g., the desired binary output vector for 1st class is $\mathrm{t} = [1, 0, 0, ..., 0]$, for 2nd class is $\mathrm{t} = [0, 1, 0, ..., 0]$, etc.). It has been shown that the continuous-valued output activations $\mathrm{y} = (y_1, y_2, ..., y_K)$ of an MLP trained by the standard back-propagation learning, which minimizes the mean squared error (MSE) between the actual outputs y and the desired binary targets t, can be directly interpreted as a least squares estimate of the posterior probabilities $\{P(C_k|\mathrm{x}),\ k = 1, ..., K\}$ [10, 13].

2.4 Maximum A Posterior Estimation of Missing Elements from Network Inversion

In a neural network based classification task, to find the missing elements $\mathrm{x}^{(m)}$, given the observed elements $\mathrm{x}^{(o)}$ and the hypothesized class C_k of the incomplete data x, the network inversion algorithm can be applied in

a straightforward manner. Basically, we perform the gradient descent search on the missing elements $\mathbf{x}^{(m)} = \{x_j^{(m)}\}$:

$$x_j^{(m)} \Longleftarrow x_j^{(m)} - \eta \frac{\partial E}{\partial x_j^{(m)}}, \quad \forall j, \tag{7}$$

while keeping the observed elements $\mathbf{x}^{(o)}$ intact during the iterative inversion of the missing elements $\{x_j^{(m)}\}$.

It is interesting to note that simply performing the iterative inversion of the missing elements maximizes the *posterior* classification probability $P(C_k|\mathbf{x}) = P(C_k|\mathbf{x}^{(m)}, \mathbf{x}^{(o)})$, but not actually maximizing the posterior probability of estimating the missing elements $P(\mathbf{x}^{(m)}|\mathbf{x}^{(o)}, C_k)$. More specifically, it can be easily shown by Bayes rule that

$$P(\mathbf{x}^{(m)}|\mathbf{x}^{(o)}, C_k) \propto P(C_k|\mathbf{x}^{(m)}, \mathbf{x}^{(o)})\ P(\mathbf{x}^{(m)}|\mathbf{x}^{(o)}). \tag{8}$$

Therefore, the iterative inversion search can be regarded as a MAP search of the missing elements $\mathbf{x}^{(m)}$ under the approximate assumption that $P(\mathbf{x}^{(m)}|\mathbf{x}^{(o)})$ is uniform. On the other hand, the standard EM algorithm finds the conditional expectation values $E[\mathbf{x}^{(m)}|\mathbf{x}^{(o)}, C_k]$ for the missing elements. In the case of the Gaussian distribution, this replacement can be regarded as being estimated via a *maximum likelihood* (ML) criterion.

A more correct MAP approach to the estimation of $\mathbf{x}^{(m)}$ would be the maximization of the product term of Eq. (8), where an adequate and differentiable formulation of $P(\mathbf{x}^{(m)}|\mathbf{x}^{(o)})$ is required.

3 EM Testing and Training via Network Inversion

We used in our simulations the IRIS data set [3], the best known database in pattern recognition literature, which contains 3 classes of 50 4-dimensional instances each, and each class refers to a type of IRIS plant.

3.1 EM Testing of IRIS Data

We trained an one-hidden layer MLP (4 inputs, 2 hidden units, and 3 outputs) with 90 IRIS complete training data (30 data for each class). The accuracy of this network reached 98.3% after 1000 sweeps of training when tested on the remaining 60 complete data (20 data for each class). Based on the trained neural network, we then tested on data with missing elements of size one or two created from these 60 complete data. To have a statistically significant testing, "all" possible missing patterns were generated. For examples, in case of one missing element, we generated 240 data (4 possible missing patterns for each testing data); while in case of two missing elements, we generated 360 data (6 possible patterns for each testing data).

Classification Accuracy	Standard EM	Inversion EM
No Missing (60 data)	98.3 %	98.3 %
One Missing (240 data)	96.5 %	96.5 %
Two Missing (360 data)	83.0 %	86.0 %

Table 1: Comparative simulation results (percentages of classification accuracy) for IRIS data classification using EM testing.

Two approaches were used in the comparative studies for EM testing. The first one is the standard EM testing, where the missing elements were replaced by the conditional expectations $E[\mathrm{x}^{(m)}|\mathrm{x}^{(o)}, C_k]$ of all three classes $k = 1, 2, 3$. The joint data distributions $P(\mathrm{x}|C_k)$ are assumed to be single-mode Gaussian distributed (it is also possible to use mixture Gaussian distribution [4] or more sophisticated nonparametric density estimator, e.g., projection pursuit density [6]), i.e.,

$$P(\mathrm{x}|C_k) = P(\mathrm{x}^{(m)}, \mathrm{x}^{(o)}|C_k) = N(\mu_k, \Sigma_k), \qquad k = 1, 2, 3, \tag{9}$$

where the mean vector μ_k and covariance matrix Σ_k were pre-estimated based on the 30 complete training data for k-th class (no further reestimations based on the incomplete data were done).

After the missing elements were replaced, we then tested the newly completed data for classification. In our simulations, we sent the data to the trained neural network to perform a *maximum a posterior* (MAP) classification, i.e., selecting the one with largest output activation value $y_k = P(C_k|\mathrm{x}^{(m)}, \mathrm{x}^{(o)})$. We could also send this newly completed data to the pre-estimated density function (e.g., Gaussian distribution) for an ML classification, selecting the one with largest likelihood $P(\mathrm{x}^{(m)}, \mathrm{x}^{(o)}|C_k)$.

The second approach was based on the proposed neural network inversion technique to obtain the estimation of the missing elements (see Eq. (7)). After the inversion process converged, trained neural network directly reported the classification posterior probabilities $y_k = P(C_k|\mathrm{x}^{(m)}, \mathrm{x}^{(o)})$ for each individual class.

The comparative performance of these two approaches is shown in Table 1. Note that, our proposed inversion EM achieves better performance when two missing elements are present. For one missing element case, due to the highly separated class distribution of IRIS data, both methods can recover the missing elements and perform the correct classification without difficulties.

3.2 EM Training of IRIS Data

For the EM training, we started with training an one-hidden layer MLP (4 inputs, 2 hidden units, and 3 outputs) with 30 IRIS complete training data (10

data for each class). This partially trained neural network was supplemented in training by additional 30 incomplete data (also 10 data for each class) with one or two missing elements of random pattern for each data.

Two approaches were again used in the comparative studies for EM training. The first one is the standard EM training, where the missing elements were "iteratively" replaced by the conditional expectations $E[\mathrm{x}^{(m)}|\mathrm{x}^{(o)}, C_k]$ of its own class C_k. The distribution parameters θ_k, $k = 1, 2, 3$, were also iteratively updated based on the originally 30 complete data and the newly completed 30 incomplete data. We again assume the joint data distributions of 3 classes are single-mode Gaussian distributed as given in Eq. (9). After the iterative replacements of the missing elements (of 30 incomplete data) converged, we combined this set with the originally complete data to retrain the neural network.

The second approach was based on the proposed neural network inversion technique to iteratively obtain the estimation of the missing elements (see Eq. (7)), starting from the neural network trained with only 30 complete data. After the inversion replacements of the missing elements (of 30 incomplete data), we then combined this set with the originally complete data set to retrain the neural network (in our case, 10 training sweeps were tried each round). Based on the retrained neural network, another round of inversion replacements of the missing elements was carried out, and another round of neural network training was then be performed. This process iterated (usually 10 to 20 rounds) until convergence.

The comparative performance of these two approaches for 90 independent and "complete" testing data is shown in Table 2. The reported classification accuracy was computed based on the average over 10 trials using different random missing patterns. Note that both EM training methods (standard and inversion) with one missing element can achieve the same performance 94.4% achieved when using 60 complete data without any missing element, and the performance is better than trained with only 30 complete data alone 88.6%.

Interesting enough to see that the inversion EM training slightly outperforms the standard EM with two missing elements, and both of them achieve better performance achieved when using 60 complete data without any missing element. This is possibly due to the outlier suppression capability of EM procedures, which replace the (noisy) missing elements with clean conditional expectations or network inversion values.

4 Conclusion

We propose EM training and testing procedures based on a neural network inversion technique. Our approach relaxes the assumption made about the probability density function, and more importantly, the missing value replacements take into account of the discrimination among classes. Simula-

Classification Accuracy	Standard EM	Inversion EM
60 complete	94.4 %	94.4 %
30 complete and 30 incomplete (missing one)	94.4 %	94.4 %
30 complete and 30 incomplete (missing two)	94.8 %	95.6 %

Table 2: Comparative simulation results (percentages of classification accuracy) for IRIS data classification using EM training.

tion results for classifying the IRIS data indicate the potential superiority of the inversion EM over the standard EM. Simulations of the proposed inversion EM is to be tested on a larger set of heart-attack data to be classified. Comparison with mixture Gaussian joint distribution and more exact MAP estimation of the missing elements will be carried in the near future.

References

[1] D. T. Davis, Z. Chen, L. Tsang, J. N. Hwang, and A.T.C. Chang. Retrieval of snow parameters by iterative inversion of a neural network. IEEE Trans. on GeoScience and Remote Sensing, 31(4):842-852, July 1993.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. J. Royal Statistical Society Series B, 39:1-38, 1977.

[3] R. O. Duda and P. E. Hart. Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[4] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In Proceedings of Advances in Neural Information Processing Systems 6, J. D. Cowan, G. Teasuro and J. Alspector (eds.), Morgan Kaufmann Publisher, San Francisco CA, 1994.

[5] J. N. Hwang and C.H. Chan. Iterative constrained inversion and its applications. 24th. Conf. on Information Systems and Sciences, Princeton, pp. 754-759, March 1990.

[6] J. N. Hwang, S. R. Lay and A. Lippman. Nonparametric multivariate density estimation: a comparative study. To appear in IEEE Trans. on Signal Processing, October 1994 (in press).

[7] B. H. Juang. Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chain. AT&T Technical Journal, 64(6):1235-1249, July-August 1985.

[8] A. Linden and J. Kindermann. Inversion of multilayer nets. In Proc. Int'l Joint Conf. on Neural Networks, II 425-430, Washington D.C., June 1989.

[9] R. J. A. Little and D. B. Rubin. Statistical Analysis with Missing Data. Wiley, New York, 1987.

[10] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate Bayesian a posterior probabilities. Neural Computation, 3(4):461-483, 1991.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Parallel Distributed Processing (PDP): Exploration in the Microstructure of Cognition, Vol. 1, Chapter 8, pp. 318-362. MIT Press, Cambridge, Massachusetts, 1986.

[12] M.L. Southcott and R.E. Bogner. Classification of incomplete data using neural networks. In Proceedings of the Fourth Australian Conference on Neural Networks (ACNN '93).

[13] E. Yair and A. Gersha. Maximum a posterior decision as evaluation of class probabilities by Boltzmann perceptron classifiers. Proceedings of the IEEE, 78(10):1620-1628, October 1990.

OSA — A TOPOLOGICAL ALGORITHM FOR CONSTRUCTING TWO-LAYER NEURAL NETWORKS

Fabio Massimo Frattale Mascioli Giuseppe Martinelli
INFO-COM Dept. - University of Rome
Via Eudossiana, 18; 00184 Roma - ITALY
Tel.: + 39-6-44585488/9; Fax: + 39-6-4873300
e-mail: mascioli@infocom.ing.uniroma1.it

Abstract— In this paper we present a constructive training algorithm for supervised neural networks: OSA (Oil-Spot Algorithm). It builds a two-layer neural network by involving successively binary examples. Its main learning rule, based on topological theorems on the cuts of a binary hypercube, is discussed. A convenient treatment of real-valued data is possible by means of a suitable real-to-binary codification. For binary target functions that have efficient halfspace union representations, the constructed networks result optimized in terms of number of neurons with respect to other constructive algorithms, as shown.

INTRODUCTION

Constructive training algorithms for supervised neural networks [1–10, 14–16] have been recently proposed in technical literature to circumvent the well-known problems of Back-Propagation, and related approaches (critical choice *a priori* of the architecture, local minima in the utilized gradient descent techniques and computational cost). In the case of two-class problems, several of them are based on the idea of trying to classify, at each step, as many example of one class as possible, keeping all the examples of the other class correctly classified. Also in the present case we apply this approach. However, with respect to the algorithms proposed in technical literature, we will mainly rely on graph theory solutions rather than learning in the usual neural sense (i.e., using local information and a simple learning rule). Namely, OSA (Oil-Spot Algorithm) is a constructive algorithm characterized by directly controlling the separating hyperplanes of the decision region. This result is obtained by relying on a topological approach, based on the representation of the mapping onto the binary hypercube of the input space and on the application of a learning rule derived by topological theorems. Our method yields the separating hyperplanes, taking account of both the training set and a smoothing generalization rule for covering the unspecified part of the mapping. Consequently, the separating hyperplanes are strictly related to the mapping of interest and they are introduced, as necessary, step-by-step under the strategy of separating at each step the maximum number of vertices. Therefore, the number of neurons, which coincides with that of these hyperplanes, is optimized. However, it is important to point out that this optimization only holds for

0-7803-2026-3/94 $4.00 © 1994 IEEE

target functions that have efficient (small) *halfspace union* representations (a counter-example is the Parity function) [10].

DEFINITIONS AND ARCHITECTURE OF THE NEURAL NETWORK

Representation of the Mapping onto the Binary Hypercube

The problem of interest is described by a training set of M examples P_r (r=1,2,..,M). The r-th example P_r is an input-output pair identified by N values $X_i^{(r)}=\{0,1\}$ (i=1,2,..,N) of the input variables and the corresponding desired output $o_r=\{0,1\}$. The mapping is therefore represented in the input space by an N-dimensional binary hypercube $C_N=\{(X_1,X_2,..,X_N)\in\{0,1\}^N\}$, whose vertex V_r (which correspond to example P_r) is labelled with the value of the r-th desired output o_r. From a topological point of view, C_N is a connected graph where the nodes correspond to the vertices and the arcs to the edges. We assume that the training set has no internal conflicts (different outputs for the same input). Since C_N has 2^N vertices and in general $M\leq 2^N$, M vertices of C_N will be labelled with a value '0' ('negative') or '1' ('positive'), while the remaining 2^N-M ones will be 'd' ('don't care') vertices.

Classification of the Hypercube Edges

A vertex V_r is represented by a vector having as components the N coordinates $V_r(i)=\{0,1\}$ (i=1,2,..,N). An edge of C_N is the closed line segment joining two contiguous vertices V_a and V_b, whose coordinates differ by only one component. The *orientation* of the edge from V_a to V_b ($V_a\rightarrow V_b$), that can be either positive or negative, is measured by:

$$V_{ab}=\sum_{i=1}^{N}\left[V_b(i)-V_a(i)\right].$$

Two edges are *parallel* if the corresponding pairs of vertices differ by the same component. Two parallel edges are <u>*congruent*</u> if their orientation coincides. We classify the edges of C_N in nine types. The edge of *type 1* joins a '0' vertex to a '1' vertex; we denote it by '0→1'. Similarly: 'd→1', '1→1', '1→0', '1→d', 'd→0', '0→d', '0→0' and 'd→d' are respectively types 2, 3, 4, 5, 6, 7, 8 and 9.

On the Positive Cuts of the Hypercube

An entirely specified mapping is represented by a complete Boolean function $f:B^N\rightarrow\{0,1\}$. In this case all the vertices of C_N are labelled with '0' or '1'. When a neural network is able to solve the given mapping, we define its decision region to be $DR=\{X\in B^N/f(X)=1\}$. DR corresponds in the hypercube C_N to the set of positive vertices. Usually, the given training set specifies only partially the mapping, i.e. 2^N-M vertices of C_N are labelled with 'd'. Let $V=V^+\cup V^-\cup V^d$ be

the set of hypercube vertices, with V^+, V^- and V^d respectively the set of positive, negative and don't care vertices (thus, the cardinality of the set V is equal to 2^N and that of the set V^+UV^- to M). We define a *positive cut* of C_N to be a subset of V^+UV^d, which can be strictly separated by a hyperplane from its complement, i.e. the set of the remaining vertices. The previous hyperplane is the boundary of the cut. Let us call <u>*boundary edges*</u> those connecting vertices of the positive cut to vertices of its complement. We remark that type 3 ('1→1') and type 8 ('0→0') can never be boundary edges. Each positive cut is therefore a binary halfspace region that contains only positive or don't care vertices. Topologically, it is a <u>*connected subgraph*</u> of C_N.

Implementation of a Generalized Decision Region with a Two-Layer Net

The union of all the positive cuts regarding a mapping defines a topological region that contains all the positive, several don't care, but none negative vertices of C_N. We will denote this region as the *generalized decision region* $GDR=\{X\in B^N / f(X)=1 \text{ or } d\}$, since it incorporates don't care vertices as a consequence of the adopted generalization rule.

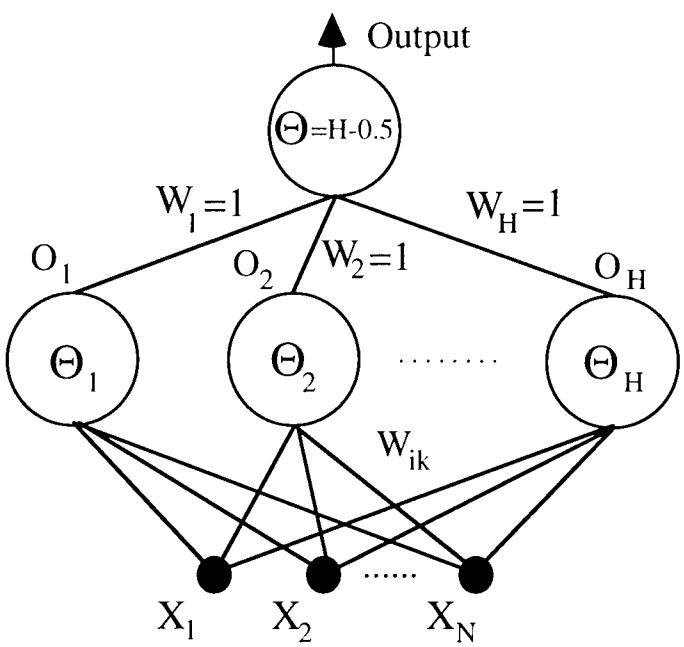


Figure 1:
Architecture of the two-layer feedforward neural network.

We consider a single hidden layer feedforward net, where all neurons are perceptron-like units (see fig. 1). There is a direct correspondence between hidden neurons and hypercube cuts. In fact, each hidden neuron implements a decision region that is the halfspace $WX+\Theta\geq 0$, where W is the vector of connection weights, Θ the threshold and X the input vector. The boundary of this

halfspace is the hyperplane WX+Θ=0. The single neuron of the second layer has the task of grouping the halfspace regions of the hidden neurons by an OR operation, in order to form the desired topological region. Therefore, the neural network implements the GDR if its hidden neurons realize all the positive cuts required by the mapping.

RATIONALE OF THE OIL-SPOT ALGORITHM

The goal of the algorithm we propose is to determine all the positive cuts (halfspaces) which constitute the generalized decision for the given problem. For achieving this result the algorithm mainly relies on four topological theorems on the cuts of a binary hypercube: Lemmas 1, 2 and 3 of ref. [11] and Lemma 7 of ref. [12]. Lemma 1 states that a boundary hyperplane can be always relocated in order to remove a vertex from a halfspace to include it in the complementary one. Lemmas 2 and 3 yield the conditions which guarantee for a set S of vertices the existence of a hyperplane HP which separates it from its complement CS. Lemma 7 has been utilized in the study of the decision regions of multilayer perceptrons and resumes the previous Lemmas. In summary, it states that any two parallel edges which cross a boundary hyperplane must do so in the same direction.

The described Lemmas give us the possibility to draw up the <u>*learning rule*</u> which is the basis of the oil-spot algorithm. It is constituted by two conditions:

1) All the vertices in a halfspace S must be the nodes of a connected subgraph of C_N.
2) Any two parallel boundary edges of S must be congruent.

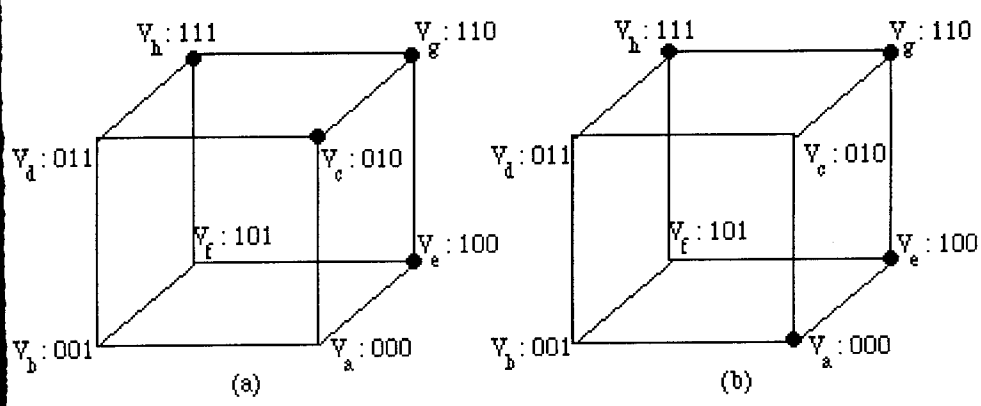


Figure 2:
The vertices of C_3 marked with black dots correspond to '1' vertices, the others to '0'. In this case all the boundary edges in figure are of type 1.
(a) It is possible to determine a positive cut which contains all the four '1' vertices.
(b) A halfspace that contains all the four '1' vertices does not exist.

Let us comment this rule with two intuitive graphical examples, where we want to determine if some vertices of a cube C_3 are contained in a positive cut S (see fig. 2).

In the first case (fig. 2.a) the halfspace which contains the vertices V_c, V_e, V_g and V_h can be strictly separated. In fact, they lie in a connected subgraph of C_3 and the three pairs of parallel boundary edges ($V_d \rightarrow V_h$ and $V_a \rightarrow V_e$; $V_f \rightarrow V_h$ and $V_a \rightarrow V_c$; $V_d \rightarrow V_c$ and $V_f \rightarrow V_e$) are congruents ($V_{dh}=V_{ae}=100$; $V_{fh}=V_{ac}=010$; and $V_{dc}=V_{fe}=-001$).

In the second case (fig. 2.b), instead, a hyperplane that separates V_a, V_e, V_g and V_h does not exist, because the two parallel boundary edges $V_f \rightarrow V_h$ and $V_c \rightarrow V_a$ are not congruents (in fact: $V_{fh}=010$ and $V_{ca}=-010$).

THE OIL-SPOT ALGORITHM (OSA)

Procedure for Determining a Positive Cut

A convenient strategy for determining a positive cut is to construct step-by-step a set of vertices, initially composed by only one positive vertex, controlling that the growth meets the two basic conditions of the learning rule.

Step 1. We choose a vertex with label '1' which will be denoted as a *candidate* vertex. It can form with the contiguous vertices three types of edges which are oriented towards the considered candidate. More precisely: type 1, type 2 and type 3 edges ('0→1', 'd→1', and '1→1'). We call *critical edges* those of type 1 ('0→1'). The critical edges can be only boundary edges, i.e. there cannot be critical edges inside a positive cut. If the chosen vertex has only '0' contiguous vertices, we can directly conclude that there exists a positive cut containing only it (in fact, a hyperplane that strictly separates a single vertex of a hypercube always exists).

Step 2. In agreement with the first basic condition, we add to the previous vertex a further contiguous one with label '1'. The new candidate vertex is also characterized by three types of critical edges. When a critical edge of the new candidate is parallel to a previously considered one, the two edges *must be congruent* in agreement with the second basic condition. Only in this case the new candidate is accepted as a member of the set we are determining, otherwise it *must be rejected*. Since the rejected candidates must remain outside the set, during a positive cut determination we relabel them with '0'.

Step 3. Excluding the vertices already visited, we repeat recursively Step 2. In this way, the candidates propagate as an "oil-spot".

Step 4. The *oil-spot propagation* terminates when all the last considered candidates have only '0' or rejected contiguous vertices. The output of the entire process is the set S which meets the two basic conditions.

Adopted Generalization Rule

It consists in considering as '1' all the 'd' vertices which have at least a contiguous one with label '1', so that the unspecified part of the mapping results as smooth as possible.

From a Positive Cut to a Hidden Neuron

The coefficients of the linear equation representing the hyperplane HP which separates S from CS can be determined by a geometrical method. We consider a system of coordinates with their origin at the center C_0 of the hypercube C_N. Let Q be the unknown vector with the origin in C_0 and orthogonal to HP. Q is characterized by being as close as possible to the vertices in S and as far as possible from the remaining vertices. Consequently, Q is the *centroid* of the vertices in S. The connection weights of the k-th hidden neuron to be determined, coincident with the coefficients of the hyperplane HP, are therefore given by:

$$W_{ik}=[\lambda_i(1)-\lambda_i(0)]\beta \qquad i=1,2,..,N; \tag{1}$$

where $\lambda_i(y)$ is the number of vertices in S which have the i-th component equal to y ($y=\{0,1\}$) and β is an arbitrary positive constant. The last parameter to be determined is the threshold Θ of the neuron. This quantity is characterized by the property that the neuron must go "on" when the input corresponds to one of the vertices of S and "off" otherwise. If the quantity entering the activation function of the neuron when the input corresponds to the vertex V_r is:

$$z_r=\sum_{i=1}^{N} W_{ik}X_i^{(r)},$$

the previous condition requires:

$z_r>\Theta_k$ when $V_r\in S$
$z_r<\Theta_k$ otherwise.

Since the set S can be separated by a hyperplane, it is sufficient to choose:

$$\Theta_k=\min_{V_r\in S}(z_r)-\frac{1}{2}\min_i|W_{ik}| \tag{2}$$

Rarely it happens that some vertices of S lie on HP together with some vertices of CS. For these vertices it results $z_r=\Theta_k$, consequently they are not strictly separated by HP. In this case OSA removes them from S and considers them as not visited. The resulting S' is properly a positive cut.

Network Construction

In general (nonlinear problem), after the construction of the first neuron several "1" vertices of C_N do not belong to the determined positive cut (rejected or not yet visited). In that case, starting from them, OSA repeats the construction of further hidden neurons until all the "1" vertices are enclosed in a positive cut. At the end of the constructive process, OSA adds the second layer neuron that

implements the OR operation (i.e., the output neuron). We remark that, given the same problem, if we change the first candidate vertex OSA constructs a different network. We have experimented by simulations that, in these cases, the number of hidden neurons is slightly affected.

Network Robustness

Since during the successive determinations of positive cuts only the first candidate must be not visited, a vertex can belong to more than one cut (i.e., an example can be satisfied by more than one neuron).

Convergence to Zero-Errors

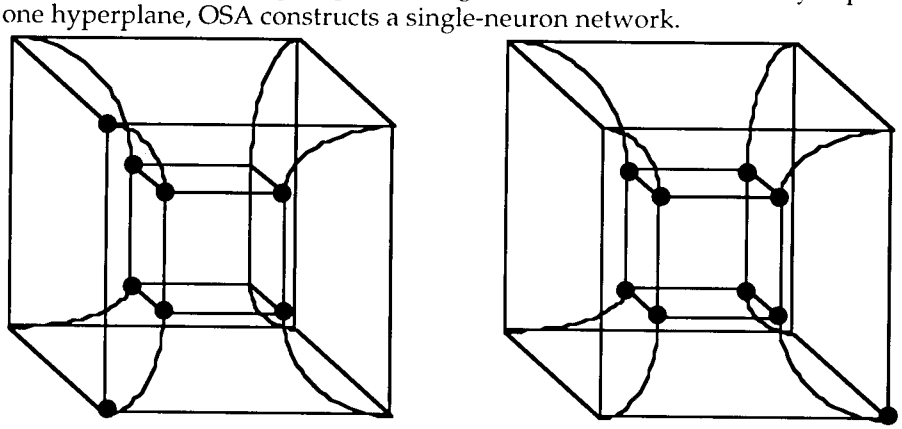
The determination of positive cuts terminates only when all the '1' vertices are visited, i.e. when all the given examples are satisfied.

Computational Cost

OSA considers as "1" the minority output in the training set. Therefore, the number of "1" is at the most M/2. The main operations of OSA are visits in a graph and algebraic additions (no multiplications). In the worst case, for determining a positive cut, the number of operations is $NOP_W = (M/2)(N-1)((M/2)-1)$. In practice, the number of operations for each constructed neuron is much less than NOP_W. In the case of small halfspace unions, OSA constructs optimal size networks, hence the total computational cost can be considered polynomial.

APPLICATIONS AND COMPARISONS

4-Cube Cuts. The problem to be solved regards the 4-cube non-isomorphic cuts [11] (fig. 3). For all the topological configurations that can be strictly separated by one hyperplane, OSA constructs a single-neuron network.



(a) (b)

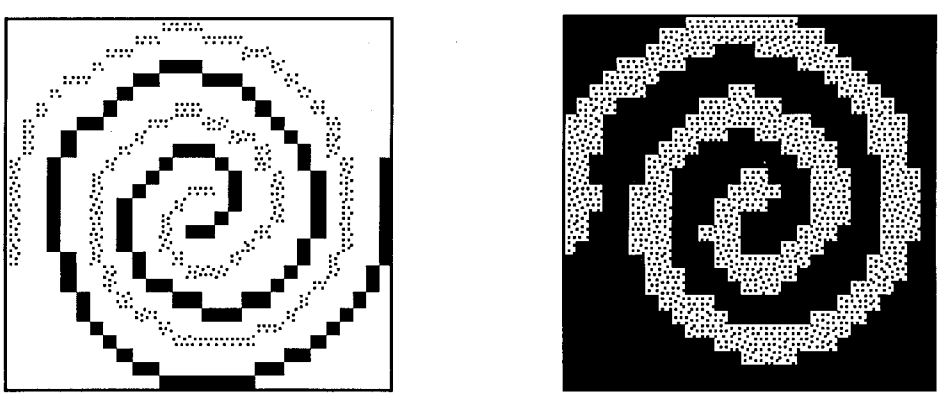
Figure 3:
An example of two non-isomorphic cuts of the 4-cube.

Random Boolean Functions. In this case the problem to be solved regards a Boolean function of 6 variables; we have generated at random 100 Boolean functions of this type. As expected, a single hidden layer network is always sufficient. The average number of hidden neurons found is 8.02±1.98, which is quite close to the one obtained by Marchand *et al.* [6] (7.28±0.82) and significantly better than the results presented in ref. [14] (20.5±3.9) and [15] (about 18 units in 4 layers).

Parity Functions. In the case of Parity functions (tested from N=2 to N=8), OSA constructs networks with $2^{(N-1)}$ hidden neurons. This result is in agreement with the well-known property which states that an exponential number of neurons is required by networks based on halfspace unions for solving Parity.

Circular Region. The problem regards in this case the approximation of a circular region of 12 pixels inside a 6x6 grid (36 pixels). The X-Y coordinates of each cell are preliminarly converted from real to binary notations; therefore the inputs are 6. OSA constructs a neural network with only 4 hidden neurons, simpler than that obtained with other algorithms, as for instance the BLTA [8].

Twin spirals. The twin spirals problem (separating 194 pixels from two interlocking spirals, see fig. 4.a) is an extremely hard problem for algorithms of the Back-Propagation family to solve [13]. By means of OSA we obtain a solution with 44 hidden neurons. We preliminarly transform the real-valued input data into binary form by a suitable codification, which preserve the neighborhood of data (i.e., two pixels which are contiguous in the real input space are coded into two contiguous vertices of the N-cube). The number of inputs is consequently 16 instead of 2. The resulting decision region is shown in fig. 4.b. It is satisfactory. The time required for building the network (with a 486-based computer) is less than ten seconds. We note that a solution of the same problem with Upstart [2] and Cascade-Correlation [4] requires about ten minutes of elaboration time in the same conditions. Finally, we remark that the only other solution to twin spirals using a single hidden layer architecture, that we are aware of, requires 50 hidden units [16].



(a) (b)

Figure 4:
(a) The twin spirals problem training set (194 pixels in a 32x32 grid);
(b) The resulting decision region obtained with a 44 hidden neuron net.

CONCLUSIONS

OSA is characterized by a learning rule which relies on a topological approach. It operates with binary data directly in the input space (binary hypercube). A suitable codification, that preserves the contiguity of data, can be adopted with good results in real-valued problems. As illustrated in simulations, the constructed nets are often simpler than those obtained with other methods. It is moreover important to note that OSA does not need of specific parameters to set or stopping criterion to use during training. Finally, we remark that in simulations the elaboration time required by the algorithm turned out to be lower than we expected. Further work in progress regards the use of the algorithm to more significant examples related with actual applications and its extension to the multiple output case.

Acknowledgements. This work was supported by MURST and CNR. The authors thank F. Cerra, F. Cincioni, A. Rizzi, and A. Sommella for software simulations.

REFERENCES

[1] J. Nadal: "Study of a growth algorithm for neural networks", Int. J. Neural Syst., Vol. 1, 1989, pp. 55-59.

[2] M. Frean: "The upstart algorithm: a method for constructing and training feedforward neural networks", Neural Computation, Vol. 2, 1990, pp. 198-209.

[3] G. Martinelli, L. Prina Ricotti, S. Ragazzini, and F.M. Mascioli: "A pyramidal delayed perceptron", IEEE Trans. on CAS, Vol. 37, 1990, pp. 1176-1181.

[4] S.E. Fahlman, C. Lebiere: "The cascade-correlation learning architecture", in Adv. in Neur. Inf. Proc. Syst. 2, D.S. Touretzky, Los Altos, Morgan-Kaufmann, 1990, pp. 524-532.

[5] E. Baum: "On learning a union of halfspaces", J. Complexity, Vol. 6, 1990, pp. 67-101.

[6] M. Marchand, M. Golea, and P. Rujàn: "A convergence theorem for sequential learning in two-layer perceptrons", Europhysics Letters, Vol. 11, 1990, pp. 487-492.

[7] G. Martinelli and F.M. Mascioli: "Cascade perceptron", IEE Electronics Letters, Vol. 28, 1992, pp. 947-949.

[8] D.L. Gray and A.N. Michel: "A training algorithm for binary feedforward neural networks", IEEE Trans. on Neural Networks, Vol. 3, 1992, pp. 176-194.

[9] G. Martinelli, F.M. Mascioli, and G. Bei: "Cascade neural network for binary mapping", IEEE Trans. on Neural Networks, Vol. 4, 1993, pp. 148-150.

[10] M. Marchand and M. Golea: "On learning simple neural concepts: from halfspaces intersections to neural decision lists", Network: computation in neural systems, Vol. 4, 1993, pp. 67-89.

[11] M.R. Emamy-Khansary: "On the cuts and cut number of the 4-cube", J. Combinatorial Theory, Series A, Vol. 41, 1986, pp. 221-227.

[12] G. J. Gibson and C. F. N. Cowan: "On the decision regions of multilayer perceptrons", Proc. of the IEEE, Vol. 78, No. 10, 1990, pp. 1590-1594.

[13] K.J. Lang and M.J. Witbrock: "Learning to tell two spirals apart", Proc. of the 1988 Connectionist Models Summer School, Morgan Kaufmann, 1988.

[14] M. Golea and M. Marchand: "A growth algorithm for neural network decision trees", Europhys Lett., 12, 1990, pp. 205-210.

[15] M. Mezard and J.P. Nadal: "Learning in feedforward layered networks: the tiling algorithm", J. Phys. A, 22, 1989, pp. 2191-2203.

[16] E.B. Baum and K.J. Lang: "Constructing hidden units using examples and queries", NIPS3, 1990, pp. 904-910.

GENERALIZATION PERFORMANCE OF REGULARIZED NEURAL NETWORK MODELS

Jan Larsen and Lars Kai Hansen
The Computational Neural Network Center
Electronics Institute, Building 349
Technical University of Denmark
DK-2800 Lyngby, Denmark

Abstract. Architecture optimization is a fundamental problem of neural network modeling. The optimal architecture is defined as the one which minimizes the generalization error. This paper addresses estimation of the generalization performance of regularized, complete neural network models. Regularization normally improves the generalization performance by restricting the model complexity. A formula for the optimal weight decay regularizer is derived. A regularized model may be characterized by an effective number of weights (parameters); however, it is demonstrated that no simple definition is possible. A novel estimator of the average generalization error (called *FPER*) is suggested and compared to the Final Prediction Error (*FPE*) and Generalized Prediction Error (*GPE*) estimators. In addition, comparative numerical studies demonstrate the qualities of the suggested estimator.

INTRODUCTION

One of the fundamental problems involved in design of neural network models is architecture optimization aiming at high generalization performance. In this paper the generalization measure is defined as the *average generalization error*, i.e., the expected squared error averaged over all possible training sets of size N, with N being the number of training samples. The average generalization error, Γ, can be decomposed into three additive components [2], [8]: $\Gamma = \sigma_\varepsilon^2 + MSME + WFP$, viz. the inherent noise variance, the mean square model error, and the weight fluctuation penalty[1]. The inherent noise variance is caused by noise on the data which – per definition – cannot be modeled.

[1]The *MSME* and the *WFP* are related to the *squared bias* and the *variance*, respectively. See [2] for a definition of bias and variance.

0-7803-2026-3/94 $4.00 © 1994 IEEE

Presence of *MSME* reflects the lack of modeling capability of the neural network for modeling the current data, i.e., the network is an *incomplete model* of the data generating system. Finally, the *WFP* reflects the increase in average generalization error caused by fluctuations in the estimated weights, which stem from the fact that the weights are estimated from a given finite training set.

Architecture optimization can be viewed as a bias/variance trade off [2], [11] or equivalently a *MSME/WFP* trade off: The *MSME* is reduced when increasing the network complexity[2] while the *WFP* typically[3] increases. The literature provides a variety of methods for performing this trade off, including architecture pruning and growing schemes, as well as regularization techniques.

TRAINING AND GENERALIZATION

Consider modeling the data generating system:

$$y(k) = g(\boldsymbol{x}(k)) + \varepsilon(k) \tag{1}$$

where k is the discrete time index, $y(k)$ is the scalar output signal, $g(\cdot)$ constitutes a nonlinear mapping of the p-dimensional input signal $\boldsymbol{x}(k)$ (column vector), and $\varepsilon(k)$ is an inherent noise signal.

Assumption 1 *The input signal $\boldsymbol{x}(k)$ is assumed to be a strongly mixing[4] strictly stationary sequence and the inherent noise $\varepsilon(k)$ is assumed to be a strictly stationary sequence independent on the input, white, with zero mean, and finite variance, σ_ε^2.*

The neural network model of the system in Eq. (1) is given by

$$y(k) = f(\boldsymbol{x}(k); \boldsymbol{w}) + e(k; \boldsymbol{w}) \tag{2}$$

where $f(\cdot; \boldsymbol{w})$ defines the mapping of the neural network parameterized by the m-dimensional weight vector $\boldsymbol{w}$, and $e(k; \boldsymbol{w})$ is the error signal.

Assumption 2 *The model is assumed* **complete** *[8, Def. 6.3], i.e., there exists a true weight vector, $\boldsymbol{w}^\circ$, so as to*

$$\forall \boldsymbol{x}: \quad f(\boldsymbol{x}; \boldsymbol{w}^\circ) \equiv g(\boldsymbol{x}) \tag{3}$$

In general, only little a priori knowledge of the data generating system is available, i.e., most neural network models are *incomplete*, which result in non-zero mean square model error. However, a multi-layer perceptron neural

[2]This statement is only true for nested families of network architectures. Moreover, *MSME* may remain unchanged when adding irrelevant complexity.

[3]It should be emphasized that it is possible to give simple examples where the *WFP* actually *decreases* when adding extra complexity [8, Ch. 6.3.4].

[4]Loosely speaking, i.e., the dependence of $\mathbf{x}(k)$ and $\mathbf{x}(k+\tau)$ vanishes as $|\tau| \to \infty$.

network with many hidden neurons is capable of approximating a large class of functions, thus $MSME$ may be small relative to $\sigma_\varepsilon^2 + WFP$, and the model may be regarded as *quasi-complete*. When dealing with cases where the complete model assumption is dubious, it is suggested to estimate the generalization performance by using the GEN estimator [7], [8].

Define the training set of N samples by $\mathcal{T} = \{\boldsymbol{x}(k); y(k)\}$, $k = 1, 2, \cdots, N$. The model is estimated by minimizing a cost function being the sum of the usual mean square cost and a weight decay regularizer[5]:

$$C_N(\boldsymbol{w}) = S_N(\boldsymbol{w}) + \boldsymbol{w}^\mathsf{T} \boldsymbol{R} \boldsymbol{w} \tag{4}$$

where $S_N(\boldsymbol{w}) = N^{-1} \sum_{k=1}^N e^2(k; \boldsymbol{w}) = N^{-1} \sum_{k=1}^N [y(k) - f(\boldsymbol{x}(k); \boldsymbol{w})]^2$ is the mean square cost and $\boldsymbol{R}$ is a $m \times m$ symmetric, positive semidefinite regularization matrix. Standard weight decay regularization is obtained by using $\boldsymbol{R} = \kappa \boldsymbol{I}$, where $\kappa \geq 0$ is the weight decay parameter and $\boldsymbol{I}$ the identity matrix. The presented theory is not restricted to the chosen cost function, thus analogous results can be obtained when e.g., using log-likelihood cost functions and more general regularizers, $r(\boldsymbol{w}; \boldsymbol{\kappa})$, where $r(\cdot)$ is a regularization function parameterized by $\boldsymbol{\kappa}$.

The weights of the estimated model are denoted the *estimated weights*, i.e.,

$$\widehat{\boldsymbol{w}} = \arg \min_{\boldsymbol{w}} C_N(\boldsymbol{w}) \tag{5}$$

Also define the *expected cost function*:

$$C(\boldsymbol{w}) = E\left\{C_N(\boldsymbol{w})\right\} = E\left\{e^2(\boldsymbol{w})\right\} + \boldsymbol{w}^\mathsf{T} \boldsymbol{R} \boldsymbol{w} \tag{6}$$

where $E\{\cdot\}$ denotes expectation w.r.t. the joint input-output probability density function. Under mild regularity conditions (see e.g., [8, Ch. 5], [12]) $\lim_{N\to\infty} C_N(\boldsymbol{w}) = C(\boldsymbol{w})$, and the estimated weight vector $\widehat{\boldsymbol{w}}$ becomes a consistent estimate ($N \to \infty$) of the *optimal weight vector*: $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w})$. Since the model is assumed complete $\boldsymbol{w}^*$ is identical to $\boldsymbol{w}^\circ$ when omitting regularization. However, regularization imposes a bias of the optimal weights towards $\mathbf{0}$.

The *generalization error* of the estimated model is defined as the expected squared error on an test sample, $[\boldsymbol{x}; y]$, independent on the training samples, i.e.,

$$G(\widehat{\boldsymbol{w}}) = E\left\{e^2(\widehat{\boldsymbol{w}})\right\} = E\left\{[y - f(\boldsymbol{x}; \widehat{\boldsymbol{w}})]^2\right\} \tag{7}$$

It turns out (see e.g., the discussion in [8, Sec. 6.3.2]) that $G(\widehat{\boldsymbol{w}})$ is not necessarily a reliable measure of the model quality since it depends on the *actual training set* through $\widehat{\boldsymbol{w}}$. In addition, it is not possible to obtain estimates of $G(\widehat{\boldsymbol{w}})$ without perfect knowledge of the joint input-output distribution. Hence, the appropriate model quality measure is the *average generalization error*, e.g., [8], [11]:

$$\Gamma = E_{\mathcal{T}}\left\{G(\widehat{\boldsymbol{w}})\right\} \tag{8}$$

[5] $^\mathsf{T}$ denotes the transpose operator.

where $E_{\mathcal{T}}\{\cdot\}$ denotes expectation over all training sets with N samples. That is, averaging is w.r.t. fluctuation in $\widehat{\boldsymbol{w}}$ due to different training sets. Define $\mathcal{T}_{\boldsymbol{x}} = \{\boldsymbol{x}(k)\}$ and $\mathcal{T}_{\varepsilon} = \{\varepsilon(k)\}$. As the noise and the input are assumed independent, the expectation w.r.t. $\mathcal{T}$ is carried out as[6]:

$$E_{\mathcal{T}}\{G\} = E_{\mathcal{T}_{\boldsymbol{x}}}\left\{E_{\mathcal{T}_{\varepsilon}}\{G \mid \mathcal{T}_{\boldsymbol{x}}\}\right\} \tag{9}$$

ESTIMATING THE AVERAGE GENERALIZATION ERROR

The objective of this presentation is to obtain an estimate of Γ defined in Eq. (8) calculated in terms of quantities derived from the estimated model. From a statistical point of view it is possible to set different quality requirements on the estimator. Here the following requirements are made:

Definition 1 *The estimator searched for, $\widehat{\Gamma}$, is required to be consistent, and unbiased to order $1/N$, i.e., $\widehat{\Gamma} \to \Gamma$ as $N \to \infty$, and $E_{\mathcal{T}}\{\widehat{\Gamma}\} = \Gamma + o(1/N)$, where $o(\cdot)$ is the order function.*

The basic tool for deriving an estimator are second order Taylor series expansions of the average training and generalization errors, as follows:

$$\begin{aligned} E_{\mathcal{T}}\left\{S_N(\widehat{\boldsymbol{w}})\right\} &\approx E_{\mathcal{T}}\left\{S_N(\boldsymbol{w}^{\circ})\right\} + E_{\mathcal{T}}\left\{\frac{\partial S_N(\boldsymbol{w}^{\circ})}{\partial \boldsymbol{w}^{\mathsf{T}}}\Delta\boldsymbol{w}\right\} \\ &\quad + E_{\mathcal{T}}\left\{\Delta\boldsymbol{w}^{\mathsf{T}}\boldsymbol{H}_N(\boldsymbol{w}^{\circ})\Delta\boldsymbol{w}\right\} \end{aligned} \tag{10}$$

$$\begin{aligned} E_{\mathcal{T}}\left\{G(\widehat{\boldsymbol{w}})\right\} &\approx E_{\mathcal{T}}\left\{G(\boldsymbol{w}^{\circ})\right\} + E_{\mathcal{T}}\left\{\frac{\partial G(\boldsymbol{w}^{\circ})}{\partial \boldsymbol{w}^{\mathsf{T}}}\Delta\boldsymbol{w}\right\} \\ &\quad + E_{\mathcal{T}}\left\{\Delta\boldsymbol{w}^{\mathsf{T}}\boldsymbol{H}(\boldsymbol{w}^{\circ})\Delta\boldsymbol{w}\right\} \end{aligned} \tag{11}$$

where $\Delta\boldsymbol{w}$ is the weight fluctuation $\Delta\boldsymbol{w} = \widehat{\boldsymbol{w}} - \boldsymbol{w}^{\circ}$, $\boldsymbol{H}_N(\boldsymbol{w})$ is the Hessian matrix of the mean square cost function, i.e.,

$$\boldsymbol{H}_N(\boldsymbol{w}) = \frac{1}{2}\frac{\partial^2 S_N(\boldsymbol{w})}{\partial\boldsymbol{w}\partial\boldsymbol{w}^{\mathsf{T}}} = \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{\psi}(k;\boldsymbol{w})\boldsymbol{\psi}^{\mathsf{T}}(k;\boldsymbol{w}) - \boldsymbol{\Psi}(k;\boldsymbol{w})e(k;\boldsymbol{w}) \tag{12}$$

defining $\boldsymbol{\psi}$ as the instantaneous gradient vector of the model output, $\boldsymbol{\psi}(k;\boldsymbol{w}) = \partial f(\boldsymbol{x}(k);\boldsymbol{w}) / \partial\boldsymbol{w}$. Finally, $\boldsymbol{\Psi}$ is the second derivative matrix of the model output, $\boldsymbol{\Psi}(k;\boldsymbol{w}) = \partial\boldsymbol{\psi}(k;\boldsymbol{w})/\partial\boldsymbol{w}^{\mathsf{T}}$. Similarly, $\boldsymbol{H}(\boldsymbol{w})$ is the Hessian matrix of the generalization error, given by

$$\boldsymbol{H}(\boldsymbol{w}) = \frac{1}{2}\frac{\partial^2 G(\boldsymbol{w})}{\partial\boldsymbol{w}\partial\boldsymbol{w}^{\mathsf{T}}} = E\left\{\boldsymbol{\psi}(\boldsymbol{w})\boldsymbol{\psi}^{\mathsf{T}}(\boldsymbol{w}) - \boldsymbol{\Psi}(\boldsymbol{w})e(\boldsymbol{w})\right\} \tag{13}$$

In order to ensure the validity of the Taylor series approximations it is required that $\Delta\boldsymbol{w}$ is sufficiently small. As mentioned above $\widehat{\boldsymbol{w}}$ is a consistent estimate

[6]Note that expectation over the training set, $\mathcal{T} = \{\mathbf{x}(k); y(k)\}$, equals expectation over input and inherent noise samples, cf. the model definition Eq. (2).

of $\boldsymbol{w}^*$; however, $\boldsymbol{w}^*$ does not collapse onto $\boldsymbol{w}^\circ$ unless $\boldsymbol{R} = \boldsymbol{0}$. Consequently, it is expected that the Taylor series are valid for sufficiently large N and sufficiently small $\boldsymbol{R}$.

The appendix below provides a brief evaluation of the individual terms of Eq. (10), (11). The result is: For $N > 2m_1 - m_2$,

$$E_{\mathcal{T}}\left\{S_N(\widehat{\boldsymbol{w}})\right\} = \sigma_\varepsilon^2\left(1 - \frac{2m_1 - m_2}{N}\right) + M' + o(1/N) \tag{14}$$

$$\Gamma = \sigma_\varepsilon^2\left(1 + \frac{m_2}{N}\right) + M + o(1/N) \tag{15}$$

where m_1, m_2 defines two different *effective number of weights*[7]:

$$m_1 = \mathrm{tr}\left[\boldsymbol{H}(\boldsymbol{w}^\circ)\boldsymbol{J}^{-1}(\boldsymbol{w}^\circ)\right], \; m_2 = \mathrm{tr}\left[\boldsymbol{H}(\boldsymbol{w}^\circ)\boldsymbol{J}^{-1}(\boldsymbol{w}^\circ)\boldsymbol{H}(\boldsymbol{w}^\circ)\boldsymbol{J}^{-1}(\boldsymbol{w}^\circ)\right] \tag{16}$$

$\boldsymbol{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w}) + \boldsymbol{R}$ is the Hessian matrix of the expected cost function which is assumed to be invertible, and tr[·] is the trace operator.

$$M' = (\boldsymbol{w}^\circ)^\top \boldsymbol{R}\boldsymbol{J}^{-1}(\boldsymbol{w}^\circ)\left(\boldsymbol{H}(\boldsymbol{w}^\circ) + \frac{-2\boldsymbol{K}_1 + \boldsymbol{K}_2}{N}\right)\boldsymbol{J}^{-1}(\boldsymbol{w}^\circ)\boldsymbol{R}\boldsymbol{w}^\circ \tag{17}$$

with $\boldsymbol{K}_1$, $\boldsymbol{K}_2$ being 4th order moments, as shown by[8]:

$$\boldsymbol{K}_1 = E\left\{\left(\boldsymbol{\psi}\boldsymbol{\psi}^\top - \boldsymbol{H}\right)\boldsymbol{J}^{-1}\left(\boldsymbol{\psi}\boldsymbol{\psi}^\top - \boldsymbol{H}\right)\right\} \tag{18}$$

$$\boldsymbol{K}_2 = E\left\{\left(\boldsymbol{\psi}\boldsymbol{\psi}^\top - \boldsymbol{H}\right)\boldsymbol{J}^{-1}\boldsymbol{H}\boldsymbol{J}^{-1}\left(\boldsymbol{\psi}\boldsymbol{\psi}^\top - \boldsymbol{H}\right)\right\} \tag{19}$$

M equals M' except that the term $\boldsymbol{K}_1$ is absent. In general, M and M' are negligible compared to the remaining terms in Eq. (14), (15) when 1) using a regularization matrix close to the optimal setting Eq. (24), and when 2) the signal-to-noise ratio, $V\{g(\boldsymbol{x})\}/\sigma_\varepsilon^2$, is reasonable large.

Neglecting M, M' and eliminating σ_ε^2 in Eq. (14), (15) leads to:

$$\widehat{\Gamma} = \frac{N + m_2}{N - 2m_1 + m_2} E_{\mathcal{T}}\left\{S_N(\widehat{\boldsymbol{w}})\right\}, \quad N > 2m_1 - m_2 \tag{20}$$

which is unbiased to $o(1/N)$. Notice that elimination of σ_ε^2 introduces terms proportional to N^{-j}, $j > 1$. This seems inconsistent; however, for practical purposes the form is convenient since $\widehat{\Gamma}$ typically is an underestimate of Γ on the average. In the case of a complete linear model which is estimated without regularization [3] and [8, Theorem 6.10] support this statement.

The suggested estimator may be viewed as an extension of the classical *FPE* estimator [1], $FPE = E_{\mathcal{T}}\left\{S_N(\widehat{\boldsymbol{w}})\right\}(N + m)/(N - m)$, in which the

[7] It is easily shown that $m_1 \geq m_2 > 0$ thus $2m_1 - m_2 > 0$. Moreover, 1) $m_1 = m_2 = m$ for $\mathbf{R} = \mathbf{0}$ and $\mathbf{H}(\mathbf{w}^\circ)$ non-singular, and 2) $m_1 \to 0$, $m_2 \to 0$ as $\|\mathbf{R}\| \to \infty$.

[8] $\mathbf{K}_1$, $\mathbf{K}_2$ are of order one, and limited by assumption. Further note that all involved quantities are evaluated at $\mathbf{w}^\circ$.

number of weights m is replaced by the different effective number of weights, m_2 and $2m_1 - m_2$. Moreover, the estimator can be interpreted as a special version[9] of the *GPE* estimator [10], [11] where the inherent noise variance is estimated by: $\sigma_\varepsilon^2 = E_{\mathcal{T}}\{S_N(\widehat{\boldsymbol{w}})\}\, N/(N - 2m_1 + m_2)$. In order to construct a Γ-estimator from observable quantities, estimation of the noise variance is indeed important. This problem is not directly addressed in [10], [11]. The estimator suggested in [10] reads: $\sigma_\varepsilon^2 = E_{\mathcal{T}}\{S_N(\widehat{\boldsymbol{w}})\}\, N/(N - m_1)$, which obviously differs from the one derived from Eq. (14). In conclusion – as suggested in [9], [11] – it is not possible to define a single quantity m_1 which expresses the effective number of weights in the model, since σ_ε^2 should be estimated from $2m_1 - m_2$ rather than m_1 effective weights.

For practical purposes the quantities in Eq. (20) are estimated from observed quantities. An unbiased $o(1/N)$ estimator within the second order Taylor series expansion Eq. (10), (11) is the the **F**inal **P**rediction **E**rror estimator for **R**egularized models, as shown by:

$$FPER = \frac{N + \widehat{m}_2}{N - 2\widehat{m}_1 + \widehat{m}_2} S_N(\widehat{\boldsymbol{w}}), \quad N > 2\widehat{m}_1 - \widehat{m}_2 \tag{21}$$

where

$$\widehat{m}_1 = \mathrm{tr}\left[\boldsymbol{H}_N(\widehat{\boldsymbol{w}})\boldsymbol{J}_N^{-1}(\widehat{\boldsymbol{w}})\right], \ \widehat{m}_2 = \mathrm{tr}\left[\boldsymbol{H}_N(\widehat{\boldsymbol{w}})\boldsymbol{J}_N^{-1}(\widehat{\boldsymbol{w}})\boldsymbol{H}_N(\widehat{\boldsymbol{w}})\boldsymbol{J}_N^{-1}(\widehat{\boldsymbol{w}})\right] \tag{22}$$

and $\boldsymbol{J}_N(\widehat{\boldsymbol{w}}) = \boldsymbol{H}_N(\widehat{\boldsymbol{w}}) + \boldsymbol{R}$ is the Hessian matrix of the cost function which is assumed to be invertible.

OPTIMIZING THE WEIGHT DECAY REGULARIZATION PARAMETER

For simplicity, consider simple weight decay regularization, i.e., $\boldsymbol{R} = \kappa \boldsymbol{I}$ where κ is the weight decay parameter. As mentioned in the introduction, trading off weight fluctuation penalty (*WFP*) and mean square model error (*MSME*) leads to an optimal setting of κ. In [6] this problem was addressed for linear models and the following may be viewed as an extension of this work.

Inspecting Eq. (15) it turns out that[10] $M = MSME$ and $WFP = \sigma_\varepsilon^2 m_2/N$. The optimal value, κ_{opt}, is found by solving:

$$\frac{\partial WFP}{\partial \kappa} + \frac{\partial MSME}{\partial \kappa} = 0 \tag{23}$$

As expected, $\lim_{N\to\infty} WFP = 0$, since it measures the contribution due to a finite training set. Consequently, in order to reach the minimal average generalization error $\Gamma = \sigma_\varepsilon^2$ the restriction $\lim_{N\to\infty} MSME = 0$ should be met. The κ-dependence of the individual elements of $\boldsymbol{K}_1$ is $(\lambda_i + \kappa)^{-1}$

[9] Notice that this coincidence is based on various important assumptions, e.g., the model being complete and the negligibleness of M.

[10] Notice when determining an optimal κ, M is *not* neglected in Eq. (15).

where λ_i is the i'th eigenvalue of $\boldsymbol{H}(\boldsymbol{w}^\circ)$. For $\boldsymbol{K}_2$ the element dependence is: $\prod_{s=1}^{2}(\lambda_{i_s}+\kappa)^{-1}$. In summary, M is a sum of addends which κ-dependence are given by: $\kappa^2 \prod_{r=1}^{\ell}(\lambda_{i_r}+\kappa)^{-1}$, $\ell \in \{2,3,4\}$. That is, to fulfill the requirement $\lim_{N\to\infty} MSME = 0$, $\lim_{N\to\infty} \kappa = 0$ should be imposed. The solution to Eq. (23) can therefore be expressed as: $\kappa_{\text{opt}} = \kappa'_{\text{opt}}/N + o(1/N)$. Expanding the addends of Eq. (23) to first order in κ and $1/N$ and solving for κ gives:

$$\kappa_{\text{opt}} = \frac{\sigma_\varepsilon^2}{N} \cdot \frac{\operatorname{tr} \boldsymbol{H}^+(\boldsymbol{w}^\circ)}{(\boldsymbol{w}^\circ)^\top \boldsymbol{H}^+(\boldsymbol{w}^\circ)\boldsymbol{w}^\circ} + o(1/N) \tag{24}$$

where $\boldsymbol{H}^+(\boldsymbol{w}^\circ)$ is the Moore-Penrose pseudo inverse. Suppose the eigenvalues of $\boldsymbol{H}(\boldsymbol{w}^\circ)$ obey: $\lambda_1 \geq \cdots \geq \lambda_n > 0$ and $\lambda_i = 0, \forall\, i \in [n+1; m]$. The associated eigenvectors are assembled (as column vectors) in the matrix $\boldsymbol{Q}$. The pseudo inverse then reads: $\boldsymbol{H}^+(\boldsymbol{w}^\circ) = \boldsymbol{Q}\,\operatorname{diag}[\lambda_1^{-1}, \cdots, \lambda_n^{-1}, 0, \cdots, 0]\,\boldsymbol{Q}^\top$.

Notice two facts concerning κ_{opt}: First, it is proportional to the inherent noise variance. If no noise is present $WFP = 0$, thus one should not introduce $MSME$ by employing a non-zero κ. Secondly, κ_{opt} is inversely proportional to the length of the optimal weight vector weighted by the elements of the Moore-Penrose pseudo inverse Hessian matrix. This is due to the fact that we regularize against the zero weight vector.

Since the optimal weights $\boldsymbol{w}^\circ$ are unknown, it is impossible to calculate κ_{opt} directly; however, in [4] *adaptive regularization* is studied for a linear one-dimensional model, and [5] presents an adaptive regularization scheme for the purpose of designing compact time series models. In addition, it is possible to show that the average generalization error is reduced when using $0 < \kappa \leq 2\kappa_{\text{opt}}$.

NUMERICAL EXPERIMENTS

To substantiate the qualities of the suggested $FPER$ estimator Eq. (21), numerical comparisons with the FPE and GPE estimators[11],

$$FPE = \frac{N+m}{N-m} S_N(\widehat{\boldsymbol{w}}) \qquad GPE = \frac{N+\widehat{m}_1}{N-\widehat{m}_1} S_N(\widehat{\boldsymbol{w}}) \tag{25}$$

is – for convenience – performed for a linear model. The linear data generating system (dimension $m = 15$) is given by:

$$y(k) = \boldsymbol{x}^\top(k)\boldsymbol{w}^\circ + \varepsilon(k) \tag{26}$$

where $\boldsymbol{x}(k)$ is an i.i.d. Gaussian distributed sequence with zero mean and, the elements of $\boldsymbol{H} = E\{\boldsymbol{x}\boldsymbol{x}^\top\}$ are selected randomly, resulting in an eigenvalue-spread approx. equal to 900. The optimal weights are drawn independently from a standard Gaussian distribution. The inherent noise is a Gaussian

[11] As regards the GPE estimator, the noise variance estimation suggested in [10] is employed.

zero mean, i.i.d. sequence which is independent of the input with variance $\sigma_\varepsilon^2 = 0.25 \cdot E\left\{\left(\boldsymbol{x}^\top(k)\boldsymbol{w}^\circ\right)^2\right\} = 0.25 \cdot (\boldsymbol{w}^\circ)^\top \boldsymbol{H}\boldsymbol{w}^\circ$. That is, the signal-to-noise ratio equals approx. 6 dB.

$Q = 2.4 \cdot 10^4$ independent training sets of size N in the interval [15; 35] were randomly generated, and the weights of the associated model were estimated using a simple weight decay regularizer with $\kappa = 2\kappa_{\text{opt}}$.

The "true" average generalization error was estimated by $\widehat{\Gamma}_G = \langle G(\widehat{\boldsymbol{w}}) \rangle$ where $\langle \cdot \rangle$ denotes the average w.r.t. the Q training sets, and

$$G(\widehat{\boldsymbol{w}}) = E\left\{\left[\varepsilon + \boldsymbol{x}^\top\left(\boldsymbol{w}^\circ - \widehat{\boldsymbol{w}}\right)\right]^2\right\} = \sigma_\varepsilon^2 + \left(\boldsymbol{w}^\circ - \widehat{\boldsymbol{w}}\right)^\top \boldsymbol{H}\left(\boldsymbol{w}^\circ - \widehat{\boldsymbol{w}}\right) \quad (27)$$

The quality of the estimators[12], $\widehat{\Gamma}(\mathcal{T}) \in \{FPER, FPE, GPE\}$, is quantified by three different measures:

$$NB = \frac{\widehat{\Gamma}(\mathcal{T}) - \widehat{\Gamma}_G}{\widehat{\Gamma}_G} \cdot 100\% \qquad NRMSE = \frac{\sqrt{\left\langle \left[\widehat{\Gamma}(\mathcal{T}) - \widehat{\Gamma}_G\right]^2 \right\rangle}}{\widehat{\Gamma}_G} \cdot 100\% \quad (28)$$

$$\Pi\left(\widehat{\Gamma}\right) = \left\langle \mu\left(\left|\widehat{\Gamma}(\mathcal{T}) - \widehat{\Gamma}_G\right| - \left|FPER(\mathcal{T}) - \widehat{\Gamma}_G\right|\right)\right\rangle \quad (29)$$

NB is the normalized bias, *NRMSE* is the normalized root mean square error, and Π is the probability that *FPER* is closer to the true estimate, $\widehat{\Gamma}_G$, than another estimator, $\widehat{\Gamma}$. Here $\mu(\cdot)$ denotes the step function. Fig. 1 shows plots of the considered measures. *NB* of *FPER* is smallest for all training set sizes; however, as the training set size approaches infinity all estimates becomes identical as $\kappa_{\text{opt}} \to 0$. For $N = 35$ *NB*(*FPER*) is approx. half the *NB*(*GPE*). The *NRMSE*'s of and *GPE* are approx. identical, thus one could claim that the normalized bias improvement of *FPER* relative to *GPE* is lost at increased variance[13]. However, the probability that *FPER* is closer than *GPE* to the true Γ is around 0.65; consequently, *FPER* should be preferred to *GPE*. *FPE* shows extremely bad performance in all figures and moreover, *FPE* is *negative*, possibly infinite when $N \leq 15$.

CONCLUSION

This paper presented an consistent and $o(1/N)$ unbiased estimator of the average generalization error for a complete neural network model, called *FPER*. The network is trained by using a cost function which is the sum of the mean square error and a quadratic regularization term. The estimator may be viewed as an extension of the *FPE* and *GPE* estimators [1], [10]. It turns out that the complexity reduction obtained by using regularization is expressed in terms of *two* distinct effective number of weights, unlike defining a single

[12] Notice, the dependence on the particular training set, $\mathcal{T}$, is emphasized.

[13] That is, mean square error the minus squared bias.

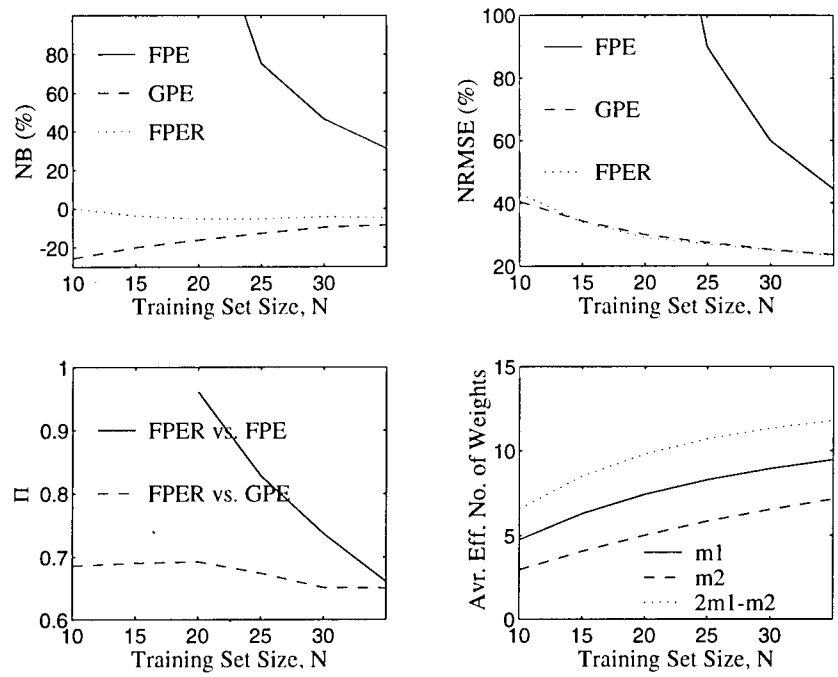


Figure 1: Comparison of *FPE*, *GPE* and *FPER*. The *FPE* curves are not calculated for $N \leq 15$, and the upper panels are cutoff at $+100\%$. The bottom right panel shows the average effective number of weights $\langle \widehat{m}_1 \rangle$, $\langle \widehat{m}_2 \rangle$ as well as $2\langle \widehat{m}_1 \rangle - \langle \widehat{m}_2 \rangle$. quantity reflecting the effective number of weights, as suggested in [9], [11]. Moreover, an expression for the optimal weight decay parameter is presented and discussed. The potential of the *FPER* estimator was demonstrated by comparative numerical studies.

ACKNOWLEDGMENTS

This work was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center.

APPENDIX

Evaluation of the terms in Eq. (10), (11) is based on two observations: First, $\partial C_N(\widehat{\boldsymbol{w}}) / \partial \boldsymbol{w} = \mathbf{0}$ since $\widehat{\boldsymbol{w}}$ minimizes $C_N(\boldsymbol{w})$. A first order Taylor series expansion of $\partial C_N(\widehat{\boldsymbol{w}}) / \partial \boldsymbol{w}$ reads[14]:

$$\frac{\partial C_N(\boldsymbol{w}^\circ)}{\partial \boldsymbol{w}} + \frac{\partial^2 C_N(\boldsymbol{w}^\circ)}{\partial \boldsymbol{w} \partial \boldsymbol{w}^\mathsf{T}} \Delta \boldsymbol{w} = \mathbf{0} \tag{30}$$

[14] Expanding beyond first order result in 3rd and higher order derivatives of the cost function which already are assumed to be negligible.

Subsequently, a few algebraic manipulations result in:

$$\Delta \boldsymbol{w} = \boldsymbol{J}_N^{-1}(\boldsymbol{w}^\circ) \left[\frac{1}{N} \sum_{k=1}^{N} \psi(k; \boldsymbol{w}^\circ)\varepsilon(k) - \boldsymbol{R}\boldsymbol{w}^\circ \right] \tag{31}$$

where $\boldsymbol{J}_N(\boldsymbol{w}^\circ)$ is the non-singular Hessian matrix of the cost function.

The second observation is an expansion of the inverse Hessian obtained by repeatedly using the matrix inversion lemma [8, App. A,B]. The result is: $\boldsymbol{J}_N^{-1}(\boldsymbol{w}^\circ) = \boldsymbol{J}^{-1}(\boldsymbol{w}^\circ) - N^{-1} \cdot \boldsymbol{J}^{-1}(\boldsymbol{w}^\circ)\boldsymbol{\Theta}\boldsymbol{J}^{-1}(\boldsymbol{w}^\circ) + \cdots$, where $\boldsymbol{\Theta} = \boldsymbol{H}_N(\boldsymbol{w}^\circ) - \boldsymbol{H}(\boldsymbol{w}^\circ)$.

REFERENCES

[1] H. Akaike, "Fitting Autoregressive Models for Prediction," Annals of the Institute of Statistical Mathematics, vol. 21, pp. 243–247, 1969.

[2] S. Geman, E. Bienenstock & R. Doursat, "Neural Networks and the Bias/Variance Dilemma," Neural Computation, vol. 4, pp. 1–58, 1992.

[3] L.K. Hansen, "Stochastic Linear Learning: Exact Test and Training Error Averages," Neural Networks, vol. 6, pp. 393–396, 1993.

[4] L.K. Hansen & C.E. Rasmussen, "Pruning from Adaptive Regularization," Preprint Electronics Institute, The Technical University of Denmark, 1993. Accepted for publication in Neural Computation.

[5] L.K. Hansen, C.E. Rasmussen, C. Svarer, & J. Larsen, "Adaptive Regularization," in Proceedings of the 1994 IEEE NNSP Workshop.

[6] A. Krogh & J.A. Hertz, "A Simple Weight Decay Can Improve Generalization," in J.E. Moody, S.J. Hanson, R.P. Lippmann (eds.) Advances in Neural Information Processing Systems 4, Proceedings of the 1991 Conference, San Mateo, California: Morgan Kaufmann Publishers, 1992, pp. 950–957.

[7] J. Larsen, "A Generalization Error Estimate for Nonlinear Systems," in S.Y. Kung, F. Fallside, J. Aa. Sørensen & C.A. Kamm (eds.) Neural Networks for Signal Processing 2: Proceedings of the 1992 IEEE-SP Workshop, Piscataway, New Jersey: IEEE, 1992, pp. 29–38.

[8] J. Larsen, Design of Neural Network Filters, Ph.D. Thesis, Electronics Institute, The Technical University of Denmark, March 1993.

[9] D. MacKay, "A Practical Bayesian Framework for Backprop Networks," Neural Computation, vol. 4, pp. 448-472, 1992.

[10] J. Moody, "Note on Generalization, Regularization, and Architecture Selection in Nonlinear Learning Systems," in B.H. Juang, S.Y. Kung & C.A. Kamm (eds.) Proceedings of the first IEEE Workshop on Neural Networks for Signal Processing, Piscataway, New Jersey: IEEE, 1991, pp. 1–10.

[11] J. Moody, "The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems," in J.E. Moody, S.J. Hanson, R.P. Lippmann (eds.) Advances in Neural Information Processing Systems 4, Proceedings of the 1991 Conference, San Mateo, California: Morgan Kaufmann Publishers, 1992, pp. 847–854.

[12] H. White, "Consequences and Detection of Misspecified Nonlinear Regression Models," Journal of the American Statistical Association, vol. 76, no. 374, pp. 419–433, June 1981.

AN APPLICATION OF IMPORTANCE-BASED FEATURE EXTRACTION IN REINFORCEMENT LEARNING

David J. Finton
Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706

Yu Hen Hu
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI 53706

Abstract—The sparse feedback in reinforcement learning problems makes feature extraction difficult. We present *importance-based feature extraction*, which guides a bottom-up self-organization of feature detectors according to top-down information as to the importance of the features; we define importance in terms of the reinforcement values expected as a result of taking different actions when a feature is recognized. We illustrate these ideas in terms of the pole-balancing task and a learning system which combines bottom-up tuning with a distributed version of Q-learning; adding importance-based feature extraction to the detector tuning resulted in faster learning.

INTRODUCTION

In reinforcement learning problems the feedback is simply a scalar value which may be delayed in time. This reinforcement signal reflects the success or failure of the entire system after it has performed some sequence of actions. Hence the reinforcement signal does not assign credit or blame to any one action (the *temporal credit assignment* problem), or to any particular node or system element (the *structural credit assignment* problem).

Since the reinforcement feedback is not an error signal for individual system elements, it gives little guidance for feature extraction, the on-line development of the system's input representation. Acting properly depends on both identifying the current context as well as selecting an action appropriate to that context, but the scalar feedback signal does not indicate which of these processes is at fault. It does not indicate whether the system should tune its feature detectors, or the weights placed on the outputs of those feature detectors, or both.

0-7803-2026-3/94 $4.00 © 1994 IEEE

IMPORTANCE-BASED FEATURE EXTRACTION

We consider reinforcement learning as the composition of two subproblems: feature extraction and "action learning" (correlating the success of actions with their context). Our approach is to solve these two subproblems with modules which are separate but loosely-coupled. The feature extraction module tunes detectors by a kind of competitive learning, and the performance module learns the value of the system actions in terms of the emerging feature detectors. A unique feature of our approach is that the detector tuning is guided by information from the performance module regarding the "importance" of the detectors.

We define "important" features to be those which enable the system to recognize situations where one action is preferable over another. One action is preferable over another if it is likely to lead to better reinforcement than the other. If the system is designed to *model* the input space, all areas of the space may be equally important. In contrast, if the learning task is a *control* task, then the only important features are those which distinguish between contexts that require different behaviors. Therefore, important areas of feature space are those for which the values of the system's actions differ greatly. Regions of feature space for which the system's actions have the same values are *un*important, no matter how frequently these regions are encountered.

In many domains the important features will occur frequently, so that feature extraction based on frequency will act much like importance-based feature extraction. But when there are important features which are infrequently seen or frequent features which are unimportant, importance-based feature extraction will enable the system to focus on the features which it needs to guide its behavior. Therefore we expect that importance-based feature extraction will outperform frequency-based feature extraction when frequency is not a reliable indicator of importance.

THE LEARNING SYSTEM

The learning system receives four inputs which indicate the system state: the position and velocity of the cart on its track, and the angular displacement and velocity of the pole. The system monitors the highest and lowest values of these inputs and automatically scales them; the result is that each input appears to the detectors to have the same size range. The scaled inputs feed into a layer of feature detector nodes, which in turn feed into a layer of effector nodes which represent system actions.

In the detector layer, the system enables the N detectors closest to the current input, and normalizes their outputs; the remaining detectors are set to 0. The

effector nodes simply calculate a weighted sum of the feature detectors. The winning effector triggers the associated system action, unless the effector values are very close, in which case the choice is made randomly.

The detectors are tuned by moving the "closest" detector toward the current point in input space. Our system is unique in defining "closest" according to a rule which combines the Euclidean distance and the *importance* of the detector. This results in a kind of importance-based feature extraction, but ordinary frequency-based feature extraction results as a special case when the importance parameter is set to 0. For the pole-balancing problem we define importance according to the difference between the two effector weights for a given detector. If these weights are equal, the detector contributes equally to the effector nodes for "push left" and "push right." Such a detector is judged "unimportant" because it has no influence on the system's behavior; therefore, our system gives precedence to retuning these unimportant detectors.

The weights between the detectors and effectors are then updated by a fuzzy version of Q-learning [9]. In our system, the output value of an effector represents a Q-value: the predicted reinforcement for taking that action from the current system state. Therefore, an important detector is one which predicts very different reinforcement values according to which effector node activates. Such detectors are valuable because they detect features which are directly relevant to choosing the best action in order to satisfy system goals.

The following sections describe the components of the system in greater detail.

Input Scaling and Detector Initialization

The learning system automatically compensates for the fact that the inputs have different ranges. During the first learning trial the system acts randomly and limits its learning to finding the extreme values of its input components. As learning progresses, the system updates these values so it can scale each input accordingly. The scaling factor used in the detector's distance calculations is $\mu_k = 2/(high_k - low_k)$, where $high_k$ and low_k are the extreme values seen for input component x_k.

The feature detectors are initialized as a lattice within the hypercube $[-0.1, 0.1]^M$, where $M = 4$ is the dimensionality of our feature space for the pole-balancing problem.

Feature Detection Module

Each detector node, i, computes its Euclidean distance to the current input vector according to

$$e_dist_i = \sqrt{\sum_k \mu_k^2 (c_k - x_k)^2}$$

Here $\mathbf{x} = (x_1, x_2, x_3, x_4)$ is the vector of system inputs, and $\mathbf{c} = (c_1, c_2, c_3, c_4)$ is the center for detector i. μ_k is the scaling parameter described above for input component k.

The output of each detector is simply the reciprocal of this Euclidean distance, truncated to the range [0, 1000000]. The system then inhibits all but the top N detectors, and scales their output values so that the sum of their squared values is normalized to unity.

The tuning rule for detector i depends on the importance of i, as well as detector i's closeness to the current input vector. We define importance in terms of the weights $w_{i,1}$ and $w_{i,2}$, which are the weights from detector i to the effector nodes for "push left" and "push right," respectively. Thus

$$imp_i = 0.5 \mid w_{i,1} - w_{i,2} \mid$$

which is in the range [0, 1], since $w_{i,j}$ is in [-1,1]. We define closeness in terms of the Euclidean distance and the importance:

$$close_i = e_dist_i(1 + \lambda imp_i),$$

with the importance factor $\lambda \geq 0$. Note that with $\lambda = 0$, the system just tunes the closest detector according to the Euclidean distance; we call this frequency-based feature extraction.

Then detector i's center is tuned by

$$\mathbf{c}^i = \begin{cases} (1-\alpha)\mathbf{c}^i + \alpha\mathbf{x} & \text{if } close_i = \min_k\{close_k\} \\ \mathbf{c}^i & \text{otherwise} \end{cases}$$

The idea of frequency-based feature extraction can be expressed in different ways, but an advantage of the simple tuning rule used here is that it allows direct comparison of importance-based and frequency-based feature extraction by simply changing the value of the parameter λ.

We used $N = 10$ in our simulations, producing a fuzzy representation of classification. Note that $N = 1$ causes the feature representation to be a *partition* of disjoint regions.

Effector Module

Each effector value represents a distributed version of a Q-value:

$$eff_j(t) = \sum_i w_{ij}(t) det_i(t)$$

where eff_j is the output value of effector j, $det_i(t)$ is the output value of detector i, and $w_{ij}(t)$ is the weight on the link between i and j.

The system evaluates its state according to two quantities:

$$goodness(t) = \max_k \{eff_k(t)\}$$

$$r(t) = \text{ reinforcement from environment at time } t$$

The weights w_{ij} from detectors to the winning effector j are updated as follows:

$$w_{ij}(t+1) = \begin{cases} w_{ij}(t) + \beta det_i(t)(r(t+1) - goodness(t)) & \text{if } r \neq 0 \\ w_{ij}(t) + \beta det_i(t)(\gamma\ goodness(t+1) - goodness(t)) & \text{otherwise} \end{cases}$$

where $\gamma \in [0, 1]$ is a discount factor. The weights to the other effectors are not updated. Note that this algorithm may be regarded as a fuzzy version of Q-learning. Indeed, if the input representation is a partition (i.e., exactly one detector is active for each input state), the predicted values, eff_k, are exactly those of Q-learning.

POLE BALANCING

The pole balancing task has been studied by Barto, Sutton and Anderson [3], Anderson [1] and others. The task involves a wheeled cart on a track, with a pole hinged to the top of the cart. At each time step (0.02 second interval) the controller must decide whether the cart should apply a force to the left or to the right, in order to keep the pole balanced vertically. The trial is judged a failure when the pole falls too far ($\geq$ 12 degrees) to either the left or the right, or when the cart falls off the track (cart position, in meters, outside the range $[-2.4, 2.4]$). The controller's input consists of five values: the four system state variables described above, and a reinforcement signal of -1 when a trial fails. The output of the controller is a binary value indicating a push on the cart either to the left or to the right.

RESULTS

We implemented the pole and cart dynamics according to the equations given in [2]. The criterion for a successful run was learning to keep the pole balanced for a trial of 100000 steps, which represents slightly more than half an hour of

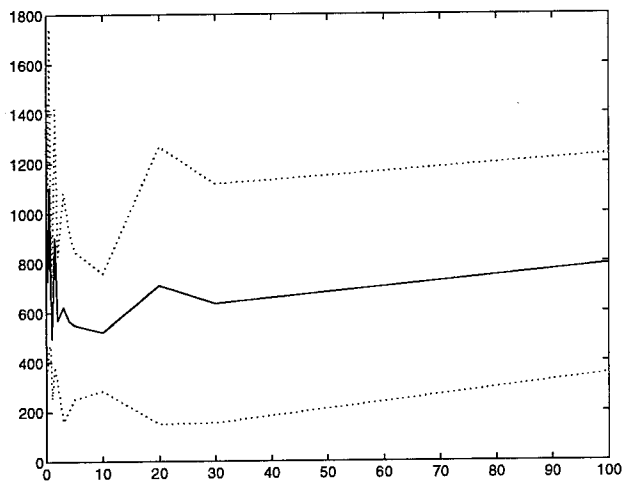


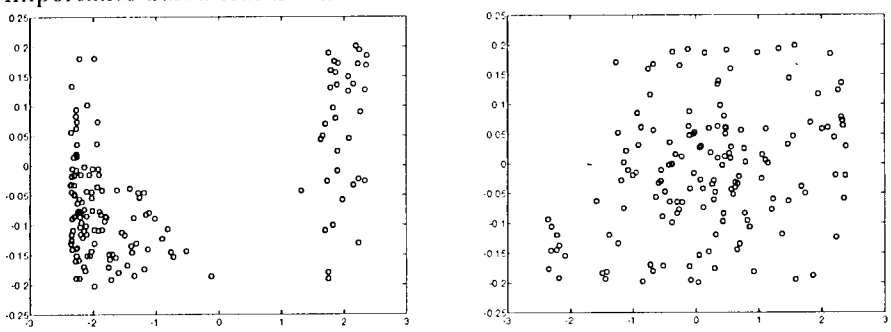
Figure 1: Number of trials vs. lambda

balancing in real-time. For each experiment, we made 15 runs with different random initialization seeds. In the first experiment, we set the value of the importance parameter, λ, to 1.0 (importance-based feature extraction). All the runs were successful, with an average of 458 failed trials until the successful trial; the standard deviation was 260 trials. The second experiment was set up like the first, but with $\lambda = 0$ (frequency-based feature extraction). Results were similar; all runs converged after an average of 537 trials, with a standard deviation of 188 trials.

For our second set of experiments we modified the simulation to initialize the pole to a six degree tilt, with the direction of tilt chosen randomly. Otherwise the experiments were set up exactly as before. Figure 1 shows the performance of the system as a function of the parameter λ. The solid line plots the average number of trials that were needed to meet the success criterion for each value of λ; the dotted lines indicate one standard deviation above and below the average. We note that with $\lambda = 0$ the system took nearly twice as long to learn to satisfy the success criterion as it did for $\lambda = 1.0$.

Figure 2 shows the final distribution of the detectors in (cart position, pole angle) space after two successful runs. The detector set on the left used frequency-based tuning, and the set on the right used importance-based tuning. We observed that our importance-based tuning created a fairly even spread of detectors through the feature space, while the frequency-based feature extraction ($\lambda = 0$) resulted in more clustering of the detectors about the extreme values of the inputs. As a result, the system may have had more difficulty achieving a tight control of the cart, because the detectors were optimized for states close to failure. A possible explanation for this effect is

Figure 2: Distribution of detectors in (cart position, pole angle) space. The left figure shows the detector set after a successful run using frequency-based tuning. The right figure shows the detector set after a successful run using importance-based feature extraction.



that small pole angles can be important but are not very frequent at first, since the system is failing often. Frequency-based feature extraction methods aim to cluster detectors according to the probability density function of the system inputs, so they may be led to give undue attention to detecting these failure points.

Our system performed input scaling, feature extraction and action learning on-line, and successfully learned the pole-balancing task. We feel that our results show the promise of combining importance-based feature extraction with temporal-difference methods such as Q-learning. In our system, importance-based feature extraction resulted in lower learning times. These results are for the simplest type of importance-based tuning, which simply selects detectors based on their importance. We are currently extending these ideas to algorithms which actively tune detectors to regions of greater importance to the system.

RELATED WORK

Most research in reinforcement learning explores feature extraction either purely in terms of the top-down feedback, or totally as a result of bottom-up self-organization. Importance-based feature extraction combines elements of both approaches. We propose that feature extraction be done by a bottom-up competitive interaction among feature detectors, but that it be guided by top-down information as to the importance of the features.

Barto, Sutton and Anderson [3], Sutton [7] and others have constructed learning systems composed of a critic module and an action module; the critic module learns to predict the reinforcement feedback, and the action module learns to perform the task based on feedback from the critic module.

Our approach differs from theirs by separating feature learning from action learning, since we feel that feature extraction is a separate problem, which requires additional information. Anderson [1] reports results for a similar system which used back propagation to learn features in the pole-balancing task. This system required around 5000 failed trials before the feature detectors emerged, with learning proceeding quickly after that point. In comparison, our importance-based feature extraction system never took more than 1000 failures to reach a successful trial in the original pole problem. Whereas Anderson's back-propagation system tuned features according to the top-down feedback, our system merely uses top-down information to guide a bottom-up competitive learning which tunes the detectors. Our position is that detector tuning requires optimization of some kind of importance measure, which is absent in typical gradient-descent tuning.

Holdaway [4] used competitive learning for feature extraction, but for a supervised learning task. His feature extraction module was trained off-line and used Kohonen's SOM [5], which produces a feature set based on the frequency of the input data points. We note that a similar kind of frequency-based feature extraction is a special case of our algorithm, obtained by setting the importance parameter, λ, to 0; however, this differs from Kohonen's SOM by not decreasing the learning rate or tuning window with time, since our system is designed to be able to continuously adapt to changing conditions. Wang and Hanson's tuning rule [8] is similar to ours, but uses a "win-rate" parameter instead of our importance parameter. Their aim is to make the detector set correspond more closely to the probability density function of the inputs by equalizing the winning rates of the detectors. In contrast, our aim is to make the detectors relevant to the system's actions and goals by retuning those detectors which are unimportant.

CONCLUSION

We feel that constructing good input representations is one of the important problems in neural network learning. A common approach is to use top-down feedback with gradient-descent to produce feature detectors in a hidden layer of nodes. This strategy works poorly for reinforcement learning problems because of their sparse, delayed feedback signals. Our work suggests that combining top-down feedback with bottom-up self-organization may be a more effective technique for feature extraction in reinforcement learning, when the effect is to guide feature extraction by the *importance* of features, rather than by their frequency. The simplicity of our model shows that such mechanisms need not be complex in order to be effective. We conclude that importance-based feature extraction is a promising basis for further studies in reinforcement learning.

REFERENCES

[1] C. W. Anderson, "Strategy learning with multilayer connectionist representations," in Proceedings of the Fourth International Workshop on Machine Learning, 1987, pp. 103–114.

[2] C. W. Anderson and W. T. Miller, III, "A challenging set of control problems," in W. T. Miller, III, R. S. Sutton and P. S. Werbos, editors, Neural Networks for Control, Cambridge, MA: MIT Press, 1990, Appendix.

[3] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-13, no. 5, pp. 834–846, September 1983.

[4] R. M. Holdaway, "Enhancing supervised learning algorithms via self-organization," in Proceedings of the International Joint Conference on Neural Networks, 1989, pp. 523–529.

[5] T. Kohonen, "Self-organized formation of topologically correct feature maps," Biological Cybernetics, vol. 43, pp. 59–69, 1982.

[6] D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning," in D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing, vol. 1, Cambridge, MA: MIT Press, 1986, ch. 5, pp. 151–193.

[7] R. S. Sutton, Temporal Credit Assignment in Reinforcement Learning, Ph.D. thesis, University of Massachusetts, Amherst, Massachusetts, 1984.

[8] L. Z. Wang and J. V. Hanson, "Competitive learning and winning-weighted competition for optimal vector quantizer design," in C. A. Kamm, G. M. Kuhn, B. Yoon, R. Chellappa and S. Y. Kung, editors, Neural Networks for Signal Processing III: Proceedings of the 1993 IEEE Workshop, IEEE Press, 1993, pp. 50–59.

[9] C. J. C. H. Watkins, Learning From Delayed Rewards, Ph.D. thesis, Cambridge University, Cambridge, England, 1989.

Multilayer Perceptron Design Algorithm

Elizabeth Wilson
Raytheon Company
Equipment Division
1001 Boston Post Road
Marlboro, MA 01752
bwilson @ sud.2.ed.ray.com
Phone: (508) 490-1769
Fax: (508) 490-3007

Donald W. Tufts
Kelley Annex
EE Department
University of Rhode Island
Kingston, RI 02881
tufts @ ele.uri.edu

May 1994

Abstract

This paper describes a design algorithm that has been developed to calculate the number of hidden nodes required and compute a good set of starting weights for the Multilayer Perceptron (MLP). There are significant advantages to being able to calculate the number of hidden nodes required. The proper choice of the number of hidden nodes results in shorter training times, better generalization, and simpler computations in implementation.

This method is then used to design an efficient, effective MLP for multiple-class decision using these simplified binary-decision neural networks. The resulting algorithmic structure has an efficient pipelined implementation. Simulations describe the application of the design algorithm and parametric classification of a transient signal. A modified wavelet feature representation is introduced as an input to the neural networks associated with arrival time discrimination.

1 Introduction

Neural networks have been used to solve a number of difficult problems and these networks are particularly useful when the statistics associated with a mapping of received inputs to a desired output are not completely known and/or are nonlinear. The Multilayer Perceptron (MLP) yields a simple feedforward network that accomplishes this mapping.

Although a popular method for training the MLP, the backpropagation algorithm [1] is often criticized for the length of time it takes to converge (if it converges) and the potential for settling into a local instead of global minimum. There have been a number of techniques proposed to improve the backpropagation algorithm by optimizing parameters, speeding up the gradient descent, pruning unnecessary weights, and using clustering algorithms to define the structure.

0-7803-2026-3/94 $4.00 © 1994 IEEE

The general mapping formulas in the literature generally lead to more nodes than necessary and longer training times. The structure of the network is critical to successful implementation, but often the number of hidden nodes in an MLP is chosen arbitrarily and modified by trial and error. Too few nodes do not properly characterize the mapping and make convergence difficult. Too many nodes will improve performance on the training set but will reduce the ability of the network to generalize to new examples [2]. The design method described here computes the number of hidden nodes required and the starting weights.

2. MLP Design Algorithm

2.1 Using Singular Value Decomposition (SVD) for Design

Let us consider the transformation from an input vector to the set of the hidden-layer node outputs as an approximate projection from the input space onto a subspace. From this point of view one suspects that the SVD can provide some insight into a better starting point for weights in Multilayer Perceptron learning algorithms. Reference [3] describes the motivation for using the SVD and the development of a test statistic [4] to consider the binary classification of a signal vector between two subspaces in the presence of white Gaussian noise. The hypotheses are:

H_1: Signal present in S: S + N (signal subspace)
H_0: Noise only A: N (alternate subspace)

The Generalized Likelihood Ratio Test can be written as

$$\underbrace{\underbrace{\hat{\mathbf{y}}^{\mathbf{H}} P_S \hat{\mathbf{y}}}_{energy\ in\ \mathbf{S}} - \underbrace{\hat{\mathbf{y}}^{\mathbf{H}} P_A \hat{\mathbf{y}}}_{energy\ in\ \mathbf{A}}}_{test\ statistic} \gtrless \underbrace{\eta_1}_{threshold} \tag{1}$$

where P_S is the projection operator for S and P_A is the projection operator for A.

This test statistic is characterized in terms of the covariance matrices corresponding to the signal subspace (represented by R_S) and the alternative subspace (R_A) and the likelihood ratio test is written as

$$\underbrace{\mathbf{y}^T \left(\mathbf{R}_A^{-1}\mathbf{R}_S - \mathbf{I}\right) \mathbf{y}}_{test\ statistic\ T} \gtrless \underbrace{\eta_2}_{threshold} \tag{2}$$

We use the following SVD of the covariance matrix product

$$R_A^{-1}R_S = U\Sigma V^T \tag{3}$$

where U is a unitary matrix and Σ is the diagonal matrix of singular values of $R_A^{-1}R_S$. The test statistic is then written as a sum

$$T = \sum_{j=1}^{N} d_j |\mathbf{u}_j^T \mathbf{y}|^2 \tag{4}$$

where d_j are the diagonal elements of Σ and u_Tj the jth column of U^T. This representation of the GLRT provides a useful tool for designing and training the MLP and computing a starting point for the weights.

2.2 Calculating the Number of Hidden Nodes

If the input-to-hidden-layer weights are viewed as approximately performing the same projection operation, the same covariance matrices can be used:

$$R_S = E\,[SS^T]$$

$$R_A = E\,[AA^T] \tag{5}$$

where S is a matrix composed of only examples from the H_1 class and A is a matrix composed of the H_0 class patterns.

The SVD of the correlation matrix product:

$$R_A^{-1}R_S = U\Sigma V^T \tag{6}$$

yields a diagonal matrix Σ. Modeling the falloff of the singular values as a simple exponential curve allows us to determine a representative time constant by taking the natural log and rearranging terms. This value is approximately the number of hidden nodes (N) required. Regardless of the technique used, the rank N of Σ can be taken to be the necessary number of hidden nodes. The rank N is less than K where K is the dimension of the input and the size of the R square matrices).

This procedure has been successful for a class of problems where there is a single output node. For the binary hypothesis test, the form of the likelihood ratio shows that only this one output node is needed. With the input layer defined by the input data, the hidden nodes calculated above, and the specification of one output, the architecture is now defined.

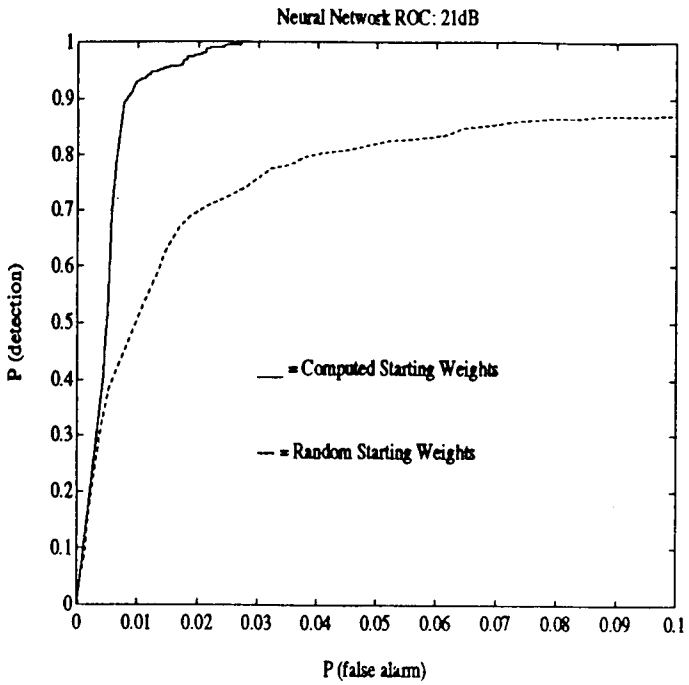


Figure 1: Improved Performance Using Design Algorithm

2.3 Computing the Starting Weights

If only the first N diagonal elements of Σ are used, then only the first N columns of U are relevant. Therefore, the matrix U in the SVD result (6) directly reveals the starting values for the input to hidden weights. Because U is a K x K square matrix, taking the first N columns results in the weight matrix $W^{(IH)}$, which is K x N as required:

$$W^{(IH)} = [u_1 \ u_2 \ \dots \ u_N] \tag{7}$$

The likelihood test statistic can be used to define starting weights from the hidden layer to the one output node. Using the diagonal elements of Σ, these starting values are:

$$W^{(H0)} = \sum_{l=j}^{N} d_j \tag{8}$$

which produces a matrix $W^{(H0)}$ that is N x 1 as desired.

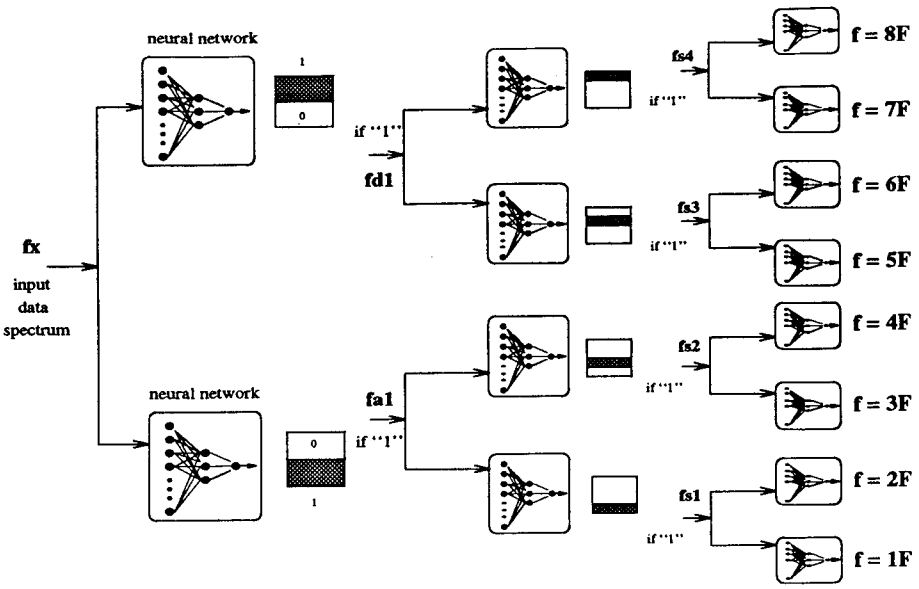


Figure 2: Multistage Implementation for Transient Frequency Feature Extraction

3. Transient Binary Classification Example

For the first binary classification example [5,6] we test for the presence of a signal component is a prescribed cell in the time-frequency sub-region.

The training set X contains examples of the time-frequency phase sampling and is separated into an S matrix comprised of the H_1 cases and an A matrix with the H_0 examples. Computing the covariance matrix product and performing the SVD described in Equation 6 and the calculations described in the algorithm yields a value of 4 for the number of hidden nodes. The first 4 columns of U are utilized as the starting weights from the input to hidden layers. The first 3 singular values are converted as shown in Equation 8 to the starting weights from the hidden layer to the output mode.

The same problem was addressed with the original training set and random weights. A network of the same size would not converge using the PDP software package [7]. Weights were added and eventually another layer was added. After extensive attempts at training an MLP the best performance was realized with a network with two hidden layers (6 nodes in the first hidden layer and 4 nodes in the second). Using fewer layers with fewer nodes in each layer yielded better performance.

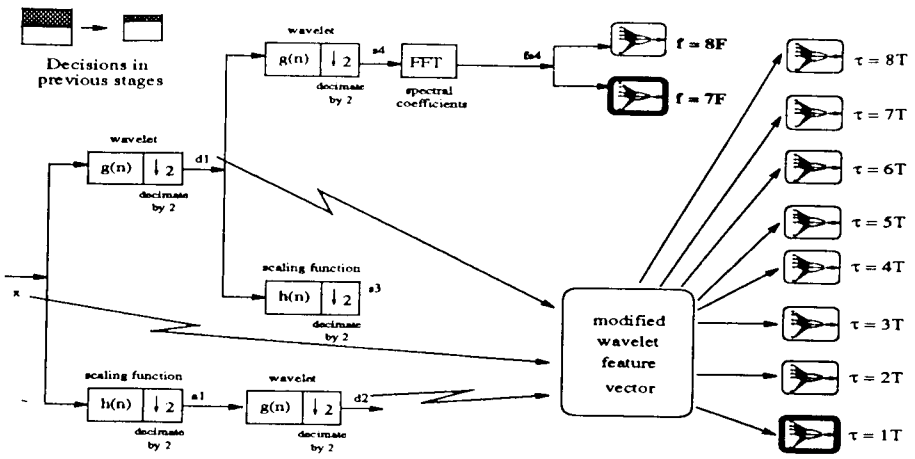


Figure 3: Multistage Implementation for t=1 and f=7 Example

The test blocks are applied to both networks and the false alarm, miss, correct H0, and correct H1 rates are computed. Figure 1 shows the resulting ROC curves plotting PFA versus PD for the random start case (dashed line) and the design algorithm case (solid line). The performance has been improved significantly and the network now has fewer hidden nodes.

4 Extension to Multiple Classes

A multistage structure is constructed to cascade binary neural networks [8]. This allows the use of smaller and simpler networks which provide for more efficient training and implementation. The pipeline architecture with parallel computations is conducive to a VLSI implementation. The design goal is to be able to train with examples that only contain one signal, but use the architecture to resolve the components of multiple signals.

The coarse stage is responsible for detecting the presence of a signal in a relatively large portion of the Time-Frequency space and transforming the input data into a more desirable form for further processing. This transformation includes the calculation of the various wavelet representations of the input signal at successive resolutions and their corresponding spectral components.

In the fine stages, only those networks associated with a detection in the coarse stage are implemented. If multiple signals are identified at the coarse stage, all of the necessary regions will be processed further. If a coarse region is classified as having no signal, no further operations will be performed on that area.

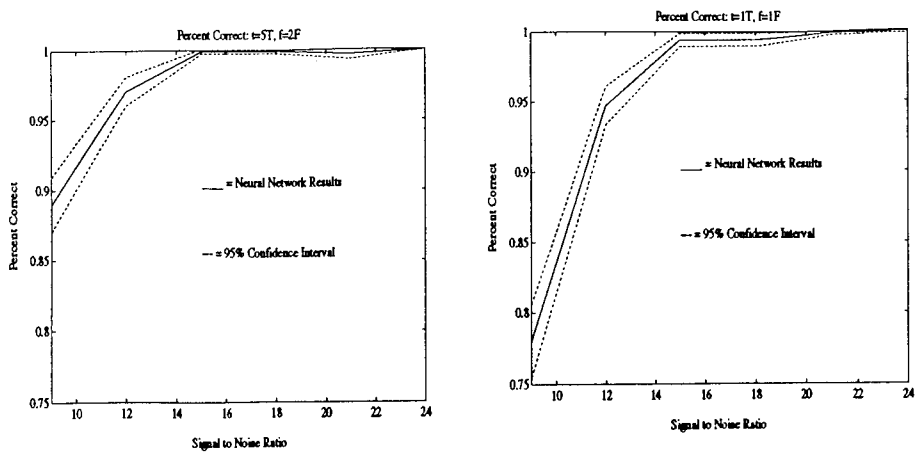


Figure 4: One Signal Performance

For both the time and frequency decision, quadrature mirror filters are used to represent the signal in the form of the Mallat wavelet transform implementation [9]. This technique has been chosen as a means of reducing the input feature size (and therefore the network sizes) from stage to stage. The filtering also removes many of the unwanted regions of the time-frequency region making the network training and implementation easier and allowing for the analysis of multiple signals with the same networks that were trained with examples of one signal. In addition, for the arrival time discrimination, the modified wavelet representation makes use of the property that the onset of the transient will line up across different scales [10,11]. The development of the modified wavelet feature vector is described in reference [3].

The implementation of the multistage architecture for the frequency feature extraction is shown in Figure 2. The first stages split the frequency band into two parts based on the network computation of the spectrum of the input signal. If a signal component is detected, the next stages uses the next resolution as its input features. Because of the decimation at each stage of the wavelet packet generation, the networks at each stage are smaller. Each network is trained separately, but the smaller network are easier to train. Also, the smaller networks use fewer weights and therefore simplify the computation during implementation. An example of the multistage implementation for t = 1T and f = 7F is shown in Figure 3. Random seeds are chosen to ensure that the test examples are different than the training examples. Test results for two of the 64 cells are shown in Figure 4.

Where the network is shown to generalize for different signal to noise ratios. The percent correct results for the two cases described at 21 dB are plotted for 24, 18, 15, 12 and 9 dB SNR levels. The dotted lines represent the 95% confidence intervals for these measurements. The performance degrades as expected at lower SNR levels, but as in the simple case the neural networks are only presented with examples having 21 dB SNR and generalize for the other cases.

References

[1] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Volume 1: Foundations*, MIT Press: Cambridge, MA, 1989.

[2] E. Leven, N. Tishby, and S.A. Solla, "A Statistical Approach to Learning and Generalization in Layered Neural Networks," *Proceedings of the IEEE*, Vol. 78, No. 10, October 1990, pp. 1568-1574.

[3] E. Wilson and D.W. Tufts, "Neural Network Design Algorithm and Multistage Structure," submitted to *IEEE Transactions on Neural Networks* 8/93.

[4] R.N. McDonough, " A Canonical Form of the Likelihood Detector for Gaussian Random Vectors," *Journal of the Acoustical Society of America*, Vol. 49, 1971, pp. 402-406.

[5] E. Wilson, S. Umesh and D.W. Tufts, "Resolving the Components of Transient Signals Using Neural Network and Subspace Inhibition Filter Algorithm," *Proceedings of the IEEE International Joint Conference on Neural Networks*, Vol. 4, 1992, pp. 283-288.

[6] E. Wilson, S. Umesh and D.W. Tufts, " Designing a Neural Network Structure for Transient Detection Using the Subspace Inhibition Filter Algorithm," *Proceedings of the IEEE Oceans Conference*, Newport, RI, Vol. 1, 1992, pp. 120-125.

[7] J.L. McClelland and D.E. Rumelhart, *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises.* MIT Press: Cambridge, MA, 1988.

[8] E. Wilson, S. Umesh and D.W. Tufts, "Multistage Neural Network Structure for Transient Detection and Feature Extraction," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, 1993, pp. 489-492.

[9] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. II, No. 7, July 1989, pp. 674-693.

[10] S. Mallat and S. Zhong, "Wavelet Transform Maxima and Multiscale Edges," in *Wavelets and their Applications*, ed. M.B. Ruskai, et al. Jones and Barlett: Boston, 1992, pp. 67-104.

[11] S. Mallat and S. Zhong, "Characterization of Signals from Multiscale Edges," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, July 1989, pp. 710-732.

MIXTURE DENSITY ESTIMATION VIA EM ALGORITHM WITH DETERMINISTIC ANNEALING

†‡Naonori UEDA †Ryohei NAKANO

†NTT Communication Science Laboratories
Hikaridai Seika-cho Soraku-gun Kyoto 619-02 Japan

‡Purdue University, Electrical Engineering Dept., PO.Box-29,
West Lafayette IN 47907 USA
Tel: 1-317-494-3378 Fax: 1-317-494-6440
E-mail: ueda@ecn.purdue.edu

Abstract- We present a new approach for the problem of estimating the parameters which determine a mixture density. The approach utilizes the *principle of maximum entropy* and *statistical mechanics analogy*. The EM process which is well known as a maximum likelihood estimation method is reformulated as the minimization problem of thermodynamic free energy. Unlike stochastic relaxation or simulated annealing, the minimization is *deterministically* performed. Moreover, the derived algorithm, unlike the conventional EM algorithm, can estimate the parameters free of initial parameter values.

INTRODUCTION

Finite mixture densities consist of a mixture of finite component densities and mixing proportions. Such densities appear as fundamental models in areas of statistical pattern recognition and classification [1][2]. In particular, Gaussian mixture model plays an important role in continuous density Hidden Markov Models [3]. The method of maximum likelihood has been regarded as the best approach for parameter estimation problems during the past three decades. However, for mixture density problems, likelihood equations become nonlinear and therfore we can not obtain an analytical solution. In other words, we should seek an approximate solution by an iterative procedure.

Dempster, Laird and Rubin proposed an iterative procedure, called EM algorithm, to numerically approximate maximum likelihood estimates [4]. The EM algorithm has been applied to a wide variety of mixture problems because of its advantage of reliable global convergence, low cost per iteration, economy of storage and ease of programming [5]. However, since maximum likelihood equations for mixture models usually have multiple roots, the EM algorithm suffers from a local maxima problem. Namely, the algorithm is highly sensitive to the initial parameter values. Indeed, the EM algorithm should be applied

0-7803-2026-3/94 $4.00 © 1994 IEEE

from a wide choice of starting values according to some ad hoc criterion.

The local maxima problem is one of the important issues in the mixture density problem. Nevertheless, as far as the information from our literature survey, no published paper has explicitly given a technique for this problem. Although several algorithms for approximating maximum likelihood estimates have recently been found in Neural Networks [6][7][8], they do not deal with this problem.

To overcome the problem, we adopt the principles of statistical mechanics. By using the principle of maximum entropy, the thermodynamic free energy is defined as an effective cost function depending on the *temperature*. The maximization of *log-likelihood* is done by minimizing the cost function. Unlike stochastic relaxation (or simulated annealing) [9], where random search is performed on the given energy surface, this cost function is *deterministically* optimized at each temperature.

Recently, such deterministic annealing has been adopted by several researchers [10][11][12]. However, in these studies, the formulation is not for the mixture density estimation problem, but for vector quantization or clustering design problem. In this paper, we first propose a deterministic annealing variant of the EM algorithm for the mixture density estimation problem.

THEORY OF MIXTURE DENSITY ESTIMATION

In this section, as the basis of our annealing EM approach, we will briefly explain the theory of the EM algorithm for the mixture density estimation.

Mixture Density Models

A parametric family of finite mixture densities consists of a finite given number, say C, of components, $\omega_1, \cdots, \omega_C$, in some proportions $\alpha_1, \cdots, \alpha_C$, respectively, where

$$\sum_{i=1}^{C} \alpha_i = 1 \quad \text{and} \quad \alpha_i \geq 0 \qquad \text{for } i = 1, \ldots, C. \tag{1}$$

Therefore, the probability density function (pdf) of a finite mixture is represented as:

$$p(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_{i=1}^{C} \alpha_i p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i), \qquad \boldsymbol{x} = (x_1, \cdots, x_d)^t \in \boldsymbol{R}^d, \tag{2}$$

where $p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i)$ is the conditional pdf corresponding to the component ω_i. The vector, $\boldsymbol{\Theta} = (\alpha_1, \cdots, \alpha_C, \boldsymbol{\theta}_1^t, \cdots, \boldsymbol{\theta}_C^t)^t$, consists of unknown parameters associated with the parametric forms adopted for these C component densities (t denotes vector transpose). For example, in the particular case of multivariate Gaussian component densities, $\boldsymbol{\theta}_i$ consists of a mean vector $\boldsymbol{\mu}_i$ and the elements of the covariance matrix $\boldsymbol{\Sigma}_i$.

Maximum Likelihood Estimation via EM Algorithm

In *maximum likelihood estimation*, the unknown parameter $\boldsymbol{\Theta}$ is estimated so that the log-likelihood (the natural logarithm of the likelihood) of the mixture pdf of (2), given by

$$L(\boldsymbol{\Theta}) = \sum_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{\Theta}), \tag{3}$$

is maximized by using a set $\boldsymbol{X}$ of observable samples drawn independently according to the probability $p(\boldsymbol{x}|\boldsymbol{\Theta})$. Accordingly, an estimate of $\boldsymbol{\Theta}$ can be obtained as a solution of the likelihood equation given by

$$\partial L(\boldsymbol{\Theta})/\partial \boldsymbol{\Theta} = 0 \tag{4}$$

Unfortunately in mixture models, likelihood equations are usually nonlinear, which means that the general analytical solution of the log-likelihood equation does not exist.

In a mixture density estimation problem, an observed sample $\boldsymbol{x}$ is *incomplete* because the information ω_i which indicates the component from which the sample originates is unobservable. The EM algorithm for the mixture density estimation problem is best regarded as a specialization of the general EM algorithm for obtaining maximum likelihood estimates from *incomplete* data. Let $L_c(\boldsymbol{\Theta})$ denote the *complete data log-likelihood.* In the mixture density given in (2), we can specify the complete log-likelihood as:

$$L_c(\boldsymbol{\Theta}) = \log \alpha_i p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i). \tag{5}$$

Note that in this context, the original log-likelihood, $L(\boldsymbol{\Theta})$, is referred to as the *incomplete data log-likelihood.*

For given $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$, the purpose of the EM algorithm is to maximize the incomplete data log-likelihood $L(\boldsymbol{\Theta})$ by using the complete data log-likelihood $L_c(\boldsymbol{\Theta})$. In the EM algorithm, the parameter $\boldsymbol{\Theta}$ is iteratively estimated by using two steps, E (for Expectation) and M (for Maximization). The E-step computes the conditional expectation of the complete data log-likelihood given $\boldsymbol{X}, \boldsymbol{\Theta}^{(k)}$,

$$\begin{aligned} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(k)}) &= E\{L_c(\boldsymbol{\Theta})|\boldsymbol{X}, \boldsymbol{\Theta}^{(k)}\} \\ &= \sum_{\boldsymbol{x}} \sum_{i} P(\omega_i|\boldsymbol{x}, \boldsymbol{\Theta}^{(k)}) \log \alpha_i p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i). \end{aligned} \tag{6}$$

In this equation, $\boldsymbol{\Theta}^{(k)}$ is an estimate value of the parameters at the k-th iteration. The M-step maximizes this $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(k)})$ function with respect to $\boldsymbol{\Theta}$ to estimate the new parameter value $\boldsymbol{\Theta}^{(k+1)}$:

$$\boldsymbol{\Theta}^{(k+1)} = \arg\max_{\boldsymbol{\Theta}} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(k)}). \tag{7}$$

The above two steps are repeatedly performed until a certain convergence criterion is met. From the maximization in the M-step, the following constraints

are derived [3]:

$$\alpha_i^{(k+1)} = \frac{1}{N} \sum_{\boldsymbol{x}} P(v\omega_i | \boldsymbol{x}, \boldsymbol{\theta}_i^{(k)}), \tag{8}$$

$$\sum_{\boldsymbol{x}} P(\omega_i | \boldsymbol{x}, \boldsymbol{\theta}_i^{(k)}) \frac{\partial}{\partial \boldsymbol{\theta}_i} \log p(\boldsymbol{x} | \omega_i, \boldsymbol{\theta}_i) = 0. \tag{9}$$

An increase in Q implies an increase in the incomplete data log-likelihood. Hence the incomplete data log-likelihood (or the original log-likelihood $L(\boldsymbol{\Theta})$) is monotonically increased, that is, $L(\boldsymbol{\Theta}^{(k+1)}) \geq L(\boldsymbol{\Theta}^{(k)})$, which means that $L(\boldsymbol{\Theta}^{(k)})$ converges to a local maximum.

$P(\omega_i | \boldsymbol{x}, \boldsymbol{\theta}_i^{(k)})$ denotes a the posterior probability that $\boldsymbol{x}$ belongs to the i-th component ω_i and plays an important role on calculating Q-function. In the mixture density problem, recall that we cannot obtain the information from which component the observed data originates. We must estimate the missing information according to the posterior probability. However, since the posterior probability is calculated by the Bayes rule as

$$P(\omega_i | \boldsymbol{x}, \boldsymbol{\theta}_i^{(k)}) = \frac{\alpha_i^{(k)} p(\boldsymbol{x} | \omega_i, \boldsymbol{\theta}_i^{(k)})}{\sum_j \alpha_j^{(k)} p(\boldsymbol{x} | \omega_j, \boldsymbol{\theta}_j^{(k)})}, \tag{10}$$

its reliability highly depends on the parameters $\alpha_i^{(k)}, \boldsymbol{\theta}_i^{(k)}$, and goes back to $\alpha_i^{(0)}, \boldsymbol{\theta}_i^{(0)}$. Therefore, the performance of the EM algorithm is sensitive to an initial parameter value $\boldsymbol{\Theta}^{(0)}$. It is, in general, extremely difficult to set a good initial parameter value and hence, the EM algorithm will be often trapped by local maxima.

DETERMINISTIC ANNEALING APPROACH

In this section, we present a new approach for attempting global maximization of the complete data log-likelifood in the EM process.

Statistical Mechanics Formulation

Let $P(\boldsymbol{x} \in \omega_i)$ be the probability that $\boldsymbol{x}$ comes from the component ω_i. Using $P(\boldsymbol{x} \in \omega_i)$, we rewrite $-Q$-function as follows:

$$E = \sum_{\boldsymbol{x}} \sum_i P(\boldsymbol{x} \in \omega_i) l_c(\boldsymbol{x}, \boldsymbol{\theta}_i), \tag{11}$$

$$\text{where} \quad l_c(\boldsymbol{x}, \boldsymbol{\theta}_i) = -\log \alpha_i p(\boldsymbol{x} | \omega_i, \boldsymbol{\theta}_i). \tag{12}$$

Note that $l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)$ is equivalent to $-L_c(\boldsymbol{\Theta})$ and is always nonnegative. Clearly, $(-E)$ is also the expectation of the complete data log-likelihood. However, it is different from Q in that the expectation is taken with respect to $P(\boldsymbol{x} \in \omega_i)$ instead of $P(\omega_i | \boldsymbol{x}, \boldsymbol{\theta}_i^{(k)})$. Therefore, if $P(\boldsymbol{x} \in \omega_i) = P(\omega_i | \boldsymbol{x}, \boldsymbol{\theta}_i^{(k)})$, then the maximization of Q is equivalent to the minimization of E.

Since originally we do not have *a priori* knowledge about $P(\boldsymbol{x} \in \omega_i)$, we apply the ***principle of maximum entropy*** to specify the probability. That is, $P(\boldsymbol{x} \in \omega_i)$ may be obtained by maximizing the entropy, H, given by

$$H = -\sum_{\boldsymbol{x}} \sum_i P(\boldsymbol{x} \in \omega_i) \log P(\boldsymbol{x} \in \omega_i), \tag{13}$$

with respect to P under constraints

$$\sum_i P(\boldsymbol{x} \in \omega_i) = 1 \quad \text{and} \quad \sum_{\boldsymbol{x}} \sum_i P(\boldsymbol{x} \in \omega_i) l_c(\boldsymbol{x}, \boldsymbol{\theta}_i) = E. \tag{14}$$

Then the augmented objective function to be maximized becomes

$$H_a = H + \lambda(\sum_i P(\boldsymbol{x} \in \omega_i) - 1) + \beta(\sum_{\boldsymbol{x}} \sum_i P(\boldsymbol{x} \in \omega_i) l_c(\boldsymbol{x}, \boldsymbol{\theta}_i) - E), \tag{15}$$

where λ and β are Lagrange multipliers. Computing $\partial H_a / \partial P = 0$, we have

$$P(\boldsymbol{x} \in \omega_i) = \exp\{1 - \lambda - \beta l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)\}. \tag{16}$$

From $\sum_i P(\boldsymbol{x} \in \omega_i) = 1$,

$$\exp\{1 - \lambda\} = 1 / \sum_i \exp\{-\beta l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)\}. \tag{17}$$

Thus, by sbstituting (17) into (16), we obtain the Gibbs distribution,

$$P(\boldsymbol{x} \in \omega_i) = \frac{1}{Z_{\boldsymbol{x}}} \exp\{-\beta l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)\}, \tag{18}$$

where $Z_{\boldsymbol{x}}$ is the ***partition function***: $Z_{\boldsymbol{x}} = \sum_j \exp\{-\beta l_c(\boldsymbol{x}, \boldsymbol{\theta}_j)\}$. From the above derivation, one can see that the parameter β corresponding to the Lagrange multiplier is determined by the value of E. From an analogy of the simulated annealing approach, $1/\beta$ corresponds to the "***computational temperature***". Since each sample $\boldsymbol{x} \in \boldsymbol{X}$ is drawn independently, the partition function for $\boldsymbol{X}$ is

$$\begin{aligned} Z(\boldsymbol{\Theta}) &= \prod_{\boldsymbol{x}} Z_{\boldsymbol{x}} \\ &= \prod_{\boldsymbol{x}} \sum_i \exp\{-\beta l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)\}. \end{aligned} \tag{19}$$

Once the total partition function is obtained explicitly, using a statistical mechanics analogy, we can define the *free energy* as an effective cost function depending on the temperature:

$$\begin{aligned} F(\boldsymbol{\Theta}) &= -\frac{1}{\beta} \log Z(\boldsymbol{\Theta}) \\ &= -\frac{1}{\beta} \sum_{\boldsymbol{x}} \log \sum_i \exp\{-\beta l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)\}. \end{aligned} \tag{20}$$

Minimizing Thermodynamic Free Energy

At equilibrium, it is known that a thermodynamic system settles into a configuration that minimizes its free energy. Moreover, statistical physics states that maximizing the entropy at a fixed temperature ($= 1/\beta$) is equivalent to minimizing the free energy. Hence, consider the following minimization problem:

$$\text{Mimimize} \quad F(\boldsymbol{\Theta}) = -\frac{1}{\beta}\sum_{\boldsymbol{x}} \log \sum_i \exp\{-\beta l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)\} \quad \text{with respect to } \boldsymbol{\Theta},$$

$$\text{subject to} \quad \sum_i \alpha_i = 1. \tag{21}$$

To solve the above problem, consider the augmented objective function given by

$$F_a(\boldsymbol{\Theta}) = F(\boldsymbol{\Theta}) + \lambda(\sum_i \alpha_i - 1), \tag{22}$$

where λ is a Lagrange multiplier. Setting the partial derivative of F_a with respect to α_i to zero,

$$\frac{\partial F_a}{\partial \alpha_i} = \sum_{\boldsymbol{x}} \frac{\exp\{-\beta l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)\}}{\sum_i \exp\{-\beta l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)\}} \frac{\partial l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)}{\partial \alpha_i} + \lambda = 0. \tag{23}$$

On the other hand,

$$\begin{aligned} \frac{\partial l_c(\boldsymbol{x}, \boldsymbol{\theta}_i)}{\partial \alpha_i} &= -\frac{\partial}{\partial \alpha_i} \log \alpha_i p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i) \\ &= -\frac{1}{\alpha_i}. \end{aligned} \tag{24}$$

Note that $p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i)$ does not depend on α_i. By subsutituting (24) into (23) and by using (18), (23) becomes simpler as follows:

$$\alpha_i = \frac{1}{\lambda}\sum_{\boldsymbol{x}} P(\boldsymbol{x} \in \omega_i). \tag{25}$$

Moreover, since $\sum_i \alpha_i = 1$,

$$\begin{aligned} 1 &= \frac{1}{\lambda}\sum_i \sum_{\boldsymbol{x}} P(\boldsymbol{x} \in \omega_i) \\ &= \frac{1}{\lambda}\sum_{\boldsymbol{x}} 1. \end{aligned} \tag{26}$$

Therefore,

$$\lambda = N. \tag{27}$$

By substituting (27) into (25), we have

$$\alpha_i = \frac{1}{N}\sum_{\boldsymbol{x}} P(\boldsymbol{x} \in \omega_i). \tag{28}$$

Next, similarly, by setting the partial derivative of F_a with respect to $\boldsymbol{\theta}_i$ to zero, we have

$$\sum_{\boldsymbol{x}} P(\boldsymbol{x} \in \omega_i) \frac{\partial}{\partial \boldsymbol{\theta}_i} \log p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i) = 0. \tag{29}$$

Comparing (8) with (9), and (28) with (29), respectively, one can see that the same equations as the results of the maximization of the Q-function have been derived, except that the posterior probability $P(\omega_i|\boldsymbol{x}, \boldsymbol{\theta}_i)$ in (8) and (9) is replaced by $P(\boldsymbol{x} \in \omega_i)$. Moreover, by substituting (12) into (18), it is shown that $P(\boldsymbol{x} \in \omega_i)$ is in fact a parameterized variant of the original posterior probability:

$$P(\boldsymbol{x} \in \omega_i) = \frac{(\alpha_i p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i))^\beta}{\sum_j (\alpha_j p(\boldsymbol{x}|\omega_j, \boldsymbol{\theta}_j))^\beta}. \tag{30}$$

Note that $P(\boldsymbol{x} \in \omega_i)$ with $\beta = 1$ completely agrees with the original posterior probability given by (10).

Annealing Variant of EM Algorithm

Let $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(k)}; \beta)$ be the conditional expectation of the complete data log-likelihood by the parameterized posterior probability $P(\boldsymbol{x} \in \omega_i)$, then the following deterministic annealing variant of the EM algorihtm can be naturally induced as follows:

[Annealing EM algorithm]

1. Set $\beta \leftarrow \beta_{min} (\ll 1)$.
2. Choose an initial estimate $\boldsymbol{\Theta}^{(0)}$ *arbitrarily*. Set $k \leftarrow 0$.
3. Iterate the following two steps until converged:
   E-step: Compute

$$Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(k)}; \beta) = \sum_{\boldsymbol{x}} \sum_i \left\{ \frac{(\alpha_i^{(k)} p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i^{(k)}))^\beta}{\sum_j (\alpha_j^{(k)} p(\boldsymbol{x}|\omega_j, \boldsymbol{\theta}_j^{(k)}))^\beta} \right\} \log \alpha_i p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i). \tag{31}$$

   M-step: Find $\boldsymbol{\Theta}^{(k+1)} = \arg\max_{\boldsymbol{\Theta}} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(k)}; \beta)$.
4. Increase β.
5. If $\beta < \beta_{max}$, set $k \leftarrow k + 1$, repeat from step 3.

An important distinction to keep in mind is that unlike stochastic relaxation, the optimization in step 3 is ***deterministically*** performed at each β. The above algorithm is the same as the original EM algorithm for the mixture density parameter estimation, except that an outer loop for the annealing process is added to the original EM algorithm. In other words, if both β_min and β_max are set to one in the annealing EM algorithm, the algorithm completely coincides with the original EM algorithm.

Consequently, in the case of the mulivariate Gaussian mixture pdf ($p(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_i \alpha_i g_i(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$), by just replacing the posterior probability in the familiar

iterative maximum likelihood estimation [3], which are obtained in M-step in the original EM algorithm, by the parameterized posterior probablity, we can easily modify the iterative estimation in the M-step as follows:

$$\alpha_i^{(k+1)} = \frac{1}{N}\sum_{\boldsymbol{x}} P(\boldsymbol{x} \in \omega_i)^{(k)},$$

$$\boldsymbol{\mu}_i^{(k+1)} = \sum_{\boldsymbol{x}} \boldsymbol{x} P(\boldsymbol{x} \in \omega_i)^{(k)} / \sum_{\boldsymbol{x}} P(\boldsymbol{x} \in \omega_i)^{(k)},$$

$$\boldsymbol{\Sigma}_i^{(k+1)} = \sum_{\boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{\mu}_i)(\boldsymbol{x} - \boldsymbol{\mu}_i)^t P(\boldsymbol{x} \in \omega_i)^{(k)} / \sum_{\boldsymbol{x}} P(\boldsymbol{x} \in \omega_i)^{(k)},$$

$$\text{where} \quad P(\boldsymbol{x} \in \omega_i)^{(k)} = (\alpha_i^{(k)} g_i(\boldsymbol{x}|\boldsymbol{\mu}_i^{(k)}, \boldsymbol{\Sigma}_i^{(k)}))^\beta / \sum_j (\alpha_j^{(k)} g_j(\boldsymbol{x}|\boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)}))^\beta.$$

DISCUSSION

What is the effect of the parameter β? How does the annealing feature help to avoid local maxima? The annealing process begins at $\beta = 0$ ($P(\boldsymbol{x} \in \omega_i)$ becomes uniform) where each $\boldsymbol{x} \in \boldsymbol{X}$ equally contributes to all components of the mixture. Clearly, at this time, the parameterized Q-function has only one (global) maximum. As a result, all components of the mixture converge to the same pdf. For example, in the case of a Gaussian mixture, $\alpha_1 = \cdots = \alpha_C = \alpha^*$, $\boldsymbol{\mu}_1 = \cdots = \boldsymbol{\mu}_C = \boldsymbol{\mu}^*$, and $\boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_C = \boldsymbol{\Sigma}^*$, which means that all components completely overlap as one component.

Then by gradually increasing β, the influence of each $\boldsymbol{x}$ is gradually localized. At $\beta > 0$ the parameterized Q function will have several local maxima. However, at each step, it can be assumed that the new global maximum is close to the previous one. Hence, the algorithm can track the global maximum at each β while increasing β. As a result, as β increases, a finer structure, which is closer to the true mixture density to be estimated, gradually emerges. The parameterized Q-function coincides with the original Q-function when $\beta = 1$, therefore $\beta_m ax = 1$ may be appropriate.

The proposed algorithm is straightforwardly applicable to learning the (Generalized) Radial Basis Function (RBF, GRBF) networks. In fact, Nowlan [5] proposes a maximum likelihood competitive learning algorithm for RBF networks. In [5], "*soft competition*" and "*hard competition*" are experimentally compared and it is shown that soft competition can give better performance. On the other hand, in our algorithm, the soft model corresponds exactly to the case $\beta = 1$, while the hard model corresponds to the case $\beta \to \infty$. Consequently, both models can be regarded as a special case in our algorithm. Furthermore, it can be regarded that the deterministic annealing approach employs a similar learning strategy to Kohonen's self-organizing feature map, in the sense that the influence of neighborhood learning is gradually reduced.

At present, we are experimenting with the proposed algorithm in a real multivariate Gaussian mixture density estimation problem to verify the usefulness of the algorithm.

ACKNOWLEDGEMENT

We thank Dr. Tsukasa Kawaoka for his encouragement.

REFERENCES

[1] Fukunaga K.,: "Introduction to Statistical Pattern Recognition", Academic Press, New York, 1972.

[2] Duda R. O. and Hart P. E.,: "Pattern Classification and Scene Analysis", John Wiley, New York, 1973.

[3] Huang X. D., Ariki Y. and Jack M. A.,: "Hidden Markov Models for Speech Recognition", Edinburgh Univ. Press, 1990.

[4] Dempster A. P., Laird N. M. and Rubin D. B.,: "Maximum-likelihood from incomplete data via the EM algorithm", *J. Royal Statist. Soc. Ser. B (methodological)*, vol.39, pp. 1–38, 1977.

[5] Render R. A. and Walker H. F.,: "Mixture densities, maximum likelihood and the EM algorithm", *Society for Industrial and Applied Math. Review*, vol.26. no.2, 1984.

[6] Nowlan S. J.,: "Maximum likelihood competitve learning", in *Advances ? Neural Information Systems*, pp. 574–582, 1990.

[7] Perlovsky L. I. and McManus M. M.,:"Maximum likelihood neural networks for sensor fusion and adaptive classification,ctions", *Neural Networks*, vol.4, pp. 89–102, 1991.

[8] Lee S. and Shimoji S.,: "BAYESNET: Bayesian classification network based on biased random competition using Gaussian kernels", in proc. *IEEE ICNN93*, pp. 1354–1359, 1993.

[9] Geman S. and Geman D.,: "Stochastic relaxation, Gibbs distribution and the Baysian restortion in images", *IEEE Trans. Pattern Anal. Machine Intell.*, vol.6, 6, pp. 721–741, 1984.

[10] Rose K, Gurewitz E. and Fox G. C.,: "Vector quantization by deterministic annealing", *IEEE Trans. Information Theory*, vol.38, no.4, pp.1249–1257, 1992.

[11] Buhmann J. and Kuhnel H.,: "Complexity optimized data clustering by competitive neural networks", *Neural Computation*, vol.5, pp.75–88, 1993.

[12] Wong Y.,: "Clustering data by melting", *Neural Computation*, vol.5, pp.89–104, 1993.

ADAPTIVE REGULARIZATION

L. K. Hansen, C. E. Rasmussen*, C. Svarer, and J. Larsen
CONNECT, Electronics Institute B349
Technical University of Denmark,
DK-2800 Lyngby, Denmark
email: lkhansen,ed,csvarer,jlarsen@ei.dtu.dk

Abstract. Regularization, e.g., in the form of weight decay, is important for training and optimization of neural network architectures. In this work we provide a tool based on asymptotic sampling theory, for iterative estimation of weight decay parameters. The basic idea is to do a gradient descent in the estimated generalization error with respect to the regularization parameters. The scheme is implemented in our *Designer Net* framework for network training and pruning, i.e., is based on the diagonal Hessian approximation. The scheme does not require essential computational overhead in addition to what is needed for training and pruning. The viability of the approach is demonstrated in an experiment concerning prediction of the chaotic Mackey-Glass series. We find that the optimized weight decays are relatively large for densely connected networks in the initial pruning phase, while they decrease as pruning proceeds.

INTRODUCTION

Learning based on the conventional feed-forward net may be analyzed with statistical methods and the result of such analysis can be applied to model optimization [5, 6, 10, 11, 12]. We have shown how pruning and regularization can be combined to design compact networks for time series prediction [11, 12]. Our "Designer Net" framework is based on the *Optimal Brain Damage* (OBD) method of Le Cun *et al.* [7] and we use simple weight decay for regularization. The benefits from compact architectures are three-fold: Their generalization ability is better, they carry less computational burden, and they are faster to adapt if the environment changes. Further, we have shown how the generalization error of the network may be estimated – without extensive cross-validation – using a modification of Akaike's *Final Prediction Error* (FPE) estimate [1]. The minimal FPE constitutes a useful stopping

*Present address: Dept. of Computer Science, University of Toronto, Canada.

0-7803-2026-3/94 $4.00 © 1994 IEEE

criterion for pruning. However, our previous work has been conditioned on the correct setting of several parameters, most prominently the weight decay parameters. *In this contribution we provide the possibility of adapting regularization parameters within the Designer Net framework.*

The results obtained can be viewed as a *sampling theory* alternative to the Bayesian or *Evidence* based techniques for adaptive regularization developed by MacKay [8, 9]. An analytical comparison of these two techniques has recently been given in [2].

LEARNING

The use of *system identification* tools for neural net learning has been pioneered by Moody (see e.g., [10]) who derived estimators for the average generalization error. The main source of uncertainty in the learning process is the shortage of training data. Other important contributions to uncertainty are: Lack of fit, noise in the training process, and non-stationarity of the data-generating environment. Lack of fit[1] was discussed in, e.g., [5], while noise in the training process has been discussed in [3]. In this presentation we will neglect these three effects. Lack of fit can be minimized by starting the pruning process from large enough networks, while noise in the training process can be relieved by careful search in weight space. Non-stationarity is a hard problem that will be pursued in future work, here we will assume stationarity.

NETWORK ARCHITECTURE AND TRAINING

The basic network is a *tapped delay line architecture* with L input units, n_H hidden sigmoid units and a single linear output unit. The initial network is fully connected between layers and implements a non-linear mapping from lag space $\mathbf{x}(k) = [x(k), ..., x(k-L+1)]$, ($L$ is the length of the tapped delay line), to the real axis:

$$\widehat{y}(k) = F_{\mathbf{u}}(\mathbf{x}(k)) \qquad \widehat{y} \in \mathcal{R}, \tag{1}$$

where $\mathbf{u} = [\mathbf{w}, \mathbf{W}]$ is the N-dimensional weight vector and $\widehat{y}(k)$ is the prediction of the target signal $y(k)$. The particular family of non-linear mappings considered here can be written as:

$$F_{\mathbf{u}}(\mathbf{x}(k)) = \sum_{j=1}^{n_H} W_j \tanh\left(\sum_{i=1}^{L} w_{ij} x(k-i-1) + w_{i0}\right) + W_0, \tag{2}$$

where n_H is the number of hidden units, W_j are the hidden-to-output weights, while w_{ij} connect the input and hidden units.

[1]Lack of fit is also sometimes described as "the teacher does not belong to student space" or "incomplete modeling".

A simulator based on *batch mode*, second order local optimization has been developed, as described in [11, 12]. The scheme is based on the *diagonal approximation* of the cost-function Hessian (the second derivative matrix). We use the sum of squared errors to measure the performance of the current network:

$$E_{\text{train}} = \frac{1}{p}\sum_{k=1}^{p}\left[y(k) - F_{\mathbf{u}}(\mathbf{x}(k))\right]^2, \tag{3}$$

where p is the number of training examples. To ensure numerical stability and for assisting the pruning procedure we augment the cost-function with a weight decay term:

$$E = E_{\text{train}} + \frac{\alpha_w}{p}\sum_{ij}^{N_w} w_{ij}^2 + \frac{\alpha_W}{p}\sum_{j}^{N_W} W_j^2, \tag{4}$$

where N_w, N_W are the numbers of weights and thresholds in hidden and output units, respectively. Further, α_w, α_W are the weight decay parameters of the hidden and output layers, respectively. The objective of the training procedure is to optimize the networks ability to predict near future values of a given time series. Hence, the network weights, $\mathbf{u}$, are trained to recognize the short time structure of the training set time series.

PRUNING

The OBD method proposed by Le Cun *et al.* [7] was successfully applied to reduce large networks for recognition of handwritten digits. The basic idea is to estimate the increase in the *training error* when deleting weights. The estimate is formulated in terms of weight *saliencies* s_l:

$$\delta E_{\text{train}} = \sum_{l\in D} s_l \equiv \sum_{l\in D}\left(\frac{2\alpha}{p} + \frac{1}{2}\frac{\partial^2 E_{\text{train}}}{\partial u_l^2}\right) u_l^2, \tag{5}$$

where u_l is a component of $\mathbf{u}$ and the sum runs over the set D of weights to be deleted. The saliency definition used here takes into account that the weight decay terms force the weights to depart from the minimum of the training set error. As in [7] we approximate the second derivative by the positive semi-definite expression:

$$\frac{\partial^2 E_{\text{train}}}{\partial u_j^2} \approx \frac{2}{p}\sum_{k=1}^{p}\left(\frac{\partial F_{\mathbf{u}}(\mathbf{x}(k))}{\partial u_j}\right)^2. \tag{6}$$

The major assumptions entering the derivation of OBD are: 1) Terms of third and higher orders in the deleted weights can be neglected. 2) The off-diagonal terms in the Hessian, $\partial^2 E_{\text{train}}/\partial u_l \partial u_{l'}$, can be neglected. Computationally, the second order (diagonal) terms, eq. (6), are reused from the training scheme. We refrain from operations involving the full Hessian, which scales poorly for large networks. The recipe allows for *ranking* the weights according to saliency.

GENERALIZATION

The generalization error is defined as the average squared error on an example from the example distribution function $P(\mathbf{x}, y)$. The examples are assumed to be generated by a *teacher* function of the same form as the model and with a set of *unknown* weights $\mathbf{u}^*$ and degraded by additive noise:

$$y(k) = F_{\mathbf{u}^*}(\mathbf{x}(k)) + \nu(k) \tag{7}$$

where the noise samples $\nu(k)$ are independent identically distributed variables of unknown variance σ^2. Further, we assume that the noise terms are independent of the corresponding inputs. The generalization error of a given network is by definition the average error on a random example. A more interesting quantity is the training set average of the generalization error, viz., the average over an ensemble of networks in which each network is provided with its individual training set. Using the diagonal approximation for the Hessian this error (also referred to as the test error) can be estimated as [6]:

$$\begin{aligned} \widehat{E}_{\text{test}} &= \left(1 + \frac{N_{\text{eff}}}{p}\right)\sigma^2 \\ &+2\sum_{ij}^{N_w} \lambda_{ij}\left(\frac{\alpha_w}{p}\cdot\frac{w_{ij}^*}{\lambda_{ij}+2\alpha_w/p}\right)^2 \\ &+2\sum_{j}^{N_W} \Lambda_j\left(\frac{\alpha_W}{p}\cdot\frac{W_j^*}{\Lambda_j+2\alpha_W/p}\right)^2 + R, \end{aligned} \tag{8}$$

with

$$N_{\text{eff}} = \sum_{ij}^{N_w}\left(\frac{\lambda_{ij}}{\lambda_{ij}+2\alpha_w/p}\right)^2 + \sum_{j}^{N_W}\left(\frac{\Lambda_j}{\Lambda_j+2\alpha_W/p}\right)^2, \tag{9}$$

where the λ's are the second derivatives already computed in eq. (6): $\lambda_{ij} \equiv \partial^2 E_{\text{train}}/\partial w_{ij}^2$, $\Lambda_j \equiv \partial^2 E_{\text{train}}/\partial W_j^2$. The rest term R contains higher order quantities and terms that do not affect the estimate of the regularization parameters, see [5, 6] for further discussion. The estimate is based on linearization of the networks as regards the fluctuations in the weights resulting from different training sets.

The generalization error estimates were also used for answering the question of how many weights it may be possible to delete in a pruning session in [11, 12]. We applied Akaike's FPE estimate [1] of the test error in terms of the training error which reads:

$$\widehat{E}_{\text{test}} = \frac{p+N}{p-N}E_{\text{train}}, \tag{10}$$

where p is the number of training samples, and N is the number of parameters in the model. The left hand side of eq. (10) is the average generalization error, averaged over all possible training sets of size p.

The relation expresses the fact that the training and test errors are biased estimates of the noise level because each parameter during training has "absorbed" noise from the training samples.

Since we have regularized the training procedure by weight-decay terms α_w, α_W, hence, suppressed the ability of the (otherwise) ill-determined parameters to model noise, we need to modify the standard FPE estimate by replacing the total number of parameters with the *effective* number of parameters, see [10, 11, 12][2]:

$$\widehat{E}_{\text{test}} = \frac{p + N_{\text{eff}}}{p - N_{\text{eff}}} E_{\text{train}}, \tag{11}$$

With the above tool we can obtain a generalization error estimate for each pruned network. By selecting the network with the lowest estimated generalization error we obtain a stopping criterion for pruning.

Note that the estimated average generalization eq. (9) error is a function of the regularization parameters, hence, it is possible to vary these and search for minimal test error. In MacKay's Evidence framework a similar strategy was adopted, however, with the purpose of maximizing the so-called Evidence. We find it more natural to optimize the quantity that is our basic objective, namely the test error. It is at present not clear what the relation between the Evidence and the generalization error is. Empirically, they have been found to be related [2, 8].

We use a simple gradient descent procedure for minimization of the generalization error:

$$\alpha(n+1) = \alpha(n) - \rho \left. \frac{\partial E_{\text{test}}}{\partial \alpha} \right|_{\alpha = \alpha(n)}, \tag{12}$$

where ρ is a gradient descent parameter, and n is the iteration index (one epoch). The Designer Net approach is based on the diagonal approximation to the Hessian. In terms of the diagonal elements the recursion above reads,

$$\alpha_w(n+1) = \alpha_w(n) + \frac{4\rho}{p^2} \sum_{ij}^{N_w} \frac{(\sigma^2 - \alpha_w(n)(w_{ij}^*)^2)\lambda_{ij}^2}{(\lambda_{ij} + 2\alpha_w(n)/p)^3}. \tag{13}$$

A similar expression applies for the hidden-to-output weight decay parameter α_W, in fact an arbitrary set of weight decay parameters can be defined and estimated using this recipe[3]. Expression (13) contains two unknown quantities: the teacher weights w_{ij}^* and the noise variance σ^2. The teacher weights are replaced by the *current estimated weights* of the network (see [2] for a

[2] In fact the notion of an effective number of parameters is quite delicate see [6].

[3] In the derivative of the test error we have kept the dependence $\lambda^2/(\lambda + 2\alpha/p)^3$ (rather than $1/\lambda$) providing a stabilizing effect similar to the Moore-Penrose pseudo inverse discussed in [6].

discussion), while the noise variance is estimated from the training error in the same approximation as in eq. (11):

$$\widehat{\sigma}^2 = \left(1 - \frac{N_{\text{eff}}}{p}\right)^{-1} E_{\text{train}}. \tag{14}$$

EXPERIMENTS

We illustrate the virtues of the adaptive regularization scheme on two time series forecasting problems. The first experiment explores the functional dependence of the derivative of the estimated test error cf. equation (13). The forecasting problem is the sunspot benchmark involving estimation of the yearly sunspot activity from the past twelve years activity (see, e.g., [11] for a detailed description of the benchmark). To simplify we consider a linear model for which the parameters are uniquely determined when using the least squares cost function. The sunspot benchmark involves three data sets: A training set and two test sets. In figure 1 we show the weight decay dependence of the two test errors and of the derivative of the *estimated* test error. Note that both test sets have shallow minima at values just below $\alpha = 0.1$, and that the derivative of the estimated test error passes through zero at a compatible value. Also note that the particular functional form of the derivative implies that the iterative scheme will converge to the zero point of the gradient, hence, provide near-optimal regularization with improved generalization errors. To further illustrate the role of adaptive regularization in the Designer Net framework we present tentative results on a standard problem of nonlinear dynamics, viz. the Mackey-Glass chaotic time series. This forecasting problem was previously studied in [12]. The Mackey-Glass attractor is a non-linear chaotic system described by the following equation:

$$\frac{dz(t)}{dt} = -bz(t) + a\frac{z(t-\tau)}{1 + z(t-\tau)^{10}} \tag{15}$$

where the constants are $a = 0.2$, $b = 0.1$ and $\tau = 17$. The series is resampled with sampling period 1 according to standard practice. We aim at identifying the underlying dynamic model, from this chaotic time series. The network configuration is $L = 16$, $n_H = 10$, with a total of 181 parameters, and we train to implement a six step ahead prediction. That is, $\mathbf{x}(k) = [z(k-6), z(k-12), \cdots, z(k-6L)]$ and $y(k) = z(k)$.

The errors are computed as:

$$E_{\text{set}} = \frac{1}{\sigma^2_{\text{total}} \cdot p_{\text{set}}} \sum_{k=1}^{p_{\text{set}}} \left[y(k) - F_{\mathbf{u}}(\mathbf{x}(k))\right]^2, \tag{16}$$

where p_{set} is the number of examples in the data (train or test) set in question, and σ^2_{total} is the total variance of $y(k)$ on the training and test set.

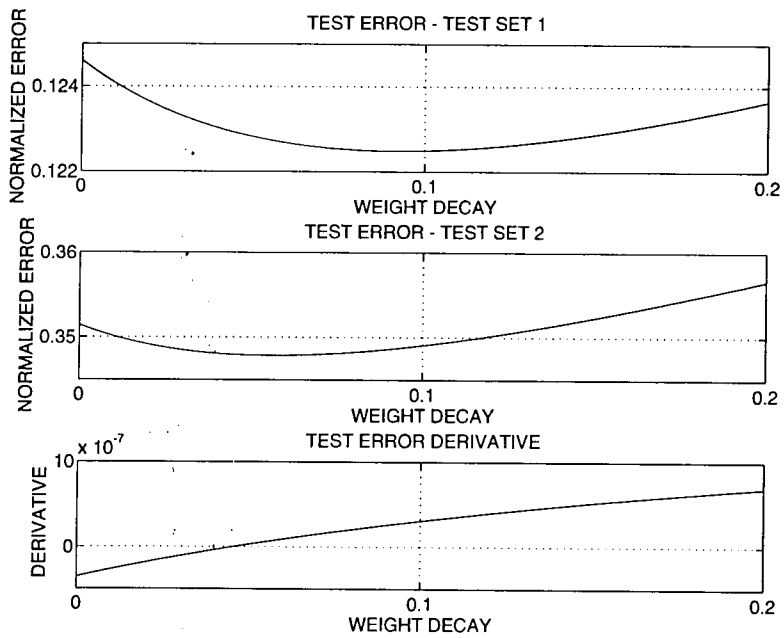


Figure 1: The role of weight decay regularization for a linear model on the sunspot benchmark series. The two upper figures show the errors (see [11] for a definition) on test sets both having shallow minima just below a weight decay of 0.1, while the bottom figure shows the derivative of the estimated test error as function of weight decay. Note that a gradient descent procedure will converge to the zero point.

There are several ways of implementing the adaptation scheme; here we initially set the weight decays to fixed values $\alpha_w = \alpha_W = 0.005$ for 100 epochs, then the network is trained with simultaneous adaptation of weight decay for 8000 epochs using eq. (13) with $\rho = 0.1$. After the initial training phase, further pruning and adaptation took place with pruning of 2% of the remaining weights per retraining round (400 epochs). In line with [12] it is seen that the stop criterion is able to select the optimal network. In figure 2 the normalized training errors, test errors cf. eq. (16), and the corresponding FPE error (after the initial training phase) are sketched for a training set size of 500 examples and the test set comprises 8500 examples. In [12] we

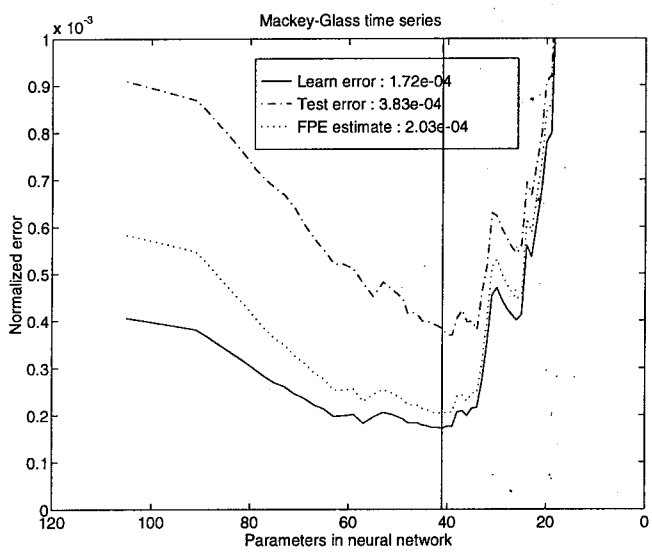


Figure 2: The evolution of training and test errors during pruning for the Mackey-Glass time series for a training set of size 500. The FPE estimate of the test error is based on eq. (11). The vertical line indicates the network for which the *estimated* test error is minimal.

compared the performances of pruned networks with those of fully connected nets, a linear model, and with a K-nearest-neighbor linear model. It was noted that the performance of the networks is similar to the nearest neighbor estimate. While the two weight decays previously were set manually we here adapt them according to equation (13). In figure 3 the development of the two weight decays is depicted as pruning proceeds. Note that the adaptive regularization scheme "chooses" relatively high regularization for the large network as should be expected. These networks have superfluous resources that could potentially harm generalization through overfitting. Eventually, at the end of the pruning session the test error estimates are rather biased (the network is underfitting) and the adaptive scheme does not provide reliable estimates. We have observed that the scheme in it present form has some

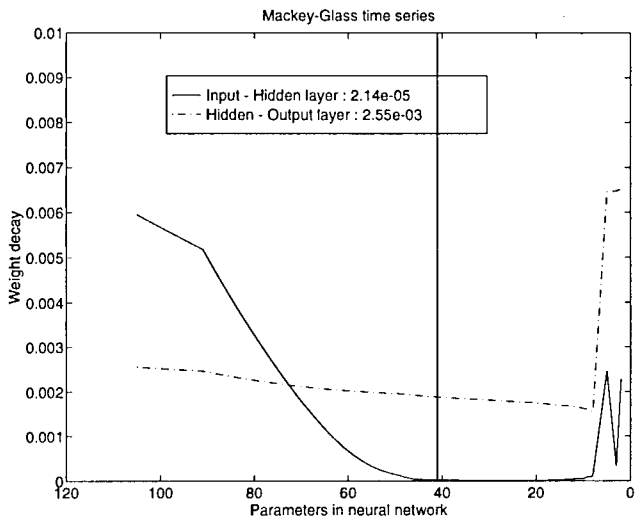


Figure 3: The evolution of weight decays during pruning for the Mackey-Glass time series. The vertical line indicates the network for which the *estimated* test error is minimal.

dependence on initialization of weights and weight decays; this is a topic for current research. The very low value of the input-to-hidden weight decay for the small networks is also in line with our earlier observations, namely that one can retrain the optimal architecture without weight decay and get slightly improved generalization [11, 12].

CONCLUSION

A scheme has been derived for adaptation of weight decay parameters. The scheme is based on asymptotic sampling theory. Two examples were given to illustrate the virtues of such adaptation. First, we showed that the functional form of the derivative of the estimated test error will provide convergence to near-optimal values for a linear model on the sunspot benchmark. Secondly, it was shown how the Designer Net framework can be applied with adaptive regularization, hence, relieving, manual tuning of these important parameters.

ACKNOWLEDGMENTS

This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT).

REFERENCES

[1] H. Akaike, "Fitting Autoregressive Models for Prediction," Ann. Inst. Stat. Mat., **21**, 243-247, (1969).

[2] L.K. Hansen and C.E. Rasmussen, "Pruning from Adaptive Regularization," Accepted for Neural Computation, CONNECT Electronics Institute, Technical University of Denmark, preprint (1993).

[3] L.K. Hansen, "Stochastic Linear Learning: Exact Test and Training Error Averages," Neural Networks **6**, 393-396, (1993).

[4] J. Hertz, A. Krogh and R.G. Palmer, Introduction to the Theory of Neural Computation, Addison Wesley, New York, (1991).

[5] J. Larsen, Design of Neural Network Filters, Ph. D. Thesis, Electronics Institute, Technical University of Denmark, March (1993).

[6] J. Larsen and L.K. Hansen, "Generalization Performance of Regularized Neural Network Models," In proceedings of the IEEE Workshop on Neural Networks for Signal Processing NNSP'94.

[7] Y. Le Cun, J.S. Denker, and S.A. Solla, "Optimal Brain Damage," In Advances in Neural Information Processing Systems 2, 598-605, Morgan Kaufman, (1990).

[8] D. MacKay, "Bayesian interpolation". Neural Computation **4** 448-472, (1992).

[9] D. MacKay, "A practical framework for backpropagation networks". Neural Computation **4** 415-447 (1992).

[10] J.E. Moody, "Note on Generalization, Regularization and Architecture Selection in Nonlinear Systems," In Neural Networks For Signal Processing Proceedings of the 1991 IEEE-SP Workshop (Eds. B.H. Juang, S.Y. Kung, and C. Kamm), IEEE Service Center, 1-10, (1991).

[11] C. Svarer, L.K. Hansen, and J. Larsen, "On Design and Evaluation of Tapped Delay Line Networks," In Proceedings of the 1993 IEEE International Conference on Neural Networks San Francisco, 46-51, (1993).

[12] C. Svarer, L.K. Hansen, and J. Larsen, and C.E. Rasmussen, "Designer Networks for Time Series Processing," Proceedings of the 1993 IEEE Workshop on Neural Networks for Signal Processing (NNSP'93) Baltimore (Eds. C.A. Kamm et al.), 78-87, (1993).

Faster and Better Training of Multi-Layer Perceptron for Forecasting Problems

R. R. Laddad, U. B. Desai and P. G. Poonacha
Department of Electrical Engineering,
Indian Institute of Technology, Bombay,
Powai, Bombay 400 076, INDIA
Fax : (+91-22) 578-3480 e-mail : ubdesai@ee.iitb.ernet.in

Abstract

New methods for training Multi-layer perceptron network for forecasting problems are presented. The first method exploits spectral characteristics of time series to get faster learning and improved prediction accuracy. A neural network scheme for real time implementation of this method is also presented. The second method suggests the use of two new weight initialization schemes which give very fast convergence besides giving better prediction. The foreign exchange time series is used to illustrate the efficacy of the proposed methods.

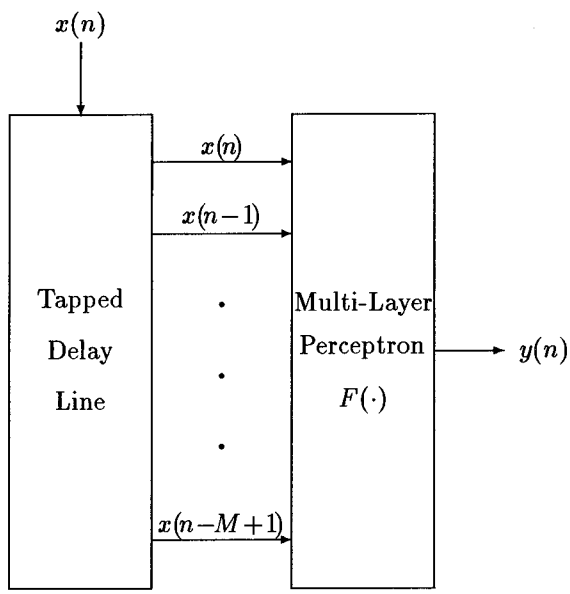
1 Introduction

In recent times Multi-Layer Perceptron (MLP) networks have been widely used in forecasting problems (see for example [2], [3], [5], [6], [9]). This approach has been vigorously pursued for time series where linear methods, like the Wiener filter, the Kalman filter, and their variants, did not give satisfactory results. The failure of the conventional methods becomes very evident when one is dealing with financial time series.

For forecasting problems, the MLP network is typically in a Time Delay Neural Network (TDNN) structure (Figure 1). From a digital signal processing perspective, the TDNN can be viewed as a nonlinear filter. Training is done using some past data; for example in case of foreign exchange rate time series, one could use previous day's data. The most popular training algorithm is the backpropagation algorithm [7], [8].

Several difficulties are encountered when one uses the raw data for training the TDNN. In a practical situation the data is very likely to be

0-7803-2026-3/94 $4.00 © 1994 IEEE



$$y(n) = F([x(n)\ x(n-1)\ \cdots\ x(n-M+1)]^T)$$

Figure 1: Time Delay Neural Network

noisy, and the MLP may learn the noise along with the signal leading to poor forecasting. The presence of noise also makes it difficult to arrive at an appropriate size of the network. Small network may not be capable of modeling the given system; on the other hand a large network may cause poor generalization. From our simulations we found this to be a difficult issue to handle. We also found that the use of raw data for training leads to slow convergence or no convergence.

It is known that if the MLP is trained with appropriately preprocessed data, it may give *better*[1] *and faster learning* [5]. A further improvement in learning speed and accuracy can be achieved by appropriately selecting the starting weights. In this paper we suggest methods for solving these two problems :

1. The preprocessing method based on the spectral decomposition of the data.

2. Initial weights selection based on the specific structure of the forecasting problem.

   (a) A fixed unity gain for the path leading from $x(n)$ to $y(n)$ in

[1] Better learning in the sense of low mean squared error

Figure 1.

(b) A Least Mean Square (LMS) filter for one path leading to the output $y(n)$.

The next two sections elaborate on both these approaches, which constitute the key contribution of this paper.

We would like to remark that in all simulations we have used the Random Optimization Algorithm (ROA) [1] for training the MLP. The reason for choosing it over backpropagation is faster learning; also we found it to give a *better minima*.

2 MLP Learning with Spectral Decomposition Technique

2.1 Proposed Scheme

The basic idea behind the proposed technique is to decompose the time series into *many simpler time series* and learn each one using a separate MLP. We propose to use the *spectral decomposition* of the data; this is achieved by passing the time series through a bank of bandpass filters each having different passband frequencies.

Figure 2 gives the schematic of the proposed scheme. Bandpass filters have cutoff frequencies such that the lower cutoff frequency of the next filter is same as the upper cutoff frequency of the previous filter. Each bandpass filter output is fed into a separate TDNN. Final prediction is a combination of the outputs of these TDNNs. The combining filter $f(\cdot)$ could be a simple summation, or a linear adaptive filter [4] or a single layer MLP (a nonlinear adaptive filter). Adaptive filter for $f(\cdot)$ may have an advantage, since it can compensate for difference in prediction accuracy of each TDNN.

2.2 Discussion

From our simulations we found the spectral decomposition technique to give much better prediction accuracy. Moreover, the method gave a significant speed advantage typically by a factor of 8. Also, the design for each TDNN becomes easier since the knowledge of the frequency contents of the respective input time series is known. In case of foreign exchange time series we found that for low frequency, a MLP with large number of input nodes and small number of hidden layer neurons is a good choice; while for medium frequency a small number of input nodes with large number of hidden layer neurons is a better choice.

In many practical situations high frequency variations are considered as noise, and in such cases one can ignore high frequencies. Removal of noise makes choosing the network size an easier problem.

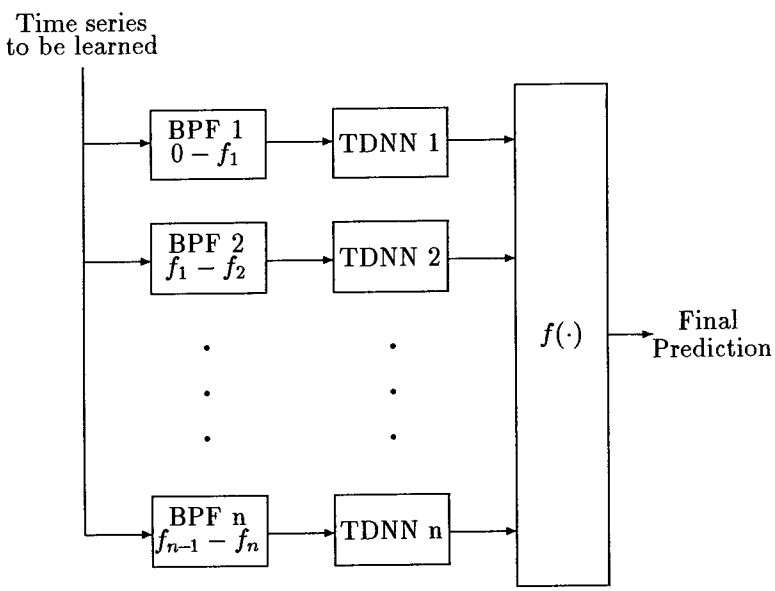


Figure 2: Schematic of Spectral Decomposition Technique

We have carried out simulations for forecasting the foreign exchange rate. Without spectral decomposition we were unable to train the TDNN to give any meaningful results; this was because in most of the cases the error obtained was unacceptable. Figure 4 gives simulation results for the spectral decomposition technique, using one and two filters. The combining filter $f(\cdot)$ used is a simple summation. In both the cases very high frequency components were ignored since they are considered unimportant in foreign exchange market.

2.3 Real Time Implementation

Any real time implementation of the band pass filter would involve a certain amount of delay. For example, a P order linear phase FIR (Finite Impulse Response filter) would have a delay of $(P+1)/2$. This delay can play *havoc* in real time forecasting. We propose a scheme wherein an approximate implementation of the FIR band pass filter is used to achieve zero delay. This scheme is depicted in Figure 3. A band pass filter, [BPF k] followed by [TDNN k] block of Figure 2 is replaced by the block $[G_{Bk}]$ followed by $[MLP_k]$ of Figure 3.

In Figure 3

$$G_{B_k} = [\ G^T_{Bk_1}\ G^T_{Bk_2}\ ]^T_{M\times(M+P)} \tag{1}$$

where G_{Bk_1} will be a $(P-1)\times(M+P)$ matrix, and G_{Bk_2} will be a

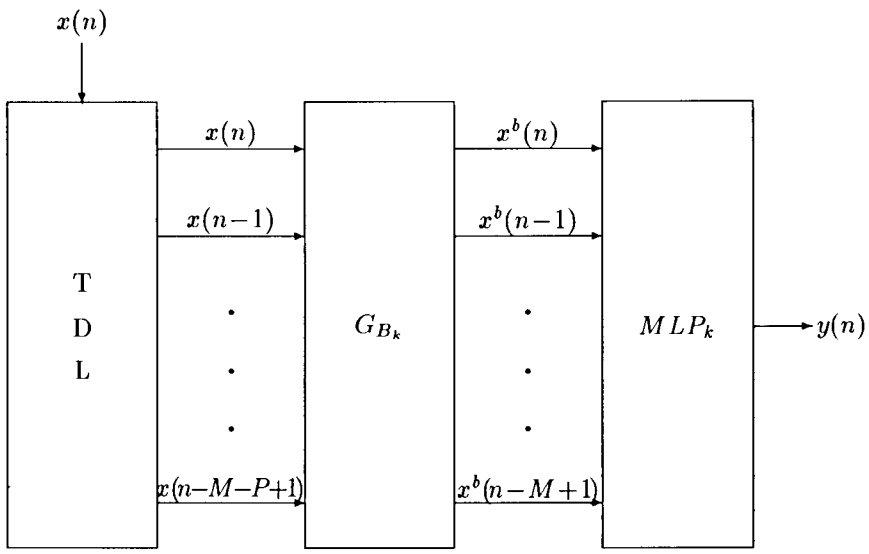


Figure 3: Schematic for Real Time Implementation of Spectral Decomposition Technique

$(M-P+1)\times(M+P)$ matrix defined as

$$G_{Bk_1} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ b^k_{2,1} & b^k_{2,2} & b^k_{2,1} & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & & & & & & & \\ b^k_{P-1,1} & \cdots & b^k_{P-1,P-1} & \cdots & b^k_{P-1,1} & 0 & \cdots & 0 \end{bmatrix} \quad (2)$$

$$G_{Bk_2} = \begin{bmatrix} b^k_1 & \cdots & b^k_P & \cdots & b^k_1 & 0 & \cdots & \cdots & 0 \\ 0 & b^k_1 & \cdots & b^k_P & \cdots & b^k_1 & 0 & \cdots & 0 \\ \vdots & & & & & & & & \\ 0 & \cdots & \cdots & 0 & b^k_1 & \cdots & b^k_P & \cdots & b^k_1 \end{bmatrix} \quad (3)$$

As seen from Figure 3, the input to the MLP_k will be

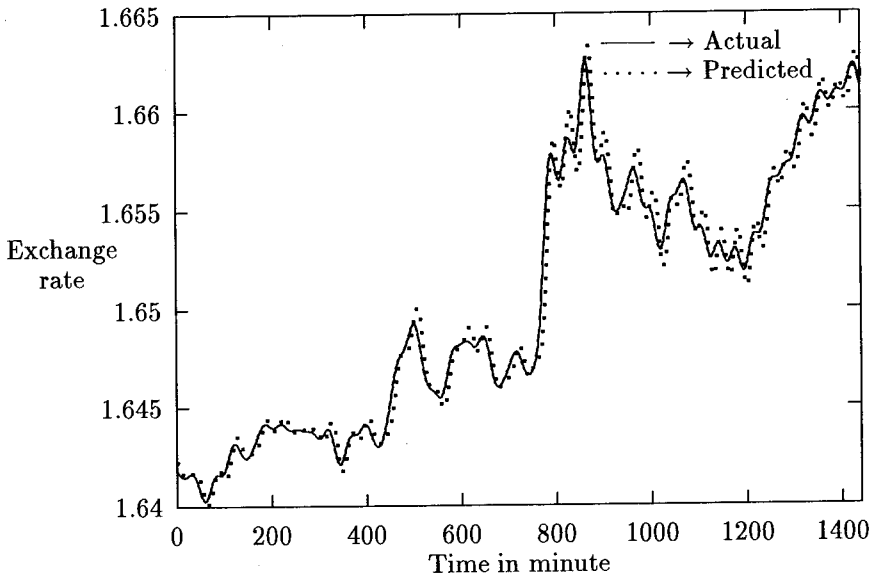
$$X^{b_k}_n = G_{B_k} X_n \quad (4)$$

where,

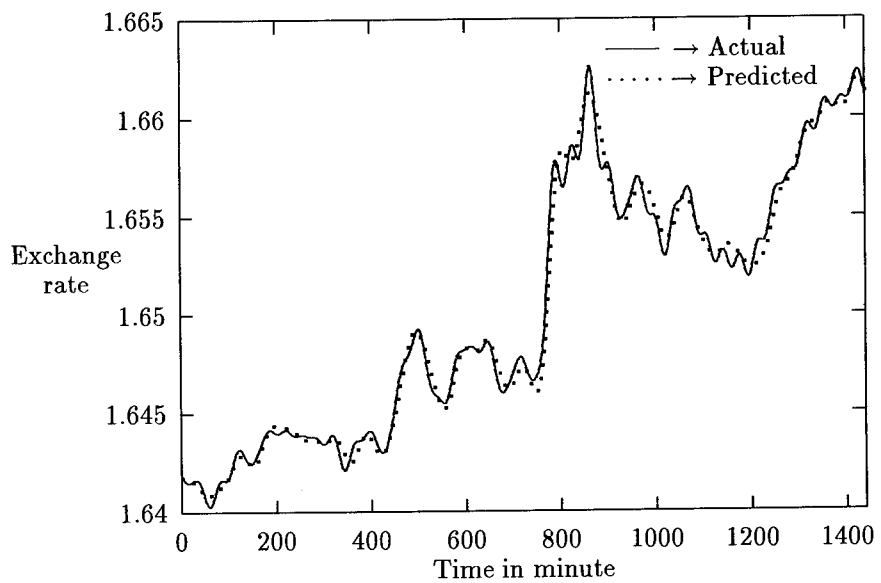
$$X_n = [x(n) \ \cdots \ x(n-M-P+1)]^T \quad (5)$$

$$X^{b_k}_n = [x^b(n) \ \cdots \ x^b(n-M+1)]^T \quad (6)$$

The specification for the band pass filter corresponds to the specification of [BPF k] in Figure 2. The elements $[b^k_{i,1}, \cdots b^k_{i,i}, \cdots b^k_{i,1}]$ are the FIR coefficient of a $(2i+1)$ order band pass filter. On the other hand $[b^k_1, \cdots b^k_P, \cdots b^k_1]$ are the FIR coefficient of $(2P+1)$ order band pass filter.



(a)



(b)

Figure 4: Results with Spectral Decomposition Technique (a) with one filter and (b) with two filters

3 Very Fast Learning in MLP for Forecasting Problems

3.1 Unity Transmission gain for One Path

Our simulations with financial time series exhibited an interesting characteristic of MLP learning. When actual and desired output is observed after each iteration, we found that learning passes through a phase where the predicted value is a delayed version of the most recent input to the MLP, i.e.

$$y(n) \approx x(n) \tag{7}$$

The MLP takes a long time to arrive at this situation. Thus the proposed scheme is as follows: Select the initial weights for the MLP, so that (7) is approximately satisfied. One way to achieve this is to select the weights such that the transmission gain between the last input (say $x(n)$ in Figure 1) and the output node is near unity. The remaining weights can be set to zero.

Our simulations overwhelmingly justify the proposed scheme for forecasting using TDNN. We carried out extensive simulations using different data sets and networks. In all cases we found 50 to 100 times faster convergence as compared to the conventional approach where the initial weights are set to zero, as ROA does not require weights to be set to a small random value as is the case with Backpropagation. Figure 6 gives simulation result which uses spectral decomposition technique of earlier section along with the above mentioned scheme of selecting initial weights. The same data and the network is used for the simulation with the conventional approach (Figure 5) and for the simulation with the new approach (Figure 6). It can be observed that the error obtained by the conventional technique after 9500 iterations is achieved by the new scheme in just 200 iterations. With the conventional approach the error practically stops decreasing after 9500 iterations and the error is more than double as compared to that obtained by the new scheme. This suggests that the new scheme is not only faster but also provides more accurate prediction.

3.2 LMS in Conjunction with MLP

The method of initializing weights as suggested above works well when used in conjunction with the spectral decomposition technique. In essence this implies the availability of noise free data for the weight initialization scheme to work. Thus a major constraint is the real time implementation of zero delay band pass filters. To avoid this constraint we propose a method where the MLP acts on the raw (noisy) time series but has one node implementing a linear adaptive prediction filter. In this paper for the purpose of illustration we have used the LMS filter. Here one

neuron in the first hidden layer along with the incoming inputs forms the FIR filter, which is trained separately using LMS algorithm. This way, for the second hidden layer we have some data which has the noise removed by a linear adaptive filter. Note the transmission gain between the output of this node to the output layer node is set to one. Once the LMS filter is trained, the MLP gives an initial output which is based on linear prediction only. After this we train the network in the regular fashion using ROA.

The simulation results for this method are depicted in Figure 7, once again for the same financial time series. In this figure the first 200 iterations corresponds to the convergence achieved using the LMS algorithm. It is to be noted that with raw data, the MLP training algorithm based on ROA or backpropagation did not give any meaningful results; thus they are not reported.

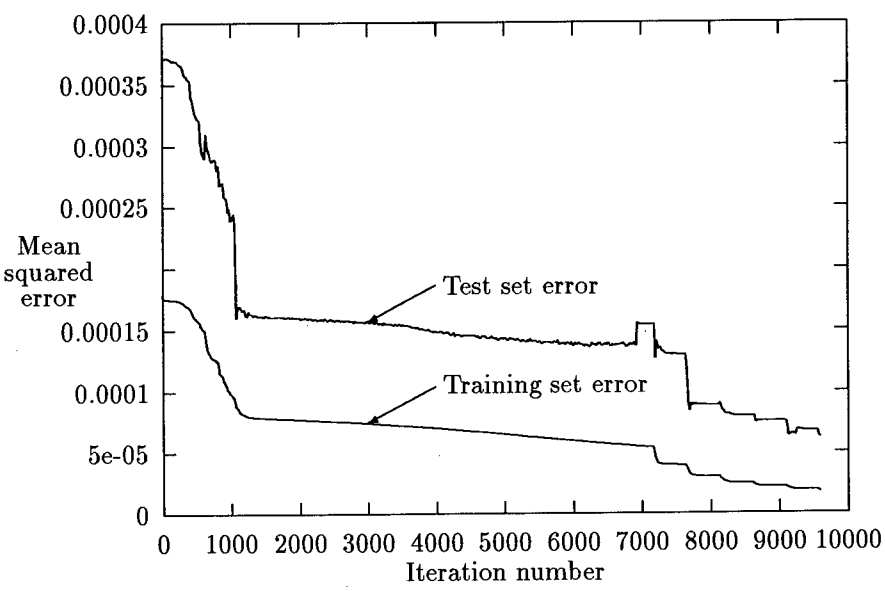


Figure 5: Error curve for conventional training

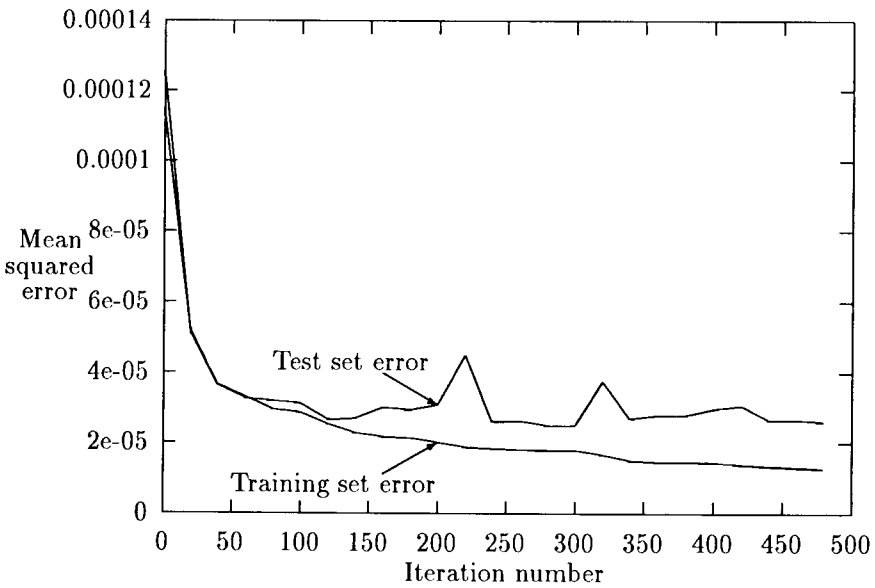


Figure 6: Error curve for training using method of Section 3.1

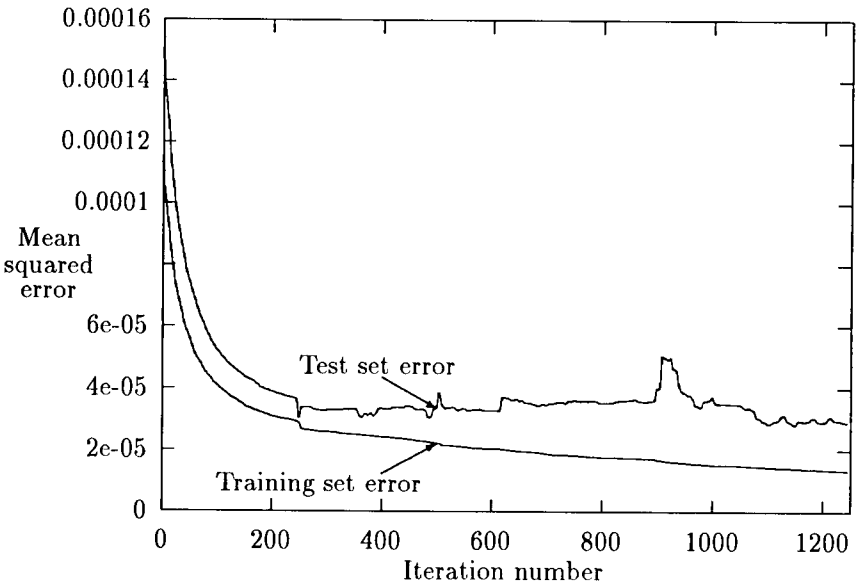


Figure 7: Error curve for LMS based training of Section 3.2

4 Conclusion

We found the combination of spectral decomposition method and the above initial weight selection approach to provide the best results both in terms of speed and accuracy. Only sample simulations are provided due to space limitation.

Even though the proposed methods are specific to the forecasting problem, nevertheless they should be applicable for the general MLP training problem.

References

[1] N.Baba, "A New Approach for finding the Global Minimum of Error Function", *Neural Networks*, Vol.2, 1989, pp. 364-373.

[2] M.J.Bramson, R.G.Hoptoff, "Forecasting the Economic Cycle: Neural Network Based Approach", *Workshop on Neural Network for Statistical and Economic Data*, Dublin, 1990, pp. 121-152.

[3] D.D. Hawley, J.D. Johnson, D. Raina, "Artificial neural systems: a new tool for financial decision making", *Financial Analysts Journals*, Nov-Dec 1990, pp. 63-72.

[4] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[5] J. Hertz, A. Krogh, R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison - Wesley, Redwood City, CA, 1991.

[6] D.R. Hush, B.G. Horne, "Progress in supervised neural networks, what's new since Lippman?", *IEEE Signal Processing Magazine*, Jan 1993, pp. 8-39.

[7] R.P. Lippman, "An introduction to computing with neural nets", *IEEE ASSP Magazine*, April 1987, pp. 4-22.

[8] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning internal representation by error propagation", In D.E. Rumelhart, J.L. Mc.Clelland, editors, *Parallel Distributed Processing : Explorations in the Microstructure of Cognition*, MIT press, Cambridge, MA, 1986, pp. 318-362.

[9] A.S. Weigend, B.A. Huberman, D.E. Rumelhart, "Predicting the future : a connectionist approach", *Int. Journal of Neural Systems*, 1(3), 1990, pp. 193-209.

An Interval Computation Approach To Backpropagation

C. E. Pedreira and E. Parente
Electrical Engineering Department
Catholic University of Rio de Janeiro PUC-Rio
C.P. 38063, Rio de Janeiro, RJ 22452-970 BRAZIL
E-mail:pedreira@ele.puc-rio.br

Abstract - A new learning algorithm is introduced to allow interval valued inputs. This procedure is based on gradient descendent and error backprogation. It has many advantages over the classical Backpropagation including the possibility of dealing with missing values attributes. We establish a framework especially useful for applications with incertitude in the input. We propose a cost function reflecting a trade-off problem between the goal of including the target value into the interval valued output, and minimizing this interval size. Simulated numerical results are presented.

INTRODUCTION

Backpropagation has been widely used in a variety of applications. No other Neural Network's paradigm has achieved such success from the practical point of view. A major restriction of this classical technique arises from the obligation of using real valued inputs. In many relevant applications one needs to train the Network by associating the output target to an interval valued input. This situation is typical in medical diagnoses problems for instance. In this type of applications, one is fact dealing with classification problems where the input patterns have a good dose of uncertainty. Another Backpropagation limitation appears in the cases when part of input information is not available.

The management of Neural Networks with missing values attributes as inputs, is a very important open problem. Again, this is an everyday occurrence in medical among other applications. Suppose that a Neural Network has been trained to emulate an automatic diagnosis system by using a data basis composed by vectors in $\Re^n$. Each of these vectors contains the information of n medical exams the patient is supposed to undertake. Suppose that after a successful training section, one or more of the n exams could not be applied to a given patient. The question then is: How can the system be used in this case? To retrain the Network for $n<m$ inputs is not, in general, a feasible solution. The Interval Neural Networks (INN), introduced in this paper, provides an adequate solution for these problems.

0-7803-2026-3/94 $4.00 © 1994 IEEE

The use of interval arithmetic's in Neural Networks was first suggested by Ishibuchi et al [1]. They proposed a classification method for two-group discriminant problems where the Network inputs are given as intervals. Their method is limited to classification problems, and it can not be directly extended to more than two groups. In this paper we propose a much more general approach to Interval Neural Networks.

Our contribution concerns the introduction of a new learning algorithm that is able to deal with interval inputs. We believe that our algorithm will find application in classification, forecasting, and pattern recognition, among other areas.

PRELIMINARIES

Let us consider a one hidden layer Neural Network with a single unit in the output layer. As it will be notice in the ensuing developments, our results can, after some algebraic manipulation, be easily extended to a more general architecture. Assume that each input unit receives an interval valued income.

Notation

Let us consider the presentation of pattern p. Let ε_{ip}^{L} and ε_{ip}^{U} , $i = 1, 2, \ldots\ldots, n$ be the lower and upper limits of the n input units. o_{jp}^{L} and o_{jp}^{U} , $j = 1, 2, \ldots, m$ denote the lower and upper limits of the outputs of the m hidden units, and O_p^L, O_p^U are the lower and upper limits of the network output. The n x m input weights are represented by w_{ji} , and the v_j are the m weights linking the hidden layer to the network output. The network target, for each pattern p, is denote by t_p. Let X^{LU} be the interval for which the lower and upper limits are X^L and X^U respectively.

Basic Definitions

We define the linear combination of the input intervals, when pattern p is presented, as:

$$net_{jp}^{LU} \equiv \sum_{i=1}^{n} w_{ji}\varepsilon_{ip}^{LU} + \theta_j \ , \text{ for each } j = 1, 2, \ldots., m \ , \tag{1}$$

while the linear combination of the hidden units outputs is given by:

$$NET_p^{LU} \equiv \sum_{j=1}^{m} v_j o_{jp}^{LU} + \theta \tag{2}$$

where θ_j and θ are the bias terms.

Our problem is to modify the Network weights such that the cost function $E \equiv \sum_p e_p$ is minimised for all patterns in the training set. The cost due to pattern p is defined as follows:

$$e_p \equiv \tfrac{1}{2}\alpha[t_p - (O_p^U + O_p^L)/2]^2 + \beta(O_p^U - O_p^L) \tag{3}$$

This cost function is intended to solve a trade off problem. Its first term pushes the centre of the output interval towards the pattern target, while its second term

minimises the interval size. This trade off problem is balanced by parameters α and β. Our main goal is to define a gradient descendent algorithm that modifies the weights w_{ji} and v_j for i=1,2,...n , j=1,2,...m such that the cost (3) is minimised.

Preliminary Results

Let $I(\Re)$ be the set of all closed real intervals. Note that real numbers $x \in \Re$ may be considered special elements $[x,x]$ of $I(\Re)$. Let $X, Y \in I(\Re)$. The following binary operations can be calculated as [2]:

$$X+Y=[X^L+Y^L, X^U+Y^U] \tag{4}$$

$$X-Y=[X^L-Y^U, X^U-Y^L] \tag{5}$$

$$X\cdot Y=[\min(X^LY^L, X^LY^U, X^UY^L, X^UY^U), \max(X^LY^L, X^LY^U, X^UY^L, X^UY^U)] \tag{6}$$

$$X+Y=[X^LX^U]\cdot[1/Y^U, 1/Y^L] \tag{7}$$

$$\begin{aligned} e^X &= [e^{X^L}, e^{X^U}] \text{ if } X^U \geq 0 \\ &= [e^{X^U}, e^{X^L}] \text{ otherwise} \end{aligned} \tag{8}$$

Here it is assumed that $0 \notin Y$ in case of division. Note that $I(\Re)$ is closed under operations (4) - (8). Furthermore, it can be proved [2] that one has: (i) commutativity and associativity for addition and multiplication; (ii) [0,0] and [1,1] are the unique neutral elements concerning addition and multiplication respectively; and (iii) $I(\Re)$ has no zero divisors.

Let $Y \in I(\Re)$ be a point interval, i.e. $Y=[y,y]$ where $y \in \Re$, then from (6) one gets

$$X\cdot Y=[\min(X^Ly, X^Uy), \max(X^Ly, X^Uy)] \quad \forall X \in I(\Re), \text{ and so}$$

$$\begin{aligned} y\cdot X &= [yX^L, yX^U] \text{ for } m \geq 0 \quad \forall X \in I(\Re) \\ y\cdot X &= [yX^U, yX^L] \text{ for } m < 0 \quad \forall X \in I(\Re) \end{aligned} \tag{9}$$

MAIN RESULTS

We are now able to establish a proceeding to calculate the weights changes in order to minimise the contribution of pattern p, e_p ,to the cost function E. Let us consider the following Delta rule:

$$\begin{aligned} \Delta_p v_j(k+1) &= \eta(-\partial e_p/\partial v_j) + \mu\Delta_p v_j(k) \\ \Delta_p w_{ji}(k+1) &= \eta(-\partial e_p/\partial w_{ji}) + \mu\Delta_p w_{ji}(k) \end{aligned} \tag{10}$$

where η and μ represent the learning and momentum parameters respectively. The remaining hardship is related to the e_p partial derivative's calculation. From (1),(2) and (9) we get:

$$net_{jp}^{LU} = \sum_{i=1}^{n} \left[w_{ji}\varepsilon_{ip}^{A}, w_{ji}\varepsilon_{ip}^{B} \right] + \theta_j \qquad \forall j = 1, 2, ..., m \ , \quad \text{and}$$

$$NET_{p}^{LU} = \sum_{j=1}^{m} \left[v_j o_{jp}^{C}, v_j o_{jp}^{D} \right] + \theta$$

where

$A = L$, $B = U$ if $w_{ji} \geq 0$, and $A = U$, $B = L$ otherwise,

$C = L$, $D = U$ if $v_j \geq 0$, and $C = U$, $D = L$ otherwise.

Now, concerning the hidden layer units we have that:

$$o_{jp}^{LU} = f(net_{jp}^{LU})$$

and for the output unit:

$$O_{p}^{LU} = f(NET_{p}^{LU})$$

where the activation function $f: I(\Re) \rightarrow I(\Re)$ is a generalised logistic function i.e:

$$f(X) = 1/(1 + \exp(-X)) \quad \forall X \in I(\Re).$$

Note that this generalised logistic function can be defined because of (4),(7) and (8).

Let us define the auxiliary parameters Φ, Ψ, K as follows:

$\Phi = L$ if $w_{ji} \geq 0$, $\Phi = U$ otherwise

$\Psi = U$ if $w_{ji} \geq 0$, $\Psi = L$ otherwise

$K = 1$ if $v_j \geq 0$, $K = 0$ otherwise

By applying the basic interval computation operations (4) - (7), and the chain rule we get:

(i) For "input to hidden" weights:

$$\frac{\partial e_p}{\partial w_{ji}} = \frac{\partial e_p}{\partial O_p^L}\frac{\partial O_p^L}{\partial NET_p^L}\left(\frac{\partial NET_p^L}{\partial o_{jp}^L}\frac{\partial o_{jp}^L}{\partial net_{jp}^L}\frac{\partial net_{pj}^L}{\partial w_{ji}} + \frac{\partial NET_p^L}{\partial o_{jp}^U}\frac{\partial o_{pj}^U}{\partial net_{jp}^U}\frac{\partial net_{jp}^U}{\partial w_{ji}}\right) +$$

$$\frac{\partial e_p}{\partial O_p^U}\frac{\partial O_p^U}{\partial NET_p^U}\left(\frac{\partial NET_p^U}{\partial o_{jp}^L}\frac{\partial o_{jp}^L}{\partial net_{jp}^L}\frac{\partial net_{jp}^L}{\partial w_{ji}} + \frac{\partial NET_p^U}{\partial o_{jp}^U}\frac{\partial o_{jp}^U}{\partial net_{jp}^U}\frac{\partial net_{jp}^U}{\partial w_{ji}}\right), \text{ and so}$$

$$\frac{\partial e_p}{\partial w_{ji}} = \left(\frac{\alpha}{\sqrt{2}}\right)(2t_p - O_p^U - O_p^L))(\xi + \varpi) + \beta(\varpi - \xi) \tag{11}$$

where

$$\xi \equiv K(1 - O_p^L)O_p^L v_j o_{jp}^L (1 - o_{jp}^L)\varepsilon_{ip}^{\Phi} + (1 - K)(1 - O_p^L)O_p^L v_j o_{jp}^U (1 - o_{jp}^U)\varepsilon_{ip}^{\Psi}$$

$$\varpi \equiv (1 - K)(1 - O_p^U)O_p^U v_j o_{jp}^L (1 - o_{jp}^L)\varepsilon_{ip}^{\Phi} + K(1 - O_p^U)O_p^U v_j o_{jp}^U (1 - o_{jp}^U)\varepsilon_{ip}^{\Psi})$$

(ii) For "hidden to output" weights:

$$\frac{\partial e_p}{\partial v_j} = \frac{\partial e_p}{\partial O_p^U}\frac{\partial O_p^U}{\partial NET_p^U}\frac{\partial NET_p^U}{\partial v_j} + \frac{\partial e_p}{\partial O_p^L}\frac{\partial O_p^L}{\partial NET_p^L}\frac{\partial NET_p^L}{\partial v_j}$$

$$= (\frac{\alpha}{\sqrt{2}}(2t_p - O_p^U - O_p^L)(\pi + \sigma) + \beta(\sigma - \pi) \qquad (12)$$

where

$$\pi \equiv o_{jp}^{\Psi} O_p^U (1 - O_p^U) \quad \text{and} \quad \sigma \equiv o_{jp}^{\Phi} O_p^L (1 - O_p^L)$$

By applying (11) and (12) in (10) one can appropriately train the Neural Network to minimize the proposed cost function E.

NUMERICAL RESULTS

To illustrate our method we simulated a hypothetical medical diagnosis decision support system. Let us suppose that four exams are applied to each patient, being the first one more relevant than the others. The Network output indicates the patient diagnostic. We assume that "0" indicates negative and "1" indicates positive, i.e a patient with an output near 1 considered to be with good health. An input equal to "1" reflects a positive exam result. For all the following examples the Neural Network has two units in the hidden layer. In our first simulation we set $\beta = 0$. The following training set was presented:

Presentation	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Input 1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Input 2	1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	0
Input 3	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	1
Input 4	1	0	0	0	1	1	0	1	0	0	0	1	1	1	1	0
Target	1	1	1	0	1	1	1	1	0	0	0	1	0	0	0	0

After a training section of approximately 4000 iterations (see figure 1) we have got the following outcome corresponding to the 16 training data:

Presentation	1	2	3	4	5	6	7	8
Outcome	0,9998	0,9996	0,9876	0,0190	0,9996	0,9996	0,9876	0,0004
Presentation	9	10	11	12	13	14	15	16
Outcome	0,0007	0,0004	0,0146	0,9833	0,0146	0,0007	0,0146	0,0007

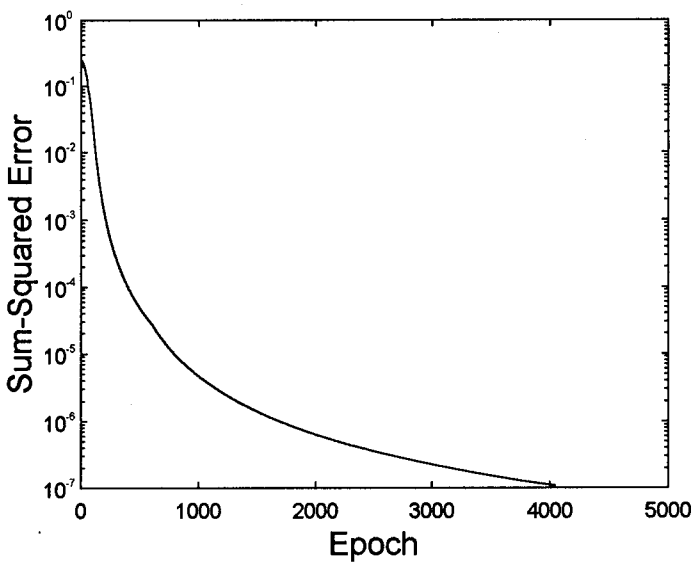


Figure 1

Note that real outcomes were obtained, i.e. lower interval bound is equal to the upper bound. This result was expected since real inputs were presented. Next we recalled the network simulating that the third exam was not undertaken, i.e. input = (1, 1 , [0,1] , 0), and we got as outcome [0,9876 ; 0,9996]. This result is also coherent since the absent exam is not the relevant one. In contrast, if we omit the first exam, i.e. input = ([0,1] , 1 , 0 , 0), we will get outcome = [0,0007 ; 0,9876], indicating an undefined response as expected. In the next experiment we set $\beta = 0,01$, and used the following training data:

Present.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Input 1	A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B
Input 2	1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	0
Input 3	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	1
Input 4	1	0	0	0	1	1	0	1	0	0	0	1	1	1	1	0
Target	1	1	1	0	1	1	1	1	0	0	0	1	0	0	0	0

where A = [0.8 , 1] and B = [0 , 0.2]. We trained the network to reach a sum squared error of approximately 10^{-7}. This network was recalled with input=([0 ; 0,4] , 1 , 1 , 0) , to get as output [0,9988 ; 1,0000].

FINAL REMARKS

In this paper we introduced a new approach to the classical Backpropagation algorithm. This method allows interval valued inputs, producing an interval valued output. The main advantages of our approach are the possibility of recalling the network even when one has missing or uncertain values attributes as input. We presented preliminary simulated numerical results obtain quite fast convergence and coherent results. Further research is now been conducted with the goal of generalising this approach to interval output targets.

REFERENCES:

[1] H.Ishibuchi, A.Miyazaki, K.Kwon and H.Tanaka , "Learning from Incomplete Training Data with Missing Values ansd Medical Application" , Proc. of Int. Joint Conf. on Neural Networks, pp1871, Nagoya, Japan, Oct 1993.

[2] G.Alefeld and J.Herzberger "Introduction to Interval Computation" , Academic Press, N.Y.,1983.

Robust Estimation for Radial Basis Functions

Adrian G. Borş I. Pitas

Department of Electrical and Computer Engineering
University of Thessaloniki
Thessaloniki 540 06, Greece

Abstract

This paper presents a new learning algorithm for radial basis functions (RBF) neural network, based on robust statistics. The extention of the learning vector quantizer for second order statistics is one of the classical approaches in estimating the parameters of a RBF model. The paper provides a comparative study for these two algorithms regarding their application in probability density function estimation. The theoretical bias in estimating one-dimensional Gaussian functions are derived. The efficiency of the algorithm is shown in modelling two-dimensional functions.

1 Introduction

Radial basis function (RBF) neural networks have been used in different applications in order to model unknown functions, providing the network with a training set [4]-[8]. RBFs have suitable properties to be used for function approximation [5], by decomposing a general function in a sum of kernels [2]. All the functions in this structure have similar parameters and can be embedded in a neural network.

The first approach considered in this paper is the second order statistics extension [7] for Learning Vector Quantization (LVQ) algorithm [3]. However, from statistical studies [2] this method is expected to give a large bias in the cases when data are long tailed distributed or contain outliers [1, 9]. In order to overcome these situations, we use an algorithm based on median type learning and called Median RBF (MRBF). Robust estimators are known to find the parameters best fitting to the bulk of the data and to identify outliers [2]. In the MRBF learning algorithm, we use the marginal median estimator in order to find the centers of the Gaussians and median of the absolute deviation for the covariance matrix parameters.

0-7803-2026-3/94 $4.00 © 1994 IEEE

The RBF network has a feed-forward topology and can be used in unsupervised as well as in supervised learning. The network can be fed with real N-dimensional vectors denoted by X and the hidden units implement a Gaussian function:

$$\phi_j(X) = \exp\left[-(\mu_j - X)' \Sigma_j^{-1} (\mu_j - X)\right],\ j = 1, \ldots, L \tag{1}$$

L is the number of hidden units, μ is the mean vector and Σ is the covariance matrix. These weights are associated with input to hidden layer connections and geometrically they represent the centers and the shape parameters for the basis functions. Each hidden unit has associated an activation region, similar with the Voronoi partition from vector quantization. In order to assign a new sample to an activation region we have assumed two different metrics: Euclidean and Mahalanobis.

The output layer implements a weighted sum of hidden unit outputs:

$$\psi_k(X) = \sum_{j=1}^{L} \lambda(k, j)\ \phi_j(X),\ k = 1, \ldots, M \tag{2}$$

where M is the number of outputs.

The outputs are binary coded and a sigmoidal function is used in order to limit the output:

$$Y^k(X) = \frac{1}{1 + \exp[-\psi_k(X)]},\ k = 1, \ldots, M \tag{3}$$

where Y^k is the kth output of the network.

2 Learning Algorithms

The weights in a RBF network can be found on-line by using a combined unsupervised-supervised technique [4]. The unsupervised part is derived from the LVQ algorithm and is similar to the adaptive k-means clustering.

In the first stage, the algorithm computes the distances from the given pattern to all the existing kernel centers. If we use Euclidean distance:

$$\text{If } \| X_i - \hat{\mu}_j \|^2 = \min_{k=1}^{L} \| X_i - \hat{\mu}_k \|^2 \text{ then } X_i \in C_j \tag{4}$$

where C_j is the kernel associated with the given pattern. Only the center of the winner class will be updated, according to the LVQ algorithm [3]:

$$\hat{\mu}_j = \hat{\mu}_j + \frac{1}{n_j}(X_i - \hat{\mu}_j) \tag{5}$$

for $j = 1, \ldots, L$, where n_j is the number of samples assigned to the cluster j. Taking the learning rate equal with the inverse of the number of samples associated with that unit we obtain a minimal output variance [10].

A similar method with (5) can be used to calculate the covariance matrix elements for each Gaussian neuron [7]:

$$\hat{\sigma}_{j,kl} = \frac{n_j - 2}{n_j - 1}\hat{\sigma}_{j,kl} + \frac{(X_i(k) - \hat{\mu}_j(l))(X_i(l) - \hat{\mu}_j(k))}{n_j - 1} \tag{6}$$

for $k, l = 1, \ldots, N$ $j = 1, \ldots, L$. The estimators in (5,6) are consistent with the classical statistical estimators for the first and the second order statistics.

The Mahalanobis square distance takes into consideration the covariance matrix for each hidden unit and can be used instead of (4):

$$\text{If } (\hat{\mu}_j - X_i)' \hat{\Sigma}_j^{-1}(\hat{\mu}_j - X_i) = \min_{k=1}^{L}(\hat{\mu}_k - X_i)' \hat{\Sigma}_k^{-1}(\hat{\mu}_k - X_i) \text{ then } X_i \in C_j \tag{7}$$

In the training stage it is desirable to avoid using patterns which may cause bias in the parameter estimation. The LVQ algorithm together with its extention in RBF network do not have robustness against the outliers or against the erroneous choices for the parameters. Robust estimators are known to provide accurate estimates when data are contaminated with outliers or have long-tailed distributions [1, 2]. The marginal median LVQ algorithm [9] can be used in order to evaluate the reference vectors for each partition region. In order to avoid increasing complexity, the samples assigned to a neuron pass through a running window W. If the data statistics change in time, then W is small. If a better evaluation of the median is desired then W is large. The learning rule is given by:

$$\hat{\mu}_j = \begin{cases} \text{med}\,\{X_0, X_1, \ldots, X_i\} & \text{if } i < W \\ \text{med}\,\{X_{i-W}, X_{i-W+1}, \ldots, X_i\} & \text{if } i \geq W \end{cases} \tag{8}$$

For the robust estimation of the scale parameter we use the median of the absolute deviation (MAD):

$$\hat{\sigma}_{j,hh} = \frac{\text{med}\,\{|X_1 - \hat{\mu}_j|, \ldots, |X_W - \hat{\mu}_j|\}}{0.6745} \tag{9}$$

where 0.6745 is a scaling factor in order to make the estimator consistent for the normal distribution [1, 2]. The cross-correlation components of the covariance matrix can be derived from the MAD calculated for $X_i(h) + X_i(l)$ and $X_i(h) - X_i(l)$ [2].

In both algorithms, for supervised learning, a second layer it is used in order to group the clusters found in the unsupervised stage. The output weights are updated as:

$$\lambda(k, j) = \lambda(k, j) + \eta\,(Y^k(X) - F^k(X))Y^k(X)(1 - Y^k(X))\phi_j(X) \tag{10}$$

for $k = 1, \ldots, M$ $j = 1, \ldots, L$ and $\eta \in (0, 1)$ is the learning rate. $F^k(X)$ is the desired output for the pattern X and it is binary coded. The formula (10) corresponds to the backpropagation for RBF network with respect to the square error cost function [8].

3 The Performance Analysis

We consider the case when we have a mixture of one-dimensional normal functions $N(\mu_j, \sigma_j)$:

$$f(X) = \sum_{j=1}^{L} \frac{\varepsilon_j}{\sqrt{2\pi}\sigma_j} \exp\left[-\frac{(X-\mu_j)^2}{2\sigma_j^2}\right] \tag{11}$$

$$\sum_{j=1}^{L} \varepsilon_j = 1 \tag{12}$$

where ε_j is the a priori probability for the function j. In the case of a mixture of multivariate normal distributions, the estimation can be done on marginal data. If we consider more complex distribution functions, they can be decomposed in sums of mixed Gaussians and reduced to the model (11).

We estimate the center for the jth Gaussian:

$$E[\hat{\mu}_j] = E[X|X \in [\hat{T}_j, \hat{T}_{j+1})] = \frac{\int_{\hat{T}_j}^{\hat{T}_{j+1}} X f(X) dX}{\int_{\hat{T}_j}^{\hat{T}_{j+1}} f(X) dX} \tag{13}$$

where $\hat{T}_j$ and $\hat{T}_{j+1}$ are the estimates of the separating boundaries for the jth Gaussian kernel and $f(X)$ is given by (11). In order to evaluate the parameters for one Gaussian from a mixture of normal functions we should also consider parts from neighboring functions which are inside the boundaries $\hat{T}_j$ and $\hat{T}_{j+1}$. Replacing (11) in (13) we derive the stationary value of the mean estimate, valid for (5).

The median is located where the *pdf* of the given data is split in two equal areas [1]. From this condition the stationary value of the center estimate for the jth Gaussian distribution can be obtained by using the median operation:

$$\sum_{i=1}^{L} \varepsilon_i \operatorname{erf}\left(\frac{E[\hat{\mu}_j^{med}] - \mu_i}{\sigma_i}\right) = \sum_{i=1}^{L} \frac{\varepsilon_i}{2}\left[\operatorname{erf}\left(\frac{T_{j+1}-\mu_i}{\sigma_i}\right) + \operatorname{erf}\left(\frac{T_j - \mu_i}{\sigma_i}\right)\right] \tag{14}$$

where we consider the definition for the erf function:

$$\operatorname{erf}(X) = \frac{1}{\sqrt{2\pi}} \int_0^X exp\left(-\frac{t^2}{2}\right) dt \tag{15}$$

The stationary value for the estimate of the variance using the classical estimator (6) is given by:

$$E[\hat{\sigma}_j^2] = E[(X - \hat{\mu}_j^{mean})^2 | x \in [\hat{T}_j, \hat{T}_{j+1})] = \frac{\int_{T_j}^{T_{j+1}} (X - E[\hat{\mu}_j^{mean}])^2 f(X) dX}{\int_{T_j}^{T_{j+1}} f(X) dX} \tag{16}$$

where $f(X)$ is from (11) and $E[\hat{\mu}_j^{mean}]$ is the stationary value of the center estimate using the mean estimator.

From similar properties like those used for (14), for the MAD estimator (9) we can derive its expected stationary value from:

$$\sum_{i=1}^{L} \varepsilon_i \left[\mathrm{erf}\left(\frac{E[\hat{\mu}_j^{med}] - \mu_i + cE[\hat{\sigma}_j^{MAD}]}{\sigma_i} \right) - \mathrm{erf}\left(\frac{E[\hat{\mu}_j^{med}] - \mu_i - cE[\hat{\sigma}_j^{MAD}]}{\sigma_i} \right) \right] =$$
$$= \frac{1}{2} \sum_{i=1}^{L} \varepsilon_i \left[\mathrm{erf}\left(\frac{\hat{T}_{j+1} - \mu_i}{\sigma_i} \right) - \mathrm{erf}\left(\frac{\hat{T}_j - \mu_i}{\sigma_i} \right) \right] \tag{17}$$

where c=0.6745.

In order to evaluate the parameters for the Gaussian kernels we must also evaluate the activation domains $V = [\hat{T}_j, \hat{T}_{j+1})$ for each Gaussian function. If the Euclidean distance is used in order to assign a new pattern to an activation region (4), we can estimate the boundary $\hat{T}_j$ between two activation regions j and $j+1$ as:

$$\hat{T}_j = \frac{\hat{\mu}_j + \hat{\mu}_{j+1}}{2} \tag{18}$$

for $j = 1, \ldots, L-1$. The first and the last boundaries are: $T_0 = -\infty$ and $T_L = \infty$.

In the case when the Euclidean distance is replaced by the Mahalanobis distance (7), then the boundary condition in one-dimensional case can be found solving the equation:

$$\left(\frac{\hat{T}_j - \hat{\mu}_j}{\hat{\sigma}_j} \right)^2 = \left(\frac{\hat{T}_j - \hat{\mu}_{j+1}}{\hat{\sigma}_{j+1}} \right)^2 \tag{19}$$

for $j = 1, \ldots, L-1$. Analytical methods can be used in order to find the boundaries as well as the model parameters.

We consider the following particular examples :

$$f(X) = \frac{1}{2} N(5, \sigma) + \frac{1}{2} N(10, \sigma) \tag{20}$$

$$f(X) = \frac{1}{3} N(3, \sigma) + \frac{1}{3} N(5, \sigma) + \frac{1}{3} N(10, \sigma) \tag{21}$$

We estimate the center and the scale parameter for the distribution $N(5, \sigma)$ using both mean and median estimators for RBF centers. The absolute errors $E[\hat{\mu}] - \mu$ are depicted in Figure 1a for the distribution (20) and in Figure 1b for the distribution (21) with respect to the scale parameter σ. The comparison results in the bias estimation for the scale parameter $E[\hat{\sigma}] - \sigma$ are presented in Figure 1c for the distribution (20) and in Figure 1d for the distribution (21). The estimation of the class means and scale parameters of (20) corresponds to the estimation of parameters of medium-tailed distribution and in the case of (21) to a short tailed distribution. All these plots show that in the cases when it is occuring a certain overlap in the functions to be estimated, the bias given by the robust algorithm it is smaller than that obtained by using classical methods.

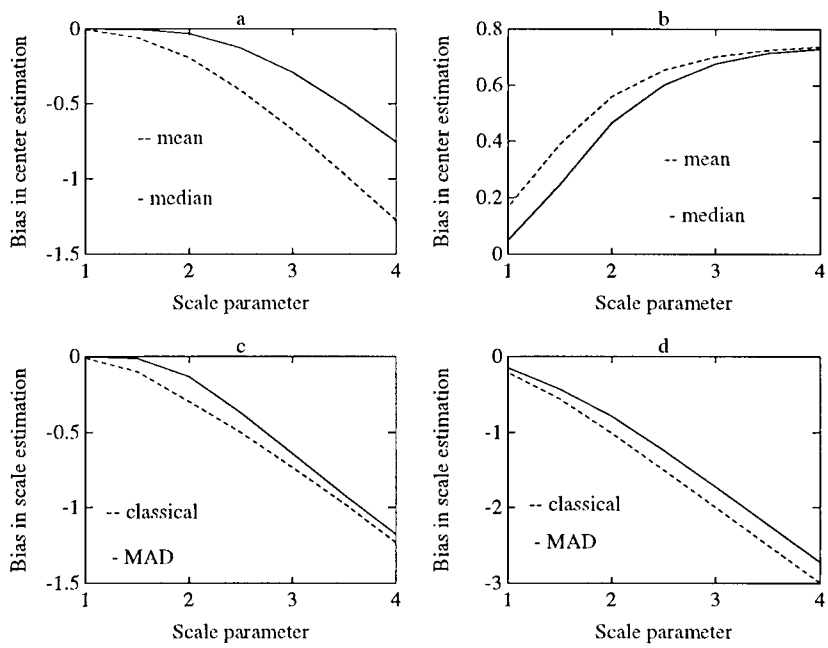


Figure 1: Theoretical analysis for robust and classical statistics estimators in evaluating the RBF parameters: a) estimation of the center for N(5,2) in a long-tailed distribution and b) in a short-tailed distribution; c) estimation of the scale parameter for N(5,2) in the first distribution and d) for the second distribution.

4 Simulation results

We have applied both algorithms presented in Section 2 and analyzed in Section 3 to the estimation of the parameters for mixed bivariate normal distributions. The first algorithm uses classical statistics estimators for finding the RBF parameters and the second uses robust estimators. In these applications we have used both Euclidean and Mahalanobis distances in order to assign a new coming pattern to a cluster.

We apply the networks for estimating the following distributions:

Distribution I: $P_1^I(X)$=N(2,1;3,1;0)+N(8,7;3,1;0)
$P_2^I(X)$=N(8,2;1,3;0)+N(2,6;1,3;0)

Distribution II: $P_1^{II}(X)$=N(6,0;4,1;0)+N(0,6;1,4;0)
$P_2^{II}(X)$=N(6,6;2,2;0)

Distribution III: $P_1^{III}(X)=\epsilon\ P_k^I+(1-\epsilon)$U([-5,15],[-5,15])

Distribution IV: $P_1^{IV}(X)=\epsilon\ P_k^{II}+(1-\epsilon)$U([-5,15],[-5,15])

where we denote a Gaussian distribution through $N(\mu_1,\mu_2;\sigma_1,\sigma_2;r)$, r is the correlation factor and a uniform distribution through U and $k \in \{1,2\}$, $\epsilon = 0.9$.

Table 1: Comparison between RBF and MRBF algorithms

Distribution	Method	Distance Measures			
		Euclidean		Mahalanobis	
		Error (%)	MSE	Error (%)	MSE
I	RBF	21.26	13.69	17.17	6.90
	MRBF	17.58	8.65	13.75	2.75
	Optimal	12.13	0.00	12.13	0.00
II	RBF	3.89	3.69	2.95	1.24
	MRBF	2.90	1.20	2.61	0.82
	Optimal	2.52	0.00	2.52	0.00
III	RBF	26.63	34.22	35.05	48.59
	MRBF	21.11	10.11	18.82	5.74
	Optimal	15.78	0.00	15.78	0.00
IV	RBF	15.28	32.36	22.21	39.61
	MRBF	8.78	5.50	7.24	2.49
	Optimal	7.18	0.00	7.18	0.00

The comparison measures are the miss-classification error and mean square error (MSE) between the true functions and those modeled by means of the neural network. The problem of multi-distribution estimation is seen as a pattern classification task. The optimal network is obtained when its parameters are identical to those of the given Gaussian distributions. The MSE is defined as:

$$MSE = \frac{1}{M}\sum_{k=1}^{M}\int_{\mathcal{D}}(Y^k(X) - \hat{Y}^k(X))^2\, dX \tag{22}$$

where the domain is $\mathcal{D} = (-\infty, \infty) \times (-\infty, \infty)$ in our case , $\hat{Y}^k(X)$ is the surface for the kth output unit and $Y^k(X)$ is the target function.

In the learning stage, we consider a window of W=401 samples (8) (for MRBF) and 4000 learning samples with equal number of samples for each cluster. The comparison results between the two methods are given in Table 1 where the same data were used for both algorithms. The simulations were repeated with different data, consistent with the same distribution functions and the presented results are the average of all these trials.

In all these cases, we have obtained a clear improvement by using the MRBF algorithm. When the mixture of bivariate normal distributions is contaminated with uniform distributed patterns the difference is very large because the median type learning is insensitive to the presence of outliers. Using the Mahalanobis distance instead of the Euclidean distance, we obtain better results, except for the classical estimators in the case of models contaminated by uniform noise.

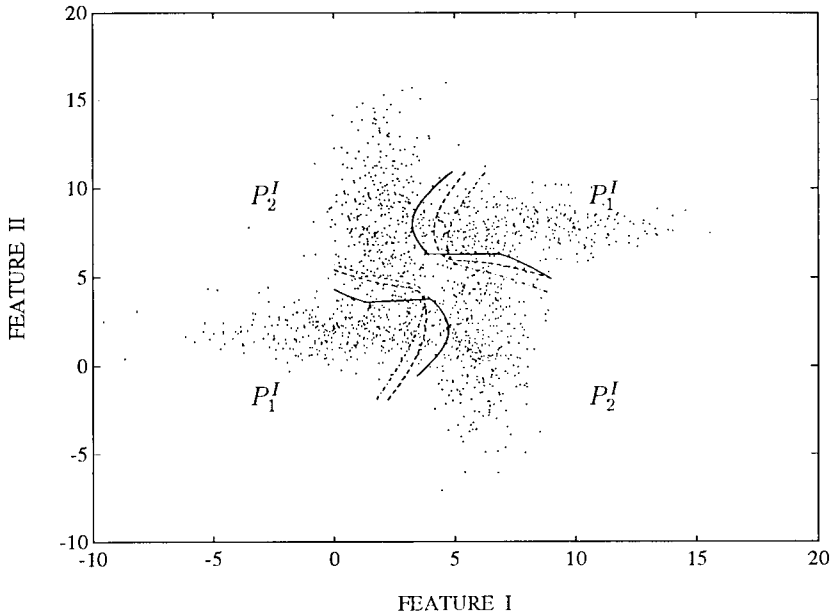


Figure 2: Samples from the distribution I and the boundaries between the classes: '-' optimal classifier, '- -' MRBF and '-·' RBF.

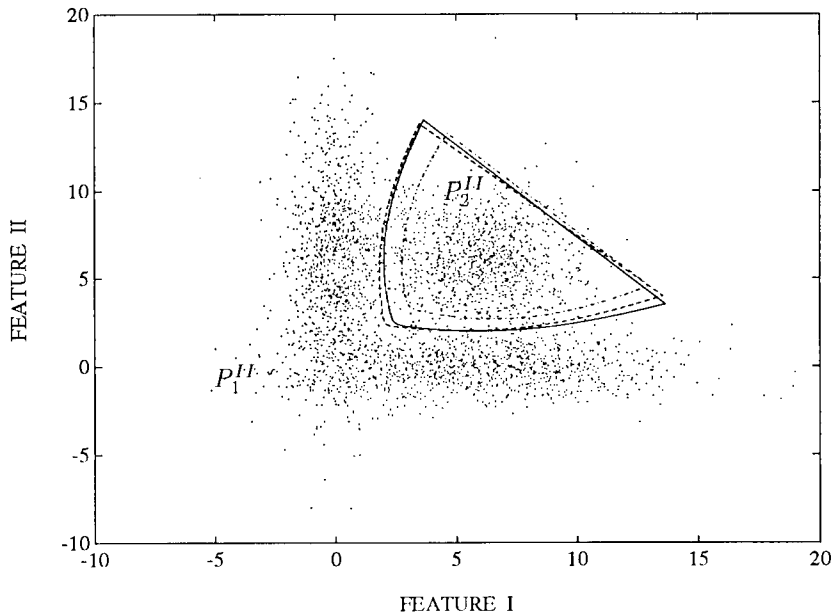


Figure 3: Samples from the distribution II and the boundaries between the classes: '-' optimal classifier, '- -' MRBF and '-·' RBF.

Samples drawn from the distributions I and II are depicted in Figures 2 and 3. The separation boundaries found by means of the RBF and MRBF networks as well as the optimal boundary are marked in these Figures. The separation boundaries are situated where two neighboring classes have equal probabilities. The decision rule for the assignment of a new pattern was based on Euclidean distance in Figure 2 and on Mahalanobis distance in Figure 3. It can be seen from these Figures that we obtain a better approximation of the optimal boundary by using MRBF compared with the classical algorithm.

In Figure 4 we evaluate the convergence of these algorithms in the case of distribution I. The learning curves represent the estimation of the *pdf* functions (MSE) with respect to the number of samples. From this plot the improvement given by MRBF compared with classical RBF learning and by using the Mahalanobis distance instead of the Euclidean distance is clear.

From the Table 1 we can see that MRBF gives better results in estimating the *pdf* functions and it is not biased by the presence of the outliers. MRBF gives more accurate approximations for the Bayesian boundaries then the classical statistical algorithms in the case of bivariate mixtures of Gaussians, as can be seen in Figures 2, 3. Median type learning applied to radial basis functions converge smoothly to a stationary value smaller than that obtained in classical estimation for RBF as can be seen in Figure 3.

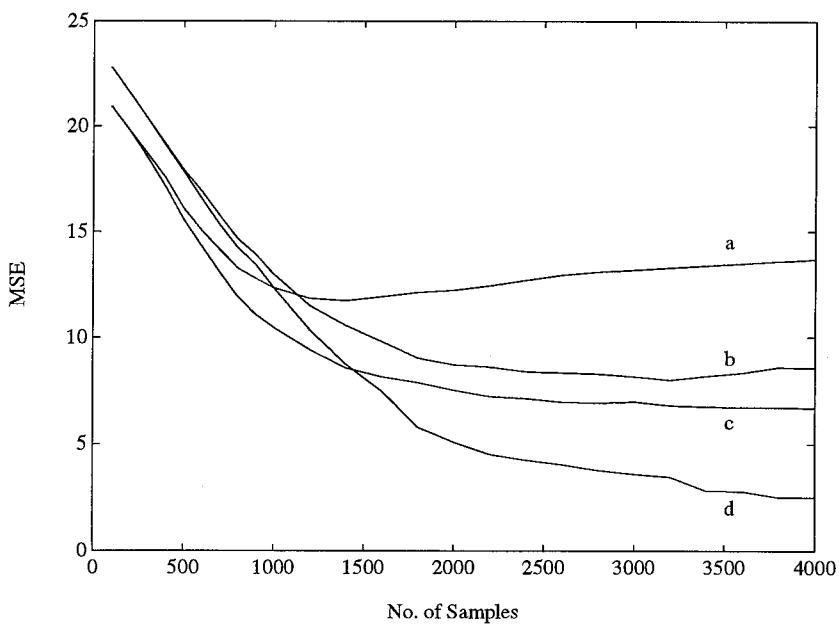


Figure 4: The learning curves in the case when the samples are drawn from the distribution I. Classical estimators are used together with the Euclidean distance for curve a and together with the Mahalanobis distance for curve c ; robust estimators are used together with the Euclidean distance for curve b and together with the Mahalanobis distance for curve d.

5 Conclusions

This paper presents a comparative study of two learning algorithms, one based on classical statistics estimators and the other on robust estimators. The algorithm derived from robust statistics and called Median RBF uses the median in order to find the centers in the network and median of absolute deviations for the estimation of the scale parameters. Both algorithms can be implemented on-line. We have derived theoretical analysis in a parameter estimation problem. The algorithm based on robust statistics is proved to give more accurate results in the one-dimensional estimation problem as well as in a two dimensional density function approximation. Possible fields of application for this algorithm are in communication systems, image processing and speech recognition.

References

[1] I. Pitas, A. N. Venetsanopoulos, *Nonlinear Digital Filters: principles and applications*, Hingham, MA: Kluwer Academic, 1990.

[2] G. Seber, *Multivariate Observations*, John Wiley, 1986.

[3] T. K. Kohonen, *Self-organization and associative memory*, 3rd edition, Berlin, Germany: Springer-Verlag, 1989.

[4] J. Moody, C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation,* vol. 1, no. 2, pp. 281-294, 1989

[5] T. Poggio, F. Girosi, "Networks for approximation and learning," *Proc. of the IEEE,* vol. 78, no. 9, pp. 1481-1497, Sep. 1990

[6] D. F. Specht, "A general regression neural network," *IEEE Trans. on Neural Networks,* vol. 2, no. 6, pp. 568-576, Nov. 1991

[7] S. Chen, B. Mulgrew, P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. on Neural Networks,* vol. 4, no. 4, pp. 570-579, Jul 1993

[8] A. G. Borş, M. Gabbouj, "Minimal topology for a radial basis functions neural network for pattern classification," to appear in *Digital Signal Processing , A review Journal*, 1994

[9] I. Pitas, P. Kiniklis, "Median learning vector quantizer," *Proc. SPIE, vol. 2180, Nonlinear Image Processing V,* San Jose, CA, pp. 23-34, 7-9 Feb. 1994

[10] E. Yair, K. Zeger, A. Gersho "Competitive learning and soft competition for vector learning quantizer design," *IEEE Trans. on Signal Processing,* vol. 40, no. 2, pp. 294-309, Feb. 1992

Network Architectures

THE USE OF RECURRENT NEURAL NETWORKS FOR CLASSIFICATION

T. L. Burrows M. Niranjan
Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ, England

Abstract–Recurrent neural networks are widely used for context dependent pattern classification tasks such as speech recognition. The feedback in these networks is generally claimed to contribute to integrating the context of the input feature vector to be classified. This paper analyses the use of recurrent neural networks for such applications. We show that the contribution of the feedback connections is primarily a smoothing mechanism and that this is achieved by moving the class boundary of an equivalent feedforward network classifier. We also show that when the sigmoidal hidden nodes of the network operate close to saturation, switching from one class to the next is delayed, and within a class the network decisions are insensitive to the order of presentation of the input vectors.

INTRODUCTION

Many classification problems depend on the context in which class data is received, *ie.* the history of previous classes. Human perception of speech is a typical example, in which coarticulation effects between adjacent phonemes are important contextual factors for correct recognition, especially in noise. The performance of a classifier can be enhanced by providing past and future context. Future context can be provided by a delay between input window and output decision. Past context can be presented within an input window which contains a fixed number of previous frames [1], and by including delayed feedback paths (recurrent connections), which provide information about previous local decisions [2]. For a fixed input window, the depth of the context *ie.* the number of frames spanned by the input, is fixed. The classifier may miss dynamic features of the class with a longer duration than that of the input window and cause smoothing of features that change rapidly within this window. For a recurrent network, the depth of the context is potentially infinite, but in practise is determined by the relative size of the recurrent connection weights.

Much experimental work *eg.* [2], has reported improved performance of recurrent networks over feed-forward networks. In a previous paper [3], we looked at how this is achieved for the system identification of time-varying patterns. In this paper, we proceed by studying how recurrent networks operate for classification of time-varying patterns. We concentrate specifically on how the recurrent connections make use of previous context during 2-class classification problems such as classification of phoneme pairs from the TIMIT database.

0-7803-2026-3/94 $4.00 © 1994 IEEE

EFFECT OF FEEDBACK ON DECISION BOUNDARY POSITION

Consider the unit delay recurrent connection around a single hidden node, with a nonlinearity $f(x) = \tanh(x)$, shown in Fig. 1. The output node is linear and the classification decision is determined by an output threshold at zero.

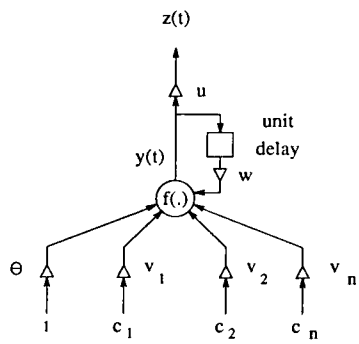


Figure 1: Single Hidden Node With Recurrent Connection

For such a network, the equations for the output of the hidden node, $y(t)$ and network output, $z(t)$, are :

$$y(t) = f\left(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x}(t) + wy(t-1) + \theta\right) \quad (1)$$

$$z(t) = uy(t) \quad (2)$$

where $\boldsymbol{v}$ is a vector of input weights, $\boldsymbol{x}(t)$ is a vector of input parameters, $[c_1, c_2, \ldots, c_n]^{\mathrm{T}}$, θ is a bias term and $(.)^{\mathrm{T}}$ denotes transpose. We used cepstral coefficeints derived from phoneme segments from the TIMIT database as an example input. The networks were trained as 2-class classifiers using back-propagation through time to minimise the mean squared-error, with class targets of -1 and 1. The decision boundary is defined by :

$$\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x}(t) + wy(t-1) + \theta = 0 \quad (3)$$

The contribution $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x}(t) + \theta$, represents a static linear decision boundary which can be interpreted as the decision boundary for a feed-forward classifier which has the same weights u, $\boldsymbol{v}$ and θ. The term $wy(t-1)$ represents a variable bias which causes the decision boundary to move parallel to $\boldsymbol{v}$. We consider a trajectory of points in class 1, Fig. 2 a), for which some of the points are incorrectly classified by the static boundary. For these errors to be corrected, the decision boundary must move away from class 1, biasing the current decision towards that of the previous classification. This occurs for positive w, which also gives stable feedback around the node. Hence for maximum classifier performance, we require a training algorithm which develops

positive w. The limits of the boundary movement lie at $\pm w$ on either side of the static boundary, and with this they divide the input space into 4 regions, A–D, as shown in Fig 2 b). Classification of points in A and D is unaffected by the position of the decision boundary and is independent of their context. A and D define a region of the input space in which the number of classification errors made by the recurrent net is predetermined. B and C define an indeterminate region of the input space, of width $2|w|$, in which the classification of points requires knowledge of their context, since movement of the decision boundary in this region causes both correction of and addition to errors made by the feed-forward net. The sensitivity of the output to the context of the input data implies that the order of presentation of the training classes is important. Different orders of presentation of the classes will not converge to the same solution, when starting from the same weight initialization.

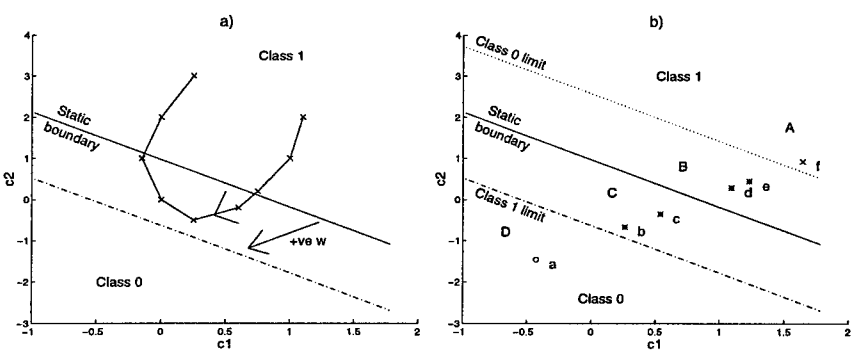


Figure 2: a) Movement of decision boundary by recurrent connection. b) Decision boundary limits and classification regions

EFFECT OF DECISION BOUNDARY MOVEMENT ON OUTPUT SWITCHING

The decision boundary movement, which biases the current decision towards that of the previous decision, gives a classifier output which exhibits a switching delay and is trajectory sensitive *ie.* is dependent on the order of presentation of input data within the current class. The magnitude of the delays and the extent of the trajectory sensitivity is determined by the relative range of the indeterminate region, B and C, and the *approximately linear* region of the node function, Fig. 3. All points in regions A and D are trajectory insensitive, since they cannot move the decision boundary. Only indeterminate points which lie within the linear region of the node function, shown hatched in Fig. 3, can cause boundary movement and are therefore trajectory sensitive. The entire indeterminate region will be trajectory sensitive if it is completely spanned by the linear region of the node function.

The variation in switching delay for different points within the indeterminate region is shown in Fig. 4. The decision boundaries and test data points for this classifier

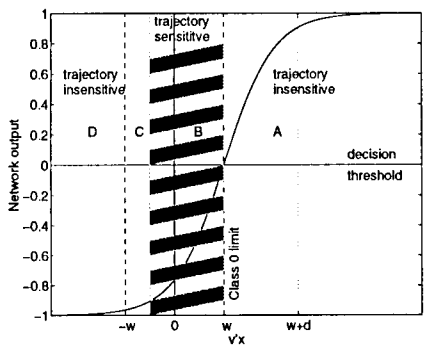


Figure 3: Trajectory sensitive region of input space when decision boundary lies at class 0 limit, for a nonlinear function which is linear over the range 2d and $\theta = 0$

are shown in Fig. 2 b). For a narrow nonlinear function, only a few indeterminate points are within the linear region and can cause switching. This switching is rapid since the boundary moves quickly across the indeterminate region to the other class limit. Most points cause saturation and no switching, Fig. 4 a). Hence most of the indeterminate region is trajectory insensitive and smoothing of the classifier output occurs *ie.* previous decisions are favoured. This is obvious in the limiting case of a step function, in which the linear width is zero. For this nonlinearity, the boundary can only lie at a class limit and only points in A or D cause switching. In this case, all indeterminate points are also trajectory insensitive and the previous decision is always chosen, giving maximum output smoothing. For a wide nonlinear function, Fig. 4 b), more of the indeterminate points fall within the linear region and are therefore trajectory sensitive, as shown in Fig. 3. The wider the linear region, the more slowly the decision boundary crosses the indeterminate region, giving longer switching delays and greater output smoothing. The actual boundary movement caused by a trajectory of points, **f** to **a**, which span the 'indeterminate region', is shown in Fig. 5 a), for $f(x) = \tanh(x)$. The boundary and data trajectory move in opposite directions, and due to the finite switching delay illustrated in Fig. 4 b), the recurrent decision lags the feed-forward decision, Fig. 5 b).

The limited trajectory sensitivity of the recurrent network is illustrated in Fig. 6, for a network with 10, 5 and 1 units in the input, hidden and output layer respectively. The network was trained as a classifier of voiced and unvoiced phonemes on sentences from the TIMIT database. In testing, adjacent input frames within a class segment were swapped and the classifier output compared with that for the normal input order. For most segments, the hidden nodes saturate, giving similar recurrent network outputs in Fig. c) and d). Only segments in which a node operates in the linear region (node output < 0.5 in Fig. 6 b).) are the network outputs very different. Smoothing of the recurrent network output within a class segment is seen in the unvoiced segments around frames 100 and 145 and a switching delay in the unvoiced/voiced class transition at frame 151.

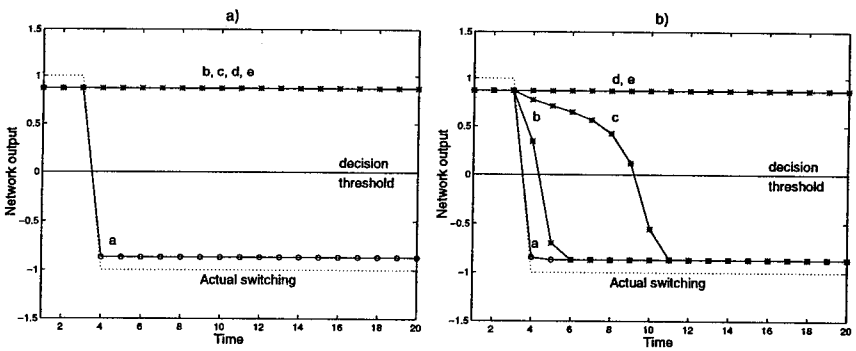


Figure 4: Delay in 1–0 switching, **f** to **a**: a) Narrow node function f(x)= tanh(10x). b) Wide node function f(x)= tanh(x)

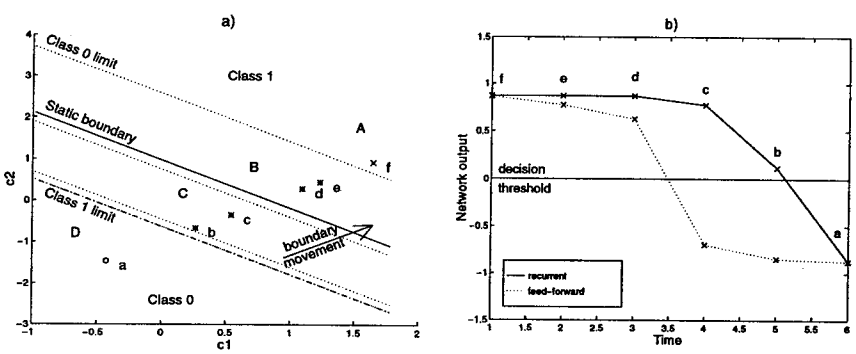


Figure 5: Movement of decision boundary due to a recurrent connection: a) Decision boundaries for 1–0 trajectory, b) Classifier output for 1–0 trajectory

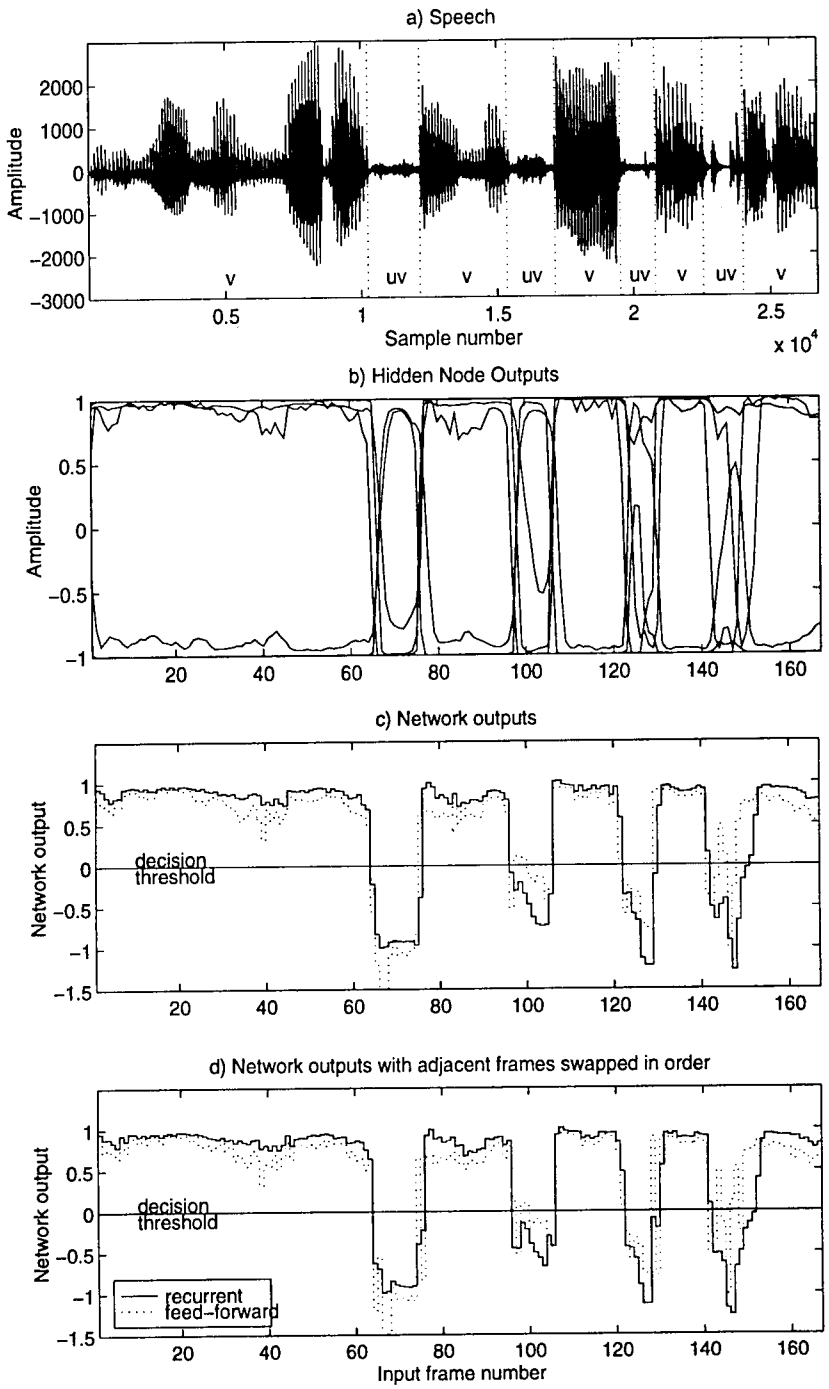


Figure 6: Voiced-unvoiced classification of phrase "a woman met a famous author"

EFFECT OF DECISION BOUNDARY MOVEMENT ON CLASSIFIER PERFORMANCE

The movement of the decision boundary has the potential to both improve and impede the performance of a recurrent net over that of a feed-forward net due to the variable classification of points in the indeterminate region, B and C. The recurrent net cannot correct errors in regions A and D. The combined effect of the trajectory sensitivity and switching delay caused by boundary movement, is to smooth output decisions of the recurrent net causing them to lag those of the feed-forward net. If the 'indeterminate region' is too narrow, Fig. 7 a), the feed-forward and recurrent outputs are almost identical and there is little difference in performance, Fig. 7 c). Conversely, if the 'indeterminate region' is too wide, Fig. 7 b), most classifications are dependent on previous decisions and over-smoothing of the output occurs, causing the performance of the recurrent net to fall below that of the feed-forward net, Fig. 7 d). Hence to minimise the additional errors of the recurrent network caused by switching delays, we require the indeterminate region to bind, as tightly as possible, any region of data overlap surrounding the static boundary.

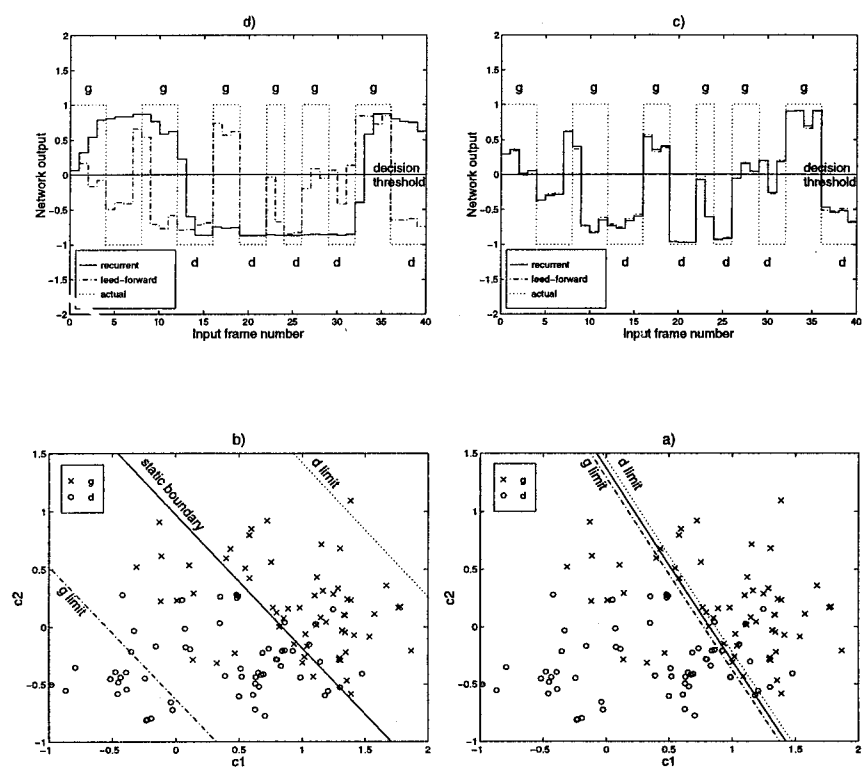


Figure 7: Test patterns and decision boundary limits for a 'g–d' classifier: a) w too large, b) w too small. Network output: c) w too large, d) w too small.

The smoothing of the recurrent network output can explain the change in relative performance of recurrent and feed-forward classifiers at different frame-rates (resolutions) [4], where recurrent networks are reported to perform better at lower frame-rates. At a higher frame-rate, there are more frames for a given phoneme duration but the parameters vary much less on a frame-to-frame basis than at a lower rate, causing saturation of the recurrent network. At a phoneme boundary, the small changes in parameter values at each frame cause the recurrent net to switch slowly, causing smoothing of the output decisions and a fall in performance below that of a feed-forward net.

DISCUSSION

Recurrent neural networks are widely used for context dependent pattern recognition. In speech recognition, for example, their application is motivated by the need to integrate acoustic cues that are distributed over time. It is generally claimed that this ability to model the temporal correlation in the data vectors gives recurrent neural network classifiers greater power than state-of-the-art acoustic models based on hidden Markov modelling. The observations reported in this paper suggest this may not be the case in practice. We have shown that the contribution of the feedback is primarily a smoothing operation. This can improve performance over a static classifier in regions of the input space where the class data may overlap, by moving the class boundary of the static classifier. The smoothing can also cause a delay in switching from one class to the next.

We also observed, that when the hidden nodes operate in the saturated regions of the sigmoid, the network outputs are not sensitive to the order of presentation of the input examples within a class. When this happens, the network is not modelling the trajectory of the input vectors and is effectively treating each data vector within a class independently, similar to a hidden Markov model state. We suggest some of the above problems can be overcome by setting the network targets (or weighting the error function) in a similar manner to Etemad [5] and Watrous [6]. These authors use a ramp-like target function over the duration of a class, say a phoneme in speech recognition, to reflect the increasing confidence of class membership as more and more data is received. Such training will force the hidden units to stay out of saturation, avoiding some of the problems we have pointed out.

For the single hidden node recurrent net, a linear decision boundary, $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x}$, is defined, with a bias of $\theta + wy(t-1)$. We have shown that the effect of w is to bias the current decision towards that of the previous decision, in a similar way to which the log prior ratio biases the decision boundary of a Bayes optimal descriminant function towards the most probable class [7]. We can interpret $wy(t-1)$ as acting like a variable prior ratio, since $wy(t-1)$ determines which class is favoured. The recurrent connection thus updates our estimate of the priors, depending on the previous context, $y(t-1)$. In [8], variation in the class priors between training and test data is accounted for by scaling the network outputs. Recent work on feed-forward nets by Gish [9] has shown that adjustment of the output biases is sufficient to adapt the classifier to the

new data. For a recurrent net, this suggests that modifications to both the recurrent weights and the biases are necessary.

For a feed-forward network (multi-layer perceptron or MLP) with a single output node, training by back-propagation is known to yield a minimum mean squared-error estimate of the Bayes optimal descriminant function [10], in which the outputs are treated as posterior probabilities. The MLP approximation is only accurate if there are sufficient hidden nodes to capture the complexity of the function. With multiple hidden nodes, the decision boundaries become nonlinear and result as a combination of local decisions by each node. For the recurrent network case, cross terms in the feed-back matrix, w determine how previous decisions in other local regions of the input space affect the current local decision. We are now studying the multiple hidden node case more closely and expect the indeterminate regions for each local decision to overlap resulting in more of the input space being context sensitive.

REFERENCES

[1] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," Tech. Rep. TR-1-0006, ATR, October 1987.

[2] A. J. Robinson and F. Fallside, "Phoneme recognition from the TIMIT database using recurrent error propagation networks," Tech. Rep. CUED/F-INFENG/TR.42., Cambridge University, England, 1990.

[3] T. L. Burrows and M. Niranjan, "The use of feed-forward and recurrent neural networks for system identification," Tech. Rep. CUED/F-INFENG/TR158., Cambridge University, England, 1993.

[4] S. Renals, M. Hochberg, and A. J. Robinson, "Learning temporal dependencies in connectionist speech recognition," in *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, 1994.

[5] K. Etemad, "Phoneme recognition based on multi-resolution and non-causal context," in *Proc. 1993 IEEE Workshop on Neural Networks for Signal Processing* (C. A. Kamm, G. M. Kuhn, B. Yoon, R. Chellappa, and S. Y. Kung, eds.), pp. 343–352, 1993.

[6] R. L. Watrous and L. Shastri, "Learning phonetic features using connectionist networks: An experiment in speech recognition," in *Proc. 1987 1st International Conference on Neural Networks*, pp. 318–388, 1987.

[7] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[8] M. D. Richard and R. P. Lippman, "Neural classifiers estimate bayesian a posteriori probabilities," *Neural Computation*, vol. 3, no. 4, pp. 461–483, 1991.

[9] H. Gish and M. Siu, "An invariance property of neural networks," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (Adelaide)*, pp. 541–544, 1994.

[10] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter, "The multilayer perceptron as an approximation to a Bayes optimal descriminant function," *IEEE Trans. on Neural Networks*, vol. 1, no. 4, pp. 296–298, 1990.

NETWORK STRUCTURES FOR NONLINEAR DIGITAL FILTERS

Ji-Nan Lin and Rolf Unbehauen
Lehrstuhl für Allgemeine und Theoretische Elektrotechnik
Universität Erlangen-Nürnberg
Cauerstrasse 7, 91058 Erlangen, Germany

Abstract **Mapping neural networks based on a piecewise-linear (PWL) function approximation scheme are useful in signal processing, i.e. nonlinear filtering. However, the traditional canonical PWL model has a drawback that limits the usefulness of these networks. To overcome this limitation, three more general PWL models with their network implementation structures are introduced in this paper. As the first application of the models in signal processing, the modelling, the unification, and the generalization of the useful nonlinear filter family, the order statistic filters are considered.**

I. INTRODUCTION

Neural networks whose input/output relation is characterized by a map, i.e. a real function of several variables, play an important role in signal processing. They are often used as a method for nonlinear digital signal filtering, which is traditionally thought a difficult problem to deal with. It is expected that the application of neural networks in signal filtering opens a new way towards generalizing or unifying existing nonlinear filters, which were mostly developed for some special purposes [1].

The most popular mapping networks may be the multilayer perceptrons motivated by a model of a biological perceptual system. Recently, mapping networks have also been developed on some function approximation schemes, e.g. the Radial-Basis-Function network and the networks that implement a piecewise-linear (PWL) function. The network structures developed in [2,3] are based on the so-called canonical PWL function proposed in [4]. It has recently been revealed that the multilayer perceptrons can also be regarded as belonging to the family of canonical PWL networks [5]. A canonical PWL network has advantages in the implementation. Theoretically, it can be used as a general model to approximate an arbitrarily given filtering operator in practice [6]. The usefulness of the canonical PWL filter has been proved in various ap-

This work is supported by the Deutsche Forschungsgemeinschaft, Bonn, Germany.

plications of nonlinear signal or image processing.

Mapping networks implementing a PWL function should have a particular meaning in nonlinear signal filtering. One notices that some useful signal processing approaches are inherently PWL characterized. For instance, an important approach of nonlinear filtering is the family of filters based on order-statistics [1]. This family is rich in members and possesses useful specialities in applications, e.g. image processing. An order statistic filter is in fact a PWL filter. This fact has not caught enough attention in the literature where specialized network structures are proposed for the realization of an order statistic filter [1,8]. It is naturally expected that a rather general PWL network structure will be meaningful in the unification and further generalizations of the family of order statistic filters.

The canonical PWL function serves as the only method at present to represent PWL functions in an explicit and compact form that is easy to implement in a network structure. However, the canonical model has a fatal drawback, i.e. it only works for a subclass of PWL functions. This drawback limits the usefulness of the canonical PWL network in applications, e.g. signal processing. Some useful PWL functions, including the function of an order statistic filter, cannot be represented by the canonical model. It is thus a meaningful research theme to overcome the limitation of the canonical model and to achieve a more general model for PWL functions.

In this paper we will consider the problem of the PWL model from the viewpoints of both the representation capability as large as possible and the suitability for the network implementation. First we will introduce a general scheme for representing all PWL functions. Two simplified models are then developed from the general scheme. Each of them has its specialities, but both are capable of representing all continuous PWL functions. Therefore, they will be more useful than the canonical PWL ones in applications of nonlinear signal filtering. The use of the models for the implementation of an order statistic filter and its generalizations are considered.

II. PWL FUNCTIONS -- A GENERAL MODEL

Let us consider a PWL function $f : \mathbb{R}^N \longrightarrow \mathbb{R}$. The domain is partitioned into a set of sub-spaces (*regions*)

$$\underline{R} := \{ R_p \subset \mathbb{R}^N \mid \bigcup_{p=1}^{P} R_p = \mathbb{R}^N, R_p \cap R_{p'} = \phi,\ p \neq p',\ p, p' \in \{1, 2, ..., P\} \} \quad (1)$$

by a finite set of Q $(N-1)$-dimensional hypersurfaces (*boundaries*)

$$\underline{H} := \{ H_q \subset \mathbb{R}^N, q \in \{1, 2, ..., Q\} \}. \quad (2)$$

Each boundary is characterized by

$$H_q := \{\boldsymbol{x} \in \mathbb{R}^N \mid \varphi_q(\boldsymbol{x}) = 0\}, \tag{3}$$

where $\varphi_q : \mathbb{R}^N \longrightarrow \mathbb{R}$ is called the *boundary function* of H_q. As usual, the boundary functions are limited only to linear ones. That means, the boundaries are all $(N-1)$-dimensional hyperplanes.

On each region $R_p, p \in \{1,2,...,P\}, f$ is represented by

$$f(\boldsymbol{x}) = f_p(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in R_p \subset \mathbb{R}^N, \tag{4}$$

where $f_p : \mathbb{R}^N \longrightarrow \mathbb{R}$ is a linear function called the *local function* of R_p.

An important class of PWL functions is that of the continuous ones, for which there is

$$f_p(\boldsymbol{x}) = f_{p'}(\boldsymbol{x}), \quad \text{where } \boldsymbol{x} \in \bar{R}_p \cap \bar{R}_{p'}, \tag{5}$$

for all $p, p' \in \{1,2,...,P\}$.

Although the PWL functions have shown their usefulness in many scientific and engineering areas, it is still a difficult task to find a compact global representation for all of them, (even only for all continuous ones.) From the viewpoint of mapping networks, a useful PWL representation which is suitable for a network realization is strongly expected.

From the above definition we can first see that any general representation of PWL functions should include two aspects, concerning the local functions and the domain partition, respectively. Through a further study we see that the topological structure of the boundaries in the domain partition plays a key role in the representation. Generally speaking, PWL functions with their domain partition topologically similar to each other, should have a similar representation.

A general scheme for representing PWL functions is given as follows

$$f(\boldsymbol{x}) = \sum_{p=1}^{P} f_p(\boldsymbol{x}) \gamma_p(s(\varphi_1(\boldsymbol{x})), s(\varphi_2(\boldsymbol{x})), ..., s(\varphi_q(\boldsymbol{x}))), \tag{6}$$

where f_p, $\varphi_q : \mathbb{R}^N \longrightarrow \mathbb{R}, s : \mathbb{R} \longrightarrow \{0,1\}$ is the hardlimit function:

$$s(\xi) := \begin{cases} 0, & \xi \leqq 0 \\ 1, & \xi > 0, \end{cases} \tag{7}$$

and γ_p for $p = \{1,2,...,P\}$ are logical operators $\gamma_p : \{0,1\}^Q \longrightarrow \{0,1\}$.

We call this scheme the $f - \varphi$ model of PWL functions, which emphases the two aspects of the local functions and the domain partition. Clearly, all PWL functions defined as above can be represented by this model. One notices that a region R_p for $p \in \{1,2,...,P\}$ can also be defined as

$$R_p = \{\boldsymbol{x} \in \mathbb{R}^N \mid \varphi_q(\boldsymbol{x}) > 0, \text{ for some } q \in \{1,2,...,P\} \text{ and}$$

$$\varphi_{q'}(\boldsymbol{x}) \leqq 0, \text{ for other } q' \in \{1,2,...,P\}\}. \tag{8}$$

The kernel part of the $f-\varphi$ model are the logical operators γ_p that carry the information about the topological property of the domain partition of the PWL function to be represented. The logical operators will not change if the $f-\varphi$ model is used to represent PWL functions that are isomorphic with each other, i.e. they are defined on isomorphic domain partitions. [1]

A network implementation of the $f-\varphi$ model is straightforward, as illustrated by Fig. 1, where the blocks of f_p and φ_q are linear combinators implementing the local and boundary functions of a PWL function, respectively. S is the hardlimit unit. Γ is a logical array implementing the logical operators, representing the topological structure of the domain partition. Beside these, the components needed by this network are simply switches.

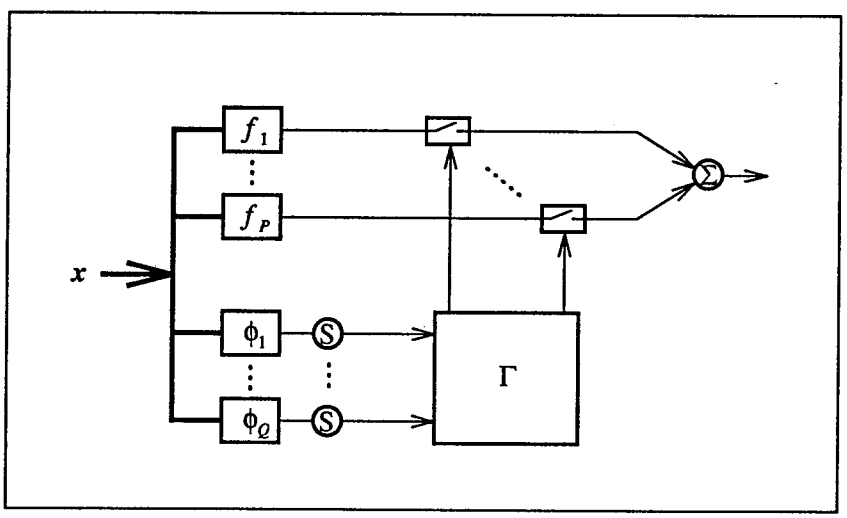


Fig. 1 Network structure of the $f-\varphi$ model

As a general scheme the $f-\varphi$ model takes all information into account which may be required in representing an arbitrary PWL function. For a concrete problem of a given PWL function, the network structure may be unnecessarily large or complicated. In practice, PWL functions are often used within a constrained subclass. Therefore, it is meaningful to study the simplification of the $f-\varphi$ model by attaching some constraints. The most widely used subclass of PWL functions may be that of the continuous ones. It has been investigated that the continuity leads to a strong constraint between the local and the boundary functions of a PWL function [9]. That means a large reduction of information should be achieved in representing a continuous PWL function. In the following we will introduce two simplified versions of the $f-\varphi$ model based on

[1] Two domain partitions are said isomorphic with each other if there exists a one-to-one correspondence between the boundaries, their intersections, intersections of intersections etc. of them.

this kind of constraint.

III. $f-f$ MODEL AND NETWORK STRUCTURE

The first simplification of the $f-\varphi$ model is defined as follows:

$$f(\boldsymbol{x}) = \sum_{p=1}^{P} f_p(\boldsymbol{x})\psi_p(r_1(f_1(\boldsymbol{x}),\ldots,f_P(\boldsymbol{x})),\ldots,r_P(f_1(\boldsymbol{x}),\ldots,f_P(\boldsymbol{x}))), \tag{9}$$

where $r_p: \mathbb{R}^P \longrightarrow \mathbb{Z}$ are rank functions defined by

$$r_p(\xi_1,\ldots,\xi_P) := k \in \{1,2,\ldots,P\},$$

$$\text{if } \xi_p \text{ is the } k\text{th largest in } \{\xi_1,\ldots,\xi_P\}, \tag{10}$$

and ψ_p for $p = \{1,2,\ldots,P\}$ are quasi-logical operators $\psi_p: \{1,2,\ldots,P\}^P \longrightarrow \{0,1\}^P$.

We call this representation scheme the $f-f$ model of PWL functions. Comparing the $f-f$ model with the $f-\varphi$ one we see that all boundary functions become implicit. That is, in the $f-f$ model the information about the domain partition is implied in that about the local functions. Accurately speaking, the $f-f$ model represents PWL functions where the domain partition is determined by the order of values of the local functions. One notices that this property is possessed by a continuous PWL functions. Therefore, we have

Theorem 1:

Any continuous PWL function may be represented by the $f-f$ model.

A network structure of the $f-f$ model is given in Fig. 2. The blocks implementing the rank functions bring about no difficulty since they can simply be expressed as:

$$r_p(\xi_1,\ldots,\xi_P) = \sum_{i \neq p} s(\xi_p - \xi_i). \tag{11}$$

The array $\boldsymbol{\psi}$ implementing the quasi-logical operators $\psi_p, p = 1,2,\ldots,P$ is also easy to realize.

The $f-f$ model may be especially effective in some cases where the PWL function to be represented has a relatively small number of local functions. A typical example is an order statistic filter. An kth order statistic filter of the signal $x(n)$ is described by

$$y(n) = x_{(k)}(n) := k\text{th largest value of } \{x(n-i), i \in \mathcal{S}\}, \tag{12}$$

where $\mathcal{S} \subset \mathbb{Z}$ is the filter window. Let S be the size of the window. An order statistic filter can be regarded as a PWL function of S variables $f(\boldsymbol{x})$, where

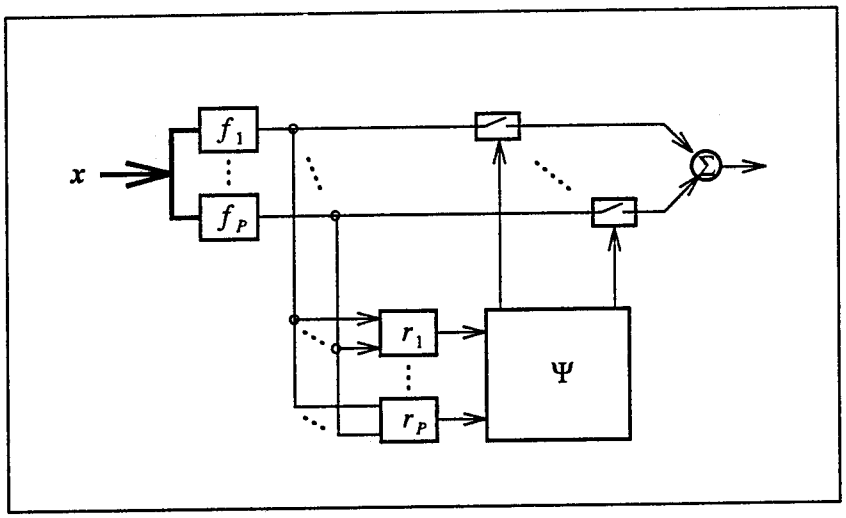


Fig. 2 Network structure of the $f-f$ model

$x_s \in \boldsymbol{x}$ for $s = 1,2,...,S$ are the values of the signal pixels within the window. f is continuous. It consists of $P = S$ local functions

$$f_p(\boldsymbol{x}) := x_s\,, \quad s \in \{1,2,...,S\} \tag{13}$$

for $p = 1,2,...,S$ and $Q = S(S-1)/2$ boundaries in the domain partition characterized by

$$\varphi_q(\boldsymbol{x}) := x_i - x_j = 0\,, \quad i > j\,,\ i,j \in \{1,2,...,S\}\,. \tag{14}$$

To represent the PWL function of the kth order statistic filter by the $f-f$ model we have simply

$$\psi_p(\boldsymbol{x}) = s(r_p(\boldsymbol{x}) - k + 1) - s(r_p(\boldsymbol{x}) - k)\,. \tag{15}$$

The network realization of this $f-f$ model is shown in Fig. 3. One notices that the network structure is similar to the OSNet proposed in [8]. The OSNet is developed as a special building block for an efficient hardware implementation of order statistic filters, while our network is as a special issue of the more general PWL model.

For all local functions $f_p(\boldsymbol{x})$ in this model being linear ones we obtain an extension of the order statistic filter. Median hybrid filters where a median filter, a special issue of the order statistic filter, is coupled to the outputs of a group of linear FIR filters [1], belong to this extension. Theoretically, any further (linear) extension can also be represented by the $f-f$ model, provided that the continuity is preserved. However, if the extension leads to a large increase of the local functions, it may be no longer efficient for the $f-f$ network realization.

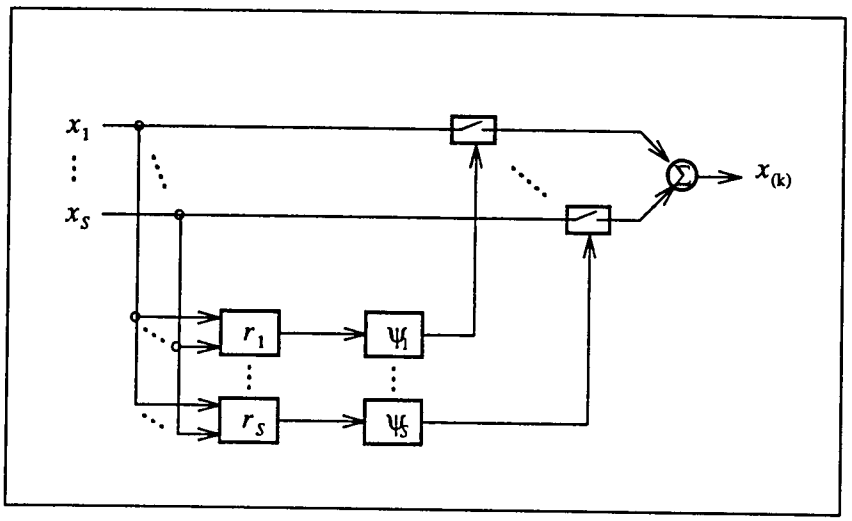


Fig. 3 Network structure of the $f-f$ model of an order statistic filter

IV. $\varphi-\varphi$ MODEL AND NETWORK STRUCTURE

Now we introduce another variation of the $f-\varphi$ model, with which a PWL function is represented by

$$f(\boldsymbol{x}) = f_0(\boldsymbol{x}) + \sum_{q=1}^{Q} \varphi_q(\boldsymbol{x}) \sum_{i_q=1}^{I_q} c_{q,i_q} \gamma_{q,i_q}(s(\varphi_1(\boldsymbol{x})), s(\varphi_2(\boldsymbol{x})), \ldots, s(\varphi_Q(\boldsymbol{x}))), \tag{16}$$

where $f_0 : \mathbb{R}^N \longrightarrow \mathbb{R}$ is a linear function, $c_{q,i_q} \in \mathbb{R}$, and γ_{q,i_q} are logical operators $\gamma_{q,i_q} : \{0,1\}^Q \longrightarrow \{0,1\}$, for $i_q = 1,2,\ldots,I_q$ and $q = 1,2,\ldots,Q$, respectively.

We call this variation scheme the $\varphi-\varphi$ model of PWL functions. It is seen that in this model all local functions are implicit. The $\varphi-\varphi$ model represents PWL functions for which each local function is expressed as a linear combination of the boundary functions. This property is fulfilled by a continuous PWL function. Thus, we have also:

Theorem 2:

Any continuous PWL function may be represented by the $\varphi-\varphi$ model.

Dual to the $f-f$ model, the $\varphi-\varphi$ model may be more suitable for representing PWL functions which have a relatively small number of boundary functions. A further study of the continuous case can reveal more details of the inner structure of the logical array $\boldsymbol{\gamma}$.

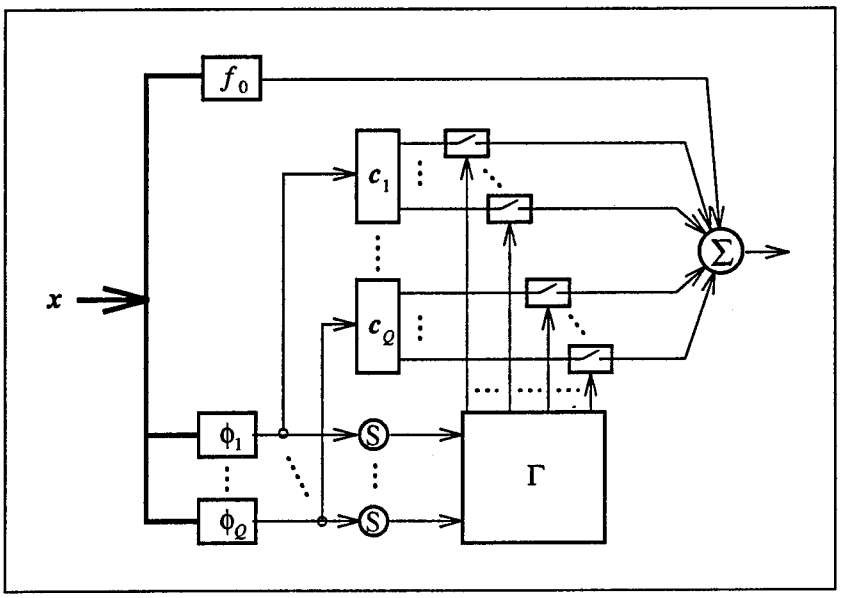


Fig. 4 Network structure of the $\varphi-\varphi$ model

Let simply $I_q = 1$ for all $q = 1,2,...,Q$ and

$$\gamma_{q,1}(s(\varphi_1(\boldsymbol{x})),s(\varphi_2(\boldsymbol{x})),...,s(\varphi_Q(\boldsymbol{x}))) = s(\varphi_q(\boldsymbol{x})),\ q = 1,2,...,Q\ . \quad (17)$$

Then, (16) becomes

$$f(\boldsymbol{x}) = f_0(\boldsymbol{x}) + \sum_{q=1}^{Q} c_q \varphi_q(\boldsymbol{x}) s(\varphi_q(\boldsymbol{x}))$$

$$= f'_0(\boldsymbol{x}) + \sum_{q=1}^{Q} c'_q \mid \varphi_q(\boldsymbol{x}) \mid \quad (18)$$

with f'_0 a linear function and $c'_q \in \mathbb{R}$. That is just the canonical PWL model given in [7], which constitutes the basis for the network structure developed in [2] and [3]. The main points of this canonical model are its explicity, compactness and that its network realization is rather simple. Unfortunately, this model is available only in a subclass of PWL functions. For the existence condition it has been proved in [7] that a continuous PWL function has a canonical representation if and only if it possesses the so-called "consistent variation property", i.e., for each boundary H_q there should exist a unique constant λ_q such that for any pair of regions R_p and $R_{p'}$ separated by H_q there is

$$f_p(\boldsymbol{x}) - f_{p'}(\boldsymbol{x}) = \lambda_q \varphi(\boldsymbol{x}). \quad (19)$$

The consistent variation property is not satisfied by some important PWL func-

tions, e.g. the PWL function of an order statistic filter. Based on the fact that the canonical model is only the simplest special issue of the $\varphi-\varphi$ model, it is expected that useful alternatives or extensions of the canonical model may be achieved through a deeper study of the $\varphi-\varphi$ structure, i.e. the logical array.

Through a study of the PWL characteristics of an order statistic filter it can be seen that the $\varphi-\varphi$ model of the filter should have $I_q = 2$ with $c_{q,1} = -1$ and $c_{q,2} = 1$ for all $q = 1,2,...,Q$. The logical structure of $\boldsymbol{\gamma}$ depends on the window size, i.e. S. We have seen in the preceding section that for an order statistic filter or the extension of median hybrid filter we have $P < Q$. That means, a $\varphi-\varphi$ network representation of the order statistic filter itself may not be efficient, in comparison to an $f-f$ one. Besides the median hybrid filter there are still other kinds of linear extensions, e.g. the L-filter and the Ll-filter which combine order statistic filters with linear FIR filters [1]. Since the combination of a PWL function with a linear function still results a PWL function, these linear generalizations are also PWL characterized. For these extensions, however, $P < Q$ is no longer tenable. Then, the $\varphi-\varphi$ model may be more suitable for representing such an extension. It can be seen that, for a given filter window, there exists a fixed $\varphi-\varphi$ structure which is common in representing all order statistic filters and all of their linear extensions, including the median hybrid filters, the L-filters, the Ll-filters [1], and even more. This is because in this case the PWL functions of these filters are all isomorphic with each other. In this sense, we may say that the $\varphi-\varphi$ model is more meaningful in the research towards generalizing order statistic filters.

V. CONCLUSIONS

In order to find a more useful network structure for nonlinear signal processing, we have developed three models for representing PWL functions. As a basic scheme, the $f-\varphi$ model provides a general model for all PWL functions. This general model is suitable for a network implementation. The other two, i.e. the $f-f$ model and the $\varphi-\varphi$ model are simplifications of the $f-\varphi$ model, by attaching a constraint between the local functions and the domain partitions. For the $f-f$ model, the information about the domain partition is implicit, while for the $\varphi-\varphi$ model, the local functions are implicit. Therefore, each of them has its specialities in applications. But both are capable for representing all continuous PWL functions. With them the limitation of the canonical model is overcome. In fact, the canonical model is just the simplest special issue of the $\varphi-\varphi$ model. As the first application, the modelling of the order statistic filter and its linear extensions has been considered. This useful family of nonlinear filters is inherently PWL characterized, but it cannot be represented by the canonical model. Our approach delivers not only a unified network structure for this family, but also a way to generalize it and, furthermore, towards the unification of this specialized family in a more general class of nonlinear filters.

The logical array carries information about the topological structure of the domain partition and plays a kernel role in the models. Its properties for a given class of PWL functions, e.g. that of the family of order statistic filters, should be an interesting theme of further study.

REFERENCES

[1] I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters*, Kluwer, 1989.

[2] J.-N. Lin and R. Unbehauen, "Adaptive nonlinear digital filter with canonical piecewise-linear structure," *IEEE Trans. Circuits and Systems*, vol. 37, pp. 347-353, 1990.

[3] R. Batruni, "A multilayer neural network with piecewise-linear structure and back-propagation learning," *IEEE Trans. Neural Networks*, vol. 2, pp.395-403, 1991.

[4] L. O. Chua and S. M. Kang, "Section-wise piecewise-linear functions: Canonical representation, properties, and applications," *Proc. IEEE*, vol. 65, no. 6, pp. 915-929, 1977.

[5] J.-N. Lin and R. Unbehauen, "Canonical PWL Network and Multilayer Perceptron-like Networks: A Unified View," *Proc. IEEE Int. Symp. Circuits and Systems*, 1993, pp. 2588-2591.

[6] J. Lin, *Mapping-Netzwerke und adaptive nichtlineare Filter*, Dissertation, University Erlangen-Nürnberg, Germany.

[7] L. O. Chua and A. C. Deng, "Canonical piecewise-linear representation," *IEEE Trans. Circuits and Systems*, vol. 35, pp. 101-111, 1988.

[8] P. Shi and R. K. Ward, "OSNet: A neural network implementation of order statistic filters," *IEEE Trans. Neural Networks*, vol. 4, pp. 234-241, 1993.

[9] T. Ohtsuki, T. Fujisawa, and S. Kumagai, "Existence theorem and a solution algorithm for piecewise-linear resistor networks," *SIAM J. Math. Anal.*, vol. 8, pp. 69-99, 1977.

LOCALLY EXCITATORY GLOBALLY INHIBITORY OSCILLATOR NETWORKS: THEORY AND APPLICATION TO PATTERN SEGMENTATION

DeLiang Wang† and David Terman‡
†Department of Computer and Information Science
and Center for Cognitive Science
‡Department of Mathematics
The Ohio State University, Columbus, Ohio 43210, USA
Telephone: 614-292-6827; Fax: 614-292-2911; Email: dwang@cis.ohio-state.edu

Abstract - An novel class of locally excitatory, globally inhibitory oscillator networks (LEGION) is proposed and investigated analytically and by computer simulation. The model of each oscillator corresponds to a standard relaxation oscillator with two time scales. The network exhibits a mechanism of selective gating, whereby an oscillator jumping up to its active phase rapidly recruits the oscillators stimulated by the same pattern, while preventing other oscillators from jumping up. We show analytically that with the selective gating mechanism the network rapidly achieves both synchronization within blocks of oscillators that are stimulated by connected regions and desynchronization between different blocks. Computer simulations demonstrate LEGION's promising ability for segmenting multiple input patterns in real time. This model lays a physical foundation for the oscillatory correlation theory of feature binding, and may provide an effective computational framework for pattern segmentation and figure/ground segregation.

1. INTRODUCTION

A basic attribute of perception is its ability to group elements of a perceived scene or sensory field into coherent clusters (objects). This ability underlies perceptual processes such as figure/ground segregation, identification of objects, and separation of different objects, and it is generally known as pattern segmentation or perceptual organization. Despite the fact that humans perform it with apparent ease, the general problem of pattern segmentation remains unsolved in the engineering of sensory processing, such as computer vision and auditory processing.

Fundamental to pattern segmentation is the grouping of similar sensory features and the segregation of dissimilar ones. Theoretical investigations of brain functions and feature binding point to the mechanism of temporal correlation as a

0-7803-2026-3/94 $4.00 © 1994 IEEE

representational framework [11, 12]. In particular, the correlation theory of von der Malsburg [11] asserts that an object is represented by the temporal correlation of the firing activities of the scattered cells coding different features of the object. A natural way of encoding temporal correlation is to use neural oscillations, whereby each oscillator encodes some feature (maybe just a pixel) of an object. In this scheme, each segment (object) is represented by a group of oscillators that shows synchrony (phase-locking) of the oscillations, and different objects are represented by different groups whose oscillations are desynchronized from each other. Let us refer to this form of temporal correlation as *oscillatory correlation*. The theory of oscillatory correlation has received direct experimental support from the cell recordings in the cat visual cortex [1, 2] and other brain regions. The discovery of synchronous oscillations in the visual cortex has triggered much interest from the theoretical community in simulating the experimental results and in exploring oscillatory correlation to solve the problems of pattern segmentation (see among others [14, 4, 8, 9, 5, 7, 13]). While several demonstrate synchronization in a group of oscillators using local (lateral) connections [4, 7, 13], most of these models rely on long range connections to achieve phase synchrony. It has been pointed out that local connections in reaching synchrony may play a fundamental role in pattern segmentation since long-range connections would lead to indiscriminate segmentation [9, 13].

There are two aspects in the theory of oscillatory correlation: (1) synchronization within the same object; and (2) desynchronization between different objects. Despite intensive studies on the subject, the question of desynchronization has been hardly addressed. The lack of an efficient mechanism for desynchronization greatly limits the utility of oscillatory correlation to perceptual organization. In this paper, we propose a new class of oscillatory networks, LEGION, and show that it can rapidly achieve both synchronization within each object and desynchronization between a number of simultaneously presented objects. LEGION is composed of the following elements: (1) A new model of a basic oscillator; (2) Local excitatory connections to produce phase synchrony within each object; (3) A global inhibitor that receives inputs from the entire network and feeds back with inhibition to produce desynchronization of the oscillator groups representing different objects. In other words, the mechanism of LEGION consists of local cooperation and global competition, thus fully encoding oscillatory correlation. This surprisingly simple neural architecture may provide an elementary approach to pattern segmentation and a computational framework for perceptual organization.

2. MODEL DESCRIPTION

The building block of LEGION, a single oscillator i, is defined in the simplest form as a feedback loop between an excitatory unit x_i and an inhibitory unit y_i:

$$\frac{dx_i}{dt} = 3x_i - x_i^3 + 2 - y_i + \rho + I_i + S_i \tag{1a}$$

$$\frac{dy_i}{dt} = \varepsilon\,(\gamma(1 + tanh(x_i/\beta)) - y_i) \tag{1b}$$

where ρ denotes the amplitude of a Gaussian noise term. I_i represents external stimulation to the oscillator, and S_i denotes coupling from other oscillators in the network. The noise term is introduced both to test the robustness of the system and to actively desynchronize different input patterns. The parameter ε is chosen to be small. In this case (1), without any coupling or noise, corresponds to a standard relaxation oscillator. The x-nullcline of (1) is a cubic curve, while the y-nullcline is a sigmoid function, as shown in Fig. 1. If $I > 0$, these curves intersect along the middle branch of the cubic, and (1) is oscillatory. The periodic solution alternates between the silent and active phases of near steady state behavior. The parameter γ is introduced to control the relative times that the solution spends in these two phases. If $I < 0$, then the nullclines of (1) intersect at a stable fixed point along the left branch of the cubic. In this case the system produces no oscillation. The oscillator model (1) may be interpreted as a model of spiking behavior of a single neuron, or a mean field approximation to a network of excitatory and inhibitory neurons.

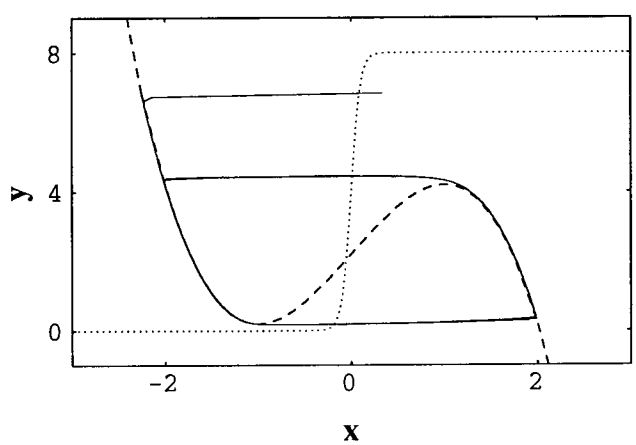


Figure 1. Nullclines and periodic orbit of a single oscillator as shown in the phase plane. The x-nullcline ($dx/dt = 0$) is shown by the dashed curve and the y-nullcline ($dy/dt = 0$) is shown by the dotted curve. In a simulation when the oscillator starts at a randomly generated point (upper middle position in the figure) in the phase plane, it quickly converged to a stable trajectory of a limit cycle. The parameters for this simulation are $I = 0.2$, $\rho = 0.02$, $\varepsilon = 0.02$, $\gamma = 4.0$, $\beta = 0.1$.

The LEGION we study here in particular is two dimensional. However, the results can easily be extended to other dimensions. Each oscillator in the LEGION is connected to only its four nearest neighbors, thus forming a 2-D grid. This is the simplest form of local connections. The global inhibitor receives excitation from each oscillator of the grid, and in turn inhibits each oscillator. This

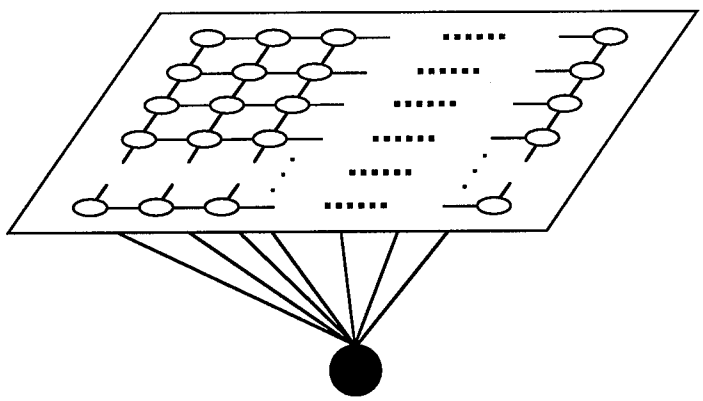


Figure 2. Architecture of a two dimensional LEGION with nearest neighbor coupling. The global inhibitor is indicated by the black circle.

architecture is shown in Fig. 2. The intuitive reason why the LEGION gives rise to pattern segmentation is the following. When multiple connected objects are mapped onto the grid, local connectivity on the grid will group together the oscillators covered by each object. This grouping will be reflected by phase synchrony within each object. The global inhibitor is introduced for desynchronizing the oscillatory responses to different objects. We assume that the coupling term S_i in (1) is given by

$$S_i = \sum_{k \in N(i)} W_{ik}\, S_\infty(x_k, \theta_x) - W_z\, S_\infty(z, \theta_{xz}) \tag{2}$$

$$S_\infty(x, \theta) = \frac{1}{1 + exp[-K(x-\theta)]} \tag{3}$$

where W_{ik} is a connection (synaptic) weight from oscillator k to oscillator i, and $N(i)$ is the set of the neighoring oscillators that connect to i. In this model, $N(i)$ is the four immediate neighbors on the 2-D grid, except on the boundaries where $N(i)$ may be either 2 or 3 immediate neighbors. θ_x is a threshold (see the sigmoid function of Eq. 3) above which an oscillator can affect its neighbors. W_z (positive) is the weight of inhibition from the global inhibitor z, whose activity is defined as

$$\frac{dz}{dt} = \phi\,(\sigma_\infty - z) \tag{4}$$

where $\sigma_\infty = 0$ if $x_i < \theta_{zx}$ for every oscillator, and $\sigma_\infty = 1$ if $x_i \geq \theta_{zx}$ for at least one oscillator i. Hence θ_{zx} represents a threshold. If the activity of every oscillator is below this threshold, then the global inhibitor will not receive any input. In this case $z \rightarrow 0$ and the oscillators will not receive any inhibition. If, on the other

hand, the activity of at least one oscillator is above the threshold θ_{zx} then, the global inhibitor will receive input. In this case $z \rightarrow 1$, and each oscillator feels inhibition when z is above the threshold θ_{zx}. The parameter ϕ determines the rate at which the inhibitor reacts to such stimulation.

In summary, once an oscillator is active, it triggers the global inhibitor. This then inhibits the entire network as described in Eq. 1. On the other hand, an active oscillator spreads its activation to its nearest neighbors, again through (1), and from them to its further neighbors. Thus, the entire dynamics of LEGION is a combination of local cooperation through excitatory coupling among neighboring oscillators and global competition via the global inhibitor. In the next section, we give a number of properties of this system.

Besides boundaries, the oscillators on the grid are basically symmetrical. Boundary conditions may cause certain distortions to the stability of synchrous oscillations. Recently, Wang [13] proposed a mechanism called *dynamic normalization* to ensure that each oscillator, whether it is in the interior or on a boundary, has equal overall connection weights from its neighbors. The dynamic normalization mechanism is adopted in the present model to form effective connections. For binary images (each pixel being either 0 or 1), the outcome of dynamic normalization is that an effective connection is established between two oscillators if and only if they are neighbors and both of them are activated by external stimulation. The network defined above can readily be applied for segmentation of binary images. For gray-level images (each pixel being in a certain value range), the following slight modification suffices to make the network applicable. An effective connection is established between two oscillators if and only if they are neighbors and the difference of their corresponding pixel values is below a certain threshold.

3. ANALYTICAL RESULTS

We have formally analyzed the LEGION. Due to space limitations, we can only list the major conclusions without proofs. The interested reader can find the details in Terman and Wang [10]. Let us refer to a *pattern* as a connected region, and a *block* be a subset of oscillators stimulated by a given pattern. The following results are about singular solutions in the sense that we formally set $\varepsilon = 0$. However, as shown in [10], the results extend to the case $\varepsilon > 0$ sufficiently small.

Theorem 1. (*Synchronization*). The parameters of the system can be chosen so that all of the oscillators in a block always jump up simultaneously (synchronize). Moreover, the rate of synchronization is exponential.

Theorem 2. (*Pattern Separation*) The parameters of the system and a constant T can be chosen to satisfy the following. If at the beginning all the oscillators of the same block synchronize with each other and the temporal distance between any two oscillators belonging to two different blocks is greater than T, then (1) Synchronization within each block is maintained; (2) The blocks activate with a fixed ordering; (3) At most one block is in its active phase at any time.

Theorem 3. (*Desynchronization*) If at the beginning all the oscillators of the system lie not too far away from each other, then the condition of Theorem 2 will be satisfied after some time. Moreover, the time it takes to satisfy the condition is no greater than N cycles, where N is the number of patterns.

The above results are true with arbitrary number of oscillators. In summary, LEGION exhibits a mechanism, referred to as *selective gating*, which can be intuitively interpreted as follows. An oscillator jumping to its active phase opens a gate to quickly recruit the oscillators of the same block due to local connections. At the same time, it closes the gate to the oscillators of different blocks. Moreover, segmentation of different patterns is achieved very rapidly in terms of oscillation cycles.

4. COMPUTER SIMULATION

To illustrate how LEGION is used for pattern segmentation, we have simulated a 20x20 LEGION as defined by (1)-(4). We arbitrarily selected four objects (patterns): two **O**'s, one **H**, and one **I**; and they form the word **OHIO**. These patterns were simultaneously presented to the system as shown in Figure 3A. Each pattern is a connected region, but no two patterns are connected to each other.

All the oscillators stimulated (covered) by the objects received an external input I = 0.2, while the others have I = -0.02. Thus the oscillators under stimulation become oscillatory, while those without stimulation remain silent. The amplitude ρ of the Gaussian noise is set to 0.02. Thus, compared to the external input, a 10% noise is included in every oscillator. Dynamic normalization results in that only two neighboring oscillators stimulated by a single pattern have an effective connection. The differential equations were solved numerically with the following parameter values: $\varepsilon = 0.02$, $\phi = 3.0$; $\gamma = 6.0$, $\beta = 0.1$, $K = 50$, θ_x = -0.5, and $\theta_{zx} = \theta_{xz} = 0.1$. The total effective connections were normalized to 6.0. The results described below were robust to considerable changes in the parameters. The phases of all the oscillators on the grid were randomly initialized.

Fig. 3B-3F shows the instantaneous activity (snapshot) of the network at various stages of dynamic evolution. The diameter of each black circle represents the x activity of the corresponding oscillator. That is, if the range of x values of all the oscillators are given by x_{min} and x_{max}, then the diameter of the black circle corresponding to an oscillator is proportional to $(x-x_{min})/(x_{max}-x_{min})$. Fig. 3B shows a snapshot of the network a few steps after the beginning of the simulation. In Fig. 3B, the activities of the oscillators were largely random. Fig. 3C shows a snapshot after the system had evolved for a short time period. One can clearly see the effect of grouping and segmentation: all the oscillators belonging to the left **O** were entrained and had large activities. At the same time, the oscillators stimulated by the other three patterns had very small activities. Thus the left **O** was segmented from the rest of the input. A short time later, as shown in Fig. 3D, the oscillators stimulated by the right **O** reached high values and were separated from the rest of the input. Fig. 3E shows another snapshot after Fig. 3D. At this time, pattern **I** had its turn to be activated and separated from the rest of the input.

Finally in Fig. 3F, the oscillators representing **H** were active and the rest of the input remained silent. This successive "pop-out" of the objects continued in a stable periodic fashion. To provide a complete picture of dynamic evolution, Fig. 3G shows the temporal evolution of each oscillator. Since the oscillators receiving no external input were inactive during the entire simulation process, they were excluded from the display in Fig. 3G. The activities of the oscillators stimulated by each object are combined together in the figure. Thus, if they are synchronized, they appear like a single oscillator. In Fig. 3G, the four upper traces represent the activities of the four oscillator blocks, and the bottom trace represents the activity of the global inhibitor. The synchronized oscillations within each object are clearly shown within just three cycles of dynamic evolution.

The exact shapes and positions of the patterns in Fig. 3 do not matter for pattern segmentation. In fact, this 2-D LEGION provides a general solution to segmentation of planar connected patterns.

5. DISCUSSION

Besides neural plausibility, oscillatory correlation has a unique feature as an computational approach to the engineering of pattern segmentation and figure/ground segregation. Due to the nature of oscillations, no single object can dominate and suppress the perception of the rest of the image permanently. The current dominant object has to give way to other objects being suppressed, and let them have a chance to be spotted. Although at most one object can dominant at any time instant, due to rapid oscillations, a number of objects can be activated over a short time period. This intrinsic dynamic process provides a natural and reliable representation of multiple segmented patterns.

The basic principles of selective gating are established for LEGION with lateral connections beyond nearest neighbors. Indeed, in terms of synchronization, more distant connections even help expedite phase entrainment. In this sense, synchronization with all-to-all connections is an extreme case of our system. With nearest-neighbor connectivity (Fig. 2), any isolated part of an image is considered as a segment. In an noisy image with many tiny regions, segmentation would result in too many small fragments. More distant connections would also provide a solution to this problem. Lateral connections typically take on the form of Gaussian distribution, with the connection strength between two oscillators falling off exponentially. Since global inhibition is superimposed to local excitation, two oscillators positively coupled may be desynchronized if global inhibition is strong enough. Thus, it is unlikely that all objects in an image form a single segment as the result of extended connections.

Due to its critical importance for computer vision, pattern segmentation, or perceptual organization as known in computer vision, has been studied quite extensively. Many techniques have been proposed in the past [3, 6]. Despite these techniques, as pointed out by Haralick and Shapiro [3], there is no underlying theory of image segmentation, and the techniques tend to be adhoc and emphasize some aspects while ignoring others. Compared to the traditional techniques for pattern segmentation, the oscillatory correlation approach offers many unique advantages. The dynamical process is inherently parallel. While conventional

computer vision algorithms are based on descriptive criteria and many adhoc heuristics, LEGION as exemplified in this paper performs computations based on only connections and oscillatory dynamics. The organizational simplicity renders LEGION particularly feasible for VLSI implementation. Also, continuous-time dynamics allows real time processing, desired by many engineering applications.

ACKNOWLEDGMENTS. DLW is supported in part by the NSF grant IRI-9211419 and the ONR grant N00014-93-1-0335. DT is supported in part by the NSF grant DMS-9203299LE.

REFERENCES

[1] R. Eckhorn, et al., "Coherent oscillations: A mechanism of feature linking in the visual cortex?" Biol. Cybern., vol. 60, pp. 121-130, 1988.

[2] C.M. Gray, P. König, A.K. Engel, and W. Singer, "Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties," Nature, vol. 338, pp. 334-337, 1989.

[3] R.M. Haralick and L.G. Shapiro, "Image segmentation techniques," Comput. Graphics Image Process., vol. 29, pp. 100-132, 1985.

[4] P. König and T.B. Schillen, "Stimulus-dependent assembly formation of oscillatory responses: I. Synchronization," Neural Comput., vol. 3, pp. 155-166, 1991.

[5] T. Murata and H. Shimizu, "Oscillatory binocular system and temporal segmentation of stereoscopic depth surfaces," Biol. Cybern., vol. 68, pp. 381-390, 1993.

[6] S. Sarkar and K.L. Boyer, "Perceptual organization in computer vision: a review and a proposal for a classificatory structure," IEEE Trans. Syst. Man Cybern., vol. 23, 382-399, 1993.

[7] D. Somers, and N. Kopell, "Rapid synchronization through fast threshold modulation," Biol. Cybern., vol. 68, pp. 393-407, 1993.

[8] H. Sompolinsky, D. Golomb, and D. Kleinfeld, "Cooperative dynamics in visual processing," Phys. Rev. A, vol. 43, pp. 6990-7011, 1991.

[9] O. Sporns, G. Tononi, and G.M. Edelman, "Modeling perceptual grouping and figure-ground segregation by means of active reentrant connections," Proc. Natl. Acad. Sci. USA, vol. 88, pp. 129-133, 1991.

[10] D. Terman and D.L. Wang, "Global competition and local cooperation in a network of neural oscillators," Submitted, 1993.

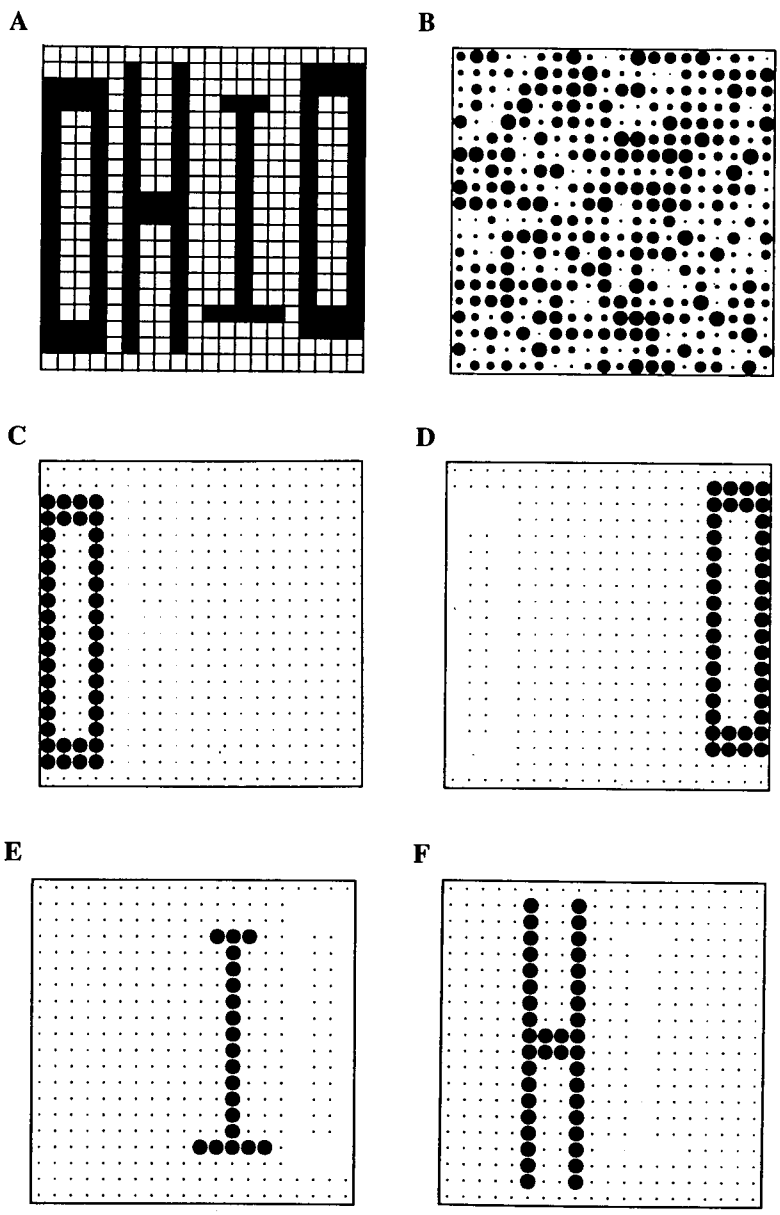
[11] C. von der Malsburg, "The correlation theory of brain functions," Internal Report 81-2, Max-Planck-Institut for Biophysical Chemistry, Göttingen, FRG, 1981.

[12] C. von der Malsburg and W. Schneider, "A neural cocktail-party processor," Biol. Cybern., vol. 54, pp. 29-40, 1986.

[13] D.L. Wang, "Modeling global synchrony in the visual cortex by locally coupled neural oscillators," Proc. 15th Ann. Conf. Cognit. Sci. Soc., 1993, pp. 1058-1063. For a more extended version, see D.L. Wang, "Emergent

synchrony in locally coupled neural oscillators," IEEE Trans. on Neural Networks, in press.

[14] D.L. Wang, J. Buhmann, and C. von der Malsburg, "Segmentation in associative memory," Neural Comput., vol. 2, pp. 94-106, 1990.



G

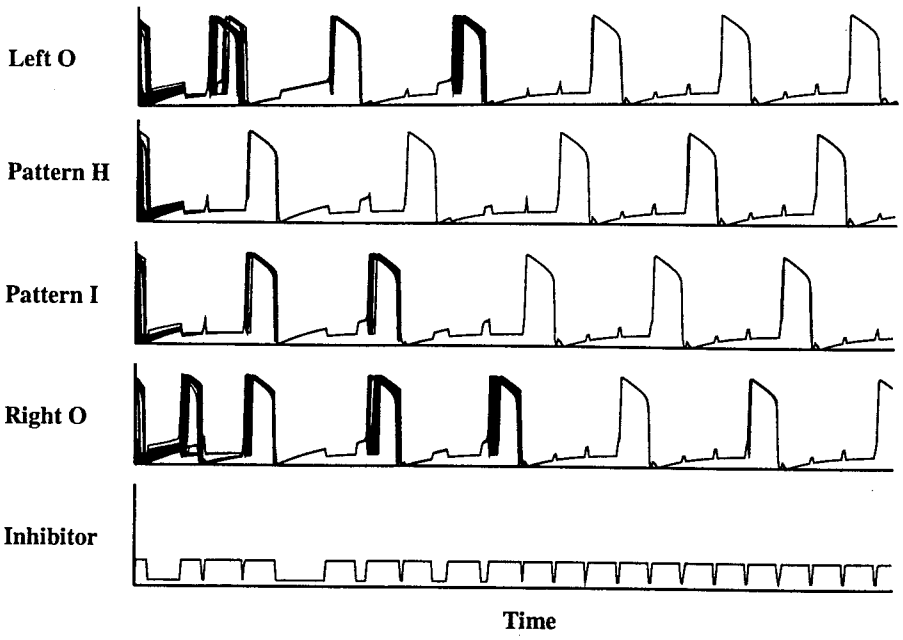


Figure 3. **A** An image composed of four patterns which were presented (mapped) to a 20x20 grid of oscillators. **B** A snapshot of the activities of the oscillator grid at the beginning of dynamic evolution. **C** A snapshot taken shortly after the beginning. **D** Another snapshot taken shortly after **C**. **E** Another snapshot taken shortly after **D**. **F** Another snapshot taken shortly after **E**. **G** The upper four traces show the combined temporal activities of the oscillator blocks representing the four patterns, respectively, and the bottom trace shows the temporal activity of the global inhibitor. The simulation took 8,000 integration steps.

A UNIFYING VIEW OF SOME TRAINING ALGORITHMS FOR MULTILAYER PERCEPTRONS WITH FIR FILTER SYNAPSES

Andrew Back*, Eric A. Wan**, Steve Lawrence*, Ah Chung Tsoi*
*Department of Electrical and Computer Engineering,
University of Queensland, St. Lucia, Queensland 4072. Australia.
**Department of Electrical Engineering and Applied Physics
Oregon Graduate Institute of Science & Technology
P.O. Box 91000, Portland, Oregon 97291, USA
back@s1.elec.uq.oz.au, ericwan@eeap.ogi.edu
lawrence@s1.elec.uq.oz.au, act@s1.elec.uq.oz.au

Abstract– Recent interest has come about in deriving various neural network architectures for modelling time-dependent signals. A number of algorithms have been published for multilayer perceptrons with synapses described by finite impulse response (FIR) and infinite impulse response (IIR) filters (the latter case is also known as *Locally Recurrent Globally Feedforward Networks*). The derivations of these algorithms have used different approaches in calculating the gradients, and in this paper we present a short, but unifying account of how these different algorithms compare for the FIR case, both in derivation, and performance. A new algorithm is subsequently presented. In this paper, results are compared for the Mackey-Glass chaotic time series against a number of other methods including a standard multilayer perceptron, and a local approximation method.

INTRODUCTION

As a means of capturing time-dependent signals in a nonlinear framework, multilayer perceptrons (MLPs) with synapses described by filters have recently been proposed [1, 2, 17]. These approaches replace the traditional scalar synaptic weights with finite impulse response (FIR) filters commonly used in digital filter theory. The architecture can be considered an extension of earlier work in which time delays were incorporated as a means of capturing time-dependent input information. For example, in the Time Delay Neural Network used by Waibel et al [20], the outputs of a layer in a feedforward network are buffered several time steps and then fed fully connected to the next layer. Lapedes and Farber's [10] use of a time-window as the input to a multilayer network for applications in time series prediction is equivalent to one layer of time delay synapses at the input. FIR networks provide a more general model for distributed time representations.

An algorithm for training networks having FIR synapses was first published by Wan [17]. A similar algorithm for the same network as well as the case for IIR synapses was published by Back and Tsoi [1, 2]. We focus on these algorithms in this paper, comparing their derivations and presenting a brief, but unifying view of them. Related work which has been presented in [4, 6, 7, 11] and [14] among others, will not be considered here. Our aim is to compare the forms of the training algorithms, and to provide an indication of how they perform on some practical prediction problems. In this brief summary, we show only one set of results, the Mackey-Glass chaotic time series which allows us to easily highlight the differences in performances of various methodologies for prediction of nonlinear time series.

The network architecture is defined below:

Definition 1. *An FIR MLP of size (L, n_w) with $N_0, N_1, ..., N_L$ nodes per layer, is defined by*

$$z_k^l(t) = f\left(\hat{x}_k^l(t)\right) \tag{1}$$

0-7803-2026-3/94 $4.00 © 1994 IEEE

$$\hat{x}_k^l(t) = \sum_{i=1}^{N_l} \hat{y}_{ik}^l(t) \tag{2}$$

$$\hat{y}_{ik}^l(t) = c_{ik}^l y_{ik}^l(t) \tag{3}$$

$$y_{ik}^l(t) = W_{ik}^l(q^{-1}) z_i^{l-1}(t), \tag{4}$$

where each neuron i in layer l has an output at time t of $z_i^l(t)$; a layer consists of N_l neurons ($l = 0$ denotes the input layer, and $l = L$ denotes the output layer, $z_{N_l}^l = 1.0$ may be used for a bias); $\hat{y}_{ik}^l(t)$ is the output of a synapse connecting from neuron i in the previous layer to neuron k in layer l; c_{ik}^l is a synaptic gain; and $f(\cdot)$ is a sigmoid function typically evaluated as $\tanh(\cdot)$. *An FIR synapse is represented by $W_{ik}^l(q^{-1}) = \sum_{j=0}^{n_w} w_{ikj}^l(q^{-j})$ where w_{ikj}^l correspond to the variable coefficients, and q^{-1} is a delay operator ($q^{-1}z(t) \triangleq z(t-1)$), and n_w is the number of delays.*

The algorithms use first order stochastic gradient descent to minimize an error function. We define the instantaneous performance criteria

$$\mathcal{E}(t) = \frac{1}{2}\sum_{k=1}^{N_L} e_k^2(t) = \frac{1}{2}\sum_{k=1}^{N_L} \left(d_k(t) - z_k^L(t)\right)^2 \tag{5}$$

where $d_k(t)$ is the desired output at time t, and the sum is taken over the output neurons. The total error or cost is given by summing the instantaneous error over all T time steps in a training sequence

$$\mathcal{E}_T = \sum_{t=0}^{T} \mathcal{E}(t). \tag{6}$$

The different forms of the training algorithms for FIR networks differ in the manner in which the gradients are calculated and on whether the instantaneous or total error is used in the calculations.

GRADIENT COMPUTATION IN FIR SYNAPSES USING AN INSTANTANEOUS COST FUNCTION

An algorithm for updating the weights in an FIR network may be obtained by considering the instantaneous error $\mathcal{E}(t)$ [1, 17]. The weight changes can be adjusted using a simple gradient method

$$w_{ikj}^l(t+1) = w_{ikj}^l(t) + \Delta w_{ikj}^l(t) \tag{7}$$

$$c_{ik}^l(t+1) = c_{ik}^l(t) + \Delta c_{ik}^l(t) \tag{8}$$

$$\Delta w_{ikj}^l(t) = -\eta \frac{\partial \mathcal{E}(t)}{\partial w_{ikj}^l(t)} \tag{9}$$

$$= -\eta \frac{\partial \mathcal{E}(t)}{\partial \hat{x}_k^l(t)} \frac{\partial \hat{x}_k^l(t)}{\partial w_{ikj}^l(t)} \tag{10}$$

$$\Delta c_{ik}^l(t) = -\eta \frac{\partial \mathcal{E}(t)}{\partial c_{ik}^l(t)}, \tag{11}$$

$$= -\eta \frac{\partial \mathcal{E}(t)}{\partial \hat{x}_k^l(t)} \frac{\partial \hat{x}_k^l(t)}{\partial c_{ikj}^l(t)} \tag{12}$$

where η is the learning rate. A derivation of the partial terms is given in [2]. In the derivation, it is necessary to define a secondary variable $\delta_k^l(t) = -\frac{\partial \mathcal{E}(t)}{\partial \hat{x}_k^l(t)}$. If we consider only the gradient at the exact time t, then we have

Algorithm IC-1 Instantaneous Cost - Instantaneous Gradient

$$\delta_k^l(t) = f'\left(\hat{x}_k^l(t)\right) \sum_{m=1}^{N_{l+1}} \delta_m^{l+1}(t) c_{km}^{l+1} w_{km0}^{l+1}. \tag{13}$$

This can be considered an approximate instantaneous gradient. This is the method adopted in [1, 2]. Note the δ terms are essentially calculated using standard backpropagation through the w_{km0} taps; the rest of the coefficients in the FIR synapse are ignored, since we only assume a relationship between $z_k^l(t)$ and $\hat{y}_{km}^{l+1}(t)$ instantaneously at time t.

A different form is achieved if we calculate the gradient over a short time period by delaying the calculation of the gradient until all contributions from feedforward delay elements can be combined.

Algorithm IC-2 Instantaneous Cost - Accumulated Gradient

$$\begin{aligned}
\delta_k^l(t) &= f'\left(\hat{x}_k^l(t)\right) \sum_{m=1}^{N_{l+1}} \sum_{d=0}^{n_w} c_{km}^{l+1} w_{kmd}^{l+1}(q^{-d}) \delta_m^{l+1}(t) \\
&= f'\left(\hat{x}_k^l(t)\right) \sum_{m=1}^{N_{l+1}} \sum_{d=0}^{n_w} c_{km}^{l+1} w_{kmd}^{l+1} \delta_m^{l+1}(t-d) \\
&= f'\left(\hat{x}_k^l(t)\right) \sum_{m=1}^{N_{l+1}} c_{km}^{l+1} W_{km}^{l+1}(q^{-1}) \delta_m^{l+1}(t).
\end{aligned} \tag{14}$$

This is similar to the second algorithm proposed by Wan in [17] (discussed in a subsequent section of this paper). In this case, we have the backpropagated error being obtained from a backward *filter* and all coefficients in the FIR synapse have an influence on the δ value.

For both cases, the final update equations for the FIR MLP are

$$w_{ikj}^l(t+1) = w_{ikj}^l(t) + \eta \delta_k^l(t) c_{ik}^l z_i^{l-1}(t-j) \tag{15}$$

$$c_{ik}^l(t+1) = c_{ik}^l(t) + \eta \delta_k^l(t) W_{ik}^l(q^{-1}) z_i^{l-1}(t), \tag{16}$$

where $\delta_k^l(t)$ may be computed by one of the two methods described above. We will discuss the relative performance of the different methods in the results section.

GRADIENT COMPUTATION IN FIR SYNAPSES USING A TOTAL COST FUNCTION

This section reviews the algorithms derived by Wan in [17, 18]. Gradient adaptation is based on the *total* squared error over the entire sequence of inputs, as opposed to the instantaneous error measure used previously. This should not be confused with the fact that in all cases, we use an on-line updating scheme which makes use of error measurement computed for that particular time instant.

Fundamentally, the weight changes in (9) and (11) are replaced by

$$\Delta w_{ikj}^l(t) = -\eta \frac{\partial \mathcal{E}_T}{\partial w_{ikj}^l(t)}, \qquad \Delta c_{ik}^l(t) = -\eta \frac{\partial \mathcal{E}_T}{\partial c_{ik}^l(t)}. \tag{17}$$

We have simply substituted the total error $\mathcal{E}_T$ for the instantaneous error $\mathcal{E}(t)$. In this case an expression for δ is obtained by maintaining the dependence over all values of the input sequence. Derivations given in [19] leads to the following algorithms. Algorithm TC-1 is very inefficient for networks with more than two layers. Algorithm TC-2 on the other hand, uses the same update equations (15) and (16) as before. In this case we derive a slightly different equation for the δ term.

Algorithm TC-1 Total Cost - Instantaneous Gradient

$$\frac{\partial \mathcal{E}(t)}{\partial w_{ikj}^l(t)} = \sum_{n=0}^{n_w} c_{ik}^l z_i^{l-1}(t-n) f'(\hat{x}_k^l(t-n)) \sum_{m=1}^{N_{l+1}} \delta_m^{l+1}(t) c_{km}^{l+1} w_{kmn}^{l+1}(t) \quad (18)$$

$$\frac{\partial \mathcal{E}(t)}{\partial c_{ik}^l(t)} = \sum_{n=0}^{n_w} y_i^l(t-n) f'(\hat{x}_k^l(t-n)) \sum_{m=1}^{N_{l+1}} \delta_m^{l+1}(t) c_{km}^{l+1}(t) w_{kmn}^{l+1}(t) \quad (19)$$

Algorithm TC-2 Total Cost - Temporal Backpropagation

$$\delta_k^l(t) = f'(\hat{x}_k^l(t)) \sum_{m=1}^{N_{l+1}} c_{km}^{l+1} W_{km}^{l+1}(q^{+1}) \delta_m^{l+1}(t). \quad (20)$$

Note that in this case, Algorithm TC-2 needs to be delayed n_w time steps to maintain causality. It can be seen as very similar to Algorithm IC-2, though the evaluation of δ and w terms occurs at different times (cf. (14)). In this algorithm, the backward filtering comes about directly as a result of using the *total* cost function over time, thereby necessitating the accumulation of gradient information. In Algorithm IC-2, gradient computations are accumulated over time after initially considering an instantaneous gradient.

SIMULATION RESULTS

As a means of comparing the different algorithms, we present some preliminary results for the application of the neural network algorithms to some time series prediction tasks. In this extended summary, only the results obtained in modelling the Mackey-Glass delay-differential equation[1] are presented, due to the widespread interest in using it as a benchmark. In the full paper, results pertaining to other time series prediction problems are considered, specifically:

1. Prediction of Mackey-Glass chaotic time series,
2. Prediction of Laser data as used in Santa Fe Time Series Prediction Competition.
3. Nonlinear speech prediction
4. Financial time series prediction

The algorithms discussed above are each trained on the data for the Mackey-Glass time series. In each case, multiple simulations were performed and the results averaged to obtain a reasonable indication of the networks performance. After some initial testing to determine suitable learning rates, we selected a specific learning rate which remained the same for each network when trained on a particular time series.

[1]The Mackey-Glass [12] equation is described by $\dot{x}(t) = -bx(t) + \frac{ax(t-\tau)}{1+x(t-\tau)^{10}}$, where τ=30, a=0.2, and b=0.1.

Our aim was to test two basic approaches to time series prediction, namely, the traditional approach of using past values of the time series directly, and secondly, the approach of embedding the time series in a phase space, and using the delay coordinates as the vector of inputs[2]. This approach, proposed by Takens [13, 15] involves sampling the time series at some *delayed* time values to create a *delay coordinate* vector. This is sometimes referred to as a phase space.

As a means of comparing each algorithm, we benchmark their relative performances againsta windowed input MLP, and a local approximation method developed by Casdagli [5] (a version of the nearest neighbor method). Obviously, there are many variations in which this could have been done. Our intent is to provide a reasonable means of quickly assessing the performance of these algorithms which may provide a starting point for anyone interested in considering them further.

The work we present here consists of benchmarking each of the above algorithms on some representative time series as listed above. We consider three main cases:

- Single-step ahead prediction using a vector of past inputs (spaced one time unit apart)
- Multi-step ahead prediction using a delay coordinate (Takens) vector of past inputs (spaced τ time units apart, where τ is the delay parameter)
- Iterated-prediction problem, using past outputs of the model as inputs for future predictions[3].

In the simulations performed, we used delay coordinate vectors with 6 elements (D=6), and a time delay of $\tau = 6$. The order of FIR filters was $n_w = 5$. The results shown in Figures 1 and 2 are for the multi-step prediction problem, and the iterated-prediction problem using a prediction time-step of 6.

These results are interesting, in that they show, for the problem at hand, the FIR MLP structure appears to be better able to model the dynamics of the chaotic time-series. The multi-step ahead prediction performance for the test set shows that each model is able to do quite well. However, when we consider the more difficult problem of iterated-prediction, we observe that the networks with FIR synapses perform much better (see for example, the generated phase space plots in fig. 3).

It is interesting also to observe the different behaviours of the algorithms possible for the FIR MLP model, indicating that while they may be better than other methods generally, there are differences between how the algorithms operate in practice. Results on the other simulation problems will be presented at the workshop.

CONCLUSIONS

The aim of this brief note was to clarify some of the issues in calculating the gradients for multilayer perceptrons with FIR synapses. This contributes to a further understanding of these types of network architectures. Results in using these networks have shown promise for a variety of nonlinear signal prediction tasks and we look forward to continued activity in this area.

[2]In our models, we only have a single input. However this is equivalent to the case where, for example, a linear predictor or multilayer perceptron has a window of inputs. In our case, the window of each filter exists in each synapse already.

[3]Our approach is to allow the models to recursively predict further and further into the future, based on the initial predictions obtained. Therefore, if we allow the model to predict 6 time-steps into the future, and we wish to see how it performs out to 400 time-steps, we allow the model to use its shorter predictions to "bootstrap" itself out. This follows the conventions adopted by Lapedes and Farber [10] and Stokbro and Umberger [16].

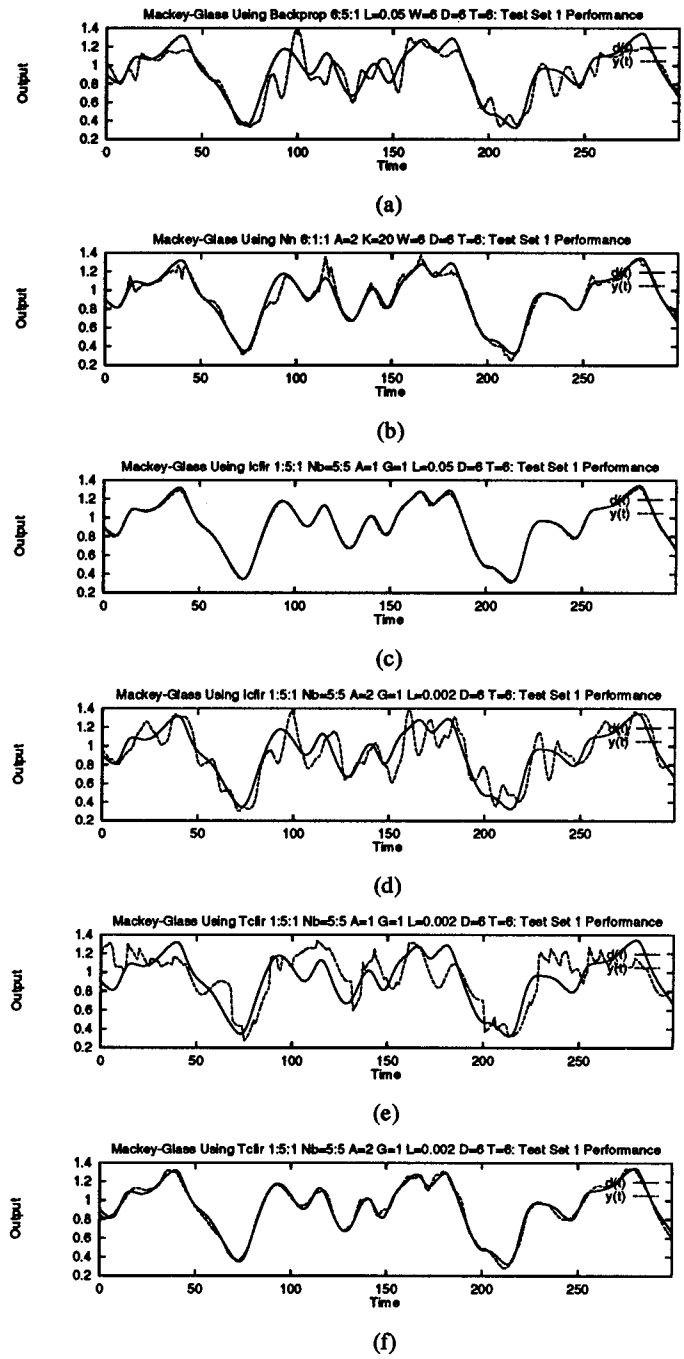


Figure 1: Test set performance on 6-step ahead prediction of the Mackey-Glass chaotic time series ($T = 30$). (a) Backpropagation (b) Nearest Neighbour ($k = 20$) (c) Algorithm IC1 (d) Algorithm IC2 (e) Algorithm TC1 (f) Algorithm TC2. (G=1 indicates synaptic gain is used, D is embedding delay, T is prediction time-step, W is input window (backpropagation only), L is learning rate, A is algorithm, Nb is FIR filter order).

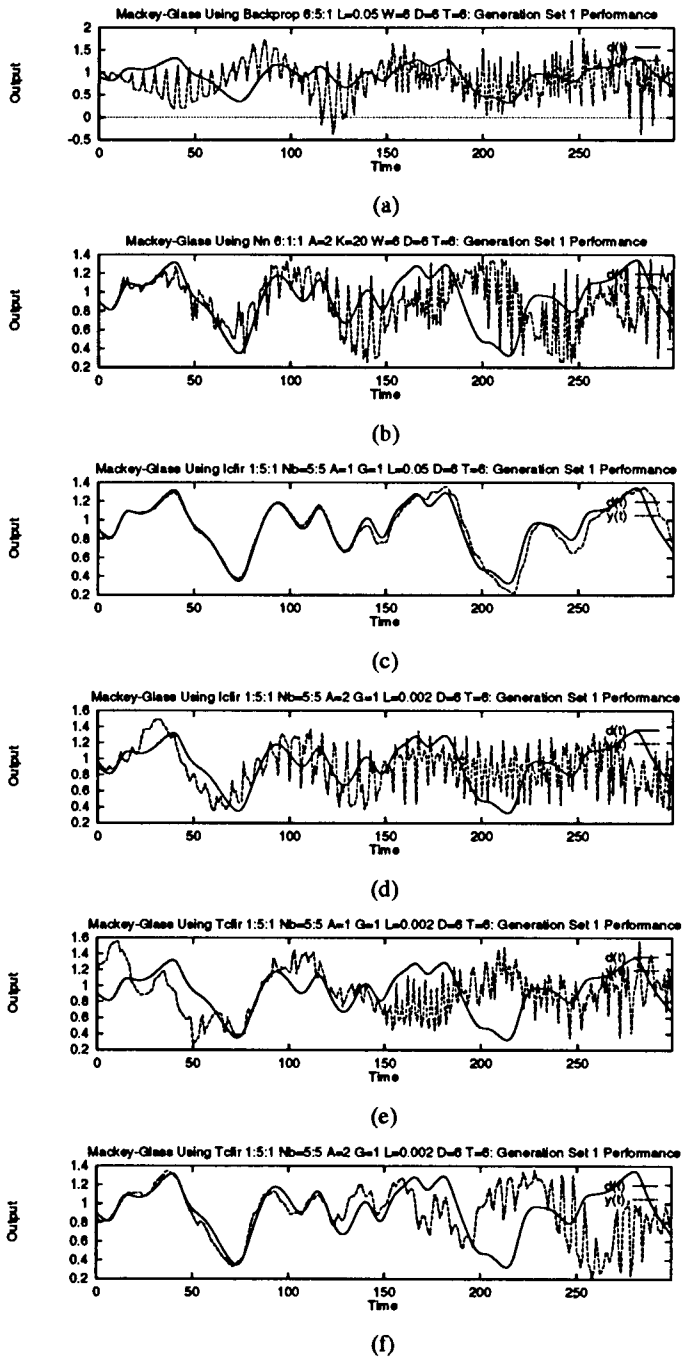


Figure 2: Iterated prediction performance on Mackey-Glass chaotic time series ($T = 30$). (a) Backpropagation (b) Nearest Neighbour ($k = 20$) (c) Algorithm IC1 (d) Algorithm IC2 (e) Algorithm TC1 (f) Algorithm TC2.

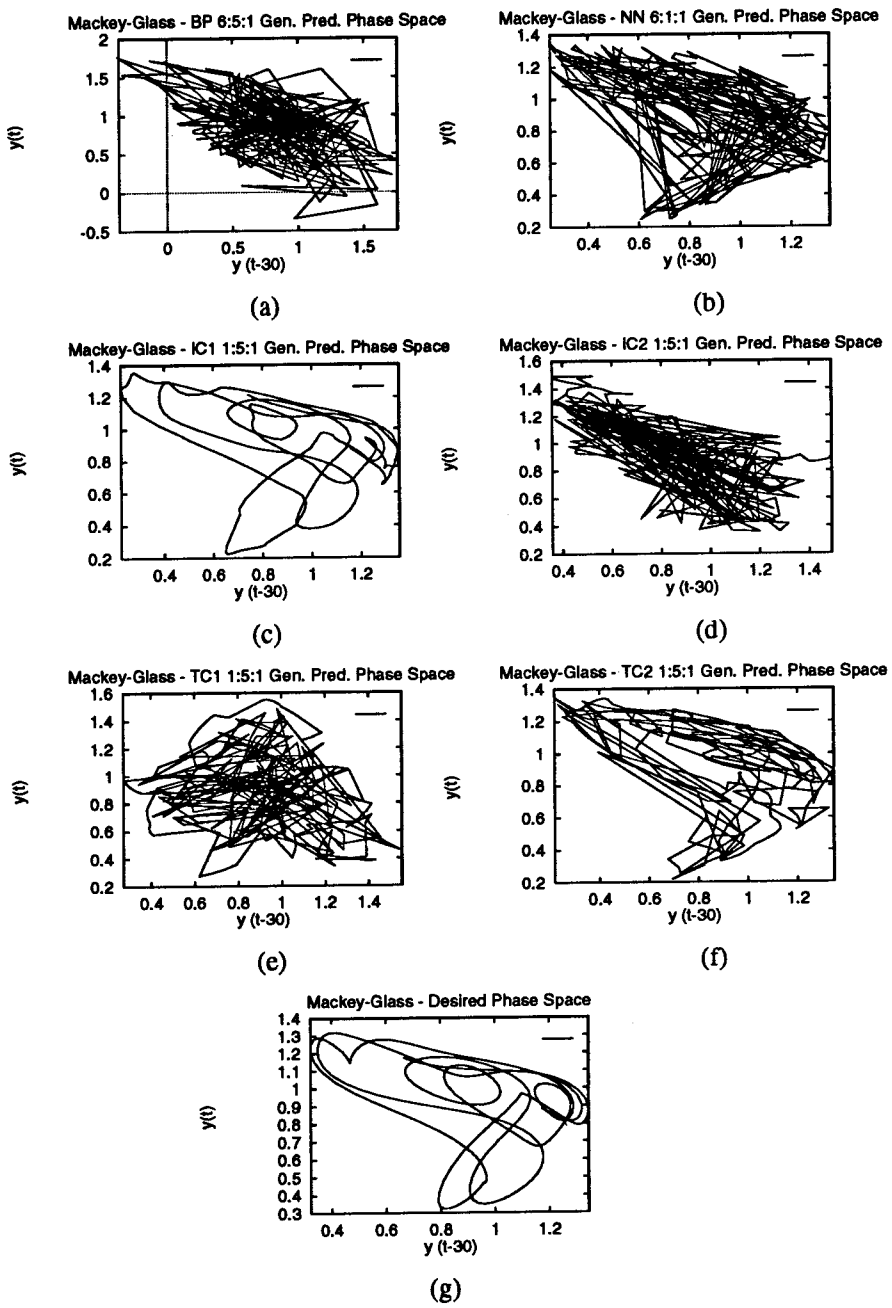


Figure 3: Phase space generation performance plotting $y(t)$ vs. $y(t-30)$ for Mackey-Glass chaotic time series ($T = 30$). (a) Backpropagation (b) Nearest Neighbour ($k = 20$) (c) Algorithm IC1 (d) Algorithm IC2 (e) Algorithm TC1 (f) Algorithm TC2. (g) Desired phase space.

Acknowledgements.

The first author acknowledges financial support from the Australian Research Council. The third author acknowledges support from the Australian Research Council and Australian Telecommunications and Electronics Research Board. The fourth author acknowledges partial support from the Australian Research Council.

References

[1] Back, A.D. and Tsoi, A.C., "A Time Series Modelling Methodology Using FIR and IIR Synapses", *Proc. Workshop on Neural Networks for Statistical and Economic Data*, Dublin, DOSES, Statistical Office of European Communities, F. Murtagh (Ed.), pp. 187-194, 1990.

[2] Back, A.D. and Tsoi,A.C., "FIR and IIR Synapses, a New Neural Network Architecture for Time Series Modelling". *Neural Computation*, vol 3. no. 3, pp. 375-385, 1991.

[3] Back, A.D. and Tsoi, A.C., "An Adaptive Lattice Architecture for Dynamic Multilayer Perceptrons", *Neural Computation*, Vol 4, No. 6, pp. 922-931, 1992.

[4] Bengio, Y. De Mori, R., Gori, M. "Learning the dynamic nature of s peech with backpropagation for sequences". *Pattern recognition Letters*. Vol. 13, pp 375 - 385, 1992.

[5] M. Casdagli, "Chaos and Deterministic versus Stochastic Non-linear Modelling", J. R. Statist. Soc. B, 1991, 54, No. 2, pp. 303-328.

[6] P. Frasconi, M. Gori and G. Soda, "Local Feedback Multilayered Networks", *Neural Computation*, vol. 4, no. 1, 1992.

[7] Gori, M., Bengio, Y., Mori, R.D. "BPS: a learning algorithm for capturing the dynamic nature of speech". *Intern. Joint Conf on Neural Networks*, Vol II, pp 417 - 423, 1989.

[8] N.A. Gershenfeld, and A.S. Weigend, "The Future of Time Series: Learning and Understanding", in Time Series Prediction: Forecasting the Future and Understanding the Past, Eds. A.S Weigend, and N.A. Gershenfeld, Addison-Wesley: Reading MA, 1993.

[9] U. Hübner, C.O. Weiss, N.B. Abraham, and D. Tang, "Lorenz-like Chaos in NH_3-FIR Lasers", in Time Series Prediction: Forecasting the Future and Understanding the Past, Eds. A.S Weigend, and N.A. Gershenfeld, Addison-Wesley: Reading MA, 1993.

[10] Lapedes, A. and Farber, R., "Nonlinear Signal Processing using Neural Networks: Prediction and System modelling", Tech Report LA-UR87-2662, Los Alamos National Laboratory, 1987.

[11] Leighton, R.R. and Conrath, B.C., "The Autoregressive Backpropagation Algorithm", Proc. Int. Joint Conf. Neural Networks, 1991.

[12] Mackey, M.C., and Glass, L., "Oscillation and Chaos in Physiological Control Systems", Science, vol. 197, pp. 287, 1977.

[13] Packard, N., Crutchfield, J., Farmer, D., and Shaw, R., "Geometry from time series", Phys. Rev. Lett., vol 45., pp. 712-716.

[14] Poddar, P. Unnikrishnan, K.P. "Memory neuron networks: A Prolegomenon". General Motors Research Laboratories Report GMR-7493, October 21, 1991.

[15] Takens F., "Detecting Strange Attractors in Turbulence", Lecture Notes in Math., vol 898, Springer-Verlag, 1981.

[16] Stokbro K. and Umberger D.K., "Forecasting with Weighted Maps", in Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity, Proc. Vol. XII, Eds. M. Cadagli, and S. Eubanks. Addison-Wesley, 1992.

[17] Wan, E.A., "Temporal backpropagation for FIR neural networks", *Proc. Int. Joint Conf. Neural Networks*, San Diego, June 1990, pp I 575-580.

[18] Wan, E.A., "Time Series Prediction by Using a Connectionist Network with Internal Delay Lines'", in A. Weigend and N. Gershenfeld, eds., *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, pages 195-218, 1994.

[19] Wan, E.A., "Finite Impulse Response Neural Networks with Applications in Time Series Prediction", *PhD Dissertation*, Stanford University, November, 1993.

[20] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K., "Phoneme recognition using time-delay neural networks", *IEEE Trans. Acoust., Speech, Signal Processing*, vol ASSP-37, March, 1989.

SPECTRAL FEATURE EXTRACTION USING POISSON MOMENTS

Samel Çelebi, Jose C. Principe
Computational Neuroengineering Lab. CSE447
University of Florida, Gainesville FL32611, USA
E-Mail: celebi@synapse.ee.ufl.edu, principe@synapse.ee.ufl.edu

Abstract. We propose to use the *Gamma filter* [1] as a feature extractor for the preprocessing of speech signals. Gamma filter which can be implemented as a cascade of identical first order lowpass filters generates at its taps the *Poisson Moments* of an input signal. These moments carry spectral information about the recent history of the input signal. They can be used to construct time-frequency representations as an alternative to the conventional methods of *short term Fourier transform, cepstrum*, etc. In this study it is shown that when the *time scale* of the Gamma filter is chosen properly, the Poisson moments correspond to the Taylor's series expansion coefficients of the input signal spectra. The appeal of the proposed method comes from the fact that in the analog domain the moments are available as a continuous time electrical signal and can be physically measured, rather than computed off-line by a digital computer. With this convenience, the speed of the discrete time processor following the preprocessor is independent of the highest frequency of the input signal, but is constrained with the stationarity duration of the signal.

INTRODUCTION

Classification of temporal patterns is one of the areas where artificial neural networks (ANNs) are frequently utilized. Speech recognition is a special case to that problem. In order to simplify the classification task undertaken by an ANN preprocessing of the temporal pattern is vital. The goal of the preprocessing should be to capture the features of the pattern and to express them in a low dimensional space. If this is achieved, then a big deal of computational and structural burden over the neural network can be removed.

One method suggested for the preprocessing of speech signals is the *Focused Gamma Network* [2][3]. This is a *generalized feedforward structure* with adjustable feedback which is responsible for changing the *time scale* (or the memory depth) of the preprocessor. Adjusting the time scale allows one to *focus* the representation space on the signal of interest such that a low dimensional, but a faithful representa-

0-7803-2026-3/94 $4.00 © 1994 IEEE

tion is obtained. Well known *Time Delay Neural Network* (TDNN) [4] is a special case of the Gamma Network where the time scale is frozen to be unity.

In their isolated word speech recognition task Tracey and Principe [2] showed that the Gamma Network is superior to TDNN both in terms of the size of the neural network required and the time it took to learn the given patterns. In this paper we analyze the Gamma Network and show that the features fed into the ANN are basically the Taylor's series expansion coefficients of the recent speech spectra. Taking into account the practicality of obtaining these features, a time-frequency representation can easily be constructed by concatenating the feature vectors of different times together. An analog implementation of the filter can be pursued in analog VLSI. Since the moment vectors are obtained in the analog domain, a digital processor that operates on these vectors is not bound by the Nyquist rate of the input signal, but by the rate the moments vary.

POISSON MOMENTS

Fairman and Shen [5] proposed that a distribution f(t) can be expanded in terms of the derivatives of Dirac's delta function as follows

$$f(t) = \sum_{i=0}^{\infty} f_i(t_o)\, e^{-\lambda(t-t_o)} \delta^{(i)}(t-t_o) \tag{1}$$

$f_i(t_o)$ is called the ith *Poisson Moment* of f(t) at $t=t_o$. It is given by

$$f_i(t_o) = f(t) * p_i(t)\big|_{t=t_o} \qquad p_i(t) = \frac{t^i}{i!} e^{-\lambda t} \qquad \lambda, t > 0 \tag{2}$$

'$\otimes$' stands for the convolution operator. $p_i(t)$ can be recognized as the impulse response of a cascade of i+1 identical lowpass filters. This structure is known as the *Gamma filter.* λ is called the *time scale* of the filter and it is responsible for adjusting the region of support of the impulse response $p_i(t)$. Equation (2) suggests that, instead of computing $f_i(t_o)$ off-line, one can physically measure it as the value at the $i+1^{st}$ tap of a Gamma filter with input f(t) (Figure 1). This convenience makes the moments computationally inexpensive.

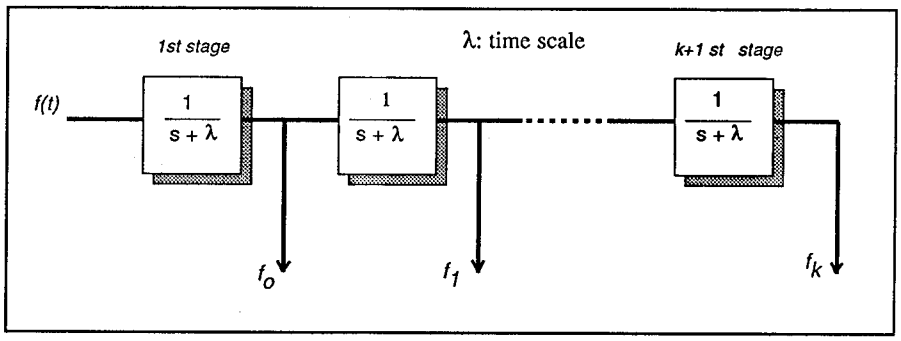


Figure 1 Poisson moments of f(t) are generated by the Gamma filter

Taking the Laplace transform of both sides of (1) and using only a finite number of terms it can be shown that this is equivalent to [4]

$$e^{st_o}F(s) \approx \sum_{i=0}^{N} f_i(t_o)\,(s+\lambda)^i \tag{3}$$

Notice the similarity between (3) and the Taylor's series expansion of $e^{sto}F(s)$ around $s=-\lambda$ given by

$$e^{st_o}F(s) \approx \sum_{i=0}^{N} c_i\,(s+\lambda)^i \tag{4}$$

where c_i's are the Taylor series coefficients.

$$c_i = \frac{1}{i!}\frac{d^i}{ds^i}\{e^{st_o}F(s)\}\Big|_{s=-\lambda} = \int_0^{\infty} f(t)\,\frac{(t_0-t)^i}{i!}\,e^{-\lambda(t_o-t)}\,dt \tag{5}$$

Taylor series coefficients carry spectral features of $e^{sto}F(s)$ around the frequencies $s=-\lambda$. If one is concerned only with the magnitude spectrum $|F(j\Omega)|$ (as with the speech signals) the term e^{sto} has no significance in the analysis since it accounts for a delay. Due to the fact that Taylor's series approximation diverges at points away from the pivot point $s=-\lambda$, feature vector $[c_0\ c_1\ \ldots\ c_N]$ carries only local frequency information. However, a piecewise representation of the entire frequency axis can be attained by concatenating the feature vectors obtained around different λ.

APPROXIMATION OF TAYLOR SERIES COEFFICIENTS IN TERMS OF THE POISSON MOMENTS

When the time scale λ and the measurement time t_o are chosen properly, Taylor series coefficients c_i can be approximated by the Poisson moments $f_i(t_o)$. If this holds, Poisson moments can be used as a feature vector to represent the spectral features of $|F(jw)|$, too. Let's denote the error of approximating the ith Taylor's coefficient by the ith Poisson moment with e_i, i.e.

$$f_i(t_o) = c_i + e_i \tag{6}$$

Assume that f(t) is a sum of complex decaying exponentials and let p_r be the real part of the pole that is closest to the imaginary axis, i.e.

$$f(t) = \sum_{k=0}^{M} b_k e^{-p_k t} u(t) \qquad \min_k \{Re[p_k]\} = p_r \tag{7}$$

Comparing (2) and (5) it can be shown that an upperbound for the absolute percent approximation error is

$$r_i = 100\left|\frac{e_i}{c_i}\right| = 100\left|\frac{e^{-\beta}}{\sum_{m=0}^{i}\frac{(-\beta)^m}{m!}}\right| \tag{8}$$

where

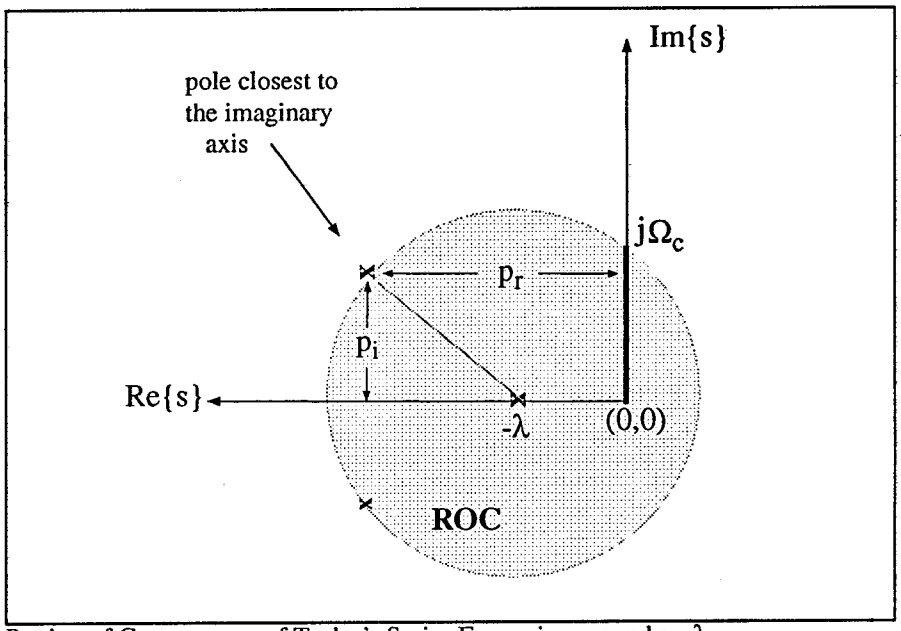
$$\beta = (p_r - \lambda)t_o \qquad p_r > 0 \tag{9}$$

The error upperbound r_i decreases monotonically as a function of β. As an example, a choice of $\beta>4.2$ guarantees that $r_i<2\%$ for moment orders up to 8. If there is no apriori information available on p_r, it can be selected to be a small positive number.

REGION OF CONVERGENCE OF THE SPECTRAL APPROXIMATION

As stated before, when the time scale λ is chosen properly, the Poisson moments correspond to the Taylor's series expansion coefficients of the input spectra around $s=-\lambda$. The choice of the time scale affects the region of convergence (ROC) of the spectral approximation as shown in Figure 2. ROC is centered around the point $s=-\lambda$ and bounded by the pole that is closest to the imaginary axis of the s-plane. If that pole has a negative real part p_r and an imaginary part p_i, it can be shown that Taylor's approximation converges for frequencies $|\Omega|<\Omega_c$ where

$$\Omega_c = \sqrt{p_r^2 + p_i^2 + 2\lambda p_r}$$



Region of Convergence of Taylor's Series Expansion around s=-λ

PIECEWISE APPROXIMATION OF THE FREQUENCY SPECTRA

Taylor's series expansion yields a local approximation. At points away from the pivot point it diverges fast. Therefore, it is not possible to globally approximate a wide bandwidth of frequencies with a finite number of coefficients. As a solution, a piecewise approximation scheme that partitions the frequency axis into several bands and approximates each band locally can be adopted. The region of support of each local approximation can be changed by selecting the pivot point $s=-\lambda$ as a complex number and varying its imaginary part so as to cover different frequency bands. Or, instead of doing the expansion around various center frequencies one can also frequency shift each band to the origin and then expand it around real λ. In practice a constant Q bandpass filter followed by a mixer can be used to shift the frequency band of interest to the origin. The baseband signal at the output of the envelope detector can be fed into the Gamma network whose tap outputs are the Poisson moments. These moments would represent the band of frequencies that the bandpass filter was tuned to. If the mixer is replaced with a cascade of a square-law device and an envelope detector, one can obtain an approximate representation for the power spectrum of the input signal (Figure 3). Actually that's what has been done in Tracey and Principe's study of simple word recognition using ANNs [2]. In preprocessing the speech signal they used a cochlea model [6] followed by a Gamma network that is used to capture the features of various bands in the form of Poisson moments. These features were further fed into an ANN for classification.

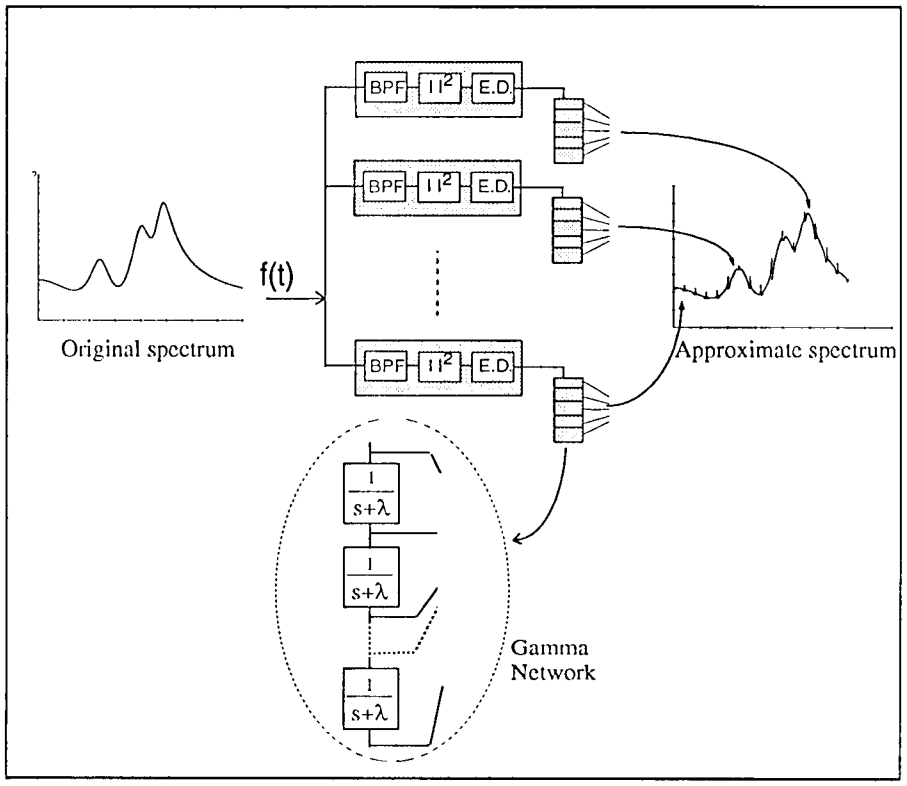


Figure 2 Piecewise approximation of the magnitude spectrum using Poisson moments

Figure 4 illustrates the magnitude spectrum of a 20 msec segment of sample word utterance 'suit' and its approximation obtained using Poisson moments. The frequency axis was divided into bands of 160 Hz each. Each band was shifted to the origin and filtered by a Gamma filter of order 4, thereby creating Poisson moment vectors of size 4. Poisson moments were further used to approximate the original magnitude spectrum. The closeness of the approximation to the original spectrum is noticeable.

CONCLUSIONS

In this study we have shown how the Gamma filter can be used to form a time-frequency representation of its input. This filter can be implemented as a cascade of identical lowpass filters. The representation is readily available at the taps of the Gamma filter in the form of Poisson moments. Compared to conventional spectral representation schemes like Fourier series or cepstral coefficients, this is a computationally very inexpensive method. The discrete time processor that operates on

the moments is not constrained by the Nyquist rate of the input signal, but by the rate moments vary. The highest frequency of the input signal affects the number of bands that need to be implemented to cover the required bandwidth with a given precision. In a sense, this method trades speed for parallelism, since each frequency band operates totally independent of the others. For spectral analysis of very high frequency signals that can not be digitized with the present technology, this method is very appealing. Analog VLSI chips can be fabricated to implement the analog bandpass filters and the Gamma structure, where the Poisson moments will be measured. The moments can be further fed into an ANN directly for tasks like classification, prediction and identification.

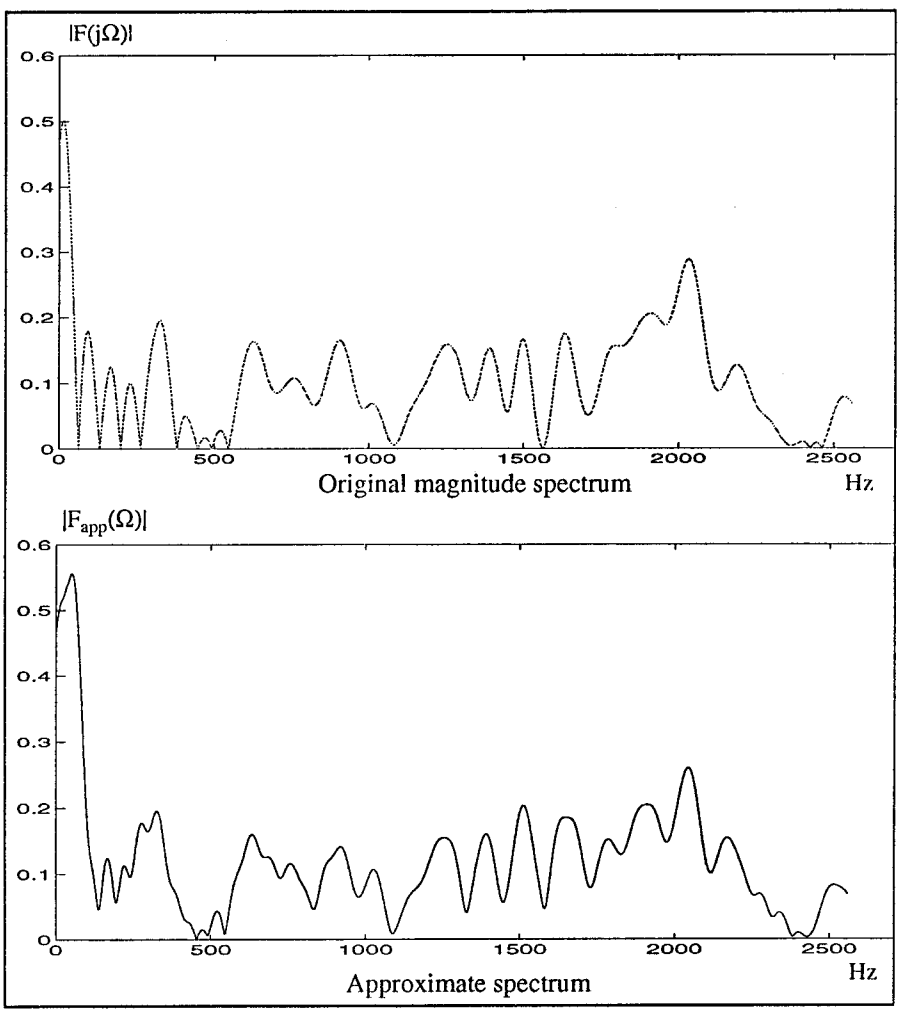


Figure 3 Original and approximate magnitude spectrums of a segment of the word utterance 'suit'.

ACKNOWLEDGEMENTS

This work has been partially supported by NSF grant ECS #920878.

REFERENCES

[1] B. de Vries, J. C. Principe, "The Gamma Model- A New Neural Model for Temporal Processing," Neural Networks, vol. 5, pp. 565-576, 1992.

[2] Tracey J.W., Principe J.C., "Isolated-Word Speech Recognition Using the Focused Gamma Neural Network," World Congress on Neural Networks, Portland, Oregon, vol III, pp. 87- 90, July 1993.

[3] Tracey J., 'Isolated Speech Recognition with the Focused Gamma Networks' Master Thesis, U. of Florida, 1992.

[4] Waibel, Alex, Hanazawa T., Hinton G., Shikano K., Lang K., "Phoneme Recognition Using Time-delay Neural Networks," IEEE Trans. on Acoustics, Speech and Signal Processing, vol. 37:3, pp. 328-339, 1989.

[5] Fairman F.W., Shen D.W.C.,"Parameter Identification for Linear Time-Varying Dynamic Processes," Proc. IEE, vol. 117, no. 10, Oct. 1970.

[6] Pickles J.O., An Introduction to the Physiology of Hearing,2nd edition, London:Academic Press, 1988.

APPLICATION OF THE FUZZY MIN-MAX NEURAL NETWORK CLASSIFIER TO PROBLEMS WITH CONTINUOUS AND DISCRETE ATTRIBUTES

A. Likas, K. Blekas and A. Stafylopatis
National Technical University of Athens
Department of Electrical and Computer Engineering
Computer Science Division
157 73 Zographou, Athens, Greece

Abstract. The fuzzy min-max classification network constitutes a promisimg pattern recognition approach that is based on hyberbox fuzzy sets and can be incrementally trained requiring only one pass through the training set. The definition and operation of the model considers only attributes assuming continuous values. Therefore, the application of the fuzzy min-max network to a problem with continous and discrete attributes, requires the modification of its definition and operation in order to deal with the discrete dimensions. Experimental results using the modified model on a difficult pattern recognition problem establishes the strengths and weaknesses of the proposed approach.

INTRODUCTION

Fuzzy min-max neural networks [2, 3] consitute one of the many models of computational intelligence that have been recently developed from research efforts aiming at synthesizing neural networks and fuzzy logic [1].

The fuzzy min-max classification neural network [2] is an on-line supervised learning classifier that is based on *hyperbox* fuzzy sets. A hyperbox constitutes a region in the pattern space that can be completely defined once the minimum and the maximum points along each dimension are given. Each hyperbox is associated with exactly one from the pattern classes and all patterns that are contained within a given hyperbox are considered to have full class membership. In the case where a pattern is not completely contained in any of the hyperboxes, a properly

0-7803-2026-3/94 $4.00 © 1994 IEEE

computed fuzzy membership function (taking values in $[0,1]$) indicates the degree to which the pattern falls outside of each of the hyperboxes. During operation, the hyperbox with the maximum membership value is selected and the class associated with the winning hyperbox is considered as the desicion of the network. Learning in the fuzzy min-max classification network is an *expansion-contraction* process that consists of creating and adjusting hyperboxes (the minimum and maximum points along each dimension) and also associating a class label to each of them.

In this work, we study the performance of the fuzzy min-max classification neural network on a pattern recognition problem that involves both discrete and continuous attributes. In order to handle the discrete attributes, the definition of a hyperbox must be modified to incorporate crisp (not fuzzy) sets in the discrete dimensions. Moreover, a modification is needed of the way the membership values are computed, along with changes in the criterion under which the hyperboxes are expanded. Besides extending the definition and operation of the fuzzy min-max network, the purpose of this work is also to gain insight into the factors that affect operation and training and test its classification capabilities on a difficult problem.

In the following section a brief description of the operation and training of the fuzzy min-max classification network is provided, while in Section 3 the modified approach is presented. Section 4 provides experimental results from the application of the approach to a difficult classification problem. It also presents results from the comparison of the method with the backpropagation algorithm and summarizes the major advantages and drawbacks of the fuzzy min-max neural network when used as a pattern classifier.

LEARNING IN THE FUZZY MIN-MAX CLASSIFICATION NETWORK

Consider a classification problem with n continuous attributes that have been rescaled in the interval $[0,1]$, hence the pattern space is I^n ($[0,1]^n$). Moreover, consider that there exist p classes and K hyperboxes with corresponding minimum and maximum values v_{ji} and w_{ji} respectively ($j = 1,\ldots,K$, $i = 1,\ldots,n$). Let also c_k denote the class label associated with hyperbox B_k.

When the hth input pattern $A_h = (a_{h1},\ldots,a_{hn})$ is presented to the

network, the corresponding membership function for hyperbox B_j is ([3])

$$b_j(A_h) = \frac{1}{n}\sum_{i=1}^{n}[1 - f(a_{hi} - w_{ji}, \gamma) - f(v_{ji} - a_{hi}, \gamma)] \tag{1}$$

where $f(x, \gamma) = x\gamma$, if $0 \leq x\gamma \leq 1$, $f(x, \gamma) = 1$ if $x\gamma > 1$ and $f(x, \gamma) = 0$ if $x\gamma < 0$. If the input pattern A_h falls inside the hyperbox B_j then $b_j(A_h) = 1$, otherwise the membership decreases and the parameter $\gamma \geq 1$ regulates the decrease rate. As already noted, the class of the hyperbox with the maximum membership is considered as the output of the network.

In a neural network formulation, each hyperbox B_j can be considered as a hidden unit of a feedforward neural network that receives the input pattern and computes the corresponding membership value. The values v_{ji} and w_{ji} can be considered as the weights from the input to the hidden layer. The output layer contains as many output nodes as the number of classes. The weights u_{jk} ($j = 1, \ldots, K$, $k = 1, \ldots, p$) from the hidden to the output layer express the class corresponding to each hyperbox: $u_{jk} = 1$ if B_j is a hyperbox for class c_k, otherwise it is zero.

During learning, each training pattern A_h is presented once to the network and the following process takes place: First we find the hyperbox B_j with the maximum membership value among those that correspond to the same class as pattern A_h and meet the expansion criterion:

$$n\theta \geq \sum_{i=1}^{n}(\max(w_{ji}, a_{hi}) - \min(v_{ji}, a_{hi})) \tag{2}$$

The parameter θ ($0 \leq \theta \leq 1$) is a user-defined value that imposes a bound on the size of a hyperbox and its value significantly affects the effectiveness of the training algorithm. In the case where an expandable hyperbox (of the same class) cannot be found, then a new hyperbox B_k is spawned and we set $w_{ki} = v_{ki} = a_{hi}$ for each i. Otherwise, the hyperbox B_j with the maximum membership value is *expanded* in order to incorporate the new pattern A_h, i.e., for each $i = 1, \ldots, n$:

$$v_{ji}^{new} = \min(v_{ji}^{old}, a_{hi}) \tag{3}$$

$$w_{ji}^{new} = \max(w_{ji}^{old}, a_{hi}) \tag{4}$$

Following the expansion of a hyperbox, an *overlap test* takes place to determine if any overlap exists between hyperboxes from different classes. In case such an overlap exists, it is eliminated by a *contraction process* during which the size of each of the overlapping hyperboxes is minimally

adjusted. Details concering the overlap test and the contraction process can be found in [2].

From the above description it is clear that the effectiveness of the training algorithm mainly depends on two factors: the value of the parameter θ and the order with which the training patterns are presented to the network.

TREATING DISCRETE ATTRIBUTES

A basic assumption concerning the application of the fuzzy min-max classification network to a pattern recognition problem is that all attributes take continuous values. Hence, it is possible to define the pattern space (union of hyperboxes) corresponding to each class by providing the minimum and maximum attribute values along each dimension. In the case of pattern recognition problems that are based on both analog and discrete attributes, it is nessecary for the discrete features to be treated in a different way. This is mainly due to the fact that it is not possible to define a meaningful ordering of the values of discrete attributes. Thus, it is not possible to apply the minimum and maximum operations on which the original fuzzy min-max neural network is based.

Consider a pattern recognition problem with n attributes (both continous and discrete). Let $\mathcal{D}$ denote the set of the indices of the discrete attributes and $\mathcal{C}$ denote the set of indices of the continuous attributes. Let also $n_C = |\mathcal{C}|$ and $n_D = |\mathcal{D}|$ denote the number of continuous and discrete attributes respectively and D^i denote the domain of each discrete feature $i \in \mathcal{D}$. A pattern $A_h = (a_{h1}, \ldots, a_{hn})$ of this problem has the characteristic that $a_{hi} \in [0,1]$ for $i \in \mathcal{C}$ and $a_{hi} \in D^i$ for $i \in \mathcal{D}$. In order to deal with problems characterized by such mixture of attributes, we consider that each hyperbox B_j is described by providing the minimum v_{ji} and maximum w_{ji} attribute values for the case of continuous features ($i \in \mathcal{C}$) and by explicitly providing a set of attribute values $D_{ji} \subseteq D^i$ for the case of discrete features $i \in \mathcal{D}$. Since it is not possible to define any distance measure between the possible values of discrete attributes, we cannot assign any fuzzy membership values to the elements of sets D_{ji}. Therefore, the sets D_{ji} are crisp sets, i.e., an element either belongs to a set or not. Taking this argument into account, equation (1) providing the membership degree of a pattern A_h to a hyperbox B_j, takes the

following form:

$$b_j(A_h) = \frac{1}{n}\{\sum_{i\in\mathcal{C}}[1 - f(a_{hi} - w_{ji}, \gamma) - f(v_{ji} - a_{hi}, \gamma)] + \sum_{i\in\mathcal{D}} m_{D_{ji}}(a_{hi})\} \quad (5)$$

where $m_S(x)$ denotes the membership function corresponding to the crisp set S, which is equal to 1 if $x \in S$, otherwise it is equal to 0.

In a neural network implementation, the continuous input units are connected to the hidden units via the two kinds of weights v_{ji} and w_{ji} as mentioned in the previous section. In what concerns the discrete attributes, we can assign one input unit to each atribute value, that is set to 1 in case this value exists in the input pattern, while the other units corresponding to the same attribute are set equal to 0. If a specific value $d_{ik} \in D^i$ belongs to the set D_{ji}, then the weight between the corresponding input unit and the hidden unit j is set equal to 1, otherwise it is 0.

During training, when a pattern A_h is presented to the network the expansion criterion has to be modified in order to take into account both the discrete and the continuous dimensions. More specifically, we have considered two distinct expansion criteria: The first one concerns the continuous dimensions and remains the same as in the original network given by equation (2) with n being replaced by n_C which denotes the number of continuous attributes. The second expansion criterion concerns the discrete features and has the following form:

$$\lambda \leq \sum_{i\in\mathcal{D}} m_{D_{ji}}(a_{hi}) \quad (6)$$

where the parameter λ ($0 \leq \lambda \leq n_D$) expresses the minimum number of discrete attributes in which the hyperbox B_j and the pattern A_h must agree in order for the hyperbox to be expanded to incorporate the pattern.

During the test for expansion process, we test whether there exist expandable hyperboxes (according to the two criteria) from the same class as A_h and we expand the hyperbox with the maximum membership. If no expandable hyperbox is found a new one B_k is spawned and we set $v_{ki} = w_{ki} = a_{hi}$ for $i \in \mathcal{C}$ and $D_{ki} = \{a_{hi}\}$ for $i \in \mathcal{D}$.

When a hyperbox is expanded, its parameters are adjusted as follows: If $i \in \mathcal{C}$

$$v_{ji}^{new} = \min(v_{ji}^{old}, a_{hi}) \quad (7)$$

$$w_{ji}^{new} = \max(w_{ji}^{old}, a_{hi}) \quad (8)$$

If $i \in \mathcal{D}$

$$D_{ji}^{new} = D_{ji}^{old} \cup \{a_{hi}\} \tag{9}$$

During overlap test and contraction the discrete dimensions are not considered and ovelap is eliminated by adjusting ony the continuous dimensions of the hyperboxes following the minimum disturbance principle as in the original network. Although it is possible to separate two hyperboxes B_j and B_k by removing common elements from some of the sets D_{ji} and D_{ki}, we have not followed this approach. The main reason is that the disturbance in the already allocated patterns would be more significant, since these sets do not contain many elements in general.

EXPERIMENTS AND CONCLUSIONS

We have studied the modified fuzzy min-max neural network classifier on a difficult classification problem concerning the assignment of credit to consumer applications. The data set (obtained from the UCI repository [5]) contains 690 examples and was originally studied by Quinlan [4] using decision trees. Each example in the data set concerns an application for credit card facilities described by 9 discrete and 6 continuous attributes, with two decision classes (either accept of reject the application). Some of the discrete attributes have large collections of possible values (one of them has 14) and there exist examples in which some attribute values are missing. As noted in [4] these data are both scanty and noisy making accurate prediction on unseen cases a difficult task.

Two series of experiments were performed. In the first series, the data set was divided into a training set of 460 examples (containing equal number of positive and negative cases) that were used to adjust the network hyperboxes, while the remaining 230 examples were used as a test set to estimate the performance of the resulting classifier. Each experiment in a series consisted of training the network (in a single pass) for certain values of θ and λ and then computing the percentage of correct classifications over the test set. Moreover, the order of presentation of the training patterns to the network was held fixed in all experiments. Best results were found for $\theta = 0.237$ and $\lambda = 8$. For these parameter values the resulting network contained 136 hyperboxes and the success rate was 87%. It must be noted that the success rate was very sensitive both on the choice of the parameter θ and on the order with which the training examples are presented. This of course constitutes a weakness of the fuzzy min-max classifier, but on the other hand, each training

experiment is very fast and the process of adjusting θ can be performed in reasonable time. We have also tested the classification performance in case the training data are presented to the network more than once and we have found that only marginal performance improvement is obtained.

We have also used the same data set to train a multilayer perceptron using the backpropagation algorithm (the on-line version). A network with one hidden layer was considered. Several experiments were conducted for different values of the number of hidden units. The best classification rate we were able to obtain was 83% for a network of 10 hidden units and with learning rate 0.1. It must be noted that the required training time was excessively long compared to the one-shot training of the fuzzy min-max network.

During experiments, we have observed that some of the examples were 'bad', in the sense that they were very difficult predict, and, in addition, when used as part of the training set, the resulting network exhibited poorer classification performance, than in the case in which these examples were not used for training. For this reason, a second series of experiments were conducted on a subset of the data set (400 examples) that resulted from the removal of the bad examples. We considered a training set and a test set of size 200, each of them containing 100 positive and 100 negative examples. Best performance was obtained for $\theta = 0.115$ and $\lambda = 8$ (112 hyperboxes) with classification rate 97.5%. Moreover, the performance was very robust with respect to the value of θ with the classification rate being more than 90% for all tested values. The best classification rate we have obtained for this data set using the backpropagation algorithm was 89.5%.

As the experiments indicate, the fuzzy min-max classification neural network constitutes a promising method for pattern recognition problems that has the advantage of fast one-shot training with its only drawback coming from its sensitivity in the parameter values used in the test for expansion criteria. Therefore, further research should be focused on developing algorithms for automatically adjusting these parameters during training.

REFERENCES

[1] IEEE Trans. on Neural Networks, Special Issue on Fuzzy Logic and Neural networks, vol. 3, No. 5, September 1992.

[2] P. K. Simpson, "Fuzzy Min-Max Neural Networks-Part 1: Classification," IEEE Trans. on Neural Networks, vol. 3, No. 5, pp. 776-786, September 1992.

[3] P. K. Simpson, "Fuzzy Min-Max Neural Networks-Part 2: Clustering," IEEE Trans. on Fuzzy Systems, vol. 1, No. 1, pp. 32-45, February 1993.

[4] J. R. Quinlan, "Simplifying Decision Trees," Int. J. Man-Machine Studies, vol. 27, pp. 221-234, 1987.

[5] P. M. Murphy and D. W. Aha, "UCI Repository of Machine Learning Databases," Irvine, CA: University of California, Department of Computer Science, 1992.

Time Signal Filtering by Relative Neighborhood Graph Localized Linear Approximation.

John Aasted Sørensen, Electronics Institute, Build. 349
Technical University of Denmark, 2800 Lyngby Denmark.

Abstract. A time signal filtering algorithm based on the relative neighborhood graph (RNG) used for localization of linear filters is proposed. The filter is constructed from a training signal during two stages. During the first stage an RNG is constructed. During the second stage, localized linear filters are associated each RNG node and adapted to the training signal. The filtering of a test signal is then carried out by inserting the test signal vectors in the RNG followed by the determination of the filter output as a function of the linear filters of the RNG nodes to which the vectors are associated. Training examples are given on a segment of a speech signal and a signal with burst structure generated from a bilinear Subba Rao model.

1 Introduction

A time signal filtering algorithm based on relative neighborhood graph (RNG) localized linear filters is proposed. The filter is constructed during two stages:
During the first stage, a training signal x_n, $n = 1, \ldots, N$ is used for generation of an RNG using an input dimension D. The RNG of a set of vectors, connect the vectors $\mathbf{x}_i^T = (x_i, \ldots, x_{i-D+1})$ and $\mathbf{x}_j$ if the intersection of the spheres with radii equal to the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ and centered in $\mathbf{x}_i$ and $\mathbf{x}_j$ does not contain any vector from the set. This intersection is also denoted the lune $\Lambda_{i,j}$ of $\mathbf{x}_i$ and $\mathbf{x}_j$. A lune thus represents a part of the input space which is mainly defined by the two vectors generating the lune. The result of the first stage is a structural representation of the input space based on the RNG. This structure is then used for localizing linear filters, adapted by a gradient algorithm to the training set during the second stage.

The filtering of a test signal t_n, $n = 1, \ldots$ is then carried out as follows: Insert test vectors $\mathbf{t}_n^T = (t_n, t_{n-1}, \ldots, t_{n-D+1})$ into the RNG, by determining all the lunes to which each $\mathbf{t}_n$ belongs. These lunes defines the neighborhood of $\mathbf{t}_n$. The filter output is then a function of the linear filters belonging to this neighborhood. In the example hereafter the filter output function is a weighted mean value of the neighborhood filters.

0-7803-2026-3/94 $4.00 © 1994 IEEE

The Relative Neighborhood Graph (RNG)

If the open sphere with center in $\mathbf{x}$ and radius r is denoted

$$B(\mathbf{x}, r) = \{\mathbf{y} \mid d(\mathbf{x}, \mathbf{y}) < r\} \tag{1}$$

where $d(\mathbf{x}, \mathbf{y})$ is the distance between $\mathbf{x}$ and $\mathbf{y}$,
then the lune $\Lambda_{i,j}$ of $\mathbf{x}_i$ and $\mathbf{x}_j$ is determined by

$$\Lambda_{i,j} = B(\mathbf{x}_i, d(\mathbf{x}_i, \mathbf{x}_j)) \cap B(\mathbf{x}_j, d(\mathbf{x}_i, \mathbf{x}_j)) \tag{2}$$

or by

$$\Lambda_{i,j} = \{\mathbf{x} \mid \max(d(\mathbf{x}_i, \mathbf{x}), d(\mathbf{x}, \mathbf{x}_j)) < d(\mathbf{x}_i, \mathbf{x}_j)\} \tag{3}$$

The lune is exemplified in Figure 1.

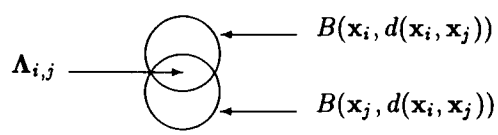


Figure 1: The relative neighborhood $\Lambda_{i,j}$.

Based on this definition of a lune [1], the RNG of an input signal x_n, $n = 1, \ldots, N$ is determined by

$$[\mathbf{P}, \mathbf{C}] = RNG(x_n,\ n = 1, \ldots, N,\ D) \tag{4}$$

where

D : Dimension of input space.
$\mathbf{P}$ is a matrix of RNG nodevectors.
$\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_R] \in \mathbf{R}^{D \times R}$
R is the number of nodes in the RNG.
$\mathbf{C}$ is the incidence matrix of the RNG.
$\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_R] \in \{0, 1\}^{R \times R}$
$\mathbf{c}_i^T = (c_{1,i}, \ldots, c_{R,i})$
$c_{i,j} = 1$ if $\Lambda_{\mathbf{p}_i, \mathbf{p}_j}$ is empty, otherwise $c_{i,j} = 0$.

2 Training Algorithm

The training algorithm is divided into 2 stages.

Stage 1: Generation of the RNG filter structure.
In the first stage the RNG is determined according to

$$[\mathbf{P}, \mathbf{C}] = RNG(x_n,\ n = 1, \ldots, N) \tag{5}$$

Stage 2: Adaptation of RNG localized linear filters.

The RNG localized linear filters are now formed by associating a FIR filter with each node of the RNG. This leads to the following filter matrix

$$\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_R] \tag{6}$$

where $\mathbf{w}_k^T = [w_{1,k}, \ldots, w_{D,k}, w_{D+1,k}]$. The term $w_{D+1,k}$ is the bias of the RNG node filter number $k = 1, \ldots, R$.

Assuming that the current augmented input signal vector at time step n is $\mathbf{z}_n^T = (x_n, x_{n-1}, \ldots, x_{n-D+1}, 1)$ and the augmented RNG node vectors are $\mathbf{r}_j^T = [\mathbf{p}_j^T, 0]$ for $j = 1, \ldots, R$, gives the following filter output, using the weighted mean of neighborhood

$$\hat{x}_n = \sum_{j=1}^{R} \frac{\gamma_{j,n}}{\gamma_n} \mathbf{w}_j^T (\mathbf{z}_n - \mathbf{r}_j) \tag{7}$$

Here $\gamma_{j,n}$ is the number of times the RNG node number j is a member of a lune to which $\mathbf{x}_n$ belongs, when $\mathbf{x}_n$ is inserted into the RNG. $\mathbf{x}_n$ is inserted in the RNG by determining the lunes to which $\mathbf{x}_n$ belongs. The total number of nodes in the lunes to which $\mathbf{x}_n$ belongs is

$$\gamma_n = \sum_{j=1}^{R} \gamma_{j,n} \tag{8}$$

The RNG node weighting matrix at time step n becomes

$$\mathbf{\Gamma}_n = \begin{bmatrix} \frac{\gamma_{1,n}}{\gamma_n} & & 0 \\ & \ddots & \\ 0 & & \frac{\gamma_{R,n}}{\gamma_n} \end{bmatrix} \tag{9}$$

In (7) $\hat{x}$ is formed as a weighted mean value of the predictions from the nodes which constitute the lunes to which $\mathbf{x}_n$ belongs. Defining the error vector between the current input vector $\mathbf{x}_n$ and RNG node number j gives $\delta_j = \mathbf{z}_n - \mathbf{r}_j$ for $j = 1, \ldots, R$. This defines the input signal matrix at time step n to the RNG nodes:

$$\mathbf{\Delta}_n = [\delta_1, \ldots, \delta_R] \tag{10}$$

From (7), (9) and (10) the filter output can be represented

$$\hat{x}_n = trace(\mathbf{W}_n^T \mathbf{\Delta}_n \mathbf{\Gamma}_n) \tag{11}$$

Using the LMS adaptation of the filter matrix $\mathbf{W}_n$, where the index n denotes the time step, leads to:

$$\mathbf{W}_{n+1} = \mathbf{W}_n + \mu e_n \mathbf{\Delta}_n \mathbf{\Gamma}_n \tag{12}$$

where $e_n = x_n - \hat{x}_n$ is the prediction error and μ is the adaptation constant.

3 Training Experiments

The training algorithm is exemplified on a speech signal segment shown in Figure 2 and on a segment of the bilinear model of Subba Rao [2]:

$$x_n = 0.8x_{n-1} - 0.4x_{n-2} + 0.6x_{n-1}e_{n-1} + 0.7x_{n-2}e_{n-1} + e_n \tag{13}$$

where e_n is white, Gaussian noise with variance 1. As shown in Figure 3, this signal exhibits burst structure.

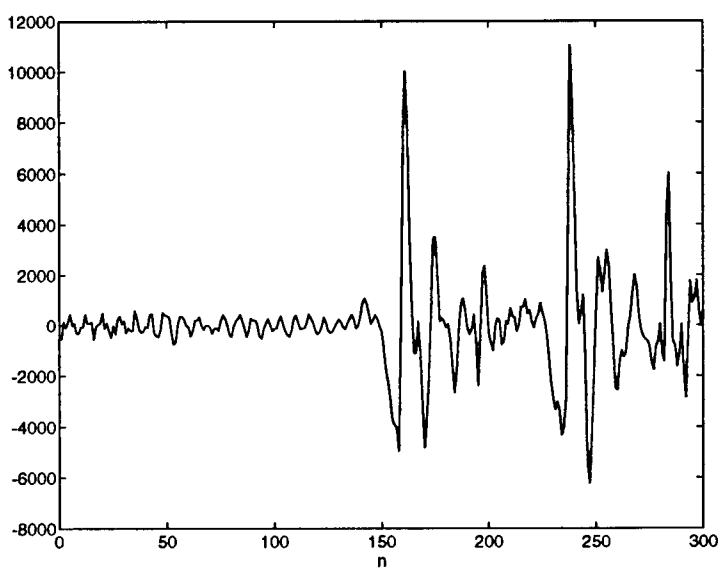


Figure 2: Speech input signal.

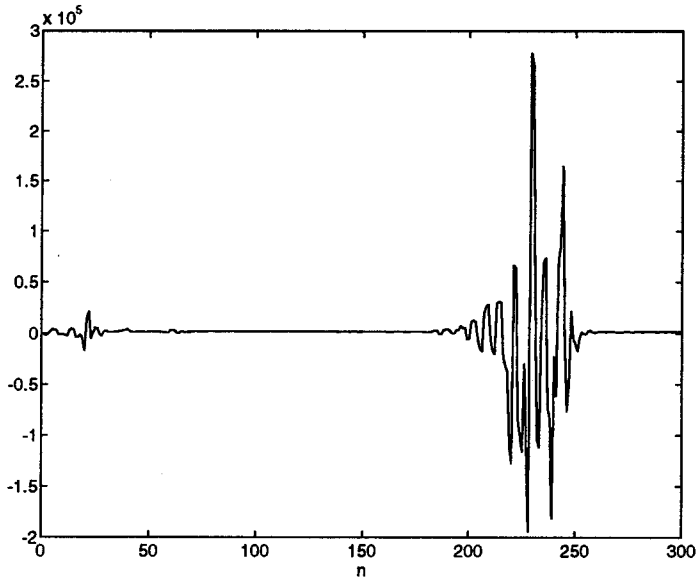


Figure 3: Input signal from the bilinear Subba Rao model.

The predictions are evaluated by the normalized, mean square error [3]

$$NMSE(x_n, \hat{x}_n) = \frac{1}{\sigma^2 N} \sum_{n=1}^{N} (x_n - \hat{x}_n)^2 \tag{14}$$

where x_n is the true value of the input signal at time step n and $\hat{x}_n$ is the predicted value. σ^2 is the variance of the input signal with N samples. Thus $NMSE$ is the ratio of the mean squared errors of the filtering method being trained and a method which predicts the mean at every time step.

In Table 1 are given training examples using the above speech and bilinear signal segments. The training is carried out on three models: The block linear filtering, the linear k-nearest neigbor filtering [4] and the relative neighborhood graph based filtering. From this it is seen that the training performance of the RNG filter structure is comparable to the performance of the k-nearest neighbor filtering at the same dimension of the input space. Furthermore it is expected that the RNG leads to a more suitable definition of neighborhood for localized filtering compared to the k-nearest nigborhood.

Filter	D	nn	R	NMSE speech	R	NMSE bilinear
Linear	12			0.137		0.691
k-nn	4	10		0.080		
	6	15		0.073		
	12	15		0.005		
	3	6				0.155
	4	6				0.032
	4	10				0.210
RNG	3		54	0.130	55	0.042
	4		66	0.051	58	0.0089

Table 1: Training results for the speech and the bilinear signal segment.
D: Dimension of input vector.
nn: The number of nearest neighbors in k-nearest neighbor.
R: The number of nodes in the RNG, determined in the first stage.

References

[1] Jerzy W. Jaromczyk, Godfried T. Tousssaint, *Relative Neighborhood Graphs and Their Relatives*, Proceedings of IEEE, Vol. 80, No. 9, September 1992.

[2] M.B. Priestly, *Non-linear and Non-stationaryTime series Analysis*, Academic Press, 1988.

[3] *Time Series Prediction*
Andreas S. Weigend, Neil A. Gershenfeld, Eds.
Proceedings Volume XV, Santa Fe Institute
Addison-Wesley Publishing Company, 1994.

[4] J.D. Farmer, J.J. Sidorowich
Exploiting Chaos to Predict the Future and Reduce Noise.
Tech. Rep. LA-UR-88-901, Los Alamos National laboratory, 1988.

CLASSIFICATION USING HIERARCHICAL MIXTURES OF EXPERTS

S.R.Waterhouse A.J.Robinson
Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ, England

Abstract–There has recently been widespread interest in the use of multiple models for classification and regression in the statistics and neural networks communities. The Hierarchical Mixture of Experts (HME) [1] has been successful in a number of regression problems, yielding significantly faster training through the use of the Expectation Maximisation algorithm. In this paper we extend the HME to classification and results are reported for three common classification benchmark tests: Exclusive-Or, N-input Parity and Two Spirals.

INTRODUCTION

Traditional Neural Network architectures such as the multi-layer perceptron have proved successful as universal function approximators and have been used in many different problems ranging from pattern classification to control engineering. Whilst there is undoubtably further valuable work to be done on such architectures, such as improving training methods, there is a considerable incentive to look in other directions for new architectures. Such architectures ideally would be statistically motivated and have parameters which are easily interpretable; they would also allow training speeds to be increased, since the gradient descent algorithm used in traditional back-propagation is typically too slow for solving real-world problems in real time.

Motivated by such concerns, a number of researchers have investigated methods of function approximation incorporating ideas from the fields of statistics and neural networks. One recurring trend in such work is the use of separate models to approximate different parts of a problem. The general approach is to divide the problem into a series of sub-problems and assign a set of function approximators or 'experts' to each sub-problem. Different approaches use different techniques to divide the problem into sub-problems and to calculate the best solution to the problem from the outputs of the experts. The architecture described in this paper, the Hierarchical Mixtures of Experts (HME) [1], employs probabilistic methods in both the way it divides the input space and the way it combines the outputs from the experts.

The paper is organised as follows. The HME architecture is described, along with the use of the Expectation Maximisation (EM) algorithm [2] which is used to estimate its parameters. The extension of the HME to classification is discussed, including the required modifications to the training algorithm. The results obtained on two classification simulations are presented: N-input Parity and the 'Two Spirals' problem.

HIERARCHICAL MIXTURES OF EXPERTS

The HME is based on the principle of 'divide and conquer' in which a large, hard to solve problem is divided into many smaller, easier to solve problems. There are

0-7803-2026-3/94 $4.00 © 1994 IEEE

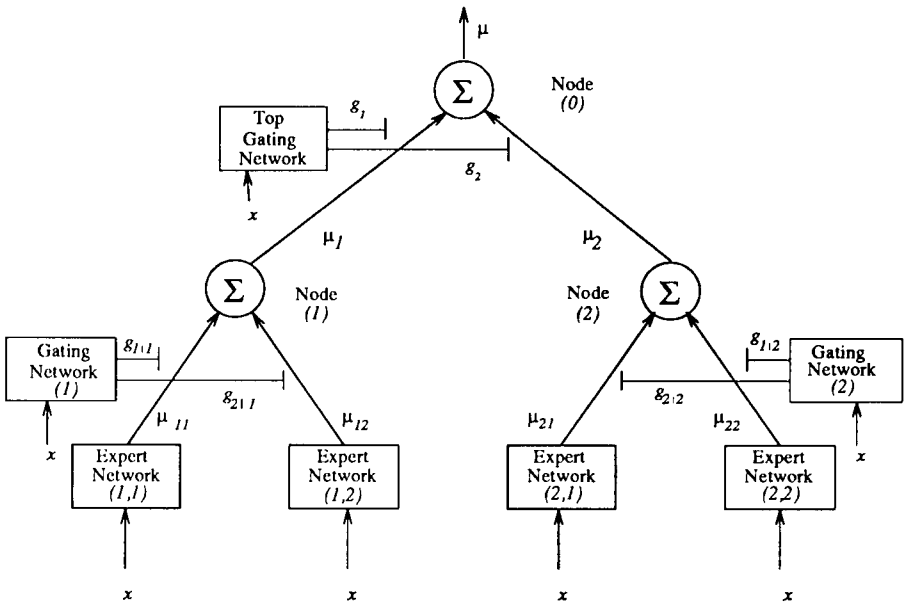


Figure 1: Hierarchical Mixture of Experts

several alternative strategies for tackling such problems. The simplest approach is to divide the problem into sub-problems having no common elements - a 'hard split' of the data. The optimum output of the experts assigned to each sub-problem may then be chosen on a 'winner-takes-all' (WTA) basis. Classification and Regression Trees (CART) [3] are based on this principle. Alternatively, the outputs of the experts may be combined in a weighted sum with weights derived from the performance of the experts in their partition of the input space; this is the principle behind Stacked Generalisation [4]. The most advanced method is to divide the problem into sub-problems which can have common elements -a 'soft split' of the input space into a series of overlapping clusters. The outputs can be chosen either using WTA or stochastically. The HME combines the ideas of soft splits of the data with stochastic selection of the outputs of the experts by the use of a gating network. A two-level HME architecture with a common branching factor of two at each level (a 'binary branching' HME) is shown in Figure 1. In the general architecture, multiple levels and branching factors are possible. In its basic form, the HME employs a fixed architecture which is pre-determined before training commences. The tree consists of non-terminal and terminal nodes which we denote by a set of indicator variables $\{Z\}$. Non-terminal node (1) is thus denoted z_1 and consists of gating network GN(1). Terminal node (1, 1) is denoted by z_{11} and consists of expert network EN(1,1). A general HME with i levels of gating networks and branching factors $b_0, b_1, ..., b_{i-1}$ is denoted by HME(i:$b_0, b_1, ..., b_{i-1}$), thus Figure 1 is HME(2:2, 2). In the original HME each expert was linear and performed a regression task [1]. In this paper, each expert is non-linear and performs multi-way classification.

Our general classification problem may be considered as follows. At time t during training, we observe an input vector $x^{(t)}$ which belongs to class n. We construct a *target* output vector $y^{(t)}$ with 1 in element j and 0 elsewhere. We wish to compute the

probability $P(y_n|\boldsymbol{x}^{(t)})$ of the correct class n being returned given the input vector at time t. [1] We do this by breaking the problem into a series of smaller problems. For example, expert network EN(1, 1) computes $P(\boldsymbol{y}|\boldsymbol{x}, z_1, z_{11})$, the probability vector of all classes given that we took the left branch of every split and ended up in terminal node (1, 1). The top level gating network GN(0) computes $P(z_1|\boldsymbol{x})$, and the second level gating network GN(1) computes $P(z_{11}|\boldsymbol{x}, z_1)$. GN(1) weights EN(1, 1) and EN(1, 2) to give the output of node (1),

$$\mu_1 = P(\boldsymbol{y}|\boldsymbol{x}, z_1) = \sum_{j=1}^{b_1} P(\boldsymbol{y}|\boldsymbol{x}, z_1, z_{1j}) P(z_{1j}|\boldsymbol{x}, z_1).$$

All these nodes are combined to give the overall output μ of the HME,

$$\begin{aligned} \mu = P(\boldsymbol{y}|\boldsymbol{x}) &= \sum_{i=1}^{b_0} P(z_i|\boldsymbol{x}) P(\boldsymbol{y}|\boldsymbol{x}, z_i) \\ &= \sum_{i=1}^{b_0} P(z_i|\boldsymbol{x}) \sum_{j=1}^{b_1} P(z_{ij}|\boldsymbol{x}, z_i) P(\boldsymbol{y}|\boldsymbol{x}, z_i, z_{ij}) \end{aligned}$$

This process may be extended to any depth and may use arbitrary branching factors at each depth. Unlike CART, the shape of the HME network is pre-determined heuristically before training.

Expert network EN(1, 1) is a single layer network with 'softmax' activation function [5] whose output is

$$\mu_{11} = P(\boldsymbol{y}|\boldsymbol{x}, z_1, z_{11}, \Theta_{11}) = \exp(\theta_{11n}^T \boldsymbol{x}) \Big/ \sum_{k=1}^{N} \exp(\theta_{11k}^T \boldsymbol{x}) \,,$$

where Θ_{11} is a parameter matrix, consisting of N independent vectors $\{\theta_{11k}\}$. The form of GN(1) is

$$P(z_1|\boldsymbol{x}, \Xi_1) = \exp(\xi_{11}^T \boldsymbol{x}) \Big/ \sum_{i=1}^{b_1} \exp(\xi_{1i}^T \boldsymbol{x})$$

where Ξ_1 is the parameter matrix for this gate, consisting of b_1 independent vectors ξ_{1i}. Therefore, the mathematical form of the gating and expert networks is the same, with the difference that the gating network is classifying the experts over the input space and the experts are classifying within the input space regions.

Training the HME

The HME is trained using the Expectation Maximisation (EM) algorithm, in which 'missing data' is specified, which if known would simplify the maximisation problem. If we had information about which node had generated the data, we could update the parameters for the gates and experts for that node. Thus the missing data for the EM algorithm applied to HMEs is the set of *indicator* variables $\{Z\}$ which indicate which node generated each output, or alternatively which node is best suited to the portion

1. For simplicity of notation, we shall now drop the superscript t on the inputs, outputs and indicator variables.

of the input space under consideration. The E-step of the EM algorithm reduces to computing the expected values of the indicator variables which gives the set of *posterior* probabilities $\{H\}$. The *conditional* posterior probability of node $(1,1)$ is the probability that EN$(1,1)$ can be considered to have generated the data based on both input and output observations, given that we are in non-terminal node (1). This is given by

$$h_{1|1} = P(z_{11}|z_1, \boldsymbol{x}, \boldsymbol{y}) = \frac{P(z_{11}|z_1, \boldsymbol{x})P(\boldsymbol{y}|\boldsymbol{x}, z_{11}, z_1, \Theta_{11})}{\sum_{j=1}^{b_1} P(z_{1j}|z_1\boldsymbol{x})P(\boldsymbol{y}|\boldsymbol{x}, z_{1j}, z_1, \Theta_{1j})},$$

where $P(\boldsymbol{y}|\boldsymbol{x}, z_{11}, z_1, \Theta_{11})$ is the probability of generating the correct output vector $\boldsymbol{y}$ from EN$(1,1)$ given the input $\boldsymbol{x}$. For 1-out-of-N classification, this is given by

$$P(\boldsymbol{y}|\boldsymbol{x}, z_{11}, z_1, \Theta_{11}) \quad = \quad \exp\left(\sum_{k=1}^{N} y_k \log \mu_{11k}\right) = \mu_{11n}$$

where μ_{11k} is the output for class k from EN$(1,1)$ and n is the correct class. In a similar way, the conditional probability of node (1) is given by

$$h_1 = P(z_1, \boldsymbol{x}, \boldsymbol{y}) = \frac{P(z_1|\boldsymbol{x})P(\boldsymbol{y}|\boldsymbol{x}, z_1)}{\sum_{i=1}^{b_1} P(z_i|\boldsymbol{x})P(\boldsymbol{y}|\boldsymbol{x}, z_i)}.$$

For node $(1,1)$ the *joint* posterior probability, h_{11} is the product of the joint posterior probability of node (1) and the conditional posterior probability of node $(1,1)$. In a deeper architecture, the joint posterior probabilities are recursively computed by multiplying the conditional posterior probabilities along a path from the root node (0) to the node in question.

The M step reduces to a set of independent *weighted* maximum likelihood problems for the experts and the gates. Thus the *weight* for GN(1), at time t, is the joint posterior probability of this node, $h_1^{(t)}$, and the *weight* for EN$(1,1)$ is the joint posterior probability of this node, $h_{11}^{(t)}$. The target outputs for the gating networks are the conditional posterior probabilities of the node in question, so that the targets for GN(1) at time t are $h_{1|1}^{(t)}$ and $h_{2|1}^{(t)}$ for outputs 1 and 2 respectively.

Once the maximum likelihood problems of the M-step have been completed, the E-step is repeated, computing a new set of posteriors $\{H\}$ for all times t which become the new weights for the M-step.

Solving the M-Step

Since each EN and GN is a simple network with a single layer of weights, we may solve the maximum likelihood problems relatively easily. We update the parameter vectors for each output of the networks independently, given the *Generalised Linear* [6] assumption that the outputs are independent. The simplest method is to use gradient ascent of the likelihood, which for parameter vector θ_i^m at iteration m for output i of an EN reduces to

$$\theta_i^{m+1} = \theta_i^m + \lambda \frac{1}{\sum_{t=1}^{T} h^{(t)}} \sum_{t=1}^{T} h^{(t)} \boldsymbol{x}^{(t)} (y_i^{(t)} - \mu_i^{(t)})$$

where $y_i^{(t)}$ is the target for class i, $\mu_i^{(t)}$ is the ith output of the EN, $h^{(t)}$ is the *weight* at time t, T is the total time, and λ is a learning rate. These equations are the same for the gating networks, with the output targets $\{y_i^{(t)}\}$ replaced by the conditional posterior probabilities of the node in question.

An alternative maximisation method, and the one adopted in this paper, is to use the Hessian or second derivative of the likelihood with respect to the parameter vectors:

$$\theta_i^{m+1} = \theta_i^m + \lambda \left(\sum_{t=1}^{T} h^{(t)} \boldsymbol{x}^{(t)} \mu_i^{(t)} (1 - \mu_i^{(t)}) \boldsymbol{x}^{(t)^T} \right)^{-1} \left(\sum_{t=1}^{T} h^{(t)} \boldsymbol{x}^{(t)} (y_i^{(t)} - \mu_i^{(t)}) \right), \quad (1)$$

where λ is once again a learning rate, which has typical values in the range 0.4 to 1.0. This method is equivalent to the Iteratively Reweighted Least Squares algorithm (IRLS) of Generalised Linear Models [6].

Implementation Issues

Variation in M-Step Iterations. Although the basic EM algorithm dictates that the M step should be iterated until convergence, the *Generalised* EM algorithm (GEM) relaxes this constraint, requiring only an increase in the likelihood in the M-step. By reducing the number of M-step iterations we can reduce the overall computation. The power of the EM algorithm lies in the E-step which repeatedly computes new weights based on the previous M steps. In our experiments the number of M step iterations was typically set to between 1 and 3.

Learning rates. The IRLS algorithm in common with conventional gradient descent algorithms, is sensitive to learning rates. Learning rates that are too large give instability, manifested in step sizes that lead to an decrease in the overall network likelihood. In practice we found that a learning rate of 0.4 for both experts and gates gave a good balance between learning speed and stability, although rates of 0.8 have proved stable with some initial conditions.

Saturation of expert and gating network outputs. If the output $\mu_i^{(t)}$ in Equation (1) of any of the networks becomes near to either 1 or 0, or if the weight $h^{(t)}$ is near 0, then the addition to the Hessian matrix for output i of that network at time t will be very small. If this occurs for a large majority of the training set, the Hessian will become singular and impossible to invert accurately. The solution to this problem is to use threshold values for the outputs of 0.9999 and 0.0001 and a floor for the weights of 0.0001. In practice, these have to be tuned to prevent instability but have no significant effect on accuracy until set to around 0.9 and 0.1.

Choice of initial parameter values. Two strategies are used to initialise the network. The first is to start all parameter vectors of experts and gates at zero and give each gate output a 'kick' so that the experts begin to separate in the input space and compute different outputs. The second is to initialise all parameter vectors to random values, in a range $-r$ to $+r$. Typically$0.1 \leq r \leq 3$. An alternative is to use random weights for the expert parameters and zero initial weights for the gates, with the choice of strategy varying with the problem. In our experiments we found that the second option gave the quickest results which were most free from local maxima, whilst the first and third options gave solutions which were drawn to local maxima or failed to separate the experts at all.

SIMULATIONS

In this paper we follow the work of [7] in using strict methods when reporting learning speeds and network performance. In particular we use a 40-20-40 threshold criterion which dictates that an output is only correct if it is greater than 0.6. We define an *epoch* as one pass through the training set. Thus, one EM cycle may consist of many epochs, depending on the number of iterations performed in the M-step.

N-Input Parity

The task of the N-input parity problem is to compute the odd parity of N binary inputs. The network must compute a 'one' if the input has an odd number of 'one' bits in the input and a 'zero' if there are an even number. The special case of 2-input parity is the Exclusive Or function (XOR) which was shown to be impossible for simple single layer networks to approximate [8]. In this paper we show that the HME can solve this problem efficiently using only three single layer networks, in the form of one GN and two ENs. We also describe solutions for 3 to 8 input parity, with learning times faster than conventional feed-forward networks. The performance of the HME on the XOR problem using a varying degree of test thresholding and averaging over 100 trials per threshold is shown in Table 1. These results were obtained using a

Threshold	Min Epochs	Max Epochs	Average Epochs	Standard Deviation
0.6	2	6	2.76	0.971
0.8	2	6	3.32	0.882
0.9	3	6	4.14	0.757
0.95	4	7	4.67	0.713
0.99	8	24	11.5	4.27

Table 1: Results for the HME on the XOR problem.

HME(1:2) with a total of 9 parameters. By way of comparison, conventional feed forward networks, using a 2-2-1 structure can solve this problem at the 0.6 threshold in an average of 19 epochs of quickprop [7]. Using the delta-bar-delta rule, an average training time of 250.4 epochs has been reported [9]. Using the HME, we reach the 0.6 threshold in an average of 2.76 epochs, and the 0.99 threshold in an average of 11.5 epochs. Of all these trials of the HME on the XOR problem, none failed to converge or had to be restarted. The results on the N-input Parity problem are shown in Table 2. By way of comparison, the best performance reported on 8-input Parity by a 2-16-1 back-propagation network is 2000 epochs of standard back-propagation [10] and 172 epochs of quickprop. The table shows the average results over 50 trials. The number of tests which failed to converge to the correct solution is shown as % NC. Using the HME, N-input parity requires at least N experts. However, in 'tight' networks with around N experts, there is an increased chance of local maxima. This effect may be seen in Figure 2 which shows the effect of different initial conditions for a HME(1:4) on the XOR problem. Since we may solve the XOR problem using only 2 experts, this network is over-specified and includes redundancy. The solutions in the figure differ in the distribution of the data between the experts. Sub-figure 2(a) is a solution in which the data is shared evenly between 2 experts with the remaining 2 experts inactive. In 2(b) the data is distributed again between 2 experts but with

N	Parameters	Architecture	Min Epochs	Max Epochs	Average Epochs	% NC
3	12	(1:3)	3	11	4.85	0
4	18	(2:2,2)	6	13	10.4	20
5	51	(2:3,3)	11	26	16.2	10
6	60	(4:2,2,2,2)	12	29	18.8	20
7	111	(5:2,2,2,2,2)	14	26	23.5	10
8	51	(2:2,4)	50	102	42	90
8	111	(5:2,2,2,2,2)	15	57	34	50
8	210	(6:2,2,2,2,2,2)	13	56	36	33

Table 2: Performance of the HME on the N-Input Parity problem.

3/4 of the data in one expert and 1/4 in the other. In 2(c) the data is distributed over 3 experts with 1/2 going to one expert and the remaining 1/2 shared between the remaining 2 experts. In 2(d) the data is distributed evenly over 4 experts. It is clear that 2 (b) is an unsatisfactory solution which would not give good generalisation, unlike (a), (c) or (d). In a series of 35 trials for HME(1:2), solution (a) occurred in 26 cases while (b) occurred in 9 cases. By using the HME(2:2, 2), solution (c) occurred 23 times, (a) 8 times, (d) 3 times and (b) only once. Thus, we have reduced the probability of solution (b) occuring by adding the extra 2 experts. For larger values

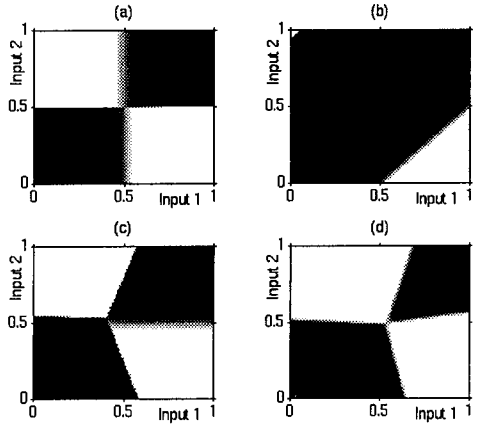


Figure 2: The effect of different initial conditions on a HME(1:4) for the XOR problem.

of N, behaviour of this sort may lead to local maxima which give us non-convergent solutions. The net result of this is that we get many more non-convergent solutions with tight networks, although there is a large advantage in terms of computation when using such a network. By relaxing the network and using more levels, and thus introducing redundancy, we create many more possible configurations which will give success, as seen by relaxing the XOR problem to using 4 experts instead of 2. Therefore the number of non-convergent solutions is reduced, and those that do occur represent states where only a small number of points remain misclassified. This effect may be seen for 8-input Parity in Table 2. By increasing the depth of the network and thus increasing the number of terminal nodes, we reduce the percentage

of non-convergent solutions.

The Two Spirals Problem

The aim of the two-spirals problem is to train a network to discriminate between two spirals in the 2-D plane. Each spiral has 97 points and coils three times around the origin and around the other spiral, without overlapping. The learning set and the evolution of the output of a binary branching HME with 10 levels is shown in Figure ??. The points in the test set are offset vertically from the points in the learning set by 0. 1. The best solutions to the spirals problem have been obtained using Cascade Correlation [11]. This is capable of approximating the problem in 1700 epochs using around 140 parameters. Back-propagation networks have been used to solve the problem [12] using a 2-5-5-1 network with shortcut connections between layers in 20,000 epochs using conventional gradient descent with momentum and 8,000 epochs using quickprop, using a similar number of parameters. Using a conventional 2-5-5-1 network without shortcuts took 60,000 epochs of quickprop.

Depth of Tree	Number of Parameters	Training Set Correct / 194	Testing Set Correct / 194	M-step Iterations	Total Epochs
10	3102	187	184	3	135
10	3102	185	184	1	140
5	111	161	159	1	30

Table 3: Results for the Two-Spirals Problem using binary branching HMEs.

The results in Table 3 demonstrate that the HME is capable of solving the two-spirals problem to a high degree of accuracy. Although the experiments performed have not resulted in a complete solution, the number of training epochs for the HME on this problem are an order of magnitude less than Cascade Correlation networks and two orders of magnitude less than conventional feed-forward back-propagation networks. We suspect that the non-convergence of the HME is due to similar effects as those proposed for the Parity problem. In terms of numbers of parameters, the HME may appear to be using far more, since for a depth of 10 and common branching factor of 2 there are 3102 parameters. This is misleading, however, since the number of terminal nodes which actually remain active is a small fraction of the total number of terminal nodes present.

CONCLUSIONS

We have described the application of the HME to classification and presented a number of results on standard benchmarks. In common with the performance of the HME on regression problems, we have found that it requires fewer epochs to learn classification problems than conventional feed-forward networks. There are however, a number of problems associated with the learning algorithm, including numerical instabilities caused by the 2nd order M-step update and the existence of local maxima within the solution sets. We have described solutions to these problems, such as the use of thresholding of outputs and weights in the M-step, choice of learning rate and initial conditions to avoid instabilities and local maxima. Future directions for this work will focus on removing the need for matrix inversion by using some form of approximation to the inverse Hessian in (1). The use of fast gradient descent methods

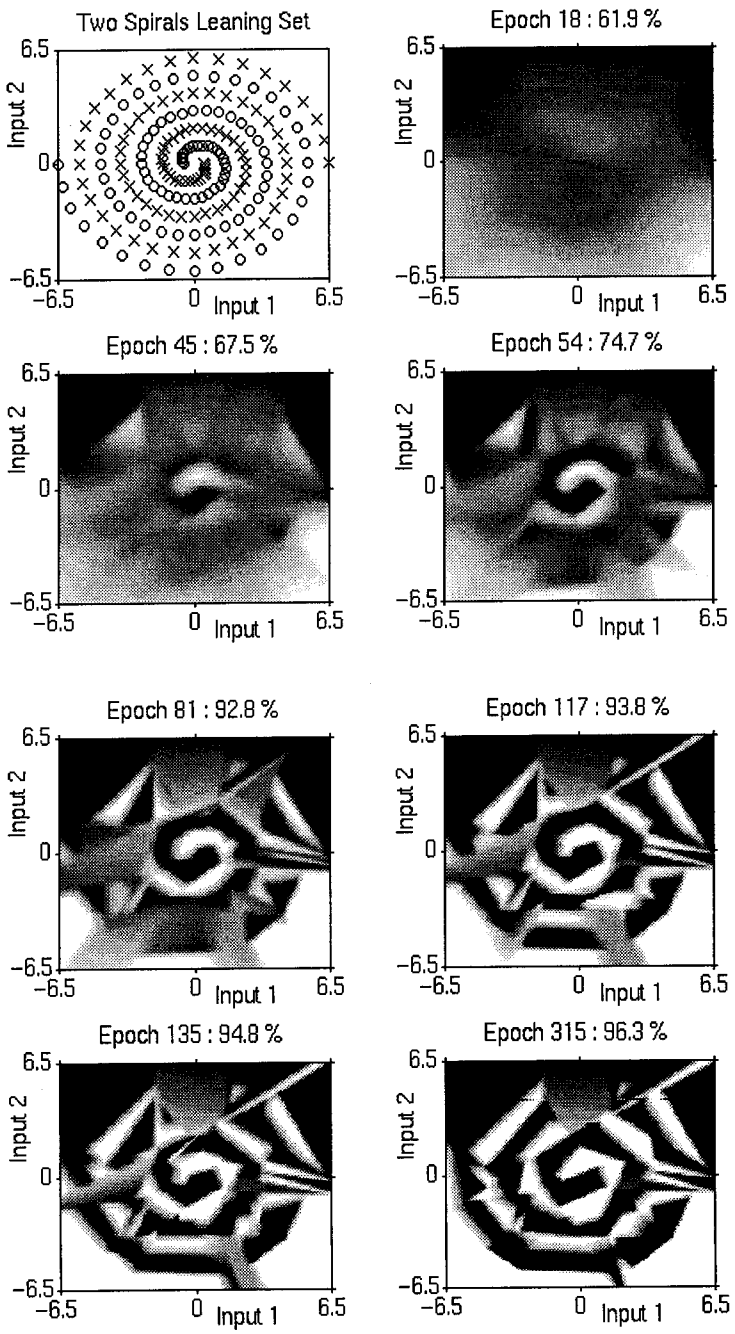


Figure 3: Learning set for the two spirals problem and evolution of the decision boundary for a binary branching HME with 10 levels.

such as quickprop [7] would move the HME closer to true connectionist methods and reduce the computational load of the M-step.

In addition we have described how the use of redundancy in the HME may reduce the chance of local maxima. In these networks, the redundant experts are typically inactive after a few epochs, which suggests that they could be 'pruned' using similar techniques to those developed for CART [3]. Alternatively we may start with a small network of, say 2 experts and grow the tree using CART principles. We anticipate that the use of such ideas will improve the performance of the HME in terms of speed and accuracy and allow us to extend the applications of classification HMEs to real world problems.

Acknowledgements

Steve Waterhouse is funded by a partnership agreement with Waterhouse Associates. Tony Robinson is supported by an EPSRC Advanced Research Fellowship.

REFERENCES

[1] M. I. Jordan and R. A. Jacobs, "Hierarchical Mixtures of Experts and the EM algorithm," *Neural Computation*, vol. 6, pp. 181–214, 1994.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society Series B*, vol. 39, pp. 1–38, June 1977.

[3] L. Breiman, J. Friedman, R. Olshen, and C. J. Stone, *Classification and Regression Trees.* Wadswoth and Brooks/Cole, 1984.

[4] D. H. Wolpert, "Stacked generalization," Tech. Rep. LA-UR-90-3460, The Santa Fe Institute, 1660 Old Pecos Trail, Suite A, Santa Fe, NM, 87501, 1993.

[5] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing: Algorithms, Architectures and Applicatations* (F. Fougelman-Soulie and J. Hérault, eds.), pp. 227–236, Springer-Verlag, 1989.

[6] P. McCullagh and J. A. Nelder, *Generalized Linear Models.* London: Chapman and Hall, 1989.

[7] S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," in *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, 1988.

[8] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry.* Cambridge, MA: MIT Press, 1969.

[9] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295–307, 1988.

[10] G. Tesauro and R. Janssens, "Scaling relationships in back-propagation learning: Dependence on predicate order," Tech. Rep. CCSR-88-1, Center for Complex Systems Research, University of Illinois at Urbana Champagne, 1988.

[11] S. E. Fahlman and C. Lebiere, "The Cascade-Correlation learning architecture," Tech. Rep. CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 1990.

[12] K. J. Lang and M. J. Witbrock, "Learning to tell two spirals apart," in *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, 1988.

A Hybrid Neural Network Architecture for Automatic Object Recognition

Thomas Fechner, Ralf Tanger

Abstract

This paper describes the application of a hybrid neural network architecture for automatic object recognition in inverse synthetic aperture radar (ISAR) imagery. The architecture employs a cascaded combination of an unsupervised and a supervised trained Neural Network. The unsupervised trained Self-Organizing Feature Map is used for object segmentation and the supervised trained Multi-Layer Perceptron classifies the segmented objects. The classification result is fed back to the Feature Map Segmentor in order to improve segmentation and classification. The functionality of this approach is demonstrated by the use of simulated noisy ISAR images from different objects.

I. Introduction

AUTOMATIC object or target recognition (ATR) with neural networks is an active research area for military and commercial applications [1]. Commonly used sensors are based on passive electro-optic or active radar technology. Electro-optic devices have the advantage of higher resolution and radar has the advantage of all-weather capability and a higher range. Although we are working with imagery from inverse synthetic aperture radar (ISAR), our aim is to develop an architecture which is capable of processing images which may be produced by different sensors.

ISAR-images are produced by radar range/doppler measurements of a rotating object. ISAR imagery differs from photographic imagery in that it only exhibits the centers of reflectivity which fluctuate depending on the aspect angle. Due to the larger wavelength of radar waves, the image resolution is much lower than photographs. Further, ISAR imagery is rather noisy. Another problem with real world ISAR imagery is that due to estimation errors the image may change its scale. In order to classify objects under such degraded conditions, powerful segmentation techniques are required. The first section of this paper describes the functional blocks of the proposed ATR-architecture. The next section demonstrates the functionality of this approach by an example.

T. Fechner and R. Tanger are with the Daimler Benz Forschungsgruppe Systemtechnik, Alt Moabit 91b, 10559 Berlin, Germany. E-mail: fechner@DBResearch-berlin.de, tanger@DBResearch-berlin.de

0-7803-2026-3/94 $4.00 © 1994 IEEE

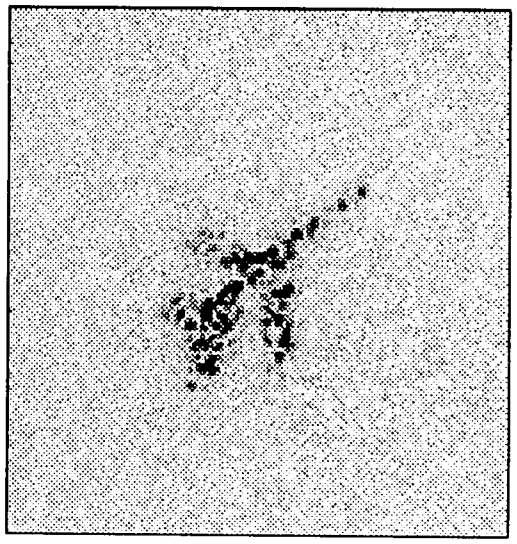


Fig. 1. Noisy input mage

II. System description

The proposed automatic object recognition system (Fig. 2) works on 2-dimensional intensity images with dimension $N \times N$. A typical input image is depicted in Fig. 1. The first processing stage is the image segmentor. The segmentor has two tasks: separating the object of interest from the background and restoring degraded regions in the segmented

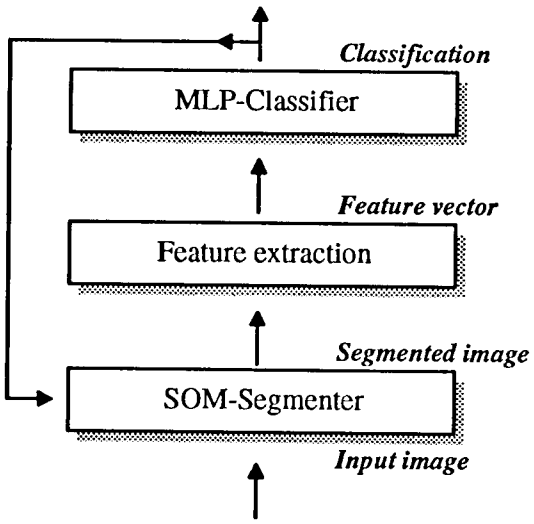


Fig. 2. Hybrid architecture or the automatic object recognition system

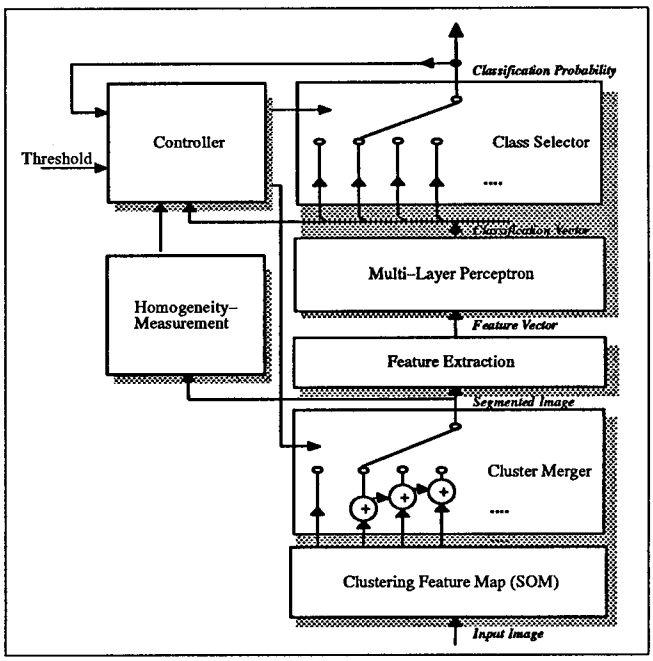


Fig. 3. Block diagram adaptive segmentation

image. The next processing stage is the feature extractor. Although some proposed neural network recognition systems work directly on the image pixels, we decided to implement a special feature extraction processing stage. The first reason is to reduce the number of input features in an effort to keep the classifier complexity low. The second reason is that we are able to integrate a-priori knowledge about the classification problem by implementing specific features. The feature vector is fed to the neural network classifier which is implemented as a multi-layer perceptron with two hidden layers. Unlike traditional feedforward classifier approaches, here the classifier output is fedback to the segmentor in order to optimize the quality of the segmented image and the sharpness of the classifier's decision.

A. Clustering Segmentation using Feature Maps

Segmentation is the process of separating the object of interest from the background. Pixels belonging to the object are set to 1 while background pixels are set to 0. Due to the simplicity, most traditional segmentation techniques are based on thresholding. In this approach, separate peaks in the intensity histogram are used to construct an optimal threshold. Unfortunately this method fails when the histogram shows a unimodal shape. Unimodal distributions are obtained when the image consists mostly of large background areas with only a small object which do not exhibits sharp boundaries. Unsupervised clustering segmentation

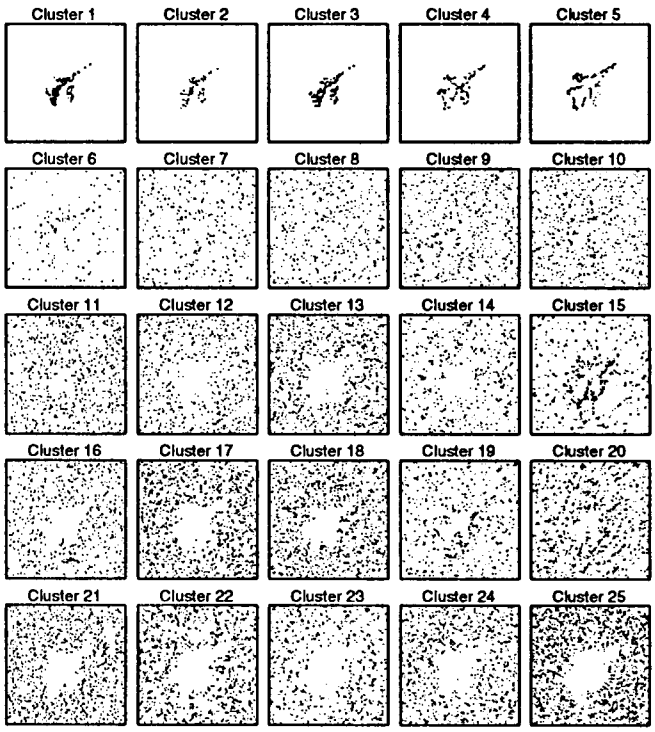


Fig. 4. Clustering the noisy input image of fig. 1, using 25 subclusters

has advantages because the segmentation relies on local features and not on global thresholds. Also, the unsupervised approach may reveal characteristics that have not been observed by the human. Clustering segmentation techniques assign a feature vector to each image pixel. The n-dimensional feature space is partitioned into M regions so as to minimize a global error function. The idea is that object and background pixels have dissimiliar features and are therefore associated to different clusters. The feature vector $x(m)$ with the features $\{x_1(m), x_2(m)\}$ is used to describe each pixel $m \in N^2$: single pixel intensity and mean intensity within a local neighborhood.

$$x(m) = \left\{ i_0(m), \sum_{j \in V_j} i_0(j) \right\} \tag{1}$$

where V_j is a set of the 24 nearest neighbors with pixel m at its center.

Although the goal is to have two clusters (one for the object pixels and one for the background) it is better to choose a number of sub-clusters which will be assembled in a hierarchical order to obtain object and background superclusters. The subclusters are represented by a set of vectors $w(l)$ with $l \in [1 \ldots M]$. This vector quantization problem can

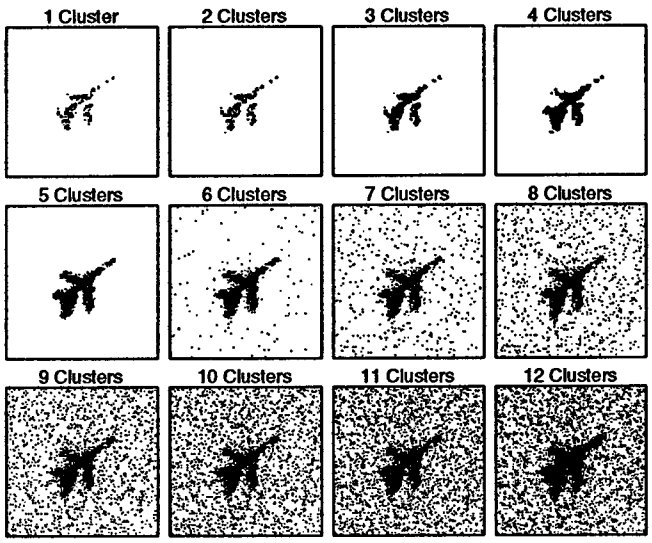


Fig. 5. Expansion of the object-supercluster by subcluster-merging (For demomstration purposes, we proceed the cluster merging above the optimum)

be solved by using either conventional vector quantization algorithms [2] or self organizing feature maps (SOM) [3][4]. SOM's have the advantage that the resulting clusters are topologically ordered. For this application the SOM is configured as a one-dimensional array of $M = 25$ laterally interconnected neuron-elements, each neuron representing one cluster. The topological order is exploited here in order to dynamically expand the object-supercluster (fig. 5) by merging neighboured subclusters (fig. 4). After clustering the segmented image s_{seg} is realized as follows:

$$s_{seg}(m) = \begin{cases} 1 & if \quad x(m) \text{ belongs to the object-supercluster} \\ 0 & else \end{cases} \tag{2}$$

B. Feature Extraction for the classifier

The primary requirements for features are compactness, informativeness and robustness. Two groups of features were extracted: statistical moments and range bin profiles (p_y). Moments have been used for many object classfication tasks e.g. aircraft-identification [5]. The main advantage is their compactness and their invariance properties. The central moments m_u are related to the center of mass and therefore they are translation invariant. By area-normalization of the central moments the resulting normalized central moments n_u are also scale invariant. Due to the binarization of the image the moments depend only on the shape of the segmented object.

$$m_{u_{i,j}} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} (x - \overline{x})^i (y - \overline{y})^j s_{seg}(x, y) \tag{3}$$

$$n_{u_{i,j}} = m_{ui,j}/m_{u_{00}}^2 \tag{4}$$

with $\overline{x} = m_{10}/m_{00}$ and $\overline{y} = m_{01}/m_{00}$.
The standard-moments m_{ij} are calculated from

$$m_{i,j} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} x^i y^j s_{seg}(x,y) \tag{5}$$

While the moments mainly describe the global object shape, the profiles provide information about the object details. For each range-bin x, the profiles describe the image variation about the center of mass [6].

$$p_x = \sum_{y=0}^{N-1} (y - \overline{y_x})^2 s_{seg}(x,y) \tag{6}$$

with

$$\overline{y_x} = \frac{\sum_{y=0}^{N-1} y s_{seg}(x,y)}{\sum_{y=0}^{N-1} s_{seg}(x,y)} \tag{7}$$

The feature extractor produces 147 features which are fed to the following processing stage.

C. Classification

Classification is the mapping of feature vectors into the decision space. Simple feedforward network structures using only input and output layers are very restricted in their capability of realizing the required mapping function. The introduction of hidden units give feedforward neural networks the potential for an arbitrary mapping. The desired mapping is trained using the backpropagation algorithm on classified examples. The neural network employed here is a multi-layer perceptron with two hidden layers (128 and 96 neurons respectively) and an output layer with 10 neurons; one output neuron for each class. The resulting output vector is normalized to unity which is required in order to interpret the outputs as probabilities. Using a training data set with 5000 manually segmented images from 10 different objects the classifier was trained for 25 iterations, resulting in an error rate of 5%. After training the generalization capability was tested by an independent data set with 5000 noise free images, which produced an error rate of 13%.

D. Closing the feedback loop: adaptive segmentation

The output o of the classifier and a measure for the homogeneity h of the segmented image is used for controlling the segmentation process. The cost function which has to be minimized is defined as follows:

$$e = (1.0 - max(o))^2 + h^2 \tag{8}$$

$$h = \left| \frac{\sum_{i=1}^{N} \sum_{j=1}^{N-1} s_{seg}(i,j) - s_{seg}(i,j+1)}{\sum_{i=1}^{N} \sum_{j=1}^{N} s_{seg}(i,j)} \right| \tag{9}$$

Starting with only one cluster the object-supercluster is expanded by merging neighboured clusters. The iterative segmentation optimization is stopped when the cost function is at its minimum.

III. Example using noisy input data

In order to test the robustness of the classifier against noise, gaussian white noise was added to the image data.

With conventional global threshold segmentation the classification performance has been shown to be very sensitive to noisy images. Even at moderate signal to noise ratios the classification error increased to 60% and above. Substitution of the global treshold segmentation by the adaptive feature map segmentation resulted in a clearly recognizable improvement of the classification performance. For training of the segmentation feature map one noisy image was sufficient. Once the feature map clusters are trained only the merging of the clusters to one object-supercluster is dynamically controlled in recall mode. The segmentation is nothing more than a two-dimensional treshold comparison. Due to the ordering of the feature map clusters the feature map weight vectors can be considered as a discrete scale of two-dimensional thresh-

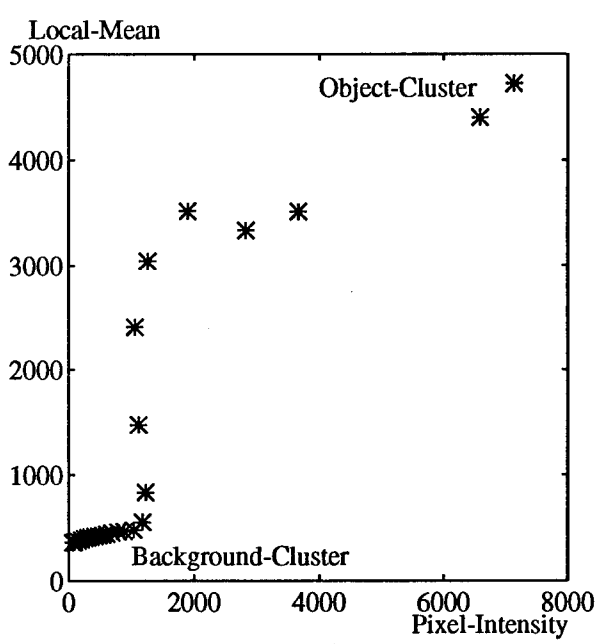


Fig. 6. 2-dimensional threshold scale given by the feature map weight vectors.

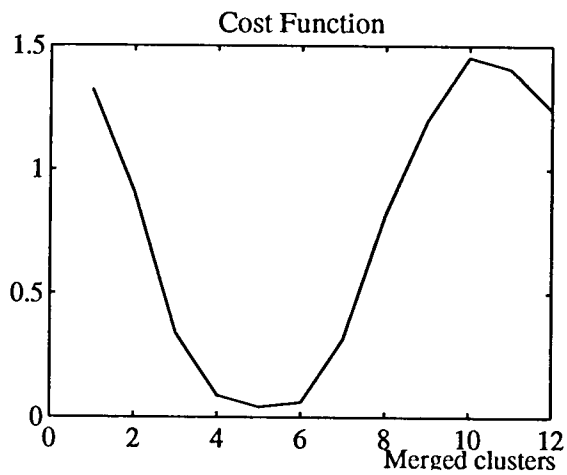


Fig. 7. Cost value for the sequence of subcluster-merging as depicted in fig. 5.

old values (pixel intensity and local mean) (fig. 6). Starting with the highest threshold given by only one cluster, the threshold is decreased by increasing the number of merged clusters until the cost function is minimized (fig. 7).

IV. Conclusion

In this paper we presented the architecture of an automatic object recognition system using a feature map for cluster segmentation and a supervised trained multi-layer perceptron for the classification. In difference to conventional feedforward classification systems, the classifier's decision is fed back to the adaptive segmentor in order to reinforce the classification result. The ordering property of the feature map provides an easily realizable adaptive cluster segmentation. While the segmentation ability was demonstrated for some few examples the feedback mechanism has to be investigated systematically for different signal to noise ratios in the future.

References

[1] M. Roth, "Neural networks for automatic target recognition", *IEEE Transactions on Neural Networks*, , no. 1, 1990.

[2] G.B. Coleman and H.C. Andrews, "Image segmentation by clustering", in *IEEE Proceedings*, 1979, vol. 67, pp. 773 – 785.

[3] J. Koh and S. Bhandakar, "A multilayer kohonen's self-organizing feature map for range image segmentation", *ICNN*, pp. 1270–1276, 1993.

[4] A. Dhawan and L. Arata, "Segmantation of medical images through competitive learning", *ICNN*, pp. 1277–1282, 1993.

[5] Sahibsingh A. Dudani, Kenneth J. Breeding, and Robert B. McGhee, "Aircraft identification by moment invariants", *IEEE Transactions on Computers*, , no. 1, January 1977.

[6] C.M. Bachmann, S. Musmann, and A. Schultz, "Lateral inibition neural networks for classification of simulated radar imagery", *IJCNN*, pp. 115–120, 1992.

TIME SERIES PREDICTION USING GENETICALLY TRAINED WAVELET NETWORKS

Aleš Procházka, Vladimír Sýs
Prague University of Chemical Technology
Department of Computing and Control Engineering
Technická 1905, 166 28 Prague 6, Czech Republic
Phone: +42-2-332 4259, Fax: +42-2-243 11082
E-mail: prochaz@vscht.cz, sysv@vscht.cz

Abstract - The paper presents a contribution to the analysis of wavelet transfer function use in neural network systems and the discussion of some possible learning algorithms of such structures. Wavelets local properties both in time and frequency domains are stated at first giving motivation for wavelet networks application and providing bases for their initial coefficient estimation described recently. The main part of the paper is devoted to the network coefficients optimization using genetic algorithms as an alternative to the gradient descent method. Principles of the evolution techniques are presented for a simple system in this part and then applied for a given time series modelling and prediction.

INTRODUCTION

Problem of nonlinear time series prediction is studied in various disciplines now including engineering, biomedicine and economics [1], [2]. Most approaches consist in the use of adaptive methods [3] and multilayer neural networks [4], [5]. Their modification include wavelet transfer function study [6], [7], [8], [9] and cascade correlation networks [10].

The paper contributes to this problem in the study of links between wavelets and neural networks presented recently [11], [8]. The emphasis is in the analysis of the learning process based upon genetic algorithm use [12], [13] as an alternative to the gradient estimate applied in the backpropagation method. The goal of the paper is to show advantages of genetic algorithm approach in this case.

Methods presented in the paper are verified for simulated series to enable their further use for real signals analysis including those stored in the Signal processing information base [14] and other physiological signals.

0-7803-2026-3/94 $4.00 © 1994 IEEE

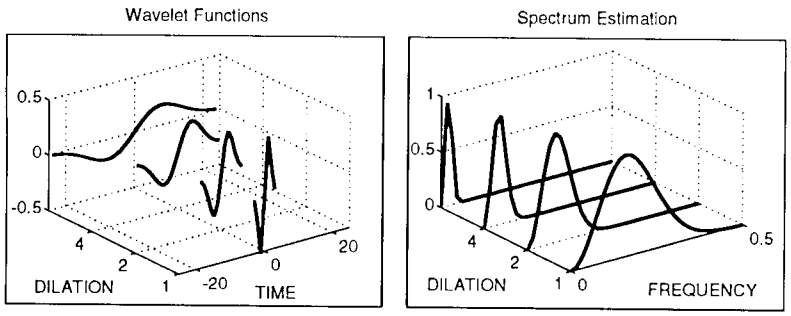


Figure 1:

An example of the wavelet set based on the given initial function $h(x) = -x\, e^{-x^2/2}$ with its spectrum estimation and relation between wavelet dilation and spectrum compression.

WAVELET NETWORK CONSTRUCTION

Wavelet functions and their application in signal processing are described in various papers published recently [15], [16], [17], [18]. These functions in close connection with the wavelet transform can be used very efficiently both for signal analysis as an alternative to the short-time Fourier transform and for signal modelling in wavelet network structures.

Basic properties of wavelet functions include their local influence both in time and frequency domains [19]. The set of continuous wavelet functions is usually derived from the initial (mother) wavelet $h(x)$ and coefficients of dilation (d) and translation (t) defining the function

$$h_{d,t}(p) = h((p-t)/d) = h(w\,p + b) \tag{1}$$

for $w = 1/d$ and $b = -t/d$. Using the signal processing point of view it is possible to consider the initial wavelet as a pass-band filter. Wavelet dilation resulting in its pass-band compression is presented in Fig. 1 for a chosen wavelet function. Properly constructed wavelets allow in this way signal decomposition into different frequency bands and their multiresolution analysis [17].

Basic two-layer neural network structure used for signal prediction is presented in Fig. 2. Signal pattern is used as a neural network input and the whole network is trained to evaluate its output close to the target signal values in the mean square sense. While the output layer transfer function is usually linear in this case the transfer function of the first layer can be nonlinear and sigmoidal functions are often applied. We shall study the alternative use of wavelet functions in such a structure now.

Mathematical description of the wavelet network $R-S1-S2$ is similar to that of sigmoidal neural network system. Summarizing network coefficients

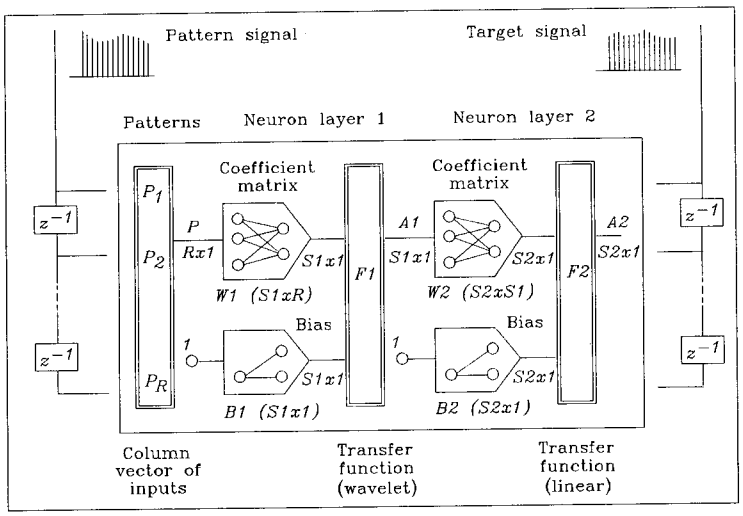


Figure 2:

Signal prediction based on the two layer neuron network having the output vector $\mathbf{A2} = F2(\mathbf{W2} * \mathbf{A1} + \mathbf{B2})$ where $\mathbf{A1} = F1(\mathbf{W1} * \mathbf{P} + \mathbf{B1})$.

in matrices

$$\mathbf{W1}_{S1,R} = \begin{bmatrix} w1_{1,1} & \cdots & w1_{1,R} \\ & \cdots & \\ w1_{S1,1} & \cdots & w1_{S1,R} \end{bmatrix}, \quad \mathbf{W2}_{S2,S1} = \begin{bmatrix} w2_{1,1} & \cdots & w2_{1,S1} \\ & \cdots & \\ w2_{S2,1} & \cdots & w2_{S2,S1} \end{bmatrix}$$

and bias vectors

$$\mathbf{b1} = [b1_1, \cdots, b1_{S1}]', \qquad \mathbf{b2} = [b2_1, \cdots, b2_{S2}]'$$

permits to employ transfer functions $F1$ and $F2$ for the first and the second layer respectively to evaluate network output

$$\mathbf{A2} = F2(\mathbf{W2} * \mathbf{A1} + \mathbf{B2}), \qquad \mathbf{A1} = F1(\mathbf{W1} * \mathbf{P} + \mathbf{B1})$$

While $F2(x) = x$ in many cases of signal prediction there is a choice of the first layer transfer function in various ways including wavelet functions as well.

During the learning process the summed square error deviations between evaluated and target values are minimized. To shorten the iterative learning process various methods of coefficients initial estimates have been suggested [5], [11]. We shall restrict our attention to the study the genetic algorithm approach to the wavelet transfer function case.

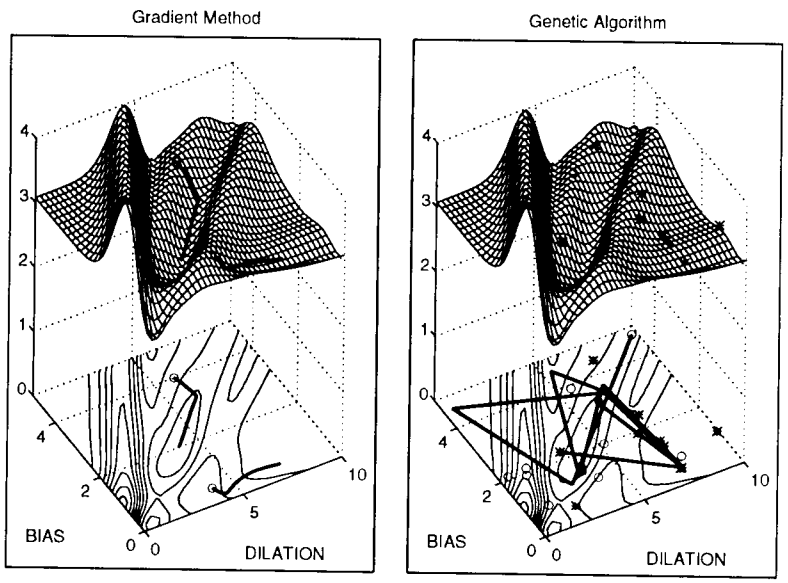


Figure 3:

Comparison of the initial part of the optimization process for the wavelet network having its structure 1−1−1 for a given nonlinear transfer function $F1(x) = -x\ e^{-x^2/2}$ using a gradient search and genetic algorithm optimization approach: **(a)** Gradient search for two different initial estimates, **(b)** Genetic search presenting the initial (o) and final (*) population and the best individual evolution.

SYSTEM COEFFICIENTS OPTIMIZATION

Analyzing basic problems of wavelet network optimization it is possible to start with signal approximation of the given pattern and target values by a single wavelet function $F1(x) = -x\ e^{-x^2/2}$ for unknown dilation a translation coefficients. Error surface in the vicinity of the problem solution is presented in Fig. 3. It is obvious that classical steepest descent approach and closely connected backpropagation method can be very inefficient even in such a simple case.

On the other hand a genetic approach [12], [20] for system coefficients optimization can provide more reliable results even for relatively complicated nonlinear problems. Its general principle resembles a very simplified version of the natural evolution giving better chance to survive to the best individuals while others die out. Basic principles of selection, reproduction, crossover and random mutation can be efficiently implemented in software systems and applied in various engineering problems as well as wavelet networks.

A space of the parameters (dilations, translations) in wavelet network

system presented above can cover the whole region of all possible solutions. A particular set of these parameters defines a network configuration or an individual (in genetic algorithm terminology). Solutions assigned to separate individuals are encoded in binary form to ease the following operations. As all solutions are expected to approximate the given signal it is possible to evaluate the mean square error for all of them standing for the fitness number and giving idea about both the population and each individual performance. Individuals with a relatively high fitness number obtain a higher probability for further reproduction and vice versa. The whole algorithm assumes

- the selection process based upon the roullete wheel providing statistically better chance for reproduction of individuals (defined by corresponding dilation and translation coefficients) with higher fitness number
- the mating operation based upon a random pairs selection and crossover application exchanging substring in selected pairs through a random position choice
- the mutation applied with a given probability and resulting in random switches between zero and one of an individual's element

For a new population fitness numbers are evaluated again and the whole process is repeated.

The process of a single wavelet neuron optimization using a genetic approach is presented in Fig. 3. Comparing initial and final population it is possible to follow the result of the population mean square error decrease. The best individual evolution is presented as well.

The probability values essential for genetic algorithm seems to be very important for the efficiency of the learning process [21], [22]. The mean square error development for two possible approaches considering constant probabilities and exponentially decreasing mutation rate and increasing crossover rate is given in Fig. 4.

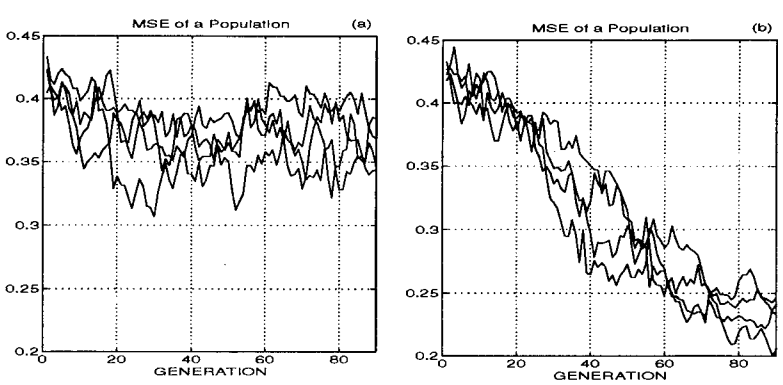


Figure 4:

Comparison of four mean square errors evolutions over the population for genetic algorithm search based on: **(a)** Constant mutation and crossover rates and **(b)** Decreasing mutation rate and increasing crossover rate.

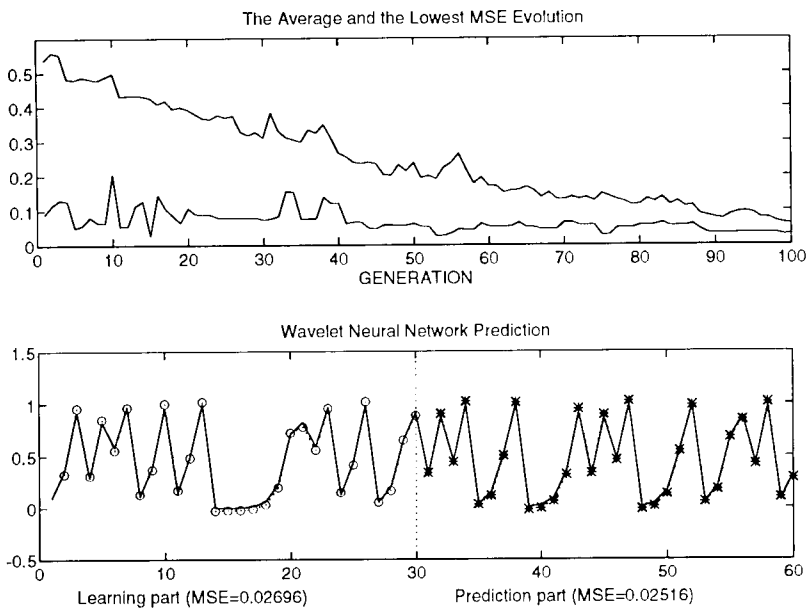


Figure 5:

The mean square error evolution over the whole population of 50 individuals and that of the best individual for a given chaotic process using the wavelet network of structure $1-3-1$ and result of one step ahead prediction during both the learning and testing phase.

SIGNAL MODELLING AND PREDICTION

Nonlinear time series modelling and prediction by wavelet networks has been tested for a common nonlinear time series mentioned in [23] and defined by relation $x(i+1) = \alpha x(i)(1-x(i))$ for $i = 1, 2, 3, \cdots$, $\alpha = 4$ and $x(1) = 0.1$ as a simple example of a deterministic chaos. Applying the network structure presented in Fig. 2 restricted to system $1-S1-1$ the values $x(i)$ as patterns and $x(i+1)$ as targets are considered. The record of the training by the genetic algorithm and prediction results are presented in Fig. 5. During several tests the lower mean square error (MSE) has been achieved during significantly lower number of generations for decreasing mutation rate and increasing crossover rate in comparison to the same results achieved by constant probabilities.

Evaluating further the mean square error in the learning and prediction part of a signal it is possible to compare results for various network structures and for different wavelet functions as well. Signal used to test methods presented in the paper implied values of the mean square error summarized in Tab. I giving these values in the learning part and the prediction part for the wavelet and sigmoidal neural network. Genetic algorithm approach assumed

TABLE I:
COMPARISON OF MEAN SQUARE ERRORS DURING THE LEARNING AND PREDICTION PARTS FOR WAVELET AND SIGMOIDAL NETWORKS.

Neural Network		*Mean Square Error*	
Type	*Structure*	*Learning part*	*Prediction part*
Wavelet Network	1-3-1	0.0270	0.0252
Sigmoidal Network	1-5-1	0.0358	0.0363

population of 50 individuals and 100 generations but similar results were achieved for other choices as well. Errors of the same order were evaluated for the wavelet network of significantly lower number of network variables in comparison with sigmoidal transfer functions use. Genetic algorithms provide in all cases the method for system coefficient estimation.

CONCLUSION

The paper presented the application of genetic algorithms for wavelet network optimization process providing the population of individuals with improving properties evaluated by their fitness numbers. Even though other algorithms including the gradient descent method may converge faster the genetic methods avoid to be captured by local minima owing to the mutation and crossover operations and they are very effective in converging to the global optimum. Using different mutation and crossover rates it is moreover possible to affect the evolution speed. The paper presents comparison of these methods for simulated signals and provides basis for real signals modelling and prediction.

ACKNOWLEDGEMENTS

The paper has been supported by the Department of Computing and Control Engineering of the Faculty of Chemical Engineering at the Prague University of Chemical Technology.

REFERENCES

[1] A. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: Prediction and system modelling. LA-UR-87-2602", Tech. Rep., Los Alamos National Laboratory, USA, 1987.

[2] M. Ohta and S. Nogaki, "Hybrid picture coding with wavelet transform and overlapped motion-compensated interframe prediction coding", IEEE Trans. Signal Processing, vol. 41, no. 12, pp. 3416–3424, December 1993.

[3] S. Haykin, Adaptive Filter Theory, Prentice Hall, Engelwood Cliffs, N.J., second edition, 1991.

[4] J. Hertz, A. Krogh, and R. G. Palmer, Introduction to the Theory of Neural Computing, Addison-Wesley Publishing Company, Redwood City, California, 1991.

[5] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights", in Proceedings of Internationale Joint Conference on Neural Networks, 1990, pp. III/21–III/26.

[6] B. R. Bakshi and G. Stephanopoulis, "Wavelets as basis functions for localized learning in multi-resolution hierarchy", in Proceedings of Internationale Joint Conference on Neural Networks, Baltimore, 1992, pp. II/140–II/145.

[7] Sathyanarayan S. Rao and Ravikanth S. Pappu, "Nonlinear time series prediction using wavelet networks", in Proceedings of World Congress on Neural Networks, Portland, Oregon, July 1993, pp. IV/613–IV/616.

[8] Q. Zhang and A. Benveniste, "Wavelet networks", IEEE Trans. on Neural Networks, vol. 3, no. 7, pp. 889–899, 1992.

[9] D. E. Newland, "Some properties of discrete wavelet maps", Probabilistic Engineering Mechanics, no. 9, pp. 59–69, September 1994.

[10] S. Fahlman et. al, "The cascade-correlation learning architecture, CMU-CS-90-100", Tech. Rep., School of Computer Science, Carnegie Mellon University, USA, 1992.

[11] Q. Zhang, "Regressor selection and wavelet network construction", Tech. Rep., IRISA, France, 1993.

[12] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[13] C. H. Chu and C. R. Chow, "A genetic algorithm approach to supervised learning for multilayered networks", in Proceedings of World Congress on Neural Networks, Portland, Oregon, July 1993, pp. IV/744–IV/747.

[14] D. H. Johnson and P. N. Shami, "The signal processing information base", IEEE SP Magazine, pp. 36–42, October 1993.

[15] O. Rioul and M. Vetterli, "Wavelets and signal processing", IEEE SP Magazine, pp. 14–38, October 1991.

[16] G. Strang, "Wavelets and dilation equations: A brief introduction", SIAM Review, vol. 31, no. 4, pp. 614–627, December 1989.

[17] M. Vetterli, "Wavelets and filter banks: Theory and design", IEEE Trans. Signal Processing, vol. 40, no. 9, pp. 2207–2232, September 1992.

[18] D. E. Newland, "Wavelet theory and applications", in Proceedings of the Third International Congress on Air- and Structure- Borne Sound and Vibrations, Montreal, Canada, 1994.

[19] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis", IEEE Trans. Inform. Theory, vol. 36, pp. 961–1005, September 1990.

[20] B. Müller and J. Reinhardt, Neural Networks, Springer-Verlag, 1991.

[21] D. Whitley and T. Hanson, "Optimizing neural networks using faster, more accurate genetic search", in Proc. 3rd International Conference on Genetic Algorithms, 1989.

[22] D. M. Etter and D. C. Dayton, "Performance characteristic of a genetic algorithm in adaptive IIR filter design", in Signal Processing II, H.W.Schussler, Ed. Elsevier Science Publishers B.V. (North-Holland), 1983.

[23] J. R. McDonnell and D. Waagen, "Evolving recurrent perceptrons for time-series modelling", IEEE Trans. Neural Networks, vol. 5, no. 1, pp. 24–38, January 1994.

A NETWORK OF PHYSIOLOGICAL NEURONS WITH DIFFERENTIATED EXCITATORY AND INHIBITORY UNITS POSSESSING PATTERN RECOGNITION CAPACITY

E.Ventouras[(1)], M.Kitsonas[(1)], S.Hadjiagapis[(1,2)], N.Uzunoglu[(1)], C.Papageorgiou[(3)], A.Rabavilas[(3)], and C.Stefanis[(3)]

(1) Microwave and Biomedical Engineering Laboratory, Dept. of Electrical and Computer Engineering, National Technical University of Athens, 42 Patission str., Athens, 10682, Greece.
Tel.:++301-3616908,Fax:++301-3691206
(2) INTRACOM S.A., Peania, P.O.Box 68, 19002, Greece.
Tel.:++301-6860311,Fax:++301-6644379
(3) Psychophysiology Laboratory, Aiginiteion Hospital, University of Athens, 74 Vassilissis Sofias, Athens, 11528, Greece.
Tel:++301-7217763,Fax:++301-7243905.

INTRODUCTION

The incorporation of properties derived from the function of physiological neurons to artificial networks is actively sought in order to enrich the range of dynamic behavior of the networks. It is subsequently hoped that such "physiological" networks will respond in a way more suitable for pattern recognition applications, than networks distanced from biological reality [1-4].

In the present work a novel neural network is proposed with different excitatory and inhibitory neural populations. This property conforms with the so-called Dale's hypothesis that applies to neurons of the mammalian brain. [5,6]. The network improves the quality of the associative memory abilities shown to exist in an earlier model, where synaptic activity derived from a neuron could be of both types [7].

ARCHITECTURE AND FUNCTION OF THE NETWORK

The model consists of N excitatory (E) and N inhibitory (I) neurons arranged in 2 "parallel" layers. Each E neuron corresponds to an I one. The network is receiving external stimuli through the Input Unit (IU), which projects input patterns to the network. There are 2N input neural units projecting through stable excitatory connections of strength $R_{ij} = \delta_{ij}R$, connecting neuron j of IU with neuron I of the network. The input patterns are identical for both the excitatory and the inhibitory levels. IU is activated with a central frequency $f_{IF} = 1/T_{IF}$, but individual input neurons have a uniformly distributed probability of firing in the temporal window $[(\nu - 1)T_{IF} - \Delta T_p, (\nu - 1)T_{IF} + \Delta T_p]$, for the νth activation cycle.

The state of each neuron i^{Δ}, Δ=E or I, is described by the membrane potential U_i at the soma of the neuron. The equations governing the evolution of the potential are:

$$\frac{dU_i^{\Delta}(t)}{dt} = -\frac{U_i^{\Delta}(t)}{T_R} \cdot \rho^{\Delta}(t)\sigma^{\Delta}(A_i^{\Delta}(t)) \quad \text{if } U_R \le U_i^{\Delta}(t) \le U_T \tag{1.1}$$

$$\frac{dU_i^{\Delta}(t)}{dt} = -\frac{U_i^{\Delta}(t)}{T_R} \quad \text{else} \tag{1.2}$$

where i=1,...,N , Δ=E or I

$\rho^{\Delta}(t)$ is the sensitivity function:

$$\rho^{\Delta}(t) = \Theta(t - t_{Li}^{\Delta} - T_F)(1 - \exp(-\frac{t - t_{Li}^{\Delta} - T_F}{T_F/2})) \tag{2}$$

where $\Theta(x) = x$ if $x \ge 0$ or $\Theta(x)=0$ if $x<0$

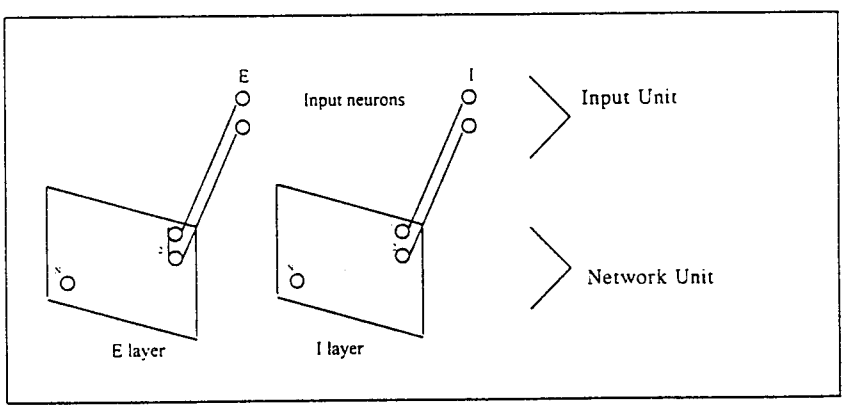


Fig.1 : The network

The component $A_i^{\Delta}(t)$ represents the function of spatiotemporal summation of incoming stimuli at the soma of neuron i^{Δ} and is given by the formula:

$$A_i^E(t) = \sum_{j=1}^{N} S_{ij}^{EE} \exp(-\frac{t - t_{Lj}^E}{T_U^E}) + R\exp(-\frac{t - t_{LPi}^E}{T_U^E}) - f^E\left\{\sum_{j\in C_i} S_{ij}^{EI} \exp(-\frac{t - t_{Lj}^I}{T_U^E})\right\} \tag{3.1}$$

$$A_i^I(t) = \sum_{j\in C_i} S_{ij}^{IE} \exp(-\frac{t - t_{Lj}^E}{T_U^I}) + R\exp(-\frac{t - t_{LPi}^I}{T_U^E}) - f^I\left\{\sum_{j\in C_i} S_{ij}^{II} \exp(-\frac{t - t_{Lj}^I}{T_U^I})\right\} \tag{3.2}$$

where $f^{\Delta}(x) = (1 - \eta^{\Delta})x + \eta^{\Delta}x^2$, Δ=E or I (3.3)

and σ^{Δ} is the activation function described as:

$$\sigma^{\Delta}(A_i^{\Delta}(t)) = \begin{cases} A_i^{\Delta}(t) / A_k^{\Delta} \cdot \lambda^{\Delta} & \text{if } |A_i^{\Delta}(t)| \le A_k^{\Delta} \quad \text{with } 0 < \lambda^{\Delta} \le 1 \\ 1 & \text{if } A_i^{\Delta}(t) > A_k^{\Delta} \\ -1 & \text{else} \end{cases} \tag{4}$$

In the above equations t_{Li}^{Δ} is the time when the last firing of neuron i^{Δ} is communicated to other neurons and t_{LPi}^{Δ} the time of the last activation of neuron i_P^{Δ} of IU. $S_{ij}^{\Delta_1\Delta_2}$ is the synaptic strength of the synapse connecting neuron j, belonging to category Δ_1, to neuron I, belonging to category Δ_2. U_T is the threshold potential and U_R the refractory potential. In case U_i reaches U_T the neuron fires and its potential is set to U_R.

Each neuron i^E may exert its influence to all other E neurons of the network but only to the I neurons which belong to its vicinity C_i, defined by the maximum acceptable distance between i^E and j^I in the parallel planes. The same restriction applies to all projections emanating from I neurons.

In the network used it was $T_U^I < T_U^E$ and $\sigma^E \neq \sigma$ which enables U^I to evolve faster than U^E. This is extremely crucial for the proper function of the network since the early firing of I neurons prevents the firing of unwanted E neurons. It should be noted that the state of the network is defined by the E neurons solely. This assumption is also physiologically motivated since in the human neocortex the main research interest concerns long range connections of excitatory neurons. The parameter $a(t) = a^E(t) = \sum_{i=1}^{N} \exp(-\frac{t - t_{Li}^E}{T_U^E})$ represents the overall activity of the network.

The synapses of the network are modified only during learning sessions. Each memory pattern is embedded in a separate session. The pattern is projected through IU. The close temporal association of the activation of IU neurons belonging to the pattern is the crucial factor for proper memorization. Furthermore neurons not belonging to the pattern should not be activated coherently with pattern neurons. The training algorithm can be classified as a generalized Hebb rule since it enables the strengthening of synapses of concurrently active neurons and decreases the strength of synapses between neurons who show uncorrelated temporal behavior.

NUMERICAL RESULTS

We present the case of two patterns described by the set of neuron A={1-10} and B={9-18}, in a network of N=100 E and I neurons. during the learning phase the parameters governing the evolution of the network are:
T_U^E=1.0ms, T_U^I=0.5ms, T_F=5.0ms, U_T=30mV, U_R=-15mV, λ^E=0.0189, λ^I=0.0114, A_k^E=15.5, A_k^I=6.05, η^E=0, η^I=0.9, P=1400, Δt_p=0.5ms, f_{IF}=50Hz.
The memorization of each pattern is achieved at approximately 600ms, and is terminated when all the synapses have converged to stable values, which can only change in subsequent learning phases.
In the recall phase the following parameters are modified: R=210, λ^E=0.087, λ^I=0.0110, f_{IF}=33.3Hz. When the input pattern (ip) was the set of neurons

(ip)={3,4,5,6,7,8,9,10,13,20,35} the recall of memory A was excellent. By denoting the emergent network pattern as (enp) it was:

$(m_{(enp),A}=1) > (m_{(ip),A}=0.95) > (m_{(ip),B}=0.85)$ where $m_{p_{1,p2}} = \sum_{i=1}^{N} i_{P1} i_{P2}$ for 2 patterns P1 and P2. In Fig.2 we present the quantity $a_E(t)$ during the recall phase.

The network proposed possesses the ability to store correlated-overlapping patterns and to categorize properly incomplete and noise corrupted inputs. Furthermore it has the ability to function at realistically low firing rates, such as those found in the cortex (15-40Hz), and to store sparse patterns [4,8,9]. Current investigations concern the assessment of the storage capacity of the network and its application in Massively Parallel Computers of the MIMD (Multiple Instructions - Multiple Data) type.

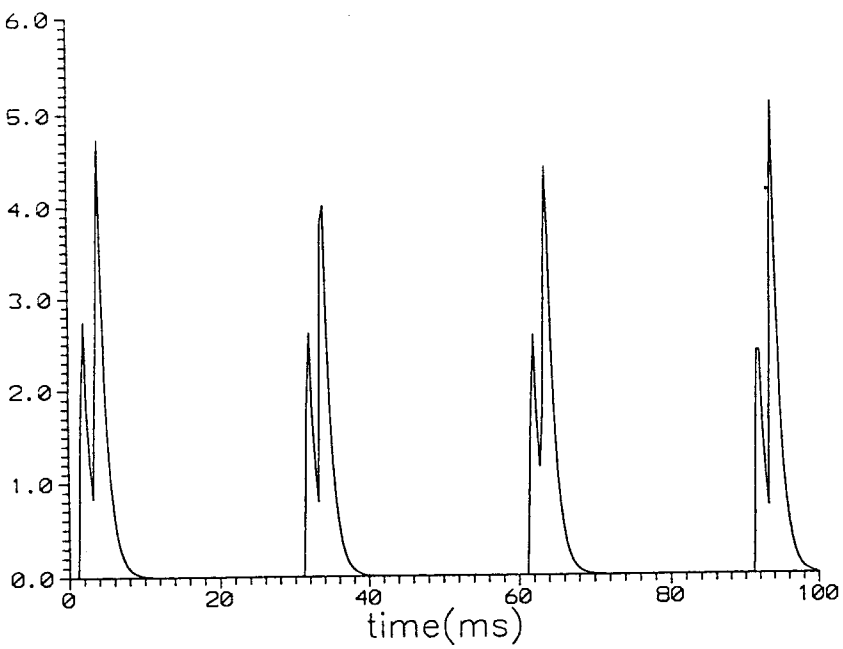


Fig.2 : Parameter $a_E(t)$ representing the overall activity of the network in the recall phase

REFERENCES

[1] D.J.Amit, *Modelling Brain Function - The World of attractor neural networks*, New York: Cambridge University Press, 1989.

[2] J.Buhmann and K.Schulten, "Associative Recognition and Storage in a Model Network of Physiological Neurons", Biol.Cybern., vol.54, pp.319-335, 1986.

[3] J.Buhmann and K.Schulten, "Influence of Noise on the Function of a "Physiological" Neural Network", Biol.Cybern., vol.56, pp.313-327, 1987.

[4] J.Buhmann, "Oscillations and low firing rates in associative memory neural networks", Phys.Rev.A, vol.40, pp.4145-4148, 1989.
[5] J.C.Eccles, "Developing Concepts of the Synapses", J.Neuroscience, vol.10(12), pp.3769-3781, 1990.
[6] E.Ventouras, C.Papageorgiou, A.Rabavilas, N.K.Uzunoglu, and C.Stefanis, "An Event-Related Potentials Generator Model Based on An Artificial Neural Architecture Incorporating Dale's Hypothesis", 9th World Congress of Psychiatry, Rio de Janeiro, 1993.
[7] E.Ventouras, C.Papageorgiou, N.K.Uzunoglu, A.Rabavilas, and C.Stefanis, "An Attractor Network Model for the Generation of Event-Related Potentials using Integrative Synapses", in ICANN'93-Proceedings of the International Conference on Artificial Neural Networks, Amsterdam, 1993, pp.698-703.
[8] B.M.Forrest and A.Loetgers, "Neural Networks with low activity levels", Z.Phys.B, vol.86, pp.309-315, 1992.
[9] C.J.P.Vincente and D.J.Amit, "Optimized network for sparsely coded patterns", J.Phys.A, vol.22, pp.559-569, 1989.

Learning With Imperfect Perception

W. Wen and M. Yokoo
NTT Communication Science Laboratories
2-2, Hikaridai, Seika-cho
Soraku-gun, Kyoto 619-02 Japan

Abstract

Machine learning algorithms which adopt a state space representation usually assume perfect knowledge of what state the system is currently in. This is to guarantee that rewards and penalties are correctly assigned to the responsible state. This assumption, however, does not hold in most real world learning problems due to imperfect perception. In this paper estimation and control theory is used to classify the systems depending on the observability of the system states. This observability determines whether the optimal control strategy of a particular system can be learned, A novel approach based on enhancing the observability is used to deal with perceptual aliasing problem. In order to *learn to perceive*, the perception actions are directly integrated into the control actions. An example is shown and further applications to robot learning is discussed.

1 Introduction

Reinforcement learning algorithms learn an optimal control policy through trial and error. The basic formulation of the problem is closely related to decision theory. Given a dynamic system described by state vector $x \in X$, a policy $a = f(x)$, where $a \in A$ is the action vector, and reward $r_t \in R$ at timestep t, the total reward(or utility) U for each $x \in X$, following the policy f can be calculated as follows, $U(x) \equiv \sum_{t=0}^{\infty} \gamma^t r_t$, where $\gamma \in (0, 1]$ is the discount factor. The problem for the decision theory is to find the optimal policy $f(\cdot)$ which gives an overall maximum reward for each state x. Apart from very simple and restricted cases, the above problem can not be solved analytically. Numerical methods such as dynamic programming can deal with such problems in an off-line manner. In the more general case in which the system must operate in real time, i.e. explore to be rewarded or penalized, no policy can be found before the system start operating. Two well known reinforcement learning algorithms are AHC(Adaptive Heuristic Critic) learning[7, 8], which learns an evaluation function of states, and Q-learning[9], which learns an evaluation function of state-action pairs. There are no essential differences between these two algorithms and their convergence can be proved given that the certain conditions are satisfied[10].

0-7803-2026-3/94 $4.00 © 1994 IEEE

In most of the real world situations, imperfect perception of the states leads to the violation of the Markov condition. These problems have been addressed in work related to active perception[12] and learning with hidden states[3]. In active perception certain state information may be lost due to the time or cost constraints of sensing operation. In problems with hidden states reward or penalty can not be correctly assigned. In both cases the Markovian process condition is violated due to the undeterministic feature of state transition. In [12] the *lion algorithm* is proposed in which a reinforcement leaning agent can learn to focus perceptual attention on the right aspect of the environment during control. Problems with hidden state can also be dealt with using memory methods.

In this paper we attempt to address these problem in the framework of statistical estimation and control. The rest of the paper is organized as follows. In section 2 the limitations of the reinforcement learning algorithms with respect to the observability and controllability of the underlining system is discussed. In section 3 a simple example is introduced to illustrate the problem of learning with imperfect perception. In section 4 a novel approach is proposed to deal with the problem of learning with perceptual aliasing and hidden states. In section 5 we propose to integrate estimation and control for reinforcement learning and show how it can be done using the same example. In section 6 we consider possible applications to real world learning problems.

2 Learning as Estimation and Control

Traditionally the estimation and control of a dynamic system has always been separated: optimal estimation algorithms are applied to perceive the real world and control strategies are designed based on the optimal estimated values. Learning algorithms aim to approximate an optimal control strategy for a dynamic system instead of designing it using certain optimal criterion. In learning, the estimation part is usually omitted through simulation or an direct approach such as the neural networks are used. In this section, we attempt to categories various learning methods in the framework of the estimation and control.

2.1 Categorizing Learning Algorithms

In real world learning problems, states must be perceived before reward or penalty can be assigned. In this paper we categorize learning algorithms into three classes depending on how perception is dealt with in the learning process:(a) real world learning; (b) simulated learning; (c) direct learning.

In the most general case, a state space representation is constructed to model the real physical process. Let X be the vector describing the real state of the world and $\hat{X}$ its estimation. It must be noted that only this estimated value of the real state is available to the learning algorithms. Perceived state and reward are fed into the control policy which in turn chooses appropriate actions to change the state of the world. This is referred to as real world learning and

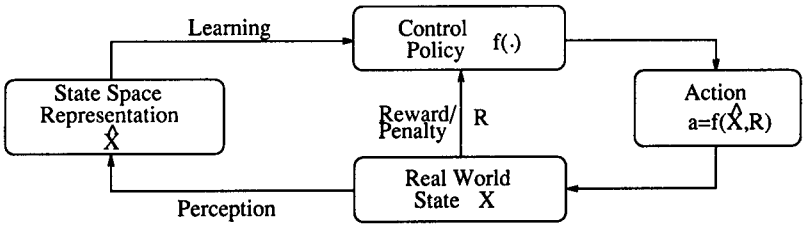


Figure 1: Real world learning

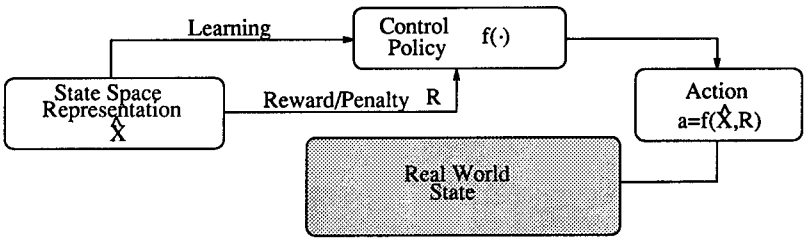


Figure 2: Simulated learning

is illustrated in Figure 1.Up to date, work in this field has been few due to the complex interaction between perception and learning[12].

Most of the algorithmatic machine learning methods assume perfect state perception. These methods have been successful in obtaining tractable algorithms with proven convergence given some restrictions. Q-learning[9] is regarded as one of the most effective algorithms for reinforcement learning. These will be referred to as the simulated learning algorithms since perfect perception can not be assumed in real world. Figure 2 shows such learning systems.

The last category is referred to as the direct learning algorithms. Neural network based approach belongs to this class. No state space representation is needed. Sensor data is directly fed to the control policy and appropriate actions are chosen to affect the state of the world. This is illustrated in Figure 3. It could be argued that the perception is somewhere embedded in the neural networks, but we will show in the next section that why this is not the case.

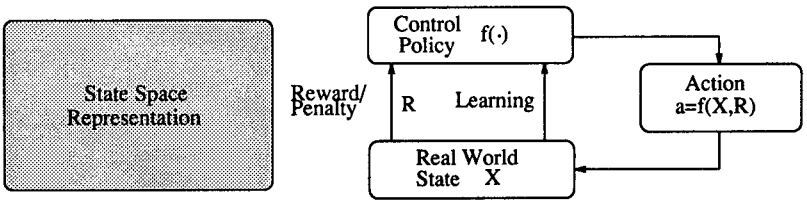


Figure 3: Direct learning

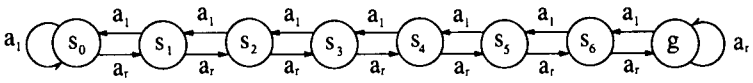


Figure 4: 1-dimensional traversing problem

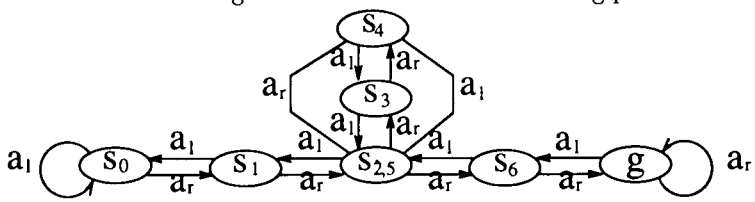


Figure 5: 1-d traversing problem with aliased states

3 Perceptual Aliasing and Hidden States Problem

The problem of learning with imperfect state perception has been investigated in [12, 3]. In both cases there exist discrepancies between the real states and perceived states. They are referred to as the perceptual aliasing problem or hidden states problem. In this paper the 1-D traversing problem proposed by Whitehead[12] will be used to illustrate the difficulties associated with learning with imperfect perception. This example is used again to show how the proposed *learn to perceive* approach in this paper can be used to solve this problem.

3.1 A Simple Example

Consider the task shown in Figure 4. In this task, the real world consists of eight states, $S_R = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, g\}$; two actions, $A = \{a_l, a_r\}$; and a deterministic transition function. Upon reaching the goal state g the agent receives an award $R(g) = 5000$. $R(s_k) = 0$ for $k = 0$ to 6. Assuming that the sensing system is unable to distinguish s_2 from s_5, how would the learning be affected? The new system is illustrated in Figure 5.

The original problem shown in Figure 4 can be learned using 1 step Q-learning algorithms. Figure 6 shows the learned Q value for action a_r and a_l for all states. A normal backpropagation three layer neural network is used. In fact, the optimal policy is obviously to take action a_r in all states.

However, if the sensor fails to distinguish s_2 and s_5, it can be shown that the 1-step Q-learning can not learn this policy although the optimal policy remains the same. In this case the learning algorithm will be unstable due to the interaction of the aliased states. For a detailed discussion see [12]. This phenomenon, however, is not limited to the simulated algorithms. In [3], backpropagation neural network based learning algorithms also fail to learn the optimal strategy in the "two-cup collection" problem.

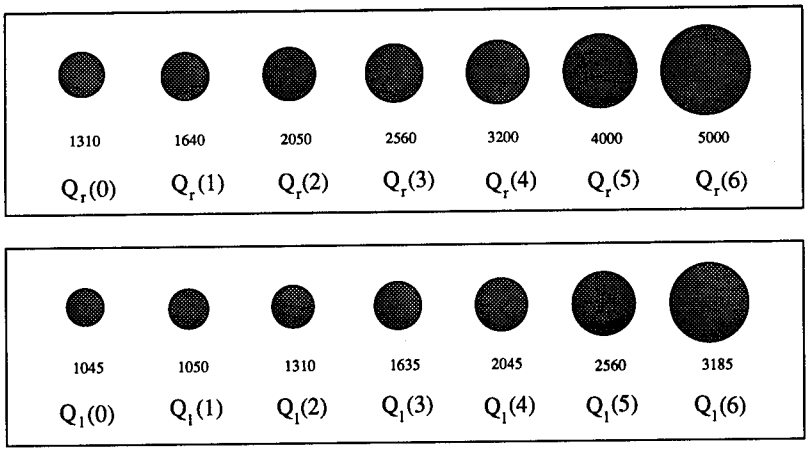


Figure 6: Learned Q-value for action a_r and a_l

3.2 Why it can not be learned?

In [12], Q-learning equations are analysed to show the unstable learning results. However, the fundamental reason for this problem lies in the estimation and control theory[4, 1, 2]. Systems such as the 1-D traversing problem shown in Figure 5 with aliased states are not fully observable[1] thus em uncontrollable in general. Assuming an optimal policy does exist but the optimal actions associated with the aliased states are different. Unlike the original example where optimal strategy exists even though aliased states exist, it is impossible to have an optimal action for the two aliased states since the two optimal actions for the real states are different.

The same rule applies to direct learning algorithms such as neural network based learning methods. If perceptual aliasing or hidden states exist, the input data for the neural networks does not contain the full dimension of the real state space, *i.e.* the real state is unobservable. In general such system is uncontrollable.

Learning algorithm learns the optimal control strategy through trial and error. However, if the system is uncontrollable, it is impossible to learn the optimal strategy even if it exists.

More often the system described by certain state space representation may be probabilistically observable. For example, a multi-sensor system provides a set of alternative perceptions. It is possible that the integration of multi-sensor data can increase the observability thus making it possible to learn the optimal or sub-optimal control strategies. Similar arguments goes to the memory methods used in [3]. Incorporation of past data into the input vector of the neural network will increase the observability of the system thus learning may become possible.

[1] The dimension of the state space representation is smaller than that of the dimension of the real problem space.

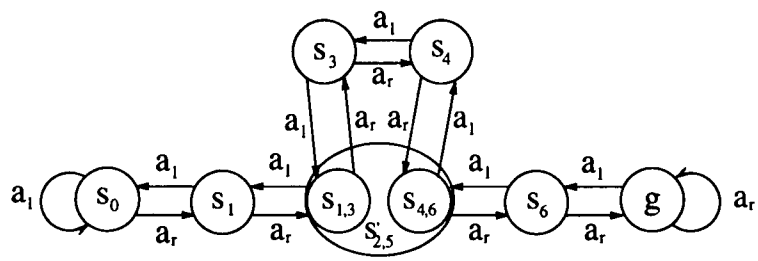


Figure 7: Observability through state space representation restructuring

4 Learning with imperfect perception

Although in [12] a *lion algorithm* is developed to deal with perceptual aliasing problem, it was not applied to the 1-D traversing problem. In fact, it is not clear how the *lion algorithm* can be used in general to deal with the perceptual aliasing problem as it is highly dependent on the sensing system. As it is clear from the point of view of the estimation and control theory, learning can only be possible if the system becomes observable.

In this section we present two novel approaches to deal with this problem. In the first approach the state space representation is restructured so that the system described by the new state space representation is fully observable. The second approach deals with the enhancement of the system observability through incorporating past observations.

4.1 Restructuring State Space Representation

In order for the system described in the 1-D traversing example to be observable, one more dimension must be added to the state space representation. Let us assume that the agent always remembers the previous state when it enters a new state. Furthermore let the aliased state $s'_{2,5}$ be divided into two states $s_{1,3}$ and $s_{4,6}$ depending on the whether the previous state is s_1, s_3 or s_4, s_6. This is illustrated in Figure 7. Note that this representation is almost identical to the original one in Figure 4 and the optimal strategy can be learned. However, if the agent is not allowed to know the previous state, the system will remain unobservable, thus no optimal strategy can be learned.

4.2 Memory Methods

The system observability can be improved by incorporating the past observations. Depending on the system dynamics, different window sizes can be used to incorporate past observations. In [3] three memory architecture are proposed to solve a learning problem with hidden states. A simple, recurrent element is added to our neural network learning algorithm to tackle the 1-D traversing problem with aliased states.

It must be noted that the level of recurrent or the size of the memory window depends on the dynamics of the system and can not be known accurately and there is no guarantee that it is always possible to construct such an neural network architecture so that the system becomes fully observable. It seems that, although the observability is a necessary condition for the optimal control strategy to be learned, it is not clear how full observability can be achieved in general case. The more likely scenario is the so-called probabilistically observable systems. In the next section we attempt to address this problem and show how learning can still be possible under such conditions.

5 Is It Possible to Learn to Perceive?

In previous sections we have shown that the state representation must satisfy the observability condition. Otherwise optimal strategies can not be learned in general. However, most of the real world learning problems can be best described as probabilistically observable. Take the 1-D traversing problem for example, assuming that the sensor can recognize state s_2 and state s_5 with certain probability, is learning possible under such condition?

As discussed previously, in the most general case, learning must deal with perception and other control actions at the same time. Mixing perceptual and control action violates the Marcov condition, hence it is not possible in general to learn the optimal strategy for such systems. In [12] Whitehead developed the so-called *lion algorithm* to learn and use an internal representation that is complete and consistent. More specifically the learning cycle is divided into two distinct phases: state identification and overt control. This falls into the traditional estimation and control paradigm, i.e. the separation of estimation and control. Inconsistent perceptual policies are detected by monitoring the sign in the estimation error in the 1-step Q-learning rule.

A detailed look at the *lion algorithm* shows that, as long as there exists a consistent representation among the alternative ones, this consistent state representation can be learned through the detecting of the inconsistent states. This observation is significant in that, even though inconsistent states are used during learning, consistent state and optimal strategy can be learned provided that there exists a consistent representation among the alternatives.

Based on the above observation, we propose an algorithm to fully integrate perception and learning and we call this *learn to perceive*. From the estimation and control theory point of view, if consistent state representation exists among the alternative representations, the system can be described as probabilistically observable. Intuitively, if we augment the action set by perturbating the alternative perceptual actions and coupling them with the real control actions, the Q-value for actions associated with inconsistent perceptual actions will lead to more costly paths. If enough trials are performed, the actions which are associated with consistent perceptual actions will be selected. From the reinforcement learning point of view, the reward is directing the choice of consistent perceptual actions as well as the choice of optimal control actions.

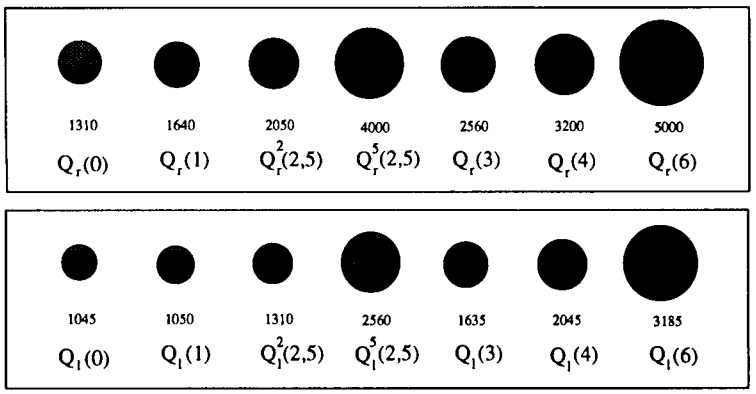


Figure 8: Utility function with perception as action

This approach represents a conceptual change in learning as well as in estimation and control theory. This shows that it is possible to fully integrate perception and control. Again we use the 1-D traversing example to illustrate the algorithm. Assuming the sensing system recognize states s_2 and s_5 correctly 50% of the time[2]. We augment the actions for $s'_{2,5}$ as follows. Instead of using $\{a_r, a_l\}$, which is the same as the actions in other states, we define $A'_{2,5} = \{a_{2,r}, a_{2,l}, a_{5,r}, a_{5,l}\}$. $a_{2,l}$ denote the action of choosing perception s_2 and traverse left and $a_{2,r}$ denote the action of choosing perception s_2 and traverse right and so on. Consequently, $Q_l^2(2,5)$ is the Q value for taking action $a_{2,l}$ in $s'_{2,5}$ and $Q_l^5(2,5)$ is the Q value for taking action $a_{5,l}$ in $s'_{2,5}$ etc. The utility function for the six states(including the aliased state) is shown in Figure 8.

6 Relevance to robot learning

Robot control can be regarded as the ultimate estimation and control problem. The robot must perceive the dynamic environment in real time and control its motors to move itself to new locations. Usually certain optimal estimation algorithm, such as the Kalman filter[11], is used to locate the robot and its surroundings. Pre-defined navigation strategies are applied to direct the robot to its goal. In reality, strategies thus constructed do not work well and hardly optimal due to the intractable interactions and uncertainties. Recently learning has been considered as a way to gradually adapt strategies in order to achieve better performance.

Our previous investigation into the problem of learning with imperfect perception has concluded that, observability is essential for the successful learning of optimal control strategies. This observability is the guarantee that consistent state representation can be found either through the incorporation of past

[2]This probability does not affect the existence of the solution but the time to find one

observations or detecting inconsistent states. For robot control problems, in which multiple sensors provide a large amount of competing and complementing information, learning to perceive is of great importance. Unlike the problem discussed previous, robot control problems concerns real physical values which are continuous. So far the question of existence and convergence of learning in continuous domain has not be satisfactorily answered. Integrating sensing with learning in continuous domain is, therefore, a very hard problem.

Various researchers have investigated robot learning problems [3, 5, 6]. In addition to the discrete maze-like problems, navigation and survival problems in continuous state spaces are also been investigated[3]. Usually the continuous state space is represented by neural networks. However, the set of actions are usually discrete. Reinforcement learning has been applied in simulated robot world but It is not clear that how learning can be conducted using a real robot.

Learning with imperfect perception is intrinsic to robot learning due to the large and diverse sensor data available to the learning agent. Can the *learn to perceive* algorithm proposed in the previous section be used in robot learning? First we must bear in mind that learning has always been associated with a finite number of strategies(value function based learning) and finite number of state-action pairs(Q-learning). Therefore it is natural to limit *learn to perceive* algorithm to deal with discrete number of perceptual strategies. Take a mobile robot fitted with sonar and infra-red sensors as an example, we propose only three sensing strategies: use sonar data; use infra-red data; use the average of both. These strategies are then perturbed and coupled with the real robot motion control action and their Q-values can be learned. Eventually the motion associated with certain sensing strategy will be selected due to the highest reward it obtains. This, however, has not been implemented and the effectiveness remains to be seen.

7 Conclusions and Future Work

In this paper we have attempted to address the problem of learning with imperfect perception from the estimation and control theoretic point of view. Essential to this problem is the observability of the state space representation. If certain aspect of the system is unobservable due to misrepresentation or sensor failure, reinforcement learning can not learn the optimal strategy for controlling such system. Two novel approaches, state space restructuring and memory method have been proposed to enhance the observability. Furthermore, an algorithm which directly integrate perception and learning is proposed. In this algorithm, perceptual actions are paired up with actual control actions and their utility values estimated by reinforcement learning. Consistent perceptual strategy as well as optimal control strategy can be learned at the same time.

However, it must be pointed out that only finite number of perceptual strategies can be used together with finite number of control strategies. One problem we propose to address in future work is whether a sub-optimal strategy can be learned if, given that the system is probabilistically observable, none of the per-

ceptual strategies alone provide consistent state space representation. Another aspect of our future work will be the implementation of the *learn to perceive* algorithm on real robot learning scenarios, such as tracking and navigation.

8 Acknowledgment

The authors wish to thank Tsukasa Kawaoka and Ryohei Nakano for their support during this work at NTT laboratories and Toru Ishida for helpful discussions.

References

[1] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association.* Academic Press, 1988.

[2] D. E. Catlin. *Estimation, Control, and the Discrete Kalman Filter.* Springer-Verlag, 1989.

[3] L. Lin. *Reinforcement Learning for Robots Using Neural Networks.* PhD thesis, Carnegie Mellon University, 1992.

[4] P. Maybeck. *Stochatic Models, Estimation, and Control,* volume 1. Academic Press, 1979.

[5] J. R. Millan and C. Torras. A reinforcement connectionist approach to robot path finding in non-maze-like environments. *Machine Learning,* 8:363–395, 1992.

[6] A. W. Moore. *Efficient Memory-Based Learning for Robot Control.* PhD thesis, Cambridge University, 1990.

[7] R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts, 1984.

[8] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning,* 3:9–44, 1988.

[9] C. J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, Cambridge University, 1989.

[10] C. J. C. H. Watkins. Technical note: Q-learning. *Machine Learning,* 8:279–292, 1992.

[11] W. Wen. *Multi-sensor Geometric Estimation.* PhD thesis, University of Oxford, 1992.

[12] S. D. Whitehead. *Reinforcement Learning for the Adaptive Control of Perception and Action.* PhD thesis, University of Rochester, 1992.

A LEARNING ALGORITHM FOR MULTI-LAYER PERCEPTRONS WITH HARD-LIMITING THRESHOLD UNITS

Rodney M. Goodman and Zheng Zeng
Department of Electrical Engineering, 116-81
California Institute of Technology
Pasadena, CA 91125
Tel: (818)395-3677, FAX: (818)568-3670
Email: rogo@micro.caltech.edu

Abstract — We propose a novel learning algorithm to train networks with multi-layer linear-threshold or hard-limiting units. The learning scheme is based on the standard back-propagation, but with "pseudo-gradient" descent, which uses the gradient of a sigmoid function as a heuristic hint in place of that of the hard-limiting function. A justification that the pseudo-gradient always points in the right down hill direction in error surface for networks with one hidden layer is provided. The advantages of such networks are that their internal representations in the hidden layers are clearly interpretable, and well-defined classification rules can be easily obtained, that calculations for classifications after training are very simple, and that they are easily implementable in hardware. Comparative experimental results on several benchmark problems using both the conventional back-propagation networks and our learning scheme for multi-layer perceptrons are presented and analyzed.

1 INTRODUCTION

Single-layer networks of linear threshold units (or hard-limiting units) known as perceptrons have been shown to have very limited learning capacity [2]. Although multi-layer systems of such units are much more powerful than single-layer ones, there has been no known learning algorithm for such networks.

In recent years, networks with continuous, nonlinear activation functions have been shown to be able to perform much more complicated tasks than single-layer perceptrons. With the differentiable activation functions, gradient descent can then be used to train such networks [4].

However, the internal representations of these networks have been hard to analyze, due to the fact that their activation spaces are continuous, and high dimensional. Multi-layer perceptron networks are thus still of interest. In addition to easily understandable internal representa-

0-7803-2026-3/94 $4.00 © 1994 IEEE

tions, classification rules can be readily obtained from trained perceptron networks, the operations of the networks after being successfully trained are extremely simple, and they are easy to implement in hardware.

In this paper, we attempt to solve the problem of training multi-layer hard-limiting-unit networks by using non-zero values for logic 0's and 1's, and by a pseudo-gradient descent learning scheme. Henceforth, these networks will be called interchangeably, as discrete networks or perceptron networks throughout this paper.

2 NETWORK ARCHITECTURE

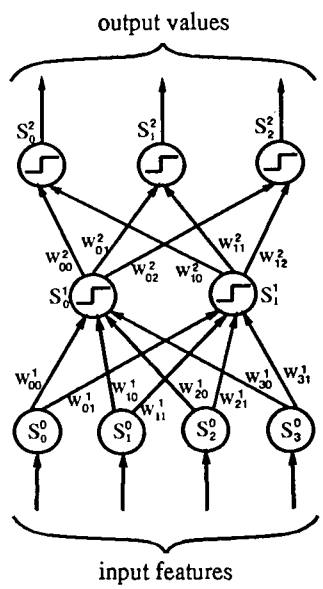


Figure 1: A network of perceptrons with a single hidden layer

Shown in Fig. 1 is a two-layer network of hard-limiting units. Note that since the output layer is "discretized", such networks are therefore used for classification or encoding problems. We use $S_i^{(l)}$ to denote the output value of unit i in layer l, where the 0th layer is defined to be the input layer, and $w_{ij}^{(l)}$ to denote the weight connecting from unit j in layer $l-1$ to unit i in layer l. The operational equations for the network are:

$$S_i^{(l)} = D_0(\sum_j w_{ij}^{(l)} S_j^{(l-1)}), \qquad \forall l, i, \tag{1}$$

$$where \;\; D_0(x) = \begin{cases} 0.8 & \text{if } x \geq 0.0 \\ 0.2 & \text{if } x < 0.0. \end{cases} \tag{2}$$

Note that the values 0.2 and 0.8 are used here instead of 0 and 1 in order for logic "0"s to have some power of influence over the next layers. These values play an important role in the pseudo-gradient learning which is explained in the following section.

3 PSEUDO-GRADIENT LEARNING AND ITS JUSTIFICATION

Our learning scheme is based on the standard back-propagation method [4], but with "pseudo-gradient" descent instead of gradient descent on the error surface. A learning method based on a similar idea for training recurrent networks was first introduced in [6, 7].

To explain the pseudo-gradient, we need to introduce another set of values for the output and hidden layers, which we will call the analog values of the units, as opposed to the discrete (hard-limited) values that are actually used during network operations:

$$h_i^{(l)} = f(net_i^{(l)}), \qquad \forall l, i, \tag{3}$$

where

$$net_i^{(l)} = \sum_j w_{ij}^{(l)} S_j^{(l-1)} \tag{4}$$

and

$$f(x) = \frac{1}{1+e^{-x}}. \tag{5}$$

From (1) to (5), it is obvious that

$$S_i^{(l)} = D(h_i^{(l)}),$$

where

$$D(x) = \begin{cases} 0.8 & \text{if } x \geq 0.5 \\ 0.2 & \text{if } x < 0.5, \end{cases}$$

For the input layer, define $h_i^{(0)} = S_i^{(0)}$ to be the ith input.

Let L be the output layer, the error function for an input pattern is defined to be:

$$E = \frac{1}{2}\sum_i (h_i^{(L)} - t_i)^2,$$

where t_i is the desired value for output unit i. For classification and encoding problems, t_i is either 0 or 1.

In a manner similar to back-propagation [4], the error "gradient" with respect to each weight is computed, but instead of the true gradient, we compute a value which we define to be the "pseudo-gradient":

$$\frac{\widetilde{\partial E}}{\partial w_{ij}^{(l)}} = \widetilde{\delta_i^{(l)}} S_j^{(l-1)}, \qquad \forall l, i, j, \tag{6}$$

where

$$\widetilde{\delta_i^{(l)}} = \frac{\widetilde{\partial E}}{\partial net_i^{(l)}}$$

$$= \begin{cases} f'(net_i^{(l)})(h_i^{(l)} - t_i) & \text{if } l = L \\ f'(net_i^{(l)}) \sum_k \widetilde{\delta_k^{(l+1)}} w_{kj}^{(l)} & \text{otherwise.} \end{cases} \tag{7}$$

Here $\frac{\widetilde{\partial x}}{\partial w_{ij}^{(l)}}$ is what we call the "pseudo-gradient" of x with respect to $w_{ij}^{(l)}$.

Note that from (1), (2) and (6), by making the possible values of $S_j^{(l-1)}$ to be 0.2 and 0.8, instead of 0 and 1, the pseudo-gradient $\frac{\widetilde{\partial E}}{\partial w_{ij}^{(l)}}$ will not be reduced to 0 when $S_j^{(l-1)}$ is in the "off" (or logic 0) mode, thus the heuristic hint provided by $\widetilde{\delta_i^{(l)}}$ will not be eliminated.

Note also that had we computed the true gradients, the only thing that would have been different in the pseudo-gradient formulae (6) and (7) is that the term $f'(net_i^{(l)})$ in the "otherwise" case in (7) should have been $D_0'(net_i^{(l)})$. However, $D_0'(x)$ is zero everywhere and non-existent at $x = 0$. By using f' instead of D_0', we provide in essence a heuristic hint of which direction in x a step up (or down) of $D_0(x)$ is, and also of how far away it is from x.

Consider the case of a single-hidden-layer network. Since for the "output layer" case, i.e., $l = L = 2$, the pseudo-gradient is in fact the same as the true gradient, the "inaccuracy" of the pseudo-gradient only exists in one layer, that is, the hidden layer ($l = 1$), thus in the "otherwise" case in (7), $\widetilde{\delta_k^{(l+1)}}$ is the true $\delta_k^{(l+1)}$ from straight back-propagation. Therefore, $\sum_k \widetilde{\delta_k^{(l+1)}} w_{kj}^{(l)}$ gives us the true value of $\sum_k \delta_k^{(l+1)} w_{kj}^{(l)}$, and since $f'(net_i^{(l)})$ is always positive, $\widetilde{\delta_i^{(l)}}$ truly gives us a good indication of the direction, distance or size of a step up (or down) in the discontinuous error surface E as a function of $net_{ij}^{(l)}$, as does $\frac{\widetilde{\partial E}}{\partial w^{(l)}{}_{ij}}$ give a similarly good indication in E as a function of $w_{ij}^{(l)}$.

4 EXPERIMENTAL RESULTS

Shown in Tables 1 through 4 are comparative experimental results of using both the proposed discrete network training method and the standard back-propagation on the following bench mark problems, respectively: exclusive or, iris data classification [1], sonar data classification [3] and NETtalk [5]. All experiments are done with two-layer networks. Detailed parameters are described in the corresponding captions.

# of hidden units	*discrete networks*		*conventional backprop*	
	# of successful runs	*avg # of epochs*	*# of successful runs*	*avg # of epochs*
2	5	5000	3	4119
3	10	2920.9	10	1154.4
4	10	1801.5	10	642.6

Table 1: Comparative results on the binary XOR problem. All networks have 2 input and 1 output units. Both the training and test data set contain all 4 instances of XOR. The learning rate is 0.5, with no momentum term and no weight decay. Error tolerance is 0.0000001, maximum number of iterations is 5000. The "number of successful runs" is obtained out of 10 runs with different random weight initializations. The "average number of epochs" is the averages over the successful runs.

The training set of the XOR problem consists of all 4 examples of the binary XOR problem. 10 runs are done with different random weight initializations for each network configuration and each of the learning schemes. In this experiment, we intend to compare the convergence speeds of the two methods. A successful run is defined to be such that the network converged within the given maximum number of epochs (in this case, 5000) during training and gives correct outputs for all 4 examples. Note that for networks with 2 hidden units, there are unsuccessful runs for both learning schemes, which means that each of the corresponding networks reached a local minimum, instead of a global one. The number of unsuccessful runs for the two are comparable: 5 for our method, and 7 for standard back-propagation.

The iris data set consists of 3 classes of 50 instances each, where each class refers to a type of iris plant. Attributes are different measurements of the flowers. 10 runs are done by partitioning the data set and using the subsets in a manner similar to cross-validation. In this experiment, we aim at investigating and comparing the effects of momentum and weight decay factors on the two learning schemes.

The sonar data set was used originally by Gorman and Sejnowski in their study of the classification of sonar signals using a neural network [3]. The task is to discriminate between sonar signals bounced off a

# of hidden units	momentum	weight decay factor	discrete networks		conventional backprop	
			avg % correct	standard deviation	avg % correct	standard deviation
2	0.5	1.0	92.0	4.99	96.0	4.42
3	0.0	1.0	96.7	5.37	97.3	4.42
3	0.5	1.0	96.0	6.11	96.7	5.37
3	0.0	.99	95.3	6.67	94.7	4.99
3	0.5	.99	96.0	5.33	97.3	4.42
4	0.0	1.0	96.7	5.37	94.7	6.53
4	0.5	1.0	96.0	5.33	94.7	5.81
4	0.5	.99	94.0	6.96	97.3	4.42

Table 2: Comparative results on the iris data classification problem. All networks have 4 input and 3 output units. The learning rate is 0.5, with different momentum and weight decay factors as shown. Error tolerance is 0.0000001, maximum number of iterations is 5000. The data set of 150 is randomly partitioned into 10 subsets, each of size 15. For each set of network parameters, 10 runs are made by leaving out each one of the subsets as the test set, and using the remaining 9 subsets as the training set. Performance is averaged over the 10 runs.

metal cylinder and those bounced off a roughly cylindrical rock. There are 208 patterns in total with 111 belonging to the "metal" class, and 97 belonging to the "rock" class. Again, for each network configuration, 13 runs are done, in a similar manner to the iris data experiment. The purpose of this experiment is to compare the performances of the two network structures with different numbers of hidden units. The network configurations of the first 5 rows in Table 3 are the same as in [3], while the last 3 rows are additional experiments we did to obtain a comparison over a wider range.

The task of the NETtalk problem is to train a network to learn to convert English text to speech. Inputs are windows of 5 letters, with the letter to be pronounced in the center. Desired outputs are encoded phonemes. Each input letter is unary encoded by a group of 27 units. The training set consists of 1000 most commonly used words. The test set consists of about 4000 words. In this case, the problem is of a particularly large size: 135 input, 22 output, and 15 to 120 hidden units, about 5600 training examples, and close to 20,000 test examples. We used this problem to test the performance of our network on very large problems.

# of hidden units	discrete networks		conventional backprop	
	avg % correct	standard deviation	avg % correct	standard deviation
2	73.08	11.60	82.69	8.55
3	72.60	8.33	85.58	6.66
6	80.77	7.93	85.58	6.19
12	85.10	9.02	86.06	6.08
24	86.06	7.00	82.21	8.79
36	83.17	7.10	82.69	10.73
48	78.85	9.35	71.63	20.89
60	77.88	11.91	56.73	21.44

Table 3: Comparative results on the sonar data set. All networks have 60 input and 2 output units. The learning rate is 0.1 for discrete networks, and 0.2 for conventional backprop, with no momentum term and no weight decay. Error tolerance is 0.001, maximum number of iterations is 300. The data set of 208 is randomly partitioned into 13 subsets, each of size 16. For each set of network parameters, 13 runs are made by leaving out each one of the subsets as the test set, and using the remaining 12 subsets as the training set. Performance is averaged over the 13 runs.

5 DISCUSSION

It can been seen that in general, the performances of the proposed discrete network are comparable to those of the conventional back-propagation network on all the benchmark problems.

From the results on the XOR problem, it is clear that the pseudo-gradient training takes longer than the conventional back-propagation, due to the inaccuracies introduced for gradient descent. However, we should note that the operations needed for one epoch of training is almost the same for pseudo-gradient as back-propagation, the only difference being the discretization operations. The experiments on all the other larger data sets were done for the same fixed number of epochs (300 to 5000) for both networks, so the comparative results shown in Tables 2 to 4 are in fact of training both networks for about the same time period.

The iris data set results indicate that adding a momentum term helps to improve the performance of the discrete network but has an opposite effect on the performance of the conventional back-propagation network. On the other hand, weight decay helps to improve the performance of the conventional network but has an opposite effect on the discrete network. The reason for the phenomena is still under investigation.

For the sonar data experiment, it is expected that the performance of either of the network structure goes up with the increase of the number of hidden units, and drops after a peak has been reached. Note that it takes

# of hidden units	discrete networks		conventional backprop	
	% correct on training set	% correct on test set	% correct on training set	% correct on test set
15	77.05	68.41	83.72	72.64
30	84.53	71.74	89.72	75.82
80	90.22	72.55	93.65	75.90
120	91.95	73.62	92.52	75.61

Table 4: Comparative results on the NETtalk data set. All networks have 135 input and 22 output units. The learning rate is 0.1, with the momentum factor being 0.9 and no weight decay. Error tolerance is 0.001, maximum number of iterations is 1000. The training set consists of 1000 most commonly used words, with 5603 letters to pronounce in total. The test set consists of about 4000 words, with 19994 letters to pronounce in total.

more hidden units for the discrete network to reach the same optimum performance as that of the conventional back-propagation network. The reason for this can be that the internal representation capacity of a discrete network is much less than that of an analog network, the former having only two possible values for each unit, and the latter having infinite values theoretically. On the other hand, for the same reason, it also takes more hidden units for the performance of the former to drop, after the optimum performance is reached, to the same level as that of the latter. That is, the discrete network overfits more slowly than the back-propagation network. Thus we gain the clear understanding of a network by losing some representational power. However, note that the performance differences of the two networks with the same appropriate number of hidden units are not significant.

The results of the NETtalk experiments show that the discrete network is able to find good solutions for such a large problem, and the performance is comparable to that of the back-propagation network, though always a little worse.

6 EXTRACTING RULES FROM THE NETWORK

Using discrete units in the network facilitates the interpretation of the network representation as discrete rules. For discrete binary inputs, classification rules are extracted from the discrete network as follows. Present the trained network with all combinations of inputs in the order of the Gray code, with one input bit change at a time. For each output unit, a truth table is thus constructed for the whole input space. Simplify

each truth table by the standard Quine-McCluskey algorithm to obtain a logic expression of a minimum number of terms. Each term is then a classification rule for the class represented by the corresponding output unit. Note this rule extraction process guarantees that all rules extracted cover every point in the input space, and are accurate descriptions of the network.

For the XOR problem, the following rules are extracted for the single output unit, with the two inputs represented by the symbols A and B, respectively:

If A=low B=high then True. If A=high B=low then True.

For larger problems with data sets containing noise, rule extraction often yields multiple high-order rules that are very specific in describing the input space region for which they can fire. This means that the network uses a very detailed partition in the input space for its classification purposes. It is expected that the less freedom (in terms of the numbers of units and adjustable weights) the network is given, the less detail such a partition will contain, and the more general the extracted rules will be. In addition, training with validation to prevent overfitting would result in less specific rules as well.

For problems with continuous input attributes, quantization can be made a priori based on domain knowledge and/or information theoretic criteria.

This rule extraction method is exhaustive, so all the rules extracted together make a full description of the network classifier over the whole input space. However, the computation grows exponentially with the dimension of the input space. Research is underway to investigate ways to efficiently generate rules according to, but not strictly based on the network, and thus allowing more general lower-order rules.

7 CONCLUSION

A pseudo-gradient learning scheme for discrete networks, or multi-layer perceptrons with hard-limiting units is proposed. For the case of single-hidden-layer networks, we showed that the proposed pseudo-gradient always points in the right down hill direction of the error surface. The experiments on different benchmark data sets show that the discrete networks have comparable performance to that of back-propagation networks. A clear understanding of the network is gained by the discrete structure at the cost of some loss of representational power. An exhaustive method to extract rules that accurately describes the network as a classifier is presented. The preliminary results are encouraging for further study of such discrete networks.

Acknowledgments

The research described in this paper was supported by ARPA under grants number AFOSR-90-0199 and NOOO14-92-J-1860.

References

[1] R.A. Fisher, "The use of multiple measurements in taxonomic problems," *Annual Eugenics,* 7, Part II, 1936.

[2] M. Minsky, S. Papert, *Perceptrons,* MIT Press, 1969.

[3] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks,* Vol. 1, 1988.

[4] D. E. Rumelhart,J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing,* MIT Press, 1986.

[5] T. J. Sejnowski, C. R. Rosenberg, "Parallel networks that learn to pronounce English text," *Complex Systems,* Vol. 1, 1987.

[6] Z. Zeng, R. Goodman, P. Smyth, "Learning finite state machines with self-clustering recurrent networks," *Neural Computation,* Vol. 5, No. 6, 1993.

[7] Z. Zeng, R. Goodman, P. Smyth, "Discrete recurrent neural networks for grammatical inference," *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, 1994.

THE SELECTION OF NEURAL MODELS OF NON-LINEAR DYNAMICAL SYSTEMS BY STATISTICAL TESTS

D. URBANI, P. ROUSSEL-RAGOT,
L. PERSONNAZ, G. DREYFUS
Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris
Laboratoire d'Electronique
10, rue Vauquelin
F - 75005 PARIS - FRANCE
Phone: 33 1 40 79 45 41 ; Fax: 33 1 40 79 44 25
e-mail: dreyfus@neurones.espci.fr

Abstract - A procedure for the selection of neural models of dynamical processes is presented. It uses statistical tests at various levels of model reduction, in order to provide optimal tradeoffs between accuracy and parsimony. The efficiency of the method is illustrated by the modeling of a highly non-linear NARX process.

INTRODUCTION

The representation of the behaviour of dynamical processes is a conceptually straightforward application of neural networks, whether feedforward or recurrent, as non-linear regressors. In practice, however, the modeling of a process requires solving several problems:
(i) the choice of the nature of the model (static model *vs* dynamic model, input-output representation *vs* state representation, ...) requires an analysis of the future use of the model (for instance, whether it will be used for predicting the future evolution of the process, or whether it will be used within a control system), and an analysis of the *a priori* knowledge on the phenomena involved in the process;
(ii) the choice of the structure of the model, defined by the number of its inputs, by the number of its outputs, by the type of input-output relationship (linear, polynomial, radial-basis function, multi-layer neural network, etc.), and by its structural parameters (degree of the polynomial approximation, number of radial basis functions, number of neurons, etc.);
(iii) the estimation of the optimal set of adjustable coefficients (synaptic weights in the case of neural net models) of the chosen structure ("identification" in automatic control, "training" in neural network parlance);

The first problem is fully application-dependent: no general statement can be made. The third problem has been investigated in great depth in the case of

0-7803-2026-3/94 $4.00 © 1994 IEEE

linear models [1]; in the case of neural network models, a variety of training algorithms is available [2], and it has been shown that the choice of a training algorithm, in the context of dynamical process modeling, is based on the nature of the noise present in the process to be modeled [3].

In the present paper, we investigate the second problem, namely, that of model selection, which is a key factor for a model to be successful [4]. We suggest a pragmatic model selection procedure for dynamical input-output non-linear models, which features three steps in succession: first, the inputs (external inputs and feedback inputs) of *linear* models of the process around operating points are selected; in a second step, the relevant inputs of the *non-linear* model are selected, thereby determining the order of the model; finally, the structural parameter of the model is determined. An optimized model of a dynamical process is thus derived.

We describe the selection procedure in the case of stable (within the range of operation for which a model is needed), single-input-single-output processes. We assume that the process is NARX:
$y_p(t) = \Phi[y_p(t-1), ..., y_p(t-\nu), u(t-1), ..., u(t-\mu)] + w(t)$
where $\{w(t)\}$ is a gaussian sequence of zero mean independent random variables, ν is the order of the assumed model, and μ is the memory span of the control sequence $\{u(t)\}$.
The following predictor is used:
$y(t) = \Psi[y_p(t-1), ..., y_p(t-n), u(t-1), ..., u(t-m)]$;
We know from [3] that such a predictor (trained with a directed, or teacher-forcing, algorithm) is optimal as a predictor for a NARX process.
If $n = \nu$, if $m = \mu$, and if $\Psi(.)$ is an accurate approximation of $\Phi(.)$, then the predictor is optimal for the process.
In the following, we describe the three steps of the procedure, in the case of a neural network model.

THE PROCEDURE

First step

In the stability domain of the process, operating points (u_i, y_i) are chosen. The process is subjected to time-dependent control sequences of length N in the ranges $[u_i + \Delta u_i, u_i - \Delta u_i]$, such that a *linear* model of the process can be considered valid in each of these ranges. For each operating point, we select, as described below, a linear model which is a satisfactory tradeoff between accuracy and parsimony. At the end of the first step, the set of all inputs which were selected is available for use in the second step of model selection.

For each operating point, we make the assumption that the process can be described as an ARX model :

$$y_p(t) = \sum_{i=1}^{\nu} \alpha_i \, y_p(t-i) + \sum_{i=1}^{\mu} \alpha_{\nu+1} \, u(t-i) + w(t) \ .$$

where ν et μ are unknow parameters.

We consider a training set of size N, and a family of predictors of the form:

$$y(t) = \sum_{i=1}^{n} \theta_i \, y_p(t-i) + \sum_{i=1}^{m} \theta_{n+i} \, u(t-i) \ .$$

The aim of the procedure is to find a predictor such that $n = \nu$, $m = \mu$.
We denote by $\mathbf{y}_p$, $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$, $\mathbf{x}_{n+1}$, ..., $\mathbf{x}_{n+m}$, $\mathbf{w}$, $\mathbf{y}$ the N-vectors, corresponding to the values $y_p(t)$, $y_p(t-1)$, ..., $y_p(t-m)$, $u(t-1)$, ..., $u(t-n)$, $w(t)$, $y(t)$, for t=1 to N; thus:

$\mathbf{y} = [\mathbf{x}_1, \ \dots \ \mathbf{x}_M] \, \theta$, where $M = m + n$.

We have to find M regressors, corresponding to M independent vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ such that the subspace spanned by these vectors is the subspace of smallest dimension containing $E[\mathbf{y}_p]$. In order to find this subspace, we start with a complete model, whose parameters n' and m' are chosen to be larger than can be expected from the *a priori* knowledge available on the process. We thus make the assumption that the subspace H spanned by the M'=n'+m' vectors contains $E[\mathbf{y}_p]$, and we expect to extract the satisfactory subset of significant regressors from the initial set. This could be achieved by computing and comparing all possible regressions; however, this method becomes too expensive for large M'.
In order to decrease the amount of computation, we build from the initial set $\{\mathbf{x}_1, \dots, \mathbf{x}_{M'}\}$ an ordered set of orthonormal vectors $\{\mathbf{p}_1, \dots, \mathbf{p}_{M'}\}$ such that the model defined by $\{\mathbf{p}_1, \dots, \mathbf{p}_k\}$, for all $1 \le k \le M'$, gives a sum of squares of errors (SSE) which is smaller than the SSE given by all other models with k regressors [5].
We first choose, among the M' vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_{M'}\}$, the vector $\mathbf{x}_j$ giving the largest square regression $|\mathbf{p}_1^T \mathbf{y}_p|^2$, with $\mathbf{p}_1 = \mathbf{x}_j / \|\mathbf{x}_j\|$. The (M'-1) remaining $\{\mathbf{x}_i\}$ vectors are orthonormalized with respect to $\mathbf{p}_1$.
Consider the kth step of the ordering procedure, where $\mathbf{p}_1, \dots, \mathbf{p}_{k-1}$ have been selected. We denote by SSE(k) the SSE obtained with the selected model having k regressors, thus :

$SSE(k-1) - SSE(k) = |\mathbf{p}_k^T \mathbf{y}_p|^2$,

with :

$SSE(0) = \|\mathbf{y}_p\|^2$.

This contribution decreases as k increases. This procedure is iterated M'-1 times for $\mathbf{p}_2$, $\mathbf{p}_3$, ... until completion of the list. Thus :

$$\|y_p\|^2 = \sum_{k=1}^{M'} |\mathbf{p}_k^T \mathbf{y}_p|^2 + SSE(M')$$

where SSE(M') is the sum of squares of errors for the complete model.

Subsequently, the above list is scanned in the inverse order of its construction, and each model is compared with the complete model, using the Log Determinant Ratio Test (LDRT). The number of models we have to take into account is at most equal to M'. Note that the comparison between these models by LDRT is easy (see Appendix for further details about this test), since the variable used to compare the k-regressor model and the complete model is :

$$X_{LDRT} = N \frac{\log[SSE(k)]}{\log[SSE(M')]}.$$

We select the smallest predictor model accepted by the test.

In order to further decrease the number of tests, we introduce a simple stopping criterion during the formation of the subset $\{\mathbf{p}_1, \ldots, \mathbf{p}_{M'}\}$: at the kth step, the procedure is terminated if $|\mathbf{p}_k^T \mathbf{y}_p|^2 < \rho \, \|\mathbf{y}_p\|^2$. The choice of ρ is not critical provided it is small (typically $\rho<10^{-8}$).

In the present work, we use LDRT, but Fisher-Snedecor test, Akaike's Information Criterion (AIC) test are also available (for a review see [4]) and lead to similar results.

Thus, for each chosen operating point, a linear model is available, which achieves a satisfactory tradeoff between accuracy and parsimony. Note that the techniques which are used in the linear context of this step are not computationally expensive, so that a large number of external inputs n and feedback inputs m can be used as a starting model for selection.

At the end of the first step, each regressor which was selected for at least one operating point is available for consideration in the second step of model selection.

Second step

In this step, the process is subjected to large-amplitude control signals corresponding to the conditions of operation which the model is expected to account for. A non-linear model is defined (e.g. a neural network), whose inputs are the set of inputs which were determined during the previous step, and whose structural parameters are deemed to be appropriate for the non-linear input-output function to be accurately approximated (e.g. a neural network with an appropriate, possibly too large, number of neurons, trained by an algorithm which allows an efficient minimization of the SSE). Such methods tend to be computationally expensive, so that the chosen number of neurons should not be excessively large. The best subset of inputs is selected by statistical tests (LDRT or AIC criterion (see appendix)) : we compare the complete non-linear model with all these sub-models with one input less. If all the models are rejected, this step of the procedure is terminated. Otherwise, the best submodel is chosen, and compared with all these sub-models having one input less, and so on.

At the end of this step, a non-linear model M_1 is available, whose inputs have been selected.

Third step

The final step aims at determining the structural parameter of the model: in the case of a neural network model, this parameter is the number of hidden neurons. Here, the accuracy/parsimony tradeoff is expressed by the fact that too large a number of hidden neurons leads to overtraining (small SSE on the training set, large SSE on the test set), whereas too small a number of neurons leads to poor approximation (large SSE on the training set itself). The model M_1 resulting from the previous two steps is considered as the complete model, and models with a smaller number of hidden neurons than M_1 are considered for selection. As in the previous steps, statistical tests are used in order to find a satisfactory tradeoff. Note that most model reduction algorithms used for neural networks aim at eliminating connections [6], whereas this final step aims at eliminating neurons.

EXAMPLE

The efficiency of the above procedure is illustrated by the modeling of a second-order, highly non-linear NARX process, which is simulated by the following equation:

$$y_p(t) = 50 \tanh\left\{2.\,10^{-3}\left[\frac{24 + y_p(t-1)}{3}\, y_p(t-1) - 8\,\frac{u(t-1)^2}{1 + u(t-1)^2}\, y_p(t-2)\right]\right\} + 0.5\, u(t-1) + w(t)\ ,$$

where w(t) is white noise with variance $(\sigma_W)^2$. The behaviour of this process is essentially that (i) of a linear first-order low-pass filter for amplitudes smaller than or on the order of 0.1, and (ii) of a second-order, oscillatory, linear ($0.1 < |u| < 0.5$), or non-linear ($0.5 < |u| < 5$) system for larger amplitudes; it becomes almost static for positive signals of very large amplitude; in addition, it is not symmetrical with respect to zero. Figure 1 shows the response of the process to steps of random amplitude in the region of interest, with $(\sigma_W)^2 = 10^{-2}$.

First step

The operating points were $u_i = \{-10, -8, -5, -2, -1, -0.5, 0.1, 1, 2, 5, 8, 10\}$. At each of these points, a uniformly distributed random sequence was added to the control input, with maximum amplitude $\Delta u_i=0.1$ ($\sigma_u^2 = 3.10^{-3}$). The initial model was chosen to have n' = m' = 100. The training sequence was of length N = 1000. The orthonormalization procedure retained 15 inputs, and the subsequent LDRT tests (with 1% risk) led to the selection of n+m = 2 to 5 inputs, depending on the operating points.

Second step

The training set was a sequence of large-amplitude steps, such as shown on Figure 1. M_1 was a fully connected neural network, with the 5 inputs (n = 3, m = 2) selected in the first step, and with 10 hidden neurons. After training, the variance of the prediction error (as estimated by SSE/N) was on the same order of magnitude as σ_W , which shows that the network was sufficiently large, and had been trained efficiently. Subsequently, the networks obtained by suppressing 1 input, then 2 inputs, etc., were trained and submitted to the LDRT procedure, as illustrated on Table 1: the full model M_1 is compared to M_2, M_3, ..., M_6. The test selected only M_2 and M_4 (the deletion of one input leads to the deletion of 11 connections; the corresponding value of the χ^2 variable for a 1% risk is 24.7). Since the SSE of M_4 was smallest, it was selected for comparison with all models smaller than M_4[1]; M_7 is the only three-input model which was selected. All models smaller than M_7 were rejected. Therefore, M_7 was finally accepted. The success of the procedure is shown by the fact that M_7 is indeed the only model which has the same inputs as the simulated process. A similar result is obtained if the AIC test is used.

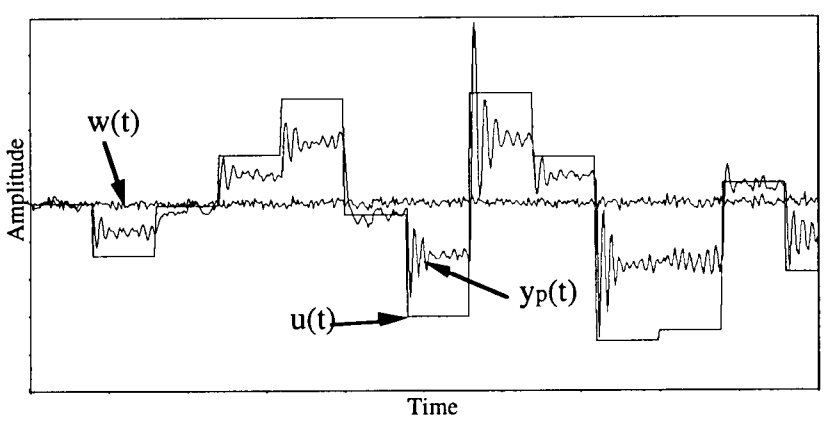


FIGURE 1
Sequence of control input and process output.

Third step

Model selection is performed on neural nets having the inputs of M_7, and 0 to 10 hidden neurons, with the same training set for all nets. The result of the selection depends on σ_W. With $\sigma_W{}^2 = 10^{-2}$, a model with 9 neurons is selected. With $\sigma_W{}^2 = 10^{-1}$, the same inputs are selected by the first two steps

[1] Actually, the SSE's of M_2 and M_4 are very close; if M_2 is selected instead of M_4, the same result is obtained, since M_7 is a sub-model of both M_2 and M_4.

Model	$y_p(t-1)$	$y_p(t-2)$	$y_p(t-3)$	$u(t-1)$	$u(t-2)$	SSE	X_{LDRT}
1	X	X	X	X	X	19.1	
2	X	X	X	X	–	19.6	11
3	X	X	X	–	X	13.0	832
4	X	X	–	X	X	19.5	10
5	X	–	X	X	X	31.6	218
6	–	X	X	X	X	31.8	221
7	X	X		X	–	19.6	1.2
8	X	X		–	X	97.7	697
9	X	–		X	X	11.8	980
10	–	X		X	X	39.4	1303
11	X	X		–		25.4	1114
12	X	–		X		18.7	978
13	–	X		X		18.2	968

TABLE1
Models labelled by boldface figures are those whose inputs include the inputs of the process.

and the third step leads to a neural network with 4 neurons. As should be expected, the procedure selects a smaller number of neurons if the noise level is high than if it is low.

CONCLUSION

A pragmatic three-step procedure for non-linear dynamical model selection has been proposed, which uses statistical tests at various levels of model reduction. It relies on the fact that efficient training procedures are available. It allows the selection of the delayed external inputs, of the feedback inputs (hence the determination of the order of the model) and of the structural parameters such as the number of hidden neurons. Its main shortcoming seems to be the fact that its application is subject to the availability of two types of data from the process, namely, small-signal responses around chosen operating points, and large-signal responses in "normal" operation. Its efficiency is shown on an illustrative example: the neural modeling of a highly non-linear NARX process.

APPENDIX

The Logarithm Determinant Ratio Test (LDRT) [4]

The problem of the selection of one model out of two can be formulated as a statistical testing problem. We suppose that an accurate model M_1, described by the vector of paramters θ, is available to explain a set of N experimental

data. The null hypothesis states that a part θ_2 of the vector parameter θ is equal to zero; if this assumption is true, $\theta = [\theta_1, \theta_2]$ can be reduced to θ_1. If the alternative hypothesis is true, then θ_2 cannot be taken equal to a zero vector. A very efficient test to solve such a problem is the Likelihood Ratio Test (LRT), but this test requires the expression of the likelihood function. In our case, with very large N, it reduces to the Log Determinant Ratio Test (LDRT) : under the null hypothesis $\theta_2=0$, with a scalar output, the distribution of the statistics :

$$X_{LDRT} = N \log \frac{SSE(\theta_1)}{SSE(\theta)}$$

converges to a chi-square distribution with $\dim(\theta_2)$ degrees of freedom.

The Akaike's Information Criterion Tests (AIC)

The AIC is an alternative way of selecting a model from a set of models, using statistical tests. For each model of the set, we compute the AIC value :
$AIC = 2\,N \log(SSE/N) + 2M$
where N is the number of data and M is the number of parameters of the model.
The model corresponding to the smallest AIC value is thus selected as the best model of the set, with respect to this criterion. This procedure requires no assumptions on the models. There exist more efficient variants of the classical AIC [4], such as the AIC*, used in this work :
$AIC^* = 2\,N \log(SSE/N) + 4\,M$

REFERENCES

[1] See for instance:
L. Ljung, System Identification: Theory for the User: Prentice Hall, 1987.
G.C. Goodwin, R.L. Payne, Dynamic System Identification: Experiment Design and Data Analysis: Academic Press,1977.

[2] O. Nerrand, P. Roussel-Ragot, L. Personnaz, G. Dreyfus, "Neural Networks and Non-linear Adaptive Filtering: Unifying Concepts and New Algorithms", Neural Computation, vol. 5, pp.165-197, 1993..

[3] O. Nerrand, P. Roussel-Ragot, D. Urbani, L. Personnaz, G. Dreyfus, "Training Recurrent Neural Networks: Why and How ? An Illustration in Dynamical Process Modeling", IEEE Transactions on Neural Networks, vol. 5, pp. 178-184, 1994.

[4] I.J. Leontaritis, S.A. Billings, "Model Selection and Validation for Non-Linear Systems", International Journal of Control, vol. 1, pp. 311-341, 1987.

[5] S. Chen, S.A. Billings, W. Luo, "Orthogonal Least Squares Methods and their Application to Non-Linear System Identification" International Journal of Control, vol. 50, pp. 1873-1896, 1989.

[6] R. Reed, "Pruning Algorithms - A Survey", IEEE Transactions on Neural Networks, vol. 4, pp. 740-747, 1993.

Speech Processing

RECURRENT NETWORK AUTOMATA FOR SPEECH RECOGNITION: A SUMMARY OF RECENT WORK

Roberto GEMELLO*, Dario ALBESANO*, Franco MANA*, Rossella CANCELLIERE*§

*CSELT - Centro Studi e Laboratori Telecomunicazioni
via G. Reiss Romoli, 274 - 10148 Torino - Italy
Tel.: +39-11-2286224 Fax: +39-11-2286207
email: gemello@cselt.stet.it

§Dipartimento di Matematica Applicata
Università di Torino
via C. Alberto, 10 - 10123 Torino - Italy

Abstract. The integration of Hidden Markov Models and Neural Networks is an important research line to obtain new speech recognition systems that join a good time-alignment capability and a powerful discrimination-based training. The Recurrent Network Automata model is a hybrid of a recurrent neural network, which estimates the state emission probability of a HMM, and a dynamic programming, which finds the best state sequence. This paper reports the last results obtained with the RNA model, after three years of research and application to speaker independent digit recognition over the public telephone network.

INTRODUCTION

This paper reports the last results of the CSELT neural network group in the field of speech recognition.

As Neural Networks (NN) are not yet able to manage well time modelling, we are presently employing them in integration with Hidden Markov Models (HMM).

This approach is currently investigated by several research teams: Franzini, Haffner and Waibel [9] [10] [13] have introduced the Connectionist Viterbi Training to enhance HMM based connected digit recognition; [4] has described Segmental Neural Networks for phonetic modelling; Bourlard et alii [5] [6] have proposed connectionist probability estimation to significantly improve a HMM based continuous speech recognition system.

Our contribution to this line was the introduction of Recurrent Network Automata (*) (RNA) [1] [2] which integrates recurrent NN with HMM word modelling, showing the advantages which can be obtained by exploiting the joint contextual information of feedback hidden units and time delayed input.

(*) patent pending

0-7803-2026-3/94 $4.00 © 1994 IEEE

Presently, we are experimenting the RNA framework both for isolated and connected word recognition [3], trying different training strategies and architectures and evaluating the impact of some recently emerged input data pre-processing like parameter high-pass filtering.

The paper reviews the HMM-NN integration proposed by the RNA model, and focuses on the last experiments.

MODEL DESCRIPTION

The RNA recognition model [1] [2] is a hybrid HMM-NN model devoted to recognise sequential patterns. Each class is described in terms of a left-to-right automaton (with self loops) as in HMM, and the emission probability of the automata states are estimated by a Simple Recurrent Network [7]. The transition probabilities among states are not considered. The RNA has an input window that comprises some contiguous frames of the sequence, one hidden layer with a self-feedback, and an output level where the activation of each unit estimates the probability of the input window to belong to an automaton state.

The hidden neuron dynamics is given by the equation:

$$y_i(t) = F(\Sigma_j w_{ij} x_j(t) + \Sigma_k w_{ik} y_k(t-1))$$

where y_i is the activation of a hidden neuron, x_j is an input unit and F is the standard logistic function. The hidden neurons are also called *state neurons* because thanks to the self-feedback they can encode a contextual information about the sequence which is being recognised. The output neurons follow the standard MLP dynamics.

SPEECH MODELING WITH RECURRENT NETWORK AUTOMATA

The RNA model was principally conceived for speech recognition, and in particular for modelling words for isolated or connected word recognition with a small vocabulary. In RNA time modelling takes place in two ways: first, by an external modelling, through the HMM like time warping ability of the dynamic programming applied to left-to-right automata corresponding to words; second, by the internal modelling of the recurrent network. In fact, the memory capability of the recurrent network allows to give the states a contextual information, inside the word automaton, and to give a more stable evolution of emission likelihoods, inside the state [1].

The architecture of RNA has many degrees of freedom: the architecture of the NN, the input window width, the number of automaton states for the different words of the vocabulary. A lot of experimental activity has been performed to optimise the architecture for the recognition of small vocabularies (10-20 words) resulting in the structure depicted in fig. 1. The input window is 3-7 frames wide, and each frame contains 26 parameters (log Energy, 12 Cepstral Coefficients, and their first derivatives). The first hidden layer is divided into three feature detectors blocks, one for the central frame, and two for the left

and right context. Each block is in its turn divided into four sub-blocks devoted to keep into account the four types of different input parameters. It was empirically found that this a priori structure is generally better than a fully connected layer. The second hidden layer has a fully connected recurrence, like in Elman's nets (the double arrow means a copy of activation values). The neurons of this layer have a twofold function: first, they represent, together with the first hidden layer neurons, a *space transform* between the input parameters and some self-organised internal features, corresponding to acoustic/phonetic characteristics (e.g. silence, stationary sounds, transitions, specific phonemes). Besides, they encode a *state information* related to the temporal context the current input is inserted in, as described in [2]. The output layer estimates the emission probabilities of the states of the word automata, and is virtually divided in several parts, each one corresponding to an automaton.

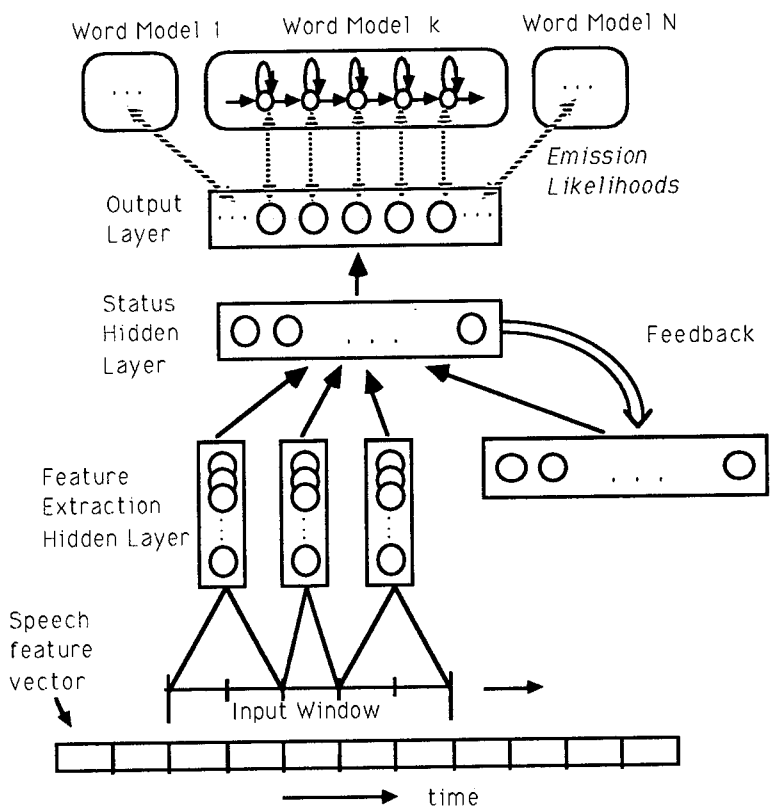


Figure 1: Architecture of a Recurrent Network Automata devoted to word recognition

Typical dimensions for a RNA devoted to recognise the ten Italian digits are:

- 7 frame input window;

- first hidden layer: central block with 24 units (divided in 2+10+2+10 for the four types of parameters), context blocks with 36 units (3+15+3+15).
- second hidden layer: 70 units, with fully connected recurrence.
- output layer: 63 units, corresponding to 63 automata states, pertaining to the 10 word automata, and divided proportionally to the average word length.

TRAINING RNA

RNA training is a complex problem, because we want simultaneously find the best segmentation of words into a given number of states and train the network to discriminate that states.

Training is an iterative procedure as follows:

Initialisation:

- initialise the RNA with small random weights;
- create the first segmentation by segmenting the training utterances uniformly.

Iterations:

- load the present segmentation;
- train the RNA some epochs to implement the automata which approximates that segmentation;
- obtain a new segmentation by applying the dynamic programming to each utterance in the training set to re-evaluate the transition points proposed by the RNA;
- update the present segmentation by using a function of itself and of the new segmentation: present_segm = F(present_segm, new_segm);
  e.g. $F(s1, s2) = \alpha s1 + (1-\alpha)s2$, with α starting from 1.0 and decreasing during the training.

The input to the RNA is a window sliding on the speech frames, including a central frame and some left and right context frames. The targets are generated according to the present segmentation, putting 1.0 for the active state of the right automaton and 0.0 otherwise. All the automata are trained into a unique net, so performing a discriminative training. The NN basic learning algorithm is the back-propagation.

Recently a variation of classical backpropagation, called Correlative Training has been developed and experimented [12]. Briefly, Correlative Training consists in changing the target definition in function of the correlation of the outputs of the considered unit and of the target unit. We redefine the target of a generic output unit k as:

$$t_k(o_k, o_h) = \begin{cases} t_h & \text{if} \quad h = k \\ o_k o_h & \text{if} \quad h \neq k \end{cases}$$

where o_k is the output of unit k, t_k the target of output unit k, h the index of the output unit with $t_h = 1.0$ This leads to a change in the $\frac{\partial E}{\partial o_j}$ term of backpropagation for output units, that becomes

$$\frac{\partial E}{\partial o_j} = (t_j(o_j, o_h) - o_j)\left(\frac{\partial(t_j(o_j, o_h))}{\partial o_j} - 1\right) = \begin{cases} -(t_j - o_j) & \text{if} \quad j = h \\ o_j(o_h - 1)^2 & \text{if} \quad j \neq h \end{cases}$$

This change in backpropagation is a simple way to adaptively soften the strength of discriminative training for classes that cannot be completely put apart, without compromising its power on separable classes. That results in an adaptive smoothing of discriminative training.

RECOGNITION EXPERIMENTS

Since three years we have experimented RNA to face a difficult real problem, i.e. the speaker independent recognition of the digits over the public telephone network. This problem has been already faced in our labs by using Continuous Density Hidden Markov Models [8], so we already have a large training database and some state of the art results to compare with. Preliminary results obtained and a comparison with HMMs were reported in [2].

Speech Database and Preprocessing

The speech database we used was collected on the Italian public telephone network, each time using a different switching circuit. It is suited for speaker independent training as about 1,000 people evenly distributed between male and female voices contributed to it. The pre-processing technique consists of a Mel-based spectral analysis followed by a Discrete Cosine Transform to obtain Cepstral coefficients. Together with the cepstral coefficients, the value of the logarithm of the total energy of each frame is retained as it provides some information about distinguishing the voiced parts of the speech input from the unvoiced ones.

A RNA network was trained on a training set containing about 500 repetition for each digit.

Input Filtering

An input pre-processing was experimented by applying a high-pass filter inspired to RASTA filter [14] directly on cepstral coefficients and energy.

The filter equation is

$$y(n) = x(n) - x(n-1) + \lambda\, y(n-1) \qquad \text{with } 0 < \lambda < 1$$

We tried several values for λ, and realised that a value of 0.99, which cuts off only the continuous frequency component of cepstral parameters is the best suitable in that case.

The results are encouraging, as can be seen in Table 1: in fact the input filtering always improves the recognition.

Comparing Feedback nets with TDNN and Feedforward nets

The basic RNA models makes use of a feedback MLP. Of course, in the same framework other MLP architectures may be inserted, like a standard feedforward MLP and a TDNN.

In this chapter we will discuss the recognition results obtained with three different architectures inserted in the RNA framework.

The first is the standard feedback MLP described in fig. 1 and in chapter 3 (18470 weights). The second is a TDNN a là Waibel, with 26 input unit (Energy, 12 Cepstral and their first derivatives) with delay = 3, 50 hidden units with delay = 5 and 63 output units, as in the other models (19650 weights). The third model is a straightforward fully connected network with input window = 3, one hidden layer of 150 units and the usual 63 units output layer (21150 weights).

From Table 1 can be seen that all the three models works pretty good inserted in the RNA framework. The high-pass filter is always useful, and in particular with feedforward and TDNN networks, where the improvement is very relevant. The last column (dist 1-2) shows the average distance between the first and the second choice of the recognizer, computed as the summation of -log(P(State | input frame)) on the best path, normalised with the length of the word. P(State | input frame) is provided by the neural network while the best path is computed by the Viterbi algorithm. From this point of view the feedback MLP is preferable because this distance is greater than in the other models. Besides, the feedback model obtains the best performance (99.2) with less weights than feedforward one.

RNA Architecture	Filtering	% Train	% Test	dist. 1-2
Feedback MLP, with feat. extr.	no	98.8	98.5	71
layer and feedback (see fig. 1)	yes	99.7	99.2	75
TDNN a là Waibel with 2 layers	no	99.1	98.4	50
of delay units with d=3 and d=5	yes	99.5	99.1	57
Feedforward fully connected	no	98.8	97.6	42
MLP with one hidden layer	yes	99.6	99.2	5

Table 1. Recogniton results for differents RNA architectures and input filterings

CONCLUSION

A hybrid speech recognition model has been described which integrates recurrent neural networks with HMM word modelling and decoding. The model has exhibited a good performance on a difficult recognition task, showing noise robustness and results comparable with the mature CDHMM technology, but with the parallelization potentiality typical of NN. The model has been widely applied to isolated words, while application to connected words is under development [3]. A preliminary analysis of its capability to reject extraneous patterns is encouraging, and could be further improved through the use of closed decision regions MLP, as described in [11]. Another interesting feature is the network scalability, currently under investigation, which seems to indicate that, thanks to the shared information of the hidden layers, the network dimension grows less that linearly with the number of words.

References

[1] D. Albesano, R. Gemello and F. Mana, "Word Recognition with Recurrent Network Automata", in *Proc. IJCNN 92*, Baltimore, June 1992, pp. 308-313.

[2] D. Albesano, R. Gemello and F. Mana, "Recurrent Network Automata for Speech Recognition", in *Proc. WCNN 93,* Portland, July 1993, vol. III, pp. 16-19.

[3] D.Albesano, R. Gemello and F. Mana, "Connected Word Recognition with Recurrent Network Automata, *Proc. of ICANN 94,* Salerno, May 1994.

[4] S. Austin, G. Zavaliagkost, J. Makhoul, and R. Schwartz, "Speech Recognition using Segmental Neural Nets", in *Proc. ICASSP*, 1992, pp. 625-628.

[5] H. Bourlard, C.J. Wellekens, "Links Between Markov Models and Multilayer Perceptrons", in IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 12, pp. 1167-1178.

[6] H. Bourlard, N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach,* Kluwer Academic Publishers, 1993.

[7] J.L. Elman, "Finding Structure in Time", CRL Technical Report #8801, University of California, San Diego, 1988.

[8] F. Canavesio, L. Fissore, M. Oreglia, R. Ruscitti "HMM modeling in the public telephone network environment: experiments and results", in *Proc. EUROSPEECH 91*, Genova, September 1991, pp. 731-734.

[9] M.A. Franzini, K.F. Lee and A. Waibel, "Connectionist Viterbi Training: A new hybrid method for continuous speech recognition", in *Proc. ICASSP*, Albuquerque,NM, April 1990, pp.425-428.

[10] M.A. Franzini, A. Waibel and K.F. Lee , "Continuous Speech Recognition with the Connectionist Viterbi Training Procedure: a summary of recent work", in *Proc. IJCNN* , Singapore, 1991, pp.1855-1860.

[11] R. Gemello and F. Mana, "An Enhancement to MLP Model to Enforce Closed Decision Regions", in *Proc. IJCNN*, Singapore, November 1991, pp.729-733.

[12] R. Gemello, D. Albesano, F. Mana, "Correlative Training and Recurrent Network Automata for Speech Recognition", in *Proc. IEEE ICNN 94,* Orlando, 1994.

[13] P. Haffner, M. Franzini, A. Waibel, "Integrating Time Alignment and Neural Networks for High performance Continuous Speech Recognition", in *Proc. ICASSP*, 1991, pp. 105-108.

[14] H. Murveit, J. Butzberger, M. Weintraub, "Reduced Channel Dependence for Speech Recognition", in *Proc. of Speech and Natural Language Workshop*, February 1992.

ACOUSTIC ECHO CANCELLATION FOR HANDS-FREE TELEPHONY USING NEURAL NETWORKS

A. N. Birkett, R. A. Goubran
Department of Systems and Computer Engineering
Carleton University, 1125 Colonel By Drive
Ottawa, Canada, K1S 5B6
Tel: (613) 788-2600 ext. 5740, Fax: (613) 788-5727
e-mail: birkett@sce.carleton.ca

Abstract: One of the limitations of linear adaptive echo cancellers in hands-free environments is their inability to effectively cancel nonlinearities which are generated mainly in the loudspeaker during large signal peaks. The soft-clipping effect encountered when large signals are applied to the loudspeaker is modelled in a neural network using a piecewise linear/sigmoid activation function. A three layer fully adaptive feedforward network is used to model the room/speakerphone transfer function using the special activation function. This network structure improves the ERLE performance by 10 dB at low to medium loudspeaker volumes compared to a NLMS echo canceller.

INTRODUCTION

A microphone placed next to a loudspeaker in a closed loop provides electro-acoustic feedback which will spontaneously oscillate at some frequency for which the modulus of the gain factor is greater than one. This arrangement exists in all hands-free telephone systems hence adaptive echo cancellation is required to prevent these oscillations while communicating in full-duplex mode.

Limitations of echo cancellers for speakerphones [4],[8] include (a) acoustic, thermal and DSP related noise, (b) inaccurate modelling of the room impulse response (c) slow convergence and dynamic tracking, (d) nonlinearities in the transfer function caused mainly due to the loudspeaker, and (e) resonances and vibration in the plastic enclosure.

To be commercially attractive, convergence times on the order of 100 ms with Echo Return Loss Enhancement (ERLE) on the order of 30 dB are necessary. Fast RLS based adaptive techniques can be used to reduce the convergence time, however, the ERLE is degraded when the input data is severely non-stationary and it has been found [4],[5] that for large filter orders and nonstationary environments, LMS type algorithms will give better overall performance than RLS type algorithms. However, nonlinear techniques must be employed to deal with system nonlinearities and IIR recursive structures must be utilized when poles exist in the room/speakerphone transfer function [6]. In this paper, a tapped delay line feedforward neural network is employed in an attempt to model only the system nonlinearities.

0-7803-2026-3/94 $4.00 © 1994 IEEE

Distortions in the Loudspeaker

A loudspeaker has several sources of nonlinearity including non-uniform magnetic field and nonlinear suspension system [1]. Nonlinear distortion is often a few percent of the output signal and it is desirable to reduce it.A loudspeaker consists of an electrical part and a mechanical part as shown in Figure 1. The electrical part is the voice coil and the mechanical part consists of the cone, the suspension system and the air load. The two parts interact through the magnetic field. The resulting equation of motion [2] is:

$$m\frac{d^2x}{dt} + r_M\frac{dx}{dt} + f_M = Bli \tag{1}$$

where B is the magnetic flux density in the air gap, l is the length of the voice coil conductor, x is the cone displacement, m is the total mass of the coil, cone and air load and f_M is the force deflection characteristic of the loudspeaker cone suspension system, usually approximated by;

$$f_M = \alpha x + \beta x^2 + \delta x^3 \tag{2}$$

where α, β and δ are modelling constants and x is the displacement of the voice coil. Suspension system nonlinearity manifests itself as soft clipping at the loudspeaker output and results in odd-order harmonics under large signal conditions.

CONVENTIONAL ADAPTIVE ECHO CANCELLER MODELS

Linear Transversal Filter Model

Figure 2a illustrates an acoustic echo canceller (AEC) utilizing a linear adaptive transversal filter to model the room impulse response to cancel the reflected signal. The reflected signal is a combination of room echoes, direct path signals, loudspeaker and microphone transfer functions, and vibration and resonances emanating through the plastics of the speakerphone as illustrated in Figure 2b. The normalized Least Mean Square (NLMS) algorithm [10] is the baseline by which performance of alternative models is measured.

Nonlinear Adaptive Volterra Model

Adaptive volterra filtering can be utilized to deal with loudspeaker nonlinearities [2], however, filter orders greater than 3 are required to effectively model the speaker transfer function and this very quickly leads to an unmanageably huge model [9]. In fact, during the course of this work, a fully connected 3rd order adaptive Volterra filter with m1=600, m2=600, and m3=50 where m1, m2 and m3 refer

to the orders of the linear, quadratic and cubic sections respectively, was constructed in an attempt to model the loudspeaker nonlinearity. The tap updates were based on the LMS algorithm presented in [9] but extended to a cubic system. It was found that no noticeable improvement in converged ERLE could be seen using this technique. Neural networks offer an alternative method of dealing with high order system nonlinearities.

NEURAL NETWORK ECHO CANCELLER MODELS

Three separate adaptive AEC networks were constructed. The first AEC uses a two layer (100,2,1) network placed in series with a 500 tap NLMS adaptive linear filter as shown in Figure 3a. The 100 inputs are obtained from a tapped delay line. The hidden layer neuron has a nonlinear activation function and the output neuron is linear.The neural network in this case is first batch trained on the first 500 points of data obtained at a medium volume and then tested on loud volume data to ensure that the network is not overtrained.

The second AEC uses the same network but in this case, the neural network is placed in parallel with the NLMS adaptive linear filter as shown in Figure 3b.

The third AEC model utilizes a fully adaptive (600,2,2,1) 3 layer feedforward neural network. The 600 inputs are obtained from a tapped delay line. The two hidden layer neurons have piecewise linear/sigmoid nonlinear activation functions and the output neuron is linear. This model is shown in Figure 3c.

In each neural network, a piecewise linear/tan-sigmoid activation function is used in order to mimic the soft clipping effect and the function response is shown in Figure 4 along with its corresponding delta function. The transfer function is linear below a user definable point and then follows a compressed hyperbolic tangent sigmoid beyond this point such that the output is squashed between +/-1.0. The linear region was set to +/- 0.75 since it was found that this gave good results.

In all cases, the backpropagation algorithm with a normalized step size is employed during the training and tracking phase. The stepsize μ is normalized [10] according to (3).

$$\mu = \frac{\alpha}{\varepsilon + \sum_{i=0}^{M-1} x_i^2} \qquad (3)$$

where α is a number between 0 and 2, and in all cases is set to 0.5. ε is a small positive constant used to prevent the stepsize from becoming too large, M is the number of delay sections in the tapped delay line (i.e. order of the input section) and x_i

is the amplitude of the ith delayed element. The stepsize μ is updated after each new sample is shifted into the tapped delay line.

EXPERIMENTAL SETUP

Figure 5 illustrates the test set-up used to obtain the data. A number of commercially available speakerphones were purchased and modified to allow access to internal signals. The modified speakerphone is placed inside a noise shielded enclosure or anechoic chamber. Filtered "reference" signals are applied to the loudspeaker and the microphone picks up the reflected or "primary" signal. Both the reference and primary data signals are recorded on a Digital Audio Tape and later sampled at 16 kHz and stored to disk for off-line processing.

TEST RESULTS

Converged ERLE for NLMS Case

The NLMS algorithm with 600 taps is applied to the measured data and a number of ERLE curves are obtained for various speaker volume levels. The algorithm is allowed to converge for 32000 samples and then the average ERLE is obtained from the last 8000 output values. The results illustrated in Figure 6, show that the converged ERLE is low for low speaker volumes where acoustic, thermal and DSP related noise are significant. This agrees with results presented in [4] and [8]. The ERLE increases as the reference signal increases but reaches a plateau. Any increase in reference signal level to the loudspeaker after this point results in a decrease in achievable ERLE. The NLMS results in Figure 6 are obtained from three different commercially available speakerphones ranging in price form $32 to $120.

Convergence Curves for Parallel and Series Models Utilizing Pretrained Neural Networks

The ERLE convergence curves of the series and parallel structures are illustrated in Figure 7. Also illustrated for comparison is the 600 tap NLMS case. The series model has a slightly superior convergence than the NLMS case but eventually settles to the same value of converged ERLE. The parallel structure has a convergence essentially the same as the NLMS case but settles to a lower value of converged ERLE. These results were obtained at a high volume of 0.25 W which is equal to the rated power handling capability of the loudspeaker.

Converged ERLE for the Fully Adaptive Three Layer Neural Network

Figure 8 illustrates the performance of the fully adaptive (600,2,2,1) structure compared to the 600 tap NLMS case. The improvement in ERLE over the NLMS case

is significant in the low and medium volume ranges and is greater than 10 dB at power levels in the vicinity of 1mW. However, the fully adaptive model does not offer significant improvement at high speaker volumes suggesting that there still exists a deficiency in the modelling of the room/speakerphone transfer function at these volume levels. A total of three speakerphones were tested. Each speakerphone yielded similar results.

DISCUSSION OF TEST RESULTS

It has been shown in this paper that a fully adaptive three layer neural network offers significant improvement in converged ERLE in the low to medium volume range where acoustic, thermal and DSP related noise are significant. However, when feedforward structures are utilized at high volume levels, little or no improvement in converged ERLE is observed for filtered noise inputs, and this is confirmed by both the Volterra models and the three neural network models presented in this paper.

It appears that the room/speakerphone transfer function may contain poles when the loudspeaker is at high volumes. This is most likely caused by resonances in the plastics of the speakerphone and to a lesser extent poles in the room transfer function [6]. In order to more accurately model the room/speakerphone transfer function, a recursive structure may be necessary and this is the thrust for future work. NARMAX [11] models, recursive neural networks, and nonlinear state-space filters [12] are all possible candidates.

It is likely that the limitation in converged ERLE at high volumes is a combination of nonlinear effects in the loudspeaker and undermodelling of resonances in the plastic enclosure, and that the limitation due to nonlinearity is being masked by the latter.

SUMMARY

Nonlinear distortions and undermodelling has been found to limit the converged ERLE of acoustic echo cancellation in handsfree terminals. Loudspeaker distortions include nonlinearity in the suspension system which will result in soft clipping at high volumes. A piecewise linear/tan-sigmoid activation function has been developed to more accurately model the soft clipping effect and offers a slight improvement in converged ERLE. A third order Volterra model and three neural network AEC models have been developed which indicate that a purely feedforward tapped delay line structure is not sufficient to accurately model the room/speakerphone transfer function at high volumes resulting in no significant improvement in converged ERLE. However, a 10 dB improvement in converged ERLE can be obtained in the low to medium volume ranges where the primary signal to noise ratio is small. It is proposed that at high volumes resonances in the plastic may be

masking the nonlinearity of the speaker and that a recursive structure incorporating poles in the transfer function may be necessary to obtain further improvements in converged ERLE.

ACKNOWLEDGEMENTS

The authors wish to thank NSERC, Carleton University and the Telecommunications Research Institute of Ontario for their financial support.

REFERENCES

[1] H.F. Olsen, Acoustical Engineering, Toronto, D. Van Nostrand Company, Inc., 1964.

[2] X. Y. Gao, W. M. Snelgrove, "Adaptive Linearization of a Loudspeaker", ICASSP 1991 Vol. 3, pp 3589-3592.

[3] O. Nerrand, P. Roussel-Ragot, L. Personnaz, G. Dreyfus, "Neural Network Training Schemes for Nonlinear Adaptive Filtering and Modelling", IJCNN 1991 pp I-61 to I-67.

[4] M. E. Knappe, Acoustic Echo Cancellation: Performance and Structures, M. Eng. Thesis, Carleton University, Ottawa, Canada, 1992.

[5] H. Yuan, Dynamic Behavior of Acoustic Echo Cancellation, M. Eng. Thesis, Carleton University, Ottawa, Canada, 1994.

[6] Y. Haneda, S. Makino, Y. Kaneda, "Common Acoustical Pole and Zero Modelling of Room Transfer Functions", I.E.E.E. Transactions on Speech and Audio Proc. Vol. 2, No. 2, April 1994, pp. 320-328.

[7] Y. Pao, Adaptive Pattern Recognition and Neural Networks, Addison-Wesley Publishing Company, Inc.

[8] M.E. Knappe, R.A. Goubran,"Steady State Performance Limitations of Full-Band Acoustic Echo Cancellers", Presented at ICASSP 1994, Australia.

[9] C. E. Davila, A. J. Welch, H.G. Rylander, "A Second Order Adaptive Volterra Filter with Rapid Convergence", I.E.E.E. Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-35, No. 9, Sept. 1987, pp. 1259-1263.

[10] S. Haykin, Adaptive Filter Theory, 2nd ed., Prentice-Hall, Toronto, 1991.

[11] S. Chen, S. A. Billings, "Representations of Nonlinear Systems: The NARMAX Model", International Journal of Control, Vol. 49, No. 3, 1989, pp. 1013-1032.

[12] D. A. Johns, W. M. Snelgrove, A. S. Sedra, "Adaptive Recursive State-Space Filters Using a Gradient-Based Algorithm", IEEE Transactions on Circuits and Systems, Vol. 37, No. 6, June 1990, pp. 673-684.

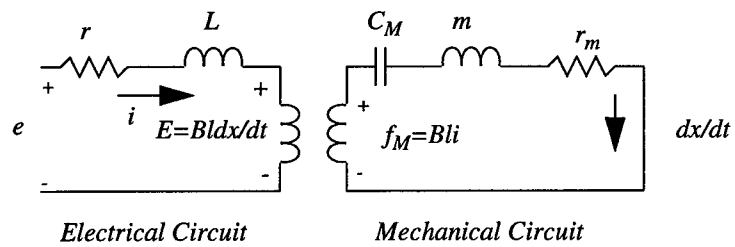
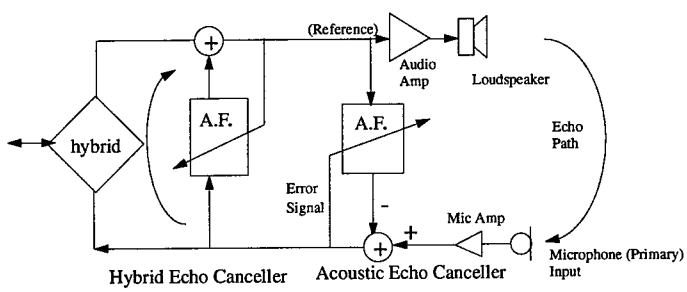


Figure 1. Loudspeaker Electro-mechanical Equivalent Model. e indicates the internal voltage of the generator, r is the total electrical resistance of the generator and voice coil, L is the inductance of the voice coil, i is the amplitude of the current in the voice coil, E is the voltage produced in the electrical circuit by the mechanical circuit. B is the magnetic flux density in the air gap, l is the length of the voice coil conductor, and x is the cone displacement. In the mechanical circuit m is the total mass of the coil, cone and air load. r_M is the total mechanical resistance due to dissipation in the air load and the suspension system. C_M is the compliance of the suspension and f_M is the force generated in the voice coil.

(a)



(b)

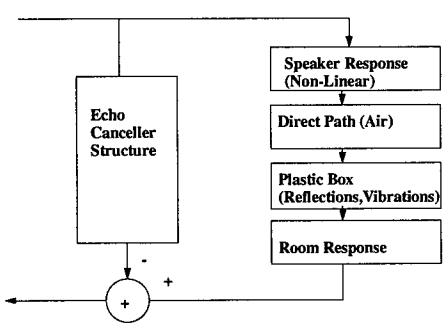


Figure 2. (a) Adaptive Echo Canceller Structure (b) Room/terminal Transfer Function is a combination of Speaker Non-linearities, Direct path, Plastic effects and Room response.

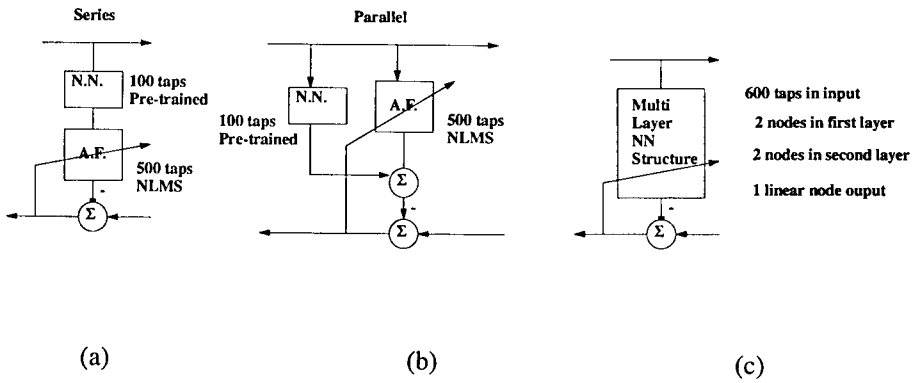


Figure 3. (a) Series model has a (100,2,1) 100 tap pretrained 2 layer network in series with the NLMS adaptive structure. (b) The parallel model has a (100,2,1) pretrained 2 layer network in parallel with the NLMS structure. (c) The fully adaptive (600,2,2,1) three layer network . In all cases the output neuron is linear and the hidden layers have a piecewise linear / sigmoidal activation function. The inputs are obtained from a tapped delay line.

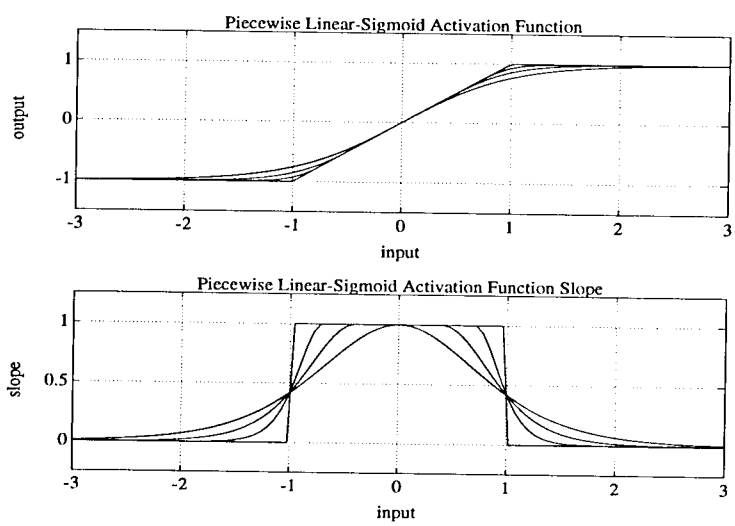


Figure 4. Piecewise linear/sigmoidal activation function and corresponding delta . The linear section with a value of +/-0.75 to +/-0.9 gave the best results in this study.

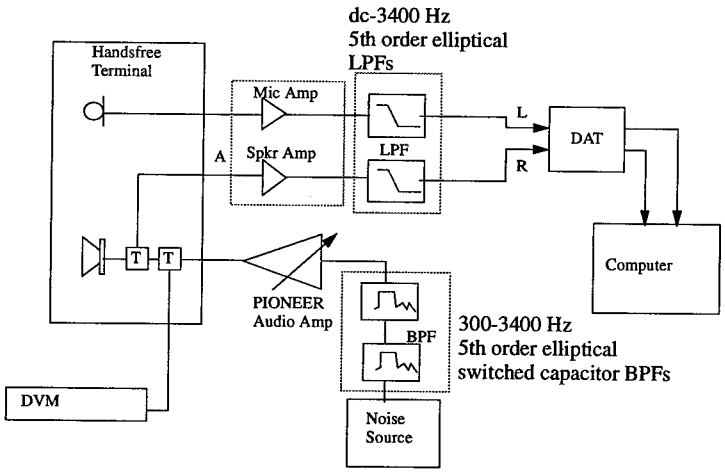


Figure 5. Experimental Setup. Primary and reference signals are recorded on DAT and later sampled to disk.

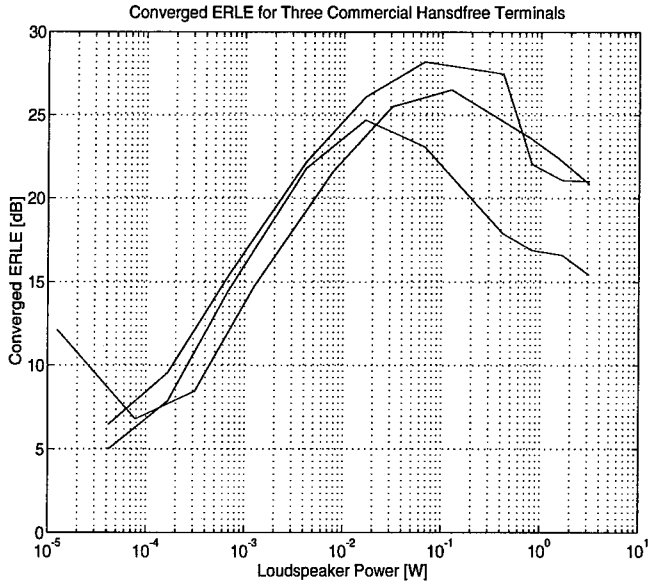


Figure 6. Converged ERLE vs. loudspeaker power for three different commercially available handsfree telephone terminals.. An AEC using the NLMS algoritm shows a decrease in ERLE as the volume increases At low volume levels, noise limits the achievable ERLE..

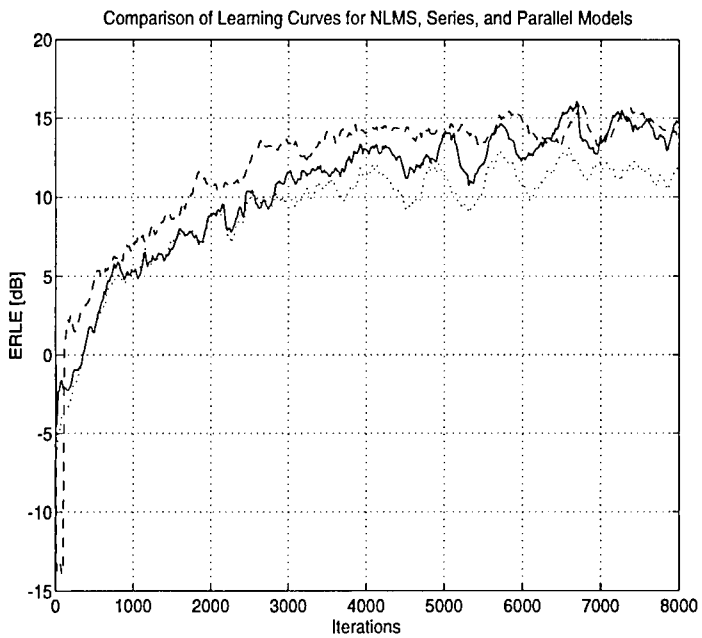


Figure 7. Convergence curves for the Series (dashed line) and Parallel (dotted line) models. The NLMS convergence curve (solid line)is shown for comparison.

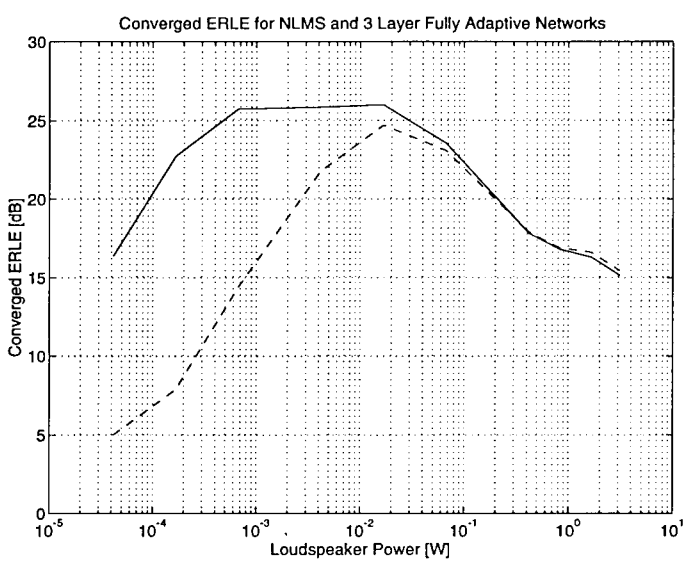


Figure 8. Converged ERLE plot vs. loudspeaker volume using a three layer fully adaptive neural network. Three layer network(solid line) shows over 10 dB improvement in ERLE at low to medium volumes. The NLMS algorithm (dashed line) is shown for comparison.

Minimum Error Training for Speech Recognition

Erik McDermott & Shigeru Katagiri

ATR Human Information Processing Research Laboratories
Hikari-dai 2-2, Seika-cho, Soraku-gun,
Kyoto 619-02, Japan
tel +81-7749-5-1055; fax +81-7749-5-1008

Abstract

In recent years several research groups have investigated the use of a new framework for minimizing the error rate of a classifier. The key idea is to define a smooth, differentiable loss function that incorporates all adaptable classifier parameters and that approximates the (non-smooth) actual performance error rate. Using a smoothed version of the actual error rate offers two main advantages: 1) the loss function can be minimized using gradient-based minimization methods, and 2) smoothing the error as calculated over a finite training set helps approximate unseen data, and thus can help generalization. This framework is applicable to a variety of classifier structures, including feed-forward neural networks, Learning Vector Quantization classifiers, and Hidden Markov Models. Here we describe a particular application in which a relatively simple distance-based classifier is trained to minimize errors in speech recognition tasks. The loss function is defined so as to reflect errors at the level of the final, grammar-driven recognition output. We show how the loss function can be made to reflect not just correctness/incorrectness at the string level, but also, for instance, a word spotting loss between the recognized string and the correct string. Thus, minimization of this loss can explicitly optimize the word spotting rate.

1 Introduction

Within the framework of Bayesian decision theory, the error rate can be defined along the following lines [4]. Suppose that in a classification task of M categories, when we observe a feature vector $\mathbf{x}$ belonging to category C_k and classify it as category C_j, we incur a loss δ_{jk}:

$$\delta_{jk} = \begin{cases} 1 & j \neq k \\ 0 & j = k \end{cases} \tag{1}$$

0-7803-2026-3/94 $4.00 © 1994 IEEE

The expected loss, or risk, corresponding to this loss is

$$R(C_j|\mathbf{x}) = \sum_{k}^{M} \delta_{jk} P(C_k|\mathbf{x}) \tag{2}$$

where $P(C_k|\mathbf{x})$ is the *a posteriori* probability of category C_k given $\mathbf{x}$. The goal is to choose the category C_j with the smallest risk. The Bayes rule for classification, in its most general form, is thus

$$\text{decide } C_j \text{ if } R(C_j|\mathbf{x}) < R(C_k|\mathbf{x}) \text{ for all } k \neq j. \tag{3}$$

For the above zero-one loss, this risk can be written as

$$R(C_j|\mathbf{x}) = 1 - P(C_j|\mathbf{x}) \tag{4}$$

and the Bayes decision rule to minimize the overall classification risk is to classify $\mathbf{x}$ as the category C_j with the largest *a posteriori* probability $P(C_j|\mathbf{x})$:

$$\text{decide } C_j \text{ if } P(C_j|\mathbf{x}) > P(C_k|\mathbf{x}) \text{ for all } k \neq j. \tag{5}$$

The implementation of the Bayes decision rule requires that we know the *a posteriori* probabilities for the categories in the problem. In most practical situations, it is difficult to estimate these probabilities, as the form of the true distributions is rarely known, and even when it is known, only a finite set of samples is available for estimation.

An approach that has arisen over the past few years is to overcome the above problems by directly formulating the classifier design problem as an error rate minimization problem. We refer to this approach alternatively as Minimum Classification Error / Generalized Probabilistic Descent (MCE/GPD) [9] and, more simply, as minimum error training. The key idea is to directly relate the design of the classifier to the quality of the actual classifier performance, or more specifically, to define in terms of the system parameters a loss function that is both 1) a close approximation of the real classification error rate, and 2) a smooth, differentiable function of the system parameters that can be used for practical optimization. In this paper we describe the theoretical framework of this approach applied to one particular classifier structure, a prototype-based system which incorporates Dynamic Time Warping to link phonetic states according to the grammar of the task at hand.

Note that the second version of the Bayes decision rule results from the use of the zero-one loss function above. Other losses may imply different uses of the *a posteriori* probabilities, according to (2). In speech recognition, and pattern recognition in general, it may be desirable to consider certain mistakes as more costly than others. In this spirit, we show how the MCE/GPD framework can accomodate losses that are more general than the 0-1 classification loss used so far, and thus capable of representing more fine-grained differences between categories.

Here, we present experimental results illustrating some of the properties of the minimum error training approach applied to a multi-state, prototype-based classifier, and show the feasibility of training using the more general loss function.

2 Prototype-Based Minimum Error Classifier

We here describe the minimum error training framework, from the perspective of a prototype-based application of this framework, the Prototype-Based Minimum Error Classifier (PBMEC), first described in [11]. This classifier uses collections of reference vectors associated with sub-phonemic states to calculate distances between a speech token and the categories of the task. The states are linked according to the grammar of the task, and a Dynamic Time Warping process will be used to find the state sequence that has the closest match to a given speech token. This state sequence will be given as the classification of the token. In the following, we represent one such token using a variable $\mathbf{x}_1^T$, which corresponds to a finite sequence of observations, $(\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$, where T is the duration of the token. T could change from token to token.

The framework for optimization used here, Generalized Probabilistic Descent (GPD) [6]-[7], is closely related to stochastic descent methods [5] and to what many researchers refer to as "online back-propagation." GPD can be described as the following adaptation process. For a given loss function $\ell_k(\mathbf{x}_1^T, \Lambda)$, where $\mathbf{x}_1^T$ is an input token belonging to category k, and where Λ represents the system parameters, we want to minimize the expectation of overall loss, $\mathcal{L}(\Lambda)$, which is the loss $\ell_k()$ integrated over all M categories and their probability densities:

$$\mathcal{L}(\Lambda) = \sum_{k}^{M} P(C_k) \int \ell_k(\mathbf{x}_1^T, \Lambda) p(\mathbf{x}_1^T | C_k) d\mathbf{x}_1^T \tag{6}$$

where $P(C_k)$ and $p(\mathbf{x}_1^T | C_k)$ are the class *a priori* and conditional probabilities respectively. For an infinite sequence of random samples $\mathbf{x}_1^T(\tau)$, and a suitably chosen step size sequence $\epsilon(\tau)$ [7], adapting the system parameters according to

$$\Lambda(\tau + 1) = \Lambda(\tau) - \epsilon(\tau) \nabla \ell_k(\mathbf{x}_1^T(\tau), \Lambda(\tau)) \tag{7}$$

has been shown to converge to a local minimum of $\mathcal{L}(\Lambda)$. Thus, though the overall loss is never directly calculated, it can be minimized by using the derivative of the local loss $\ell_k()$. In practice, as we don't have an infinite number of training tokens, random samples from the available training data are presented over and over for a pre-set number of

iterations, and the training target is to minimize the empirical error rate,

$$\mathcal{L}_1(\Lambda) = \frac{1}{N}\sum_{k}^{M}\sum_{i=1}^{N_k}\ell_k(\mathbf{x}_1^T(i,k),\Lambda) \tag{8}$$

where $\mathbf{x}_1^T(i,k)$ is the i-th token of category k, N is the total number of training samples and N_k is the number of training samples for each category k.

Given this approach to optimization, the next question is the nature of the local loss function $\ell_k(\mathbf{x}_1^T,\Lambda)$ to use in GPD optimization. If the goal is accurate pattern classification, Bayes decision theory suggests a 0-1 (0 for correct, 1 for incorrect) loss function, related to δ_{jk} described above. However, GPD requires that the loss function be a smooth (i.e. first order differentiable) function of the input token $\mathbf{x}_1^T$ and the system parameters Λ. The loss function defined in [6]-[7] thus uses a smoothed version of the "ideal", non-smooth, 0-1 loss. The use of this particular loss function is referred to as Minimum Classification Error (MCE) learning; for a smoothed loss that closely reflects the actual 0-1 loss, minimizing this loss using GPD will ideally yield a classifier that closely obeys the Bayes decision rule in its classifications, and thus minimizes the expected classification error rate.

Defining the zero-one classification loss function first involves defining, for each category j, a discriminant function $g_j(\mathbf{x}_1^T)$ reflecting the extent to which token $\mathbf{x}_1^T$ belongs to the category. This function is determined by the classifier structure and parameters Λ (we drop the use of the general term Λ in the following, and instead refer to specific classifier parameters). The PBMEC discriminant function for each category is defined in terms of a DTW procedure to link reference vector based phoneme models together according to the grammar of the task at hand. At the lowest level, the phoneme models are taken to consist of a connected sequence of sub-phonemic states, illustrated in Figure 1. Each state is assigned a number of reference vectors, analogous to the mean vectors used in a continuous Hidden Markov Model. These are used to generate an L_p norm-based *state distance* $e(\mathbf{x}_t,s)$, which is a function of a single feature vector $\mathbf{x}_t$ at time t and reference vectors belonging to the state s:

$$e(\mathbf{x}_t,s) = \left[\sum_{i=1}^{I_s}[(\mathbf{x}_t-\mathbf{r}_i^s)^t(\Sigma_i^s)^{-1}(\mathbf{x}_t-\mathbf{r}_i^s)]^{-\zeta}\right]^{-\frac{1}{\zeta}}, \tag{9}$$

where $\mathbf{r}_i^s$ denotes the (adaptable) i-th reference vector of state s, Σ_i^s is an adaptable positive definite matrix corresponding to $\mathbf{r}_i^s$, and I_s is the number of the reference vectors assigned to s. For a large ζ, the state distance becomes the distance to the closest reference vector, and each state can then be seen to correspond to a category in a Learning Vector Quantization classifier [8] [11]. A matrix of distances $D_{j,T,S}$ is defined to

be a matrix where each position (t, s) contains $e(\mathbf{x}_t, s)$ for the states of category j. The discriminant function for each phoneme/word/phrase j can then be defined as:

$$g_j(\mathbf{x}_1^T) = \left[\sum_{\theta} [V_\theta(D_{j,T,S})]^{-\xi} \right]^{\frac{1}{-\xi}} \tag{10}$$

where $V_\theta(D_{j,T,S})$ represents an accumulated sum, or *path distance*, along a possible DTW path θ through a region of $D_{j,T,S}$, and where S is the total number of states in category j. The decision rule here (rather than that described in (3) will be to choose the category with the smallest discriminant function value:

$$\text{decide } C_j \text{ if } g_j(\mathbf{x}_1^T) < g_k(\mathbf{x}_1^T) \text{ for all } k \neq j. \tag{11}$$

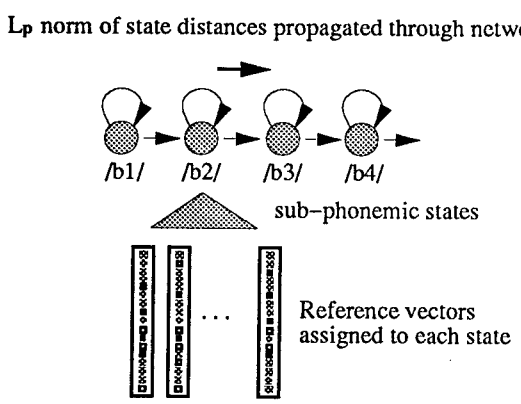


Figure 1: Structure of classifier at finest grain

Assuming that a token $\mathbf{x}_1^T$ of category k is presented to the classifier for training, and with M as before denoting the number of categories in the problem, a misclassification measure can then be defined as:

$$d_k(\mathbf{x}_1^T) = g_k(\mathbf{x}_1^T) - \left[\frac{1}{M-1} \sum_{j \neq k} g_j(\mathbf{x}_1^T)^{-\psi} \right]^{-\frac{1}{\psi}} \tag{12}$$

For a large ψ, this function will be negative for correct classifications ($g_k(\mathbf{x}_1^T) < g_j(\mathbf{x}_1^T)$), and positive for incorrect classifications ($g_k(\mathbf{x}_1^T) > g_j(\mathbf{x}_1^T)$). Depending on the grammar being used, the number of categories may be extremely large; thus, it is practical to assume a large ψ here, and only consider the top incorrect category, or top few incorrect categories, in calculating the misclassification measure.

A loss function can now be defined in terms of the misclassification measure $d_k = d_k(\mathbf{x}_1^T)$. Many choices exist for this function; for instance, a simple zero-one sigmoid:

$$\ell_k(d_k) = \frac{1}{1 + e^{-\alpha d_k}} \tag{13}$$

with a large α. One can see that this loss approximates an ideal binary loss function well and is continuous.

The adaptation process is then to perform GPD according to equation 7. By using different grammars, PBMEC can recognize and optimize at the level of a large variety of speech units, including connected words and phrases in continuous speech.

3 Practical Application of PBMEC to Continuous Speech Recognition

3.1 Use of Finite State Machine

A grammar-generated finite state machine linking phonetic states together is used to embed a grammar into the DTW matching procedure. This makes it possible to apply PBMEC to a variety of problems in continuous speech recognition. An example of a finite state machine is given in Figure 2. The categories of the task are taken to be all strings allowed by the finite state machine. The discriminant functions g_j for each category j are defined over one particular set of paths through the finite state machine. In practice, though, only the top matching categories need to be considered.

A time-synchronous DTW pass through the network similar to the Token Passing algorithm [14] was used, in tandem with an A* based N-best algorithm. During the learning phase, the top matching N incorrect categories will be pushed away in proportion to the derivative of the loss, while the correct category is pulled closer, also in proportion to the derivative of the loss; furthermore, categories will be pushed away only along the best within-category path.

3.2 A more general MCE/GPD loss function incorporating lexical / syntactic / semantic differences between categories

In MCE/GPD so far, whenever a category is mis-recognized as another category, the (ideal) loss is considered to be 1. This corresponds to the zero-one loss δ_{jk} mentioned above in the description of the Bayes decision rule. However, other losses can be used in the same Bayesian decision framework, and there are reasons to believe that the zero-one loss may be somewhat limited as a measure of performance. It may be

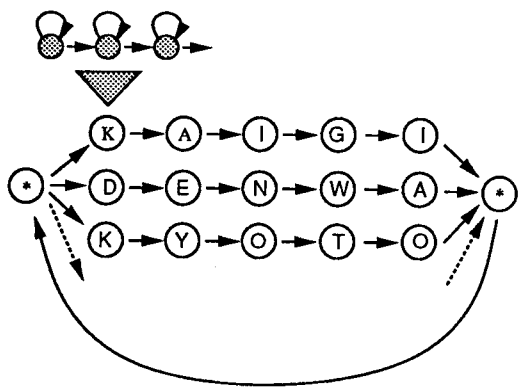


Figure 2: The grammar represented in a Finite State Machine determines how to concatenate different linguistic symbols. This defines the superstructure of the classifier. With the simple grammar illustrated here, any word can follow any other word, with the possible intervention of a garbage model, '*'.

desirable to consider more revealing error counts when comparing correct and incorrect strings, and to make the MCE/GPD classifier sensitive to these differences. We refer to these differences as inter-category symbolic distances. They could represent distances between syntactic parse trees, or word spotting distances between two strings of phonemes/words, using one of the usual ways of calculating deletion, insertion and substitution errors.

We thus take usual zero-one δ_{jk} loss to represent a more general loss, corresponding to an inter-category symbolic distance of the kind just described. The general loss δ_{jk} can then be used to weight the MCE/GPD loss in a continuous manner. For an input token $\mathbf{x}$ of category k with discriminant function $g_k(\mathbf{x})$, we have the usual misclassification measure (Equation 12) and typical loss function $\ell_k(\mathbf{x})$ (Equation 13). Then, we consider the expression

$$\frac{g_i(\mathbf{x})^{-\psi}}{\sum_{j \neq k}^{M} g_j(\mathbf{x})^{-\psi}} \tag{14}$$

as a measure of the extent to which the incorrect category i is represented in the bracketed term of the misclassification measure $d_k(\mathbf{x})$, from (12) above. For a large ψ, this expression will be close to one for the top matching incorrect category, and close to 0 for all other categories. Multiplying this expression, for each incorrect category i, by the inter-category symbolic distance δ_{ik}, and summing over all incorrect categories thus gives an aggregate, weighted symbolic distance between the correct category k and all other categories. Multiplying this aggregate

distance by the usual loss $\ell_k()$ gives the new loss $\ell_{k,\sigma}()$:

$$\ell_{k,\sigma}(d_k(\mathbf{x})) = \left[\frac{\sum_{i \neq k}^{M} g_i(\mathbf{x})^{-\psi} \delta_{ik}}{\sum_{j \neq k}^{M} g_j(\mathbf{x})^{-\psi}} \right] \ell_k(d_k(\mathbf{x})) \tag{15}$$

In the simplest case, if we take ψ to be very large, the new loss reduces to

$$\ell_{k,\sigma}(d_k(\mathbf{x})) = \delta_{ik} \ell_k(d_k(\mathbf{x})), \tag{16}$$

where δ_{ik} is the symbolic distance between the correct category k and the best matching incorrect category i. The new loss can be used for MCE/GPD adaptation in the same manner as before. Note that if all the inter-category distances, $\delta_{ik}, i \neq k$, are set to 1, the new loss becomes equal to the old loss.

3.3 Experiments in Continuous Speech Recognition

The experiments we describe here concerned the recognition of speech from a database of continuously spoken sentences from the ATR conference registration task, spoken by 9 female speakers. We describe experiments with this task to illustrate the nature of minimum error training, and to contrast training at the level of word-spotting, using the extended MCE loss, with string-level training using the usual zero-one loss.

Input speech was sampled at 12 kHz, Hamming windowed using a window size of 20 ms, and a 256-point FFT computed every 5 ms. Sixteen Melscale coefficients were then generated from the power spectrum for each frame. Seventy words were selected from the conference registration task vocabulary as keywords. A garbage model was used to describe sequences of phonemes not covered by this 70 word vocabulary. The grammar of the task was then defined as suggested above, allowing any word to follow any other word, with the possible intervention of garbage phoneme sequences. MCE/GPD training was performed subject to two separate criteria: 1) optimal string (sentence) recognition and 2) optimal word spotting recognition. In the latter mode, the inter-category symbolic distance δ_{jk} was set to be the word-spotting string distance between the strings j and k, calculated using a DP procedure. Thus, δ_{jk} as used in the second criterion is a finer measure of string similarity than the all or nothing, 0-1 δ_{jk} used in the first criterion.

One hundred and ten sentences were available for each of the nine speakers; half of these sentences were used for training and half for testing. The reference vectors were initialized using K-means clustering. The training procedure was performed for both criteria of optimal string recognition and optimal word spotting recognition.

Figure 3 shows the actual string accuracy, word spotting accuracy, and modelled accuracy (derived from the empirical error rate defined in (8)), for both training and testing data, for PBMEC trained using both string and word level training criteria. The results show good matches

Training criterion	Training data			Testing data	
	string accuracy	word accuracy	modelled accuracy	string accuracy	word accuracy
string–level	56.2	88.2	56.3	14.2	69.8
word–level	48.4	88.9	89.0	14.8	70.3

Figure 3: Recognition accuracies for string- and word-level criteria

between the actual string accuracy and the modelled loss on the one hand, and between the actual word spotting accuracy and the modelled loss, on the other.

4 Conclusion

We have described the practical application of MCE/GPD to a prototype-based classifier. The classifier incorporates a grammar-generated finite state machine linking different phonetic states together, and can be used to classify continuously spoken sentences. We have shown how this classifier can be trained to minimize the classification loss using MCE/GPD optimization at the level of the final, grammar driven recognition output. We have also described how loss functions reflecting the lexical, syntactic or semantic significance of different classification mistakes can be integrated into the MCE/GPD framework. In particular, we have illustrated the feasibility of directly optimizing the word spotting rate. A word spotting accuracy of 70% was obtained for a difficult continuous speech recognition task. Incorporation of additonal constraints into the classifier, such as penalties for word transitions, may improve this accuracy, and reveal greater differences between training to optimize word spotting performance, and training to optimize sentence recognition.

References

[1] Amari, S. (1967). *A Theory of Adaptive Pattern Classifiers.* IEEE Transactions, EC-16, No. 3, pp. 299-307.

[2] Chang, P.-C. and Juang, B.-H. (1992). *Discriminative Template Training for Dynamic Programming Speech Recognition.* Proc. IEEE ICASSP-92, pp. I:493-6.

[3] Chou, W., Juang, B.-H. and Lee, C.-H. (1992). *Segmental GPD Training of HMM Based Speech Recognizer.* Proc. IEEE ICASSP-92, pp. I:473-476.

[4] Duda, R.O. & Hart, P.E. (1973). *Pattern Classification and Scene Analysis.* John Wiley & Sons.

[5] Fu, K.-S. (1968). *Sequential Methods in Pattern Recognition and Machine Learning.* Academic Press, 1968.

[6] Juang, B.-H. and Katagiri, S. (1992). *Discriminative Learning for Minimum Error Classification.* IEEE Transactions on Signal Processing, vol. 40, No. 12, December 1992.

[7] Katagiri, S., Lee, C.-H. and Juang, B.-H. (1990). *A Generalized Probabilistic Descent Method.* Proc. of Acoustical Society of Japan, Fall Meeting, pp. 141-142.

[8] Katagiri, S., Lee, C.-H. and Juang, B.-H. (1991). *Discriminative Multilayer Feedforward Networks.* Proc. 1991 IEEE Workshop on Neural Networks for Signal Processing, August 1991, pp 11-20.

[9] Katagiri, S., Biem, A., and Juang, B.-H. (1993). *Discriminative Feature Extraction.* in "Artificial Neural Networks for Speech and Vision," Chapter 18. Ed. R. Mammone, Chapman & Hall. pp. 278-293.

[10] Komori, T, and Katagiri, S. (1992). *Application of GPD Method to Dynamic Time Warping-based Speech Recognition.* Proc. IEEE ICASSP-92, pp. I:497-500.

[11] McDermott, E. & Katagiri, S. (1992). *Prototype-Based Discriminative Training for Various Speech Units.* Proc. IEEE ICASSP-92, pp. I:417-420.

[12] Rainton, D. and Sagayama, S. (1992). *Minimum Error Classification Training of HMMs- Implementation Details and Experimental Results.* Journal of the Acoustic Society of Japan, Vol.13, No.6, pp. 379-387, November 1992.

[13] Su, K.-Y. and Lee, C.-H. (1991). *Robustness and Discrimination Oriented Speech Recognition Using Weighted HMM and Subspace Projection Approaches.* Proc. IEEE ICASSP-91, pp. I:541-544.

[14] Young, S., Russell, N.H. and Thornton J.H.S. (1991) *The use of syntax and multiple alternatives in the VODIS voice operated database inquiry system.* Computer Speech and Language 5 , 65-80.

CONNECTIONIST MODEL COMBINATION FOR LARGE VOCABULARY SPEECH RECOGNITION

M. M. Hochberg G. D. Cook S. J. Renals A. J. Robinson
Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ, England

Abstract–Recent reports in the statistics and neural networks literature have expounded the benefits of merging multiple models to improve classification and prediction performance. The Cambridge University connectionist speech group has developed a hybrid connectionist-hidden Markov model system for large vocabulary, talker independent speech recognition. The performance of this system has been greatly enhanced through the merging of connectionist acoustic models. This paper presents and compares a number of different approaches to connectionist model merging and evaluates them on the TIMIT phone recognition and ARPA Wall Street Journal word recognition tasks.

INTRODUCTION

An acoustic pre-processor or *front-end* is a common feature of all large vocabulary speech recognition systems. The front-end maps the sampled waveform onto a lower-dimensional representation of the acoustic signal. Typically, the specific mapping is selected as the front-end which performs best on some development test set. Since different front-ends may provide better representations for different acoustic events (e.g., phoneme class, talker, etc.), it would seem advantageous to sensibly merge multiple front-ends and their associated models.

There has been speech recognition research into merging multiple sources of information. For example, work at BBN has addressed merging the parameters of speaker-dependent hidden Markov models (HMMs) to obtain a speaker-independent system [1] and Cohen and Franco at SRI have merged a conventional HMM and multi-layer perceptron [2]. Recently, model combination has been shown to be a promising area of neural network research. Techniques such as *Generalized Stacking* [3] and Bayesian approaches [4] have been explored as a means to most effectively utilize all the available information. This paper presents an application of connectionist model merging to speech recognition. Multiple acoustic representations are merged resulting in a significant reduction in the recognition error rate.

THE HYBRID CONNECTIONIST-HMM

The hybrid connectionist-HMM employs the same basic framework as described in [5], but utilizes a different connectionist component. The speech recognition sys-

0-7803-2026-3/94 $4.00 © 1994 IEEE

tem uses a recurrent network to map a sequence of acoustic vectors to a sequence of posterior phone probabilities. The network outputs are used as estimates of the observation probabilities within an HMM framework, i.e., the observations are considered as a stochastic process on a non-observable, first-order Markov chain. Given new acoustic data and the connectionist-HMM framework, the maximum *a posteriori* phone or word sequence is then extracted using standard Viterbi decoding techniques.

The basic acoustic modeling system is illustrated in Figure 1. At each 16ms frame, an acoustic vector, $\mathbf{u}(t)$, is presented at the input to the network along with the previous state vector, $\mathbf{x}(t-1)$. These two vectors are passed through a single-layer, fully-connected, feed-forward network to give the output vector, $\mathbf{y}(t)$, and the next state vector, $\mathbf{x}(t)$. Forward acoustic context is modeled by expanding the input vector to cover additional frames and by delaying the target. The state vector provides the mechanism for modeling the dynamics of the acoustic signal in various contexts.

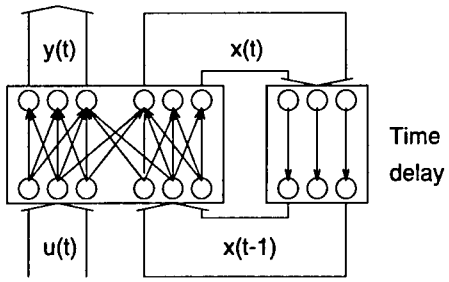


Figure 1: The recurrent net used for phone probability estimation.

Each output channel corresponds to a particular phone in the phone set. The use of the *softmax* nonlinearity for the output nodes with the cross-entropy training criterion implies that the outputs can be considered estimates of the posterior probability of the phones given the (local in time) acoustic data. This network is trained by back-propagation through time. (A more complete description of the network may be found in [6].)

THE MODELS

Because the goal of this work is to reduce the recognition error rate through merging multiple recurrent networks, it is important that each portion of the speech signal can be modeled by at least one of the individual networks. In the experiments presented here, the parameters for each network are estimated on the same speech data, but processed with different front-ends. Two successful spectral representations have been found to be a 20 channel mel-scaled filter bank with voicing features [7] and 12th order cepstral coefficients derived from perceptual linear prediction [8]. The filter bank and cepstra are referred to in this paper as MEL+ and PLP, respectively. In addition, because the recurrent network is time asymmetric, training the network to classify

forward in time will result in different dynamics than training to classify backwards in time. Based on the above considerations, four networks were constructed from the possible representations; FORWARD MEL+, BACKWARD MEL+, FORWARD PLP, and BACKWARD PLP.

MODEL COMBINATION

Probability Domain Merging

The most straightforward approach to merging the recurrent networks is through a linear combination of the model outputs. In the most general framework, the merged estimate of the posterior probability of phone i given the acoustic data up to time t is given by

$$y_i(t) = \sum_{k=1}^{K} \beta_{ik} y_i^{(k)}(t) \tag{1}$$

where $y_i^{(k)}(t)$ is the estimate of the kth model and β_{ik} are the merging weights. Note that the weights can be dependent on the input data, e.g., $\beta_{ik} = \beta_{ik}(\mathbf{u}(t))$. Sufficient conditions on the βs to guarantee a statistical interpretation of the output are that they are *tied* across phones (i.e., $\beta_{ik} = \beta_k$), *sum-to-one* (i.e., $\sum_k \beta_{ik} = 1$), and are *non-negative*. With these conditions, the merged output will meet the constraints needed for interpretation of the output as the posterior phone probabilities. As is seen in the results section, relaxing these constraints does not necessarily lead to poorer performance.

Log-Probability Domain Merging

For computational reasons, the mapping of the phone probabilities into recognized word strings is usually performed in the log-probability domain. This has led to experiments evaluating merges performed after the conversion of the network output to the log domain, i.e.,

$$\log y_i(t) = \sum_{k=1}^{K} \beta_{ik} \log y_i^{(k)}(t). \tag{2}$$

With this approach, it is difficult to assign a probabilistic interpretation to the merged outputs. However, if the models are assumed to be independent, then the estimated joint likelihood of the different data is proportional to the product (or sum in the log-domain) of the network outputs.

Merge Criteria

Given the connectionist-HMM framework, there are number of different approaches to determine the βs. In all cases where training data was required to learn the merge

parameters, the data was taken from an independent development set. Although the amounts of data in the training set was quite large, this approach was taken to further reduce the chance of obtaining a merge with substantial bias.

Uniform. The first attempt at combining networks assumed the merge weights were independent of the data with uniform probabilities, i.e., $\beta_{ik} = 1 / K$. This approach maintains the probabilistic interpretation of the merged output in the probability domain. Good initial results using this simple merging approach [9] has led to the evaluation of more complex merging techniques.

Linear Regression. Recent work has shown that merging regression predictors through linear regression (referred to as *Stacked Regressions*) produce an estimator that is better than any of the individual estimates [10]. The regression approach determines the βs through minimizing the sum-squared error

$$\sum_t \sum_i \left(\hat{y}_i(t) - \sum_{k=1}^{K} \beta_{ik} y_i^{(k)}(t) \right)^2 \tag{3}$$

on a development set. Here, $\hat{y}$ is the desired target and the regression parameters, β_{ik}, are assumed to be fixed after training. In [10], Breiman found that constraints on the βs improved performance. In this paper, the regression merging is evaluated with and without constraints such that the merge weights are tied across models and/or sum-to-one. It was rarely found that any of the merge parameters were ever less than zero.

Mixture of Experts. This framework (see Figure 2) employs a gating network to determine data-dependent merge parameters. The approach is equivalent to Jordan and Jacob's *mixture of experts* [11] with fixed experts. The data-dependent merging coefficients can be determined by maintaining a probabilistic interpretation and employing the expectation-maximization (EM) algorithm [12]. Let $U = \{\mathbf{u}(t)\}$ be the set of acoustic training data for each frame and let $C = \{c(t)\}$ be the corresponding phone. Assuming each frame is independent results in the likelihood $L(U)$ given as

$$L(U) = p(U|C, Y) = \prod_{t=1}^{T} p(\mathbf{u}(t)|c(t), \mathbf{y}^{\mathcal{M}}(t)) \tag{4}$$

where $Y = \{\mathbf{y}^{\mathcal{M}}(t)\}$ represents the outputs of all the models, i.e, $\mathbf{y}^{\mathcal{M}}(t) = \{\mathbf{y}^{(m)}(t)\}$. The merging comes about by assuming that $p(\mathbf{u}(t)|c(t), \mathbf{y}^{\mathcal{M}}(t))$ is a mixture density of the form

$$p(\mathbf{u}(t)|c(t), \mathbf{y}^{\mathcal{M}}(t)) = \sum_{k=1}^{K} p(\mathcal{M}_k|c(t), \mathbf{y}^{\mathcal{M}}(t))\, p(\mathbf{u}(t)|\mathcal{M}_k, c(t), \mathbf{y}^{\mathcal{M}}(t)) \tag{5}$$

where $\mathcal{M}_k$ represents the kth model. Here, the mixing coefficients $p(\mathcal{M}_k|i, \mathbf{y}^{\mathcal{M}}(t)) = \beta_{ik}(\mathbf{u}(t))$. As in [11], a generalized linear model is used as the gating network to compute $\beta_{ik}(\mathbf{u}(t))$.

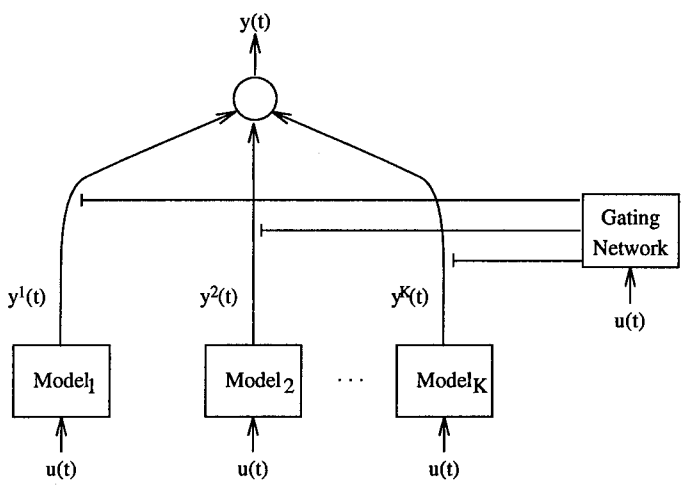


Figure 2: Mixture-of-experts framework.

The generalized EM algorithm [12] is an iterative approach used to compute the maximum likelihood estimates of the gating network parameters. Each iteration applies two conceptual steps. The E-step computes the posterior probability of each model

$$p(\mathcal{M}_k|\mathbf{u}(t), c(t)) = \frac{\beta_{ik}(\mathbf{u}(t))y_{c(t)}^{(k)}(t)}{\sum_{n=1}^{K} \beta_{in}(\mathbf{u}(t))y_{c(t)}^{(n)}(t)} \tag{6}$$

for each pair $\{\mathbf{u}(t), c(t)\}$ in the development set. The M-step estimates the parameters of the generalized linear model using the Iteratively Re-weighted Least Squares procedure (IRLS) [11] with $\mathbf{u}(t)$ as the inputs and $p(\mathcal{M}_k|\mathbf{u}(t), c(t))$ as the desired outputs. This procedure results in a method for learning the parameters of the gating network for each phone. The procedure insures that the merging weights do sum-to-one.

The standard mixture of experts approach has the weights tied across models. This is accomplished by assuming $p(\mathcal{M}_k|c(t), \mathbf{u}(t)) = p(\mathcal{M}_k|\mathbf{u}(t))$ and results in many fewer free parameters. A variation of this approach is to replace the input of the gating network with the output of one of the networks. In the experiments described later in the paper, the gating network inputs were either three contiguous frames of the acoustic feature vector or a single frame of a network output.

In addition to the above variations, the case where there are no inputs was also considered. In this case, the gating network outputs constant values and the EM algorithm [12] specifies an iterative solution for the maximum likelihood coefficients. The parameter update equation becomes simply

$$\hat{\beta}_{ik} = \frac{1}{T}\sum_{t=1}^{T} \frac{\beta_{ik}y_i^{(k)}(t)}{\sum_{n=1}^{K} \beta_{in}y_i^{(n)}(t)}\delta_{c(t),i} \tag{7}$$

where $\hat{\beta}$ represents the updated estimate and δ is the Kronecker delta function.

EXPERIMENTAL RESULTS

Recognition Tasks

TIMIT. TIMIT is one of the standard speech corpora for the evaluation of phone recognition systems. It is divided into 462 training speakers and 168 test speakers. Each speaker utters two calibration sentences and eight sentences that are used in these evaluations, giving a training set of 3696 sentences and 1344 test sentences. In the experiments described here, 1152 of the test sentences were used for cross-validation estimation of the merging parameters and 192 (the core test) sentences were used for testing.

Wall Street Journal. The Wall Street Journal (WSJ) is the current ARPA large-vocabulary recognition task. The training data used was the short-term speakers from the WSJ0 corpus consisting of 84 speakers uttering a total of 7,200 sentences. The November 1993 spoke 5 development test data was used for estimation of the merging parameters. This data was collected from 10 talkers and 216 sentences using a Sennheiser microphone. Results are reported for the November 1993 spoke 6 development test. This test has 202 sentences from the same 10 talkers as spoke 5. The test are from a closed 5,000 word, non-verbalized punctuation vocabulary using the standard bigram language model [13].

Results and Analysis

Tables 1 and 2 show the TIMIT and WSJ results for the various approaches to model merging. In the tables, *frame rate* is the classification rate of the merged system on the development data, *error rate* is the phone or word recognition error rate on the test set computed as

$$100 \times \frac{\#\text{ insertions} + \#\text{ deletions} + \#\text{ substitutions}}{\#\text{ phones}} \tag{8}$$

and *improvement* is measured relative to the average error rate. For the EXPERTS merges, ACOUS., PROB., MEL+, and PLP indicate the type of inputs to the gating network. For the TIMIT experiment, only the FORWARD AND BACKWARD MEL+ front-ends were merged.

The tables clearly show the benefits of model merging. Each of the networks trained on different front-ends have similar performance, but the frame rate is substantially improved by merging the network outputs. This improvement is reflected in the error rate by a reduction of 9% and 27% for the TIMIT and WSJ tasks, respectively. For both tasks, the simple uniform merging accounts for most of the improvement and the best results were achieved by merging in the log-probability domain.

For the regression merge approach, not much variation in either the frame rate or the recognition error rate is observed across the different types of constraints.

Merge Type	Constraints	Frame Rate %	Error Rate %	Improv. %
FORWARD ONLY	-	65.9	31.7	-
BACKWARD ONLY	-	65.7	31.8	-
AVERAGE	-	65.8	31.8	-
UNIFORM	Tied, Sum	69.3	29.4	7.5
UNIFORM (LOG)	Tied	69.2	29.0	8.8
REGRESSION	Tied, Sum	69.3	29.3	7.9
REGRESSION	Sum	69.3	29.3	7.9
REGRESSION	Tied	69.3	29.1	8.5
REGRESSION		69.7	29.3	7.9
EXPERTS (ACOUS.)	Tied, Sum	69.3	29.2	8.2
EXPERTS (ACOUS.)	Sum	69.5	29.4	7.5
EXPERTS (PROB.)	Tied, Sum	69.4	29.1	8.5
EXPERTS (PROB.)	Sum	69.0	29.5	7.2

Table 1: TIMIT phone recognition results for different merge approaches. Frame rate is computed on development data and error rate is computed on test data.

Merge Type	Constraints	Frame Rate %	Error Rate %	Improv. %
FORWARD MEL+	-	78.1	15.0	-
FORWARD PLP	-	76.6	15.1	-
BACKWARD MEL+	-	73.8	15.5	-
BACKWARD PLP	-	76.1	14.4	-
AVERAGE	-	76.2	15.0	-
UNIFORM	Tied, Sum	82.5	11.4	24.0
UNIFORM (LOG)	Tied	82.8	11.0	26.7
REGRESSION	Tied, Sum	82.5	11.5	23.3
REGRESSION	Sum	82.8	11.3	24.7
REGRESSION	Tied	82.6	11.7	22.0
REGRESSION		83.1	11.4	24.0
EXPERTS (MEL+)	Tied, Sum	82.7	11.4	24.0
EXPERTS (PLP)	Tied, Sum	82.7	11.4	24.0

Table 2: WSJ word recognition results for different merge approaches. Frame rate is computed on development data and error rate is computed on test data.

This indicates that over-fitting of the training data does not seem to be a problem. Examination of the sum-squared error obtained from (3) in the merge process also shows little variation for the different constraints or from the uniform case. This implies that – at least for these networks – little improvement over the uniform merge can be expected.

TIMIT results obtained with the mixture of experts approach show that a single gating network achieves better performance than a set of separate gating networks for each phone. This is most likely due to insufficient training data to estimate the multiple gating network parameters. Even with large amounts of training data, some phones occur very infrequently which makes it difficult to estimate the parameters of a gating network. Conditioning the mixture of experts gating network on the acoustic signal or network output achieved similar performance on TIMIT. For WSJ, using MEL+ or PLP features as inputs to the gating network had no effect on the recognition results.

As indicated in Tables 1 and 2, simple model merging improves performance but the use of more complex merging strategies does not significantly improve the recognition results. Analysis of the TIMIT task indicates that the different merge types are all reasonably close to the optimal merge. Figure 3 shows the results of a line search on the merge parameter with the tied and sum-to-one constraints. It

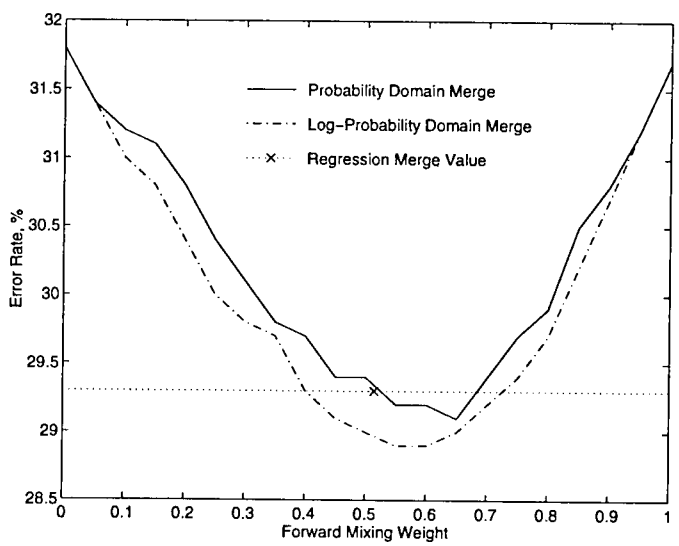


Figure 3: Error rate versus forward network mixing coefficient for probability and log-probability domain mixing on the TIMIT task.

is easy to see that the best performance is certainly in the region around 0.5 (the uniform merge). The regression estimate of the merge parameter shown in the figure is 0.51 and the mixture of experts has a mean value of 0.52 with variance 0.005. This implies that better/additional acoustic models are necessary to greatly improve the

TIMIT results.

To determine which front-end merge provides the most improvement, forward and backward models with the same spectral representation were merged. Similar merges were performed across spectral representations with the same time indexing. The results are shown in Table 3 and indicate that both the variation in spectral representation and processing of different data are important to the merging process. In addition, merging all front-ends resulted in better performance than any of the subset of the front-end merges.

Merge Type	ins. %	sub. %	del. %	errors %
AVERAGE FRONT-END MERGE	1.0	8.5	3.4	12.8
AVERAGE TIME MERGE	0.8	8.3	3.6	12.7

Table 3: Connectionist model subset merging results on the WSJ word recognition task.

DISCUSSION

This paper investigated various approaches to merging multiple, different acoustic models within the hybrid connectionist-HMM framework. Given the chosen acoustic models (recurrent networks), it was found that

- merging results in a significant reduction in error rate,
- the uniform, linear regression, and mixture of experts approaches all had similar performance, and
- the log-probability domain merging gave consistently better results.

The results presented here indicate the potential of this model merging approach. The fact that the linear regression and mixture of experts approaches did not do much better than the uniform merge may be a result of the selected networks. These techniques should show more significant gains when merging networks with different performance levels. As Figure 3 shows, the uniform merge of the log-domain probabilities may not be the best choice and research is planned in this area. In conclusion, this work shows model merging within the hybrid connectionist-HMM framework to be a very powerful mechanism for improving speech recognition performance. TIMIT results obtained with the merged system are the best known to the authors. Even with orders of magnitude fewer parameters, the merged system is competitive with state-of-the-art HMM systems on the WSJ task.

ACKNOWLEDGEMENTS

This work was partially funded by ESPRIT project 6487 (WERNICKE). Two of the authors (T.R. and S.R.) are supported by SERC fellowships. The authors would like to acknowledge MIT Lincoln Laboratory for providing the language model and Dragon Systems for providing the pronunciation lexicon for the WSJ task.

REFERENCES

[1] F. Kubala and R. Schwartz, "A new paradigm for speaker-independent training," in *1991 International Conference on Acoustics, Speech, and Signal Processing*, (Toronto, Canada), pp. 833–836, IEEE, May 1991.

[2] S. Renals, N. Morgan, M. Cohen, and H. Franco, "Connectionist probability estimation in the Decipher speech recognition system," in *1992 International Conference on Acoustics, Speech, and Signal Processing*, (San Francisco, California), pp. 601–604, IEEE, Mar. 1992. Volume 1.

[3] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[4] W. Buntine, "Learning classification trees," in *Artificial Intelligence Frontiers in Statistics III* (D. J. Hand, ed.), pp. 182–201, Chapman & Hall, 1993.

[5] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach.* The Kluwer International Series in Engineering and Computer Science. VLSI, Computer Architecture, and Digital Signal Processing, Boston, Massachusetts: Kluwer Academic Publishers, 1994.

[6] A. J. Robinson, "An application of recurrent nets to phone probability estimation," *IEEE Transactions on Neural Networks*, vol. 5, pp. 298–305, Mar. 1994.

[7] T. Robinson, "Several improvements to a recurrent error propagation network phone recognition system," Tech. Rep. CUED/F-INFENG/TR.82, Cambridge University Engineering Department, Sept. 1991.

[8] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *Journal of the Acoustical Society of America*, vol. 87, pp. 1738–1752, 1990.

[9] M. M. Hochberg, S. J. Renals, and A. J. Robinson, "ABBOT: The CUED hybrid connectionist-HMM large-vocabulary recognition system," in *Proc. of Spoken Language Systems Technology Workshop*, ARPA, Mar. 1994.

[10] L. Breiman, "Stacked regressions," Tech. Rep. 367, Department of Statistics, University of California, Berkeley, August 1992.

[11] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, pp. 181–214, Mar. 1994.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm (with discussion)," *J. Roy. Statist. Soc.*, vol. B39, pp. 1–38, 1977.

[13] D. B. Paul and J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proc. Fifth DARPA Speech and Natural Language Workshop*, (Harriman, New York), pp. 357–362, DARPA, Morgan Kaufman Publishers, Inc., Feb. 1992.

NEURAL TREE NETWORK/VECTOR QUANTIZATION PROBABILITY ESTIMATORS FOR SPEAKER RECOGNITION

Kevin Farrell, Stephen Kosonocky, and Richard Mammone
CAIP Center, Rutgers University
Core Building, Frelinghuysen Road
Piscataway, New Jersey 08855-1390
Phone: (908) 445-0573, FAX: (908) 445-4775
email: farrell@caip.rutgers.edu

Abstract - A new classification system for text-independent speaker recognition is presented. This system combines the output probabilities of distortion-based classifiers and a discriminant-based classifier. The distortion-based classifiers are the vector quantization (VQ) classifier and Gaussian mixture model (GMM). The discriminant-based classifier is the neural tree network (NTN). The VQ and GMM classifiers provide output probabilities that represent the distortion between the observation and the model. Hence, these probabilities provide an *intraclass* measure. The NTN classifier is based on discriminant training and provides output probabilities that represent an *interclass* measure. Since, these two classifiers base their decision on different criteria, they can be effectively combined to yield improved performance. Two combining methods are evaluated for several speaker recognition tasks, including speaker verification and closed set speaker identification. The results show the both methods to yield advantages for the speaker recognition tasks.

INTRODUCTION

Speaker recognition refers to the capability of recognizing a person based on his or her voice. Specifically, this consists of either speaker verification or speaker identification. The objective of speaker verification is to verify a person's claimed identity based on a sample of speech from that person. The

0-7803-2026-3/94 $4.00 © 1994 IEEE

objective of speaker identification is to use a person's voice to identify that person among a predetermined set of people. An additional characteristic of speaker identification systems is whether they are *closed set* or *open set*. Closed set speaker identification refers to the case where the speaker is known *a priori* to be a member of a set of N speakers. Open set speaker identification includes the additional possibility where the speaker may be from outside the set of N speakers. Open set speaker identification requires an additional thresholding step to determine if the speaker is "out of set". This paper provides results for text-independent speaker verification and closed set speaker identification.

The classification stage for text-independent speaker recognition is typically implemented by modeling each speaker with an individual classifier. Given a specific feature vector, a speaker model associates a number corresponding to the degree of match with that speaker. The stream of numbers obtained for a set of feature vectors can be used to obtain a likelihood score for that speaker model. For speaker identification, the feature vectors for the test utterance are applied to all speaker models and the corresponding likelihood scores are computed. The speaker is selected as having the largest score. For speaker verification, the feature vectors are applied only to the speaker model for the speaker to be verified. If the likelihood score exceeds a threshold, the speaker is verified or else is rejected.

The more common methods of constructing speaker models for text-independent speaker recognition use unsupervised training methods. These approaches include vector quantization [1], hidden Markov models [2], and Gaussian mixture models [3]. Here, only the data for that speaker is used to train the that speaker's model. Alternative methods for building speaker models use supervised training, such as that in multilayer perceptrons [4], radial basis functions [5], and neural tree networks [6]. Here, a speaker's model is trained with the data from possibly all speakers in the population. Speaker models based on supervised training capture the differences of that speaker to other speakers (interspeaker variability), whereas models based on unsupervised training use a similarity measure (intraspeaker variability).

The fact that supervised and unsupervised classifiers base their discrimination on different criteria leads one to believe that they can be combined for improved performance. This paper evaluates two methods for combining the output probabilities of the NTN and VQ classifiers and demonstrates these methods to improve performance for the speaker recognition task. The following sections review the VQ and NTN classifiers along with their application to speaker recognition. The hybrid NTN/VQ systems are then presented. This section is followed by experimental results and the conclusion of this paper.

VECTOR QUANTIZATION

The unsupervised classifier considered here is vector quantization (VQ). VQ is a clustering algorithms, which falls under the category of unsupervised training, i.e., the class label is not used. The VQ algorithm uses clustering to automatically group the training data into its individual modes or classes. Numerous VQ algorithms exist, including the Linde-Buzo-Gray (LBG) [7] method, which is used here.

The VQ classifier can be used for speaker recognition [1] as follows. Given the extracted feature vectors from a speaker, a codebook is constructed for that speaker. This process is repeated for all speakers in the population. For speaker identification, the feature vectors from a test utterance are applied to each of the codebooks. For a given codebook, the centroid, which is closest to the test vector is found and the distance to this centroid is accumulated for that codebook. The speaker is selected as corresponding to the codebook with the minimum accumulated distance. For speaker verification [8], the test vectors are only applied to the model for the speaker to be verified. The accumulated minimum distance is computed and normalized to the number of testing vectors. This normalized distance is compared to a threshold to decide if the speaker will be rejected or accepted.

NEURAL TREE NETWORK

The supervised classifier considered here is the neural tree network (NTN) [9]. The NTN is a hierarchical classifier that uses a tree architecture to implement a sequential linear decision strategy. Each node at every level of the NTN divides the input training vectors into a number of exclusive subsets of this data. The leaf nodes of the NTN partition the feature space into homogeneous subsets, i.e., a single class at each leaf node. The NTN is recursively trained as follows. Given a set of training data at a particular node, if all data within that node belongs to the same class, the node becomes a leaf. Otherwise, the data is split into several subsets, which become the children of this node. This procedure is repeated until all the data is completely uniform at the leaf nodes, else some pruning criteria is satisfied [9, 10]. Each leaf is assigned a label belonging to the class majority at that leaf. During testing, a feature vector is directed through the tree until it arrives at a leaf. The vector is then assigned the label of that leaf.

The NTN as presented in [9] is strictly a classification tree. Here, the output of the classifier consists of only a class label. Thus, the posterior probability estimate of class C_i provided by the NTN is a binary value:

$$P_{NTN}(C_i|x) = \{1, 0\}, \tag{1}$$

where x is a vector to be classified.

In [6], a modified NTN (MNTN) was presented that allows for a discrete estimate of the posterior probability. The assignment of probability measures

occurs within a technique called *forward* pruning. The forward pruning algorithm consists of simply truncating the growth of the tree beyond a certain level. For the leaves at the truncated level, a vote is taken and the leaf is assigned the label of the majority. In addition to a label, the leaf is also assigned a confidence. The confidence is computed as the ratio of the number of elements for the vote winner to the total number of elements. The confidence provides a measure of confusion for the different regions of feature space. The confidence, or posterior probability estimate, provided by the MNTN is:

$$P_{MNTN}(C_i|x) = \frac{k_{ij}}{\sum_{l=1}^{M} k_{lj}}, \tag{2}$$

where k_{ij} is the number of samples of class i in leaf j (as determined from the training data) and the denominator term corresponds to the number of samples at that leaf.

A MNTN can be trained for each speaker in the population as follows [6]. The MNTN for each speaker is presented with the data for all speakers. Here, the extracted feature vectors for that speaker are labeled as "one" and the extracted feature vectors for everyone else are labeled as "zero". A binary MNTN for speaker i is then trained with this data. This procedure is repeated for all speakers in the population. A trained MNTN can be applied to speaker recognition as follows. First a sequence of feature vectors x are extracted from the test utterance. The confidence and labels of this sequence can then be scored to obtain a likelihood for a given speaker model [6].

HYBRID SYSTEM

Two hybrid systems are evaluated in this paper. The first system embeds the capabilities of the GMM within the NTN, thus creating a *continuous* density NTN. The second system combines the output probabilities of both classifiers as a weighted sum. This technique is known as *data fusion* [11]. The details of both implementations are provided in the following subsections.

Continuous Density NTN

The MNTN described in [6] uses a discrete model for the posterior probability estimate. This model can be further extended to use a continuous density estimate, thus yielding a continuous density NTN (CDNTN). Here, a GMM is used to model the probability density function within the local regions of feature space, namely the leaves. For a given feature vector x and leaf j, a GMM can approximate the posterior probability estimate $P(C_i|x)$ of class C_i as follows:

$$P_{CDNTN}(C_i|x) = \frac{P_j(C_i)P_j(x|C_i)}{\sum_{n=1}^{N} P_j(C_n)P_j(x|C_n)} \tag{3}$$

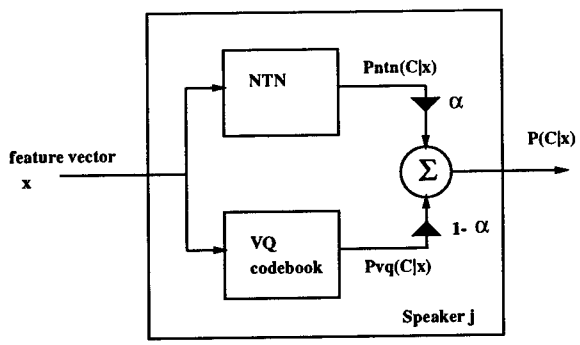


Figure 1: Data Fusion Approach

where

$$P_j(x|C_i) = \sum_{m=1}^{M} c_{im} N(x, \mu_{im}, \Sigma_{im}), \tag{4}$$

and

$$N(x, \mu, \Sigma) = \left((2\pi)^{-d/2}|\Sigma|^{-\frac{1}{2}}\right) exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]. \tag{5}$$

Here, c_{im}, μ_{im}, and Σ_{im} are the weight, mean, and variance, respectively, of the mth mixture for class i, N is two for a binary problem, and d is the dimension of the feature space. The CDNTN can be used for speaker recognition in the same manner as the MNTN. However, the CDNTN will obtain the posterior class probabilities from estimated density functions as opposed to being selected from a discrete set of probabilities.

NTN/VQ Data Fusion Model

The hybrid system considered in this paper is based on data fusion [11]. Data fusion has been used to combine inputs from multiple sensors in various applications including robotics [12] and handwriting recognition [13]. Numerous methods exist for combining the data of several classifiers. One method is to take a vote among the concurrent outputs of several classifiers and use the vote winner as the assigned label [13]. For probabilistic outputs, a weighted sum of the outputs of the different classifiers can be used, as illustrated in Figure 1. This is the method that will be used to fuse the two classifiers for this paper. Here, the outputs of the NTN and VQ classifiers are multiplied by α and $1-\alpha$, respectively, where α lies between zero and one. Hence, when $\alpha = 0$ the system consists of solely the VQ classifier and likewise when $\alpha = 1$, only the NTN is used.

To enable the VQ and NTN classifiers to be used in such a system, several normalization steps must be used. First, the VQ distortion must be converted

to a probability. The method used here is to simply use:

$$P_{vq}(C_i|x) = e^{-(x-c_j)^2}, \tag{6}$$

where c_j is the centroid closest to x. The NTN must then be modified such that it outputs a probability instead of a confidence and a label. The confidence, which lies between 0.5 and 1.0, along with the label can be converted to probability as follows:

$$P_{ntn}(C_i|x) = \begin{cases} 0.5 * (1.0 + confidence), & if\ label = 1 \\ 0.5 * (1.0 - confidence), & if\ label = 0 \end{cases}. \tag{7}$$

This system will be evaluated for text-independent speaker identification and verification in the following section.

EXPERIMENTAL RESULTS

The database considered for the speaker recognition experiments is a subset of the DARPA TIMIT database. This set consists of 100 male speakers taken from the second and third dialect regions. A VQ codebook and MNTN are trained for each of 50 speakers. The remaining 50 speakers are reserved as imposters for testing the speaker verification system.

The preprocessing of the TIMIT speech data consists of several steps. First, the speech is downsampled from 16KHz to 8 KHz sampling frequency. The downsampling is performed to obtain a toll quality signal. The speech data is processed by a silence removing algorithm followed by the application of a pre-emphasis filter H(z) $= 1 - 0.95z^{-1}$. A 30 ms Hamming window is applied to the speech every 10 ms. A twelfth order linear predictive (LP) analysis is performed for each speech frame from which twelve cepstral coefficients are derived.

There are 10 utterances for each speaker in the selected set. Five of the utterances are concatenated and used for training. The remaining five sentences are used individually for testing. The duration of the training data ranges from 7 to 13 seconds per speaker and the duration of each test utterance ranges from 0.7 to 3.2 seconds. It is noted that the duration of the test utterances is relatively short and that performance can be improved by increasing the duration.

Speaker Identification

The data fusion technique as applied to the NTN and VQ classifiers is evaluated for closed set speaker identification. Each VQ classifier consists of a seven bit codebook trained using the Linde-Buzo-Gray (LBG) [7] algorithm. It was found that the seven bit codebook performed better than the six or eight bit codebooks. Each NTN is trained with the data for all 50 speakers.

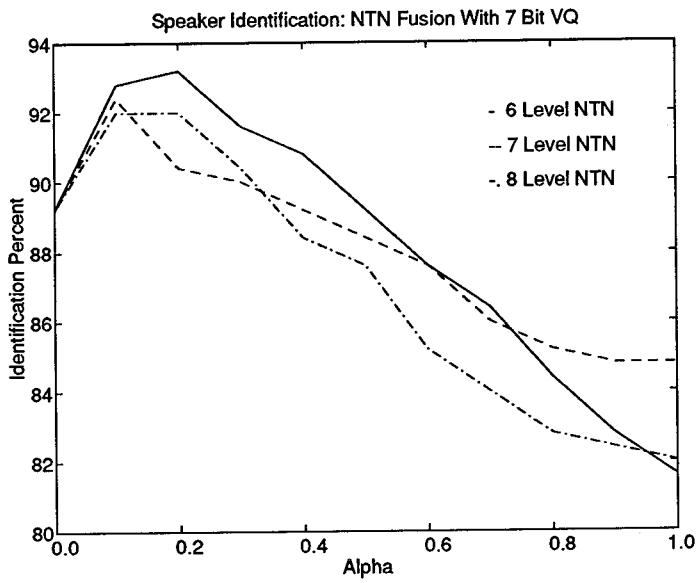


Figure 2: Speaker Identification Using Data Fusion

The maximum NTN level is varied from six to eight levels. The performance is evaluated as a function of α and is shown in Figure 2.

The individual performance of the VQ and NTN classifiers can be seen as those points corresponding to $\alpha = 0$ and $\alpha = 1$, respectively. Here, it can be seen that the optimal choice of α for the six level/six bit system yields a classifier that performs at roughly 93%, which is 4% better than either the VQ or NTN classifiers used individually.

The CDNTN is also evaluated for the identical closed set speaker identification experiment. Two sets of CDNTNs are evaluated, where the number of parameters are held constant for both. Hence, a CDNTN with 128 parameters can consist of either a seven level tree with zero mixtures/leaf, or a six level tree with one mixture/leaf (for each class), etc. The performance of the CDNTN is shown in Figure 3. Here, it is seen that the 128 and 256 parameter CDNTNs perform at roughly 85% and 86%, respectively.

Speaker Verification

The next experiments are performed for speaker verification. Here, the 50 speaker models correspond to the "enrolled" speakers and the 50 remaining speakers are used as imposters. The data fusion system is evaluated with the seven bit VQ codebook and the six and seven level NTN. The equal error rate, i.e., the point when $P(false\ accept) = P(false\ reject)$, for the data

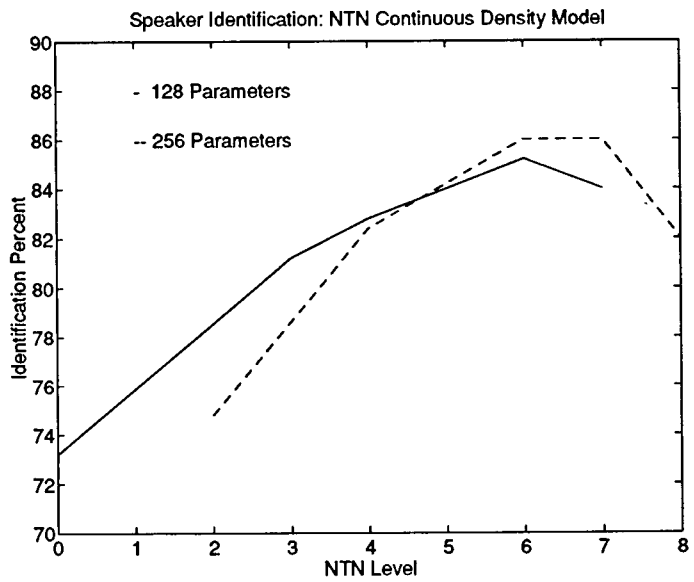


Figure 3: Speaker Identification Using CDNTN

fusion system is evaluated as a function of alpha and shown in Figure 4.

Here, it is seen that the system performance is similar for both systems and the optimal performance occurs when the system uses just the NTN. The CDNTN is also evaluated for the identical experiment for 128 and 256 parameters. The results are shown in Figure 5. Here, the best equal error rate is obtained with the 256 parameter model that uses a seven level NTN with one mixture/leaf. Note that the plot for the 128 parameter model only extends to the seventh level, since the eighth level would require 256 parameters.

CONCLUSION

Two methods are evaluated for combining classifiers based on distortion and discriminant measures. The first system embeds the capabilities of the GMM within the NTN, thus creating a *continuous* density NTN (CDNTN). The second system evaluates both classifiers independently and combines the output probabilities as a weighted sum. This technique is known as data fusion. Both methods are evaluated for several text-independent speaker recognition tasks, including closed set speaker identification and speaker verification. For closed set speaker identification, the data fusion technique provides the best performance, which is superior to that of either classifier used individually. For speaker verification, the CDNTN yields the best performance for a model that uses a NTN with one Gaussian mixture model per leaf.

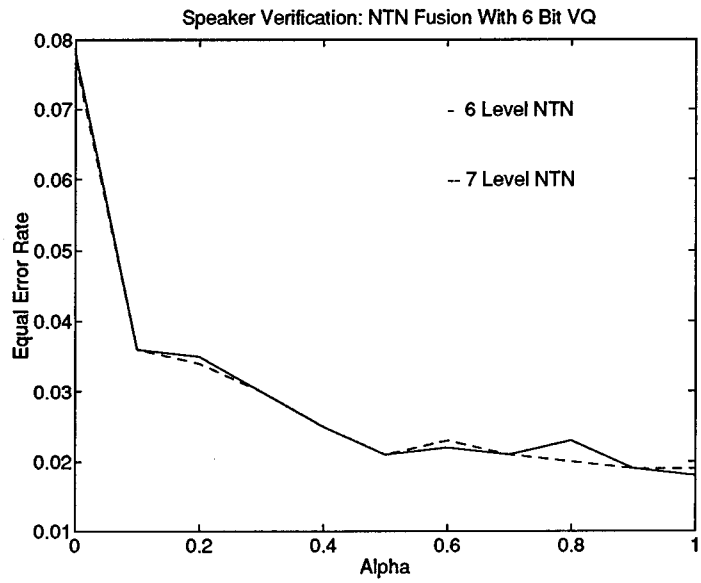


Figure 4: Speaker Verification Using Data Fusion

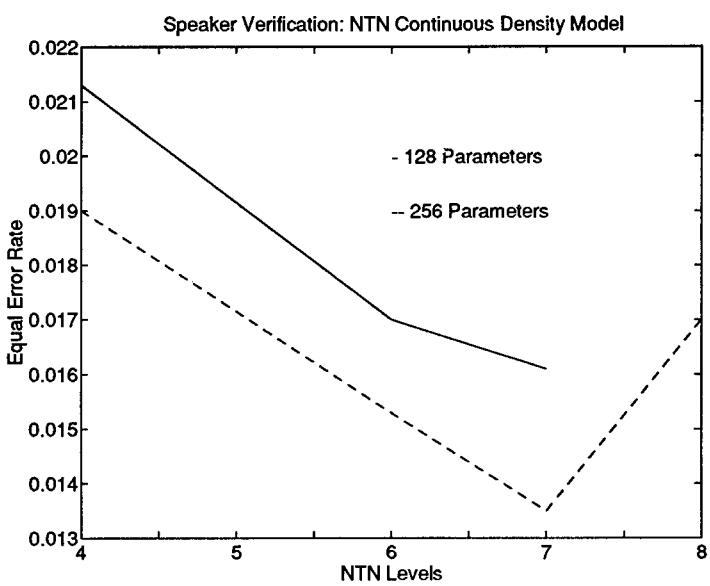


Figure 5: Speaker Verification Using CDNTN

References

[1] F.K. Soong, A.E. Rosenberg, L.R. Rabiner, and B.H. Juang. A vector quantization approach to speaker recognition. In *Proc. ICASSP*, pages 387–390, 1985.

[2] A.E. Rosenberg, C.H. Lee, and F.K. Soong. Sub-word unit talker verification using hidden Markov models. In *Proceedings ICASSP*, pages 269–272, 1990.

[3] D. Reynolds. *A Gaussian Mixture Modeling Approach to Text-Independent Speaker Identification.* PhD thesis, Georgia Institute of Technology, February 1993.

[4] J. Oglesby and J.S. Mason. Optimization of neural models for speaker identification. In *Proceedings ICASSP*, pages 261–264, 1990.

[5] J. Oglesby and J.S. Mason. Radial basis function networks for speaker recognition. In *Proceedings ICASSP*, pages 393–396, 1991.

[6] K.R. Farrell, R.J. Mammone, and K.T. Assaleh. Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on Speech and Audio Processing*, 2:194–205, January 1994.

[7] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28:84–95, 1981.

[8] A.E. Rosenberg and F.K. Soong. Evaluation of a vector quantization talker recognition system in text independent and text dependent modes. *Computer, Speech, and Language*, 22:143–157, 1987.

[9] A. Sankar and R.J. Mammone. Growing and pruning neural tree networks. *IEEE Transactions on Computers*, C-42:221–229, March 1993.

[10] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth international group, Belmont, CA, 1984.

[11] J.J. Clark and A.L. Yuille. *Data Fusion for Sensory Information Processing Systems.* Kluwer, Norwell, MA, 1990.

[12] Henderson and Weitz. Multisensor knowledge systems. *International Journal of Robotics Research*, 7(6):114–137, June 1988.

[13] L. Xu, A. Krzyzak, and C.Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, May 1992.

Parallel Training of MLP Probability Estimators for Speech Recognition: A Gender-Based Approach

Nikki Mirghafori, Nelson Morgan, and Hervé Bourlard
International Computer Science Institute, Berkeley, California
1947 Center St, Suite 600
Berkeley, CA. 94704
Tel: (510) 642-4274, FAX: (510) 643-7684
{nikki, morgan, bourlard}@icsi.berkeley.edu

Abstract

In this paper we explore the averaging of mixtures of multiple neural network probability estimators in speech recognition. We experiment with different ways of dividing up the speaker space. A division based on gender seems to be the most important. The division based on a priori knowledge (in our case, rate of speech) resulted in lower error rates than the use of k-means clustering. The overall accuracy of the Parallel Net architecture is about the same as the monolithic probability estimator, but communication costs on parallel machines can be expected to be significantly lower. Additionally, the overall product of patterns times parameters is lower with such a partitioning, resulting in reduced training time even on serial machines.

1 Introduction

In previous work, we have examined the factorization of Multi-layer Perceptrons (MLPs) that are viewed as probabilistic estimators. In two particular cases, we partitioned out the influence of phonemic context [Bourlard & Morgan, 1992], and of speaker gender [Konig & Morgan, 1993]. These partitionings permitted the evaluation of the posterior probabilities of a large number of classes without the explicit computation of a huge output layer.

In our current work, we are interested in partitioning not only the network estimators, but also the training data. This is of increasing relevance as we move to larger and larger data sets. Cycling through these data requires more than a linear increase in computation, as the estimators themselves should (ideally) be expanded to a greater number of parameters in order to take advantage of the increased coverage in the training materials.

Parallelism is a potential remedy for this increased computational burden, but depending on the machine and the algorithm, communications costs can overwhelm any advantage due to numerical parallelization. Training set parallelism is a potential cure for this difficulty. If multiple estimators are trained

0-7803-2026-3/94 $4.00 © 1994 IEEE

on disjoint elements in the training set, and then combined in some manner, communication is minimized. Additionally, there is some hope that the right partitioning and weighting of the separate estimates could provide some improvement in performance; for instance, in the gender case, separating male and female training data has some demonstrable advantages.

There is some evidence that data splitting should at least provide equivalent performance. In one report at a recent SRS meeting, R. Schwartz of BB&N described an experiment in which Hidden Markov Model (HMM) Gaussian mixture parameters were separately estimated for individual speakers and then averaged [Schwartz, 1993]. The resulting system was comparable in performance to a more standard estimator that was trained on the pooled data from all speakers. Of course, this experiment reported the estimation of data likelihoods and the averaging of Gaussian parameters, and this does not necessarily show that a posterior estimator like a Multi-layer Perceptron (MLP) will permit a similar parallelization. However, it suggests that a test is worthwhile. Recent results in applying the split net strategy in control theory [Jacobs & Jordan, 1993] are another indication that such approach may be advantageous.

Another related effort was that of the Meta-Pi network [Hampshire & Waibel, 1990]. In this approach, speaker-dependent estimators for voiced stop consonant probabilities were weighted and summed with gating elements trained with error back-propagation. For a source dependent speaker (i.e., one of the six training speakers), the performance of the Meta-Pi architecture on a six speaker three phone (/b,d,g/) task was comparable to a speaker dependent system. For a source independent (i.e., unknown) speaker, however, the error rate was almost tripled.

In the work described here, we also are using an MLP trained with back-propagation; however, these estimators are trained to be discriminant for the 61 phone set of TIMIT. We have focused our efforts on the speaker-independent case. In other words, we would like our mixture of estimators to perform at least as well as a monolithic estimator (which was trained on all of the data) when tested on an unknown speaker (which was not present in the training data).

2 Approach

In our experiments we use estimators that are based on the hybrid HMM/MLP method as explained in [Bourlard & Morgan, 1994]. The main idea in this method is to train an MLP (using a squared error or relative entropy criterion) for phonemic classification; such a net can be used as an estimator of posterior class probabilities, and when divided by class priors can estimate scaled likelihoods. The MLP estimator has the potential advantage (over standard Gaussian or Gaussian mixture estimators) of the ability to estimate

highly-dimensioned joint probabilities, such as the scaled likelihood of the data including a large acoustic context. Additionally, the MLP training is inherently discriminant, making effective use of parameters for limited training data, and estimating relatively detailed densities without strong parametric assumptions. Over the last few years, we have observed in a number of instances that the direct substitution of such an estimator for a tied-mixture module has resulted in significant improvements.

We are currently using a recognizer called Y0 (described in [Robinson et al., 1993]), which uses a single density per phone with repeated states for a simple durational model. The densities are trained with no explicit incorporation of phonemic context (e.g., triphones). Our current results on DARPA Resource Management (RM)[1] test sets show a performance that is comparable to that of much more complex context-dependent systems; the recognition word error on the February 1989 test set of the baseline hybrid HMM/MLP system, used here for comparison, was equal to 5.1% (including insertions, deletions, and substitutions).

In the current experiments, we train multiple networks on separate partitions of the training set. If M_i represents cluster i, let $P(M_i)$ be the probability that M_i is a better match than M_k, $\forall k \neq i$. If all M_i's are mutually exclusive, and cover all possible cases, $\sum_{i=1}^{n} P(M_i) = 1$, we can calculate the likelihoods (within a constant factor $P(x)$) by:

$$P(x|q_k) = \frac{\sum_{i=1}^{n} P(M_i|x)P(q_k|x, M_i)}{\sum_{i=1}^{n} P(M_i)P(q_k|M_i)} \tag{1}$$

where q_k is an HMM state, M_i represent each of the n MLPs, and $P(M_i|x)$ is the probability that M_i is the "correct" estimator of the sound class, given the data x. For instance, in the case of a male/female partition, this probability would be the probability that the speaker is male or female (2 probabilities that sum to 1). As an alternative, we calculate a weighted average of the scaled likelihoods:

$$P(x|q_k) = \sum_{i=1}^{n} P(M_i|x) \frac{P(q_k|x, M_i)}{P(q_k|M_i)} \tag{2}$$

In some of our experiments, we have used a simplified form of the above formula and inserted an equal weight averaging factor of $1/n$ instead of $P(M_i|x)$. This amounts to the following two assumptions: that $P(M_i|x)$ is independent of the data, and that all priors of M_i are equal (i.e., male prior equal to female prior).

Mixture of experts approaches are most effective when each expert has different statistical properties and biases. Therefore, each of our sub-systems

[1] This is a speaker-independent continuous speech recognition task that has a vocabulary of roughly 1000 words and uses a word-pair grammar with a perplexity 60; it is described in many places in the speech literature, including [Price et al, 1988].

(nets) should be trained on subsets of data with different statistical properties. In all the experiments reported here, we use a speaker-dependent split in the training data.

3 Pilot Experiments

3.1 Using Pretrained Nets

In our first pilot experiment, we used twelve (five female and seven male systems) *pretrained* speaker dependent (SD) estimators, each of which were trained on data from one speaker of the RM SD November 1989. By *pretrained* we mean that the nets were previously trained to maximize the accuracy of SD recognition. Each net had 1000 hidden units, 61 outputs, 234 input units (= 26 PLP and delta PLP features[2] × 9 frame window size), and trained with phonetic labels which had gone through two iterations of forced viterbi realignment. 500 of the SD sentences were used for training, and 100 were held out for cross-validation. The nets were trained starting with a learning rate of 0.008, and the training took 4-6 epochs. The word recognition error rate of each system on the same speaker's test data (RM January 1990 SD evaluation — 25 sentences per speaker) ranges from 1.8% to 11.3%[3], while the error rate on the RM February 1989 SI evaluation test set (300 sentences, 10 speakers, 4 of which are female and 6 male) ranges from 64.6% to 82.0%.

We averaged (equal weighting) the scaled likelihoods of each of the SD subsystems using equation (2) and got 22.6% word recognition error, which is better than the performance of *both* the best SD sub-system *and* the average of all the SD sub-systems. However, this error rate is not comparable to that of a monolithic SI system (a net trained on RM November 1989 SI data), which has an error rate of about 5.1% for the same SI recognition task.

Upon analyzing the results, we came across striking gender effects, as shown in Table 1. Sub-systems trained on male speech generalized better to male speech than to female speech; vice versa for female nets. This provided motivation for another experiment: if the test speaker's gender is female (male), we only allowed the probabilities generated by female (male) systems to take part in calculating the average scale likelihood. Since this was a pilot experiment, the gender of the test speakers were *known* to the system. It is possible, however, to build a gender detector which reliably (approx. 98% accuracy) detects the gender of the test speaker [Konig & Morgan, 1993]. Table 1 shows the strength of this effect. Averaging in a gender-based way further decreases the overall word recognition error to 16%.

[2]PLP stands for Perceptual Linear Predictive analysis [Hermansky, 1990]

[3]It should also be noted that these *pretrained* nets were trained approximately two years ago so that the raw error numbers are probably somewhat higher than our current systems would achieve.

Gender Effects on Percent Word Error		
Training Speaker	Test Speaker	
	Male	Female
Male	45.4	111.7
Female	106.4	38.1

Table 1: Gender Effects on Word Error. This table shows the average error rate of SD Female (Male) nets when tested on SI Female (Male) data. Error rates of higher than 100% are due to counting insertions, deletions, and substitutions as errors.

There was an interesting unexpected result: the SD system with the worst recognition score on its own data generalized best to the speech of unknown speakers. On the other hand, a system which was almost perfectly tuned to speech from the same speaker generalized the worst. While not all the systems obeyed this rule, it was a general trend. This suggested that we should use SD nets that are not as fully tuned to the same speaker's data.

3.2 Retraining the Experts

In the next group of pilot experiments we examined the effect of using speaker-independent cross-validation to avoid overfitting to the speaker-dependent training data. We also reduced the number of parameters in contrast to the first experiment (again to combat over-fitting). We changed the size of the hidden layer from 1000 hidden units to 256 hidden units for each net. In order to reduce the training time, we bootstrapped each of the nets from a 256 hidden units net that was previously trained on the hand-labeled SI TIMIT database. Our training data for each net was 600 SD sentences. Same-gender SI data for cross-validation was chosen from the RM November 1989 SI training set: 460 sentences with 23 speakers for the female set, and 490 sentences with 49 speakers for the male set. We used a lower learning rate ($\alpha = 0.004$) than for the *pretrained* nets. Each net went through only 1-2 epochs of training before cross-validation performance indicated that training should be stopped.

To estimate $P(M_i|x)$, we trained a *gating* network [Hampshire & Waibel, 1990]. We used a net with 10 hidden units, 234 input units, and n output units (where n is the number of nets). It was trained with back-prop to associate each feature vector with the label of the training speaker. In each of the experiments below, we have run Viterbi decoding on the output of each SD net and reported the results.

Based on the strong gender dependencies that we observed, we chose one gender (the female set) for the following experiments. The female set comprises five female SD systems (RM SD training data), and four female unknown test

speakers (from RM February 1989 SI evaluation set, as mentioned above).

For a fair comparison between the parallel architecture and a single-net system, we trained *one* net on the aggregate training data of the five SD systems, and cross-validated the training using female SI data (as explained above). We chose a 1000 hidden unit net that has about an equivalent number of parameters as the five female nets altogether. We bootstrapped this net from a TIMIT net in order to reduce the total training time. The initial learning rate was 0.008.

A net that was trained on the aggregate data of the SD nets, the female SI net, has an error rate of 7.4%. All of our experimental nets performed worse than this, but given a small data set of only 5 speakers, the results were not considered conclusive. However, the average error rate for the Parallel Nets using eqn (2) with an equal weight for each SD system is fairly close to the female SI net, with the Parallel Net having 13.5% more relative word error than female SI (8.4% versus 7.4%). This is not a statistically significant difference at the $p < .05$ level for this test set, so that in some sense there is no demonstrable difference in performance. The average performance of the Parallel Net architecture is better than *both* the average error of the SD systems' (13.0%) *and* the best SD system (9.7%) (significantly so for the average case).

Comparing the performance of our Female SI net with our baseline hybrid HMM/MLP system, we observe that Female SI has about 40% more relative error (7.4% versus 5.3%, which is significant at the $p < .05$ level) than the gender-independent SI net. This is unsurprising, since the baseline SI net is trained on over 30 speakers and is trained longer. The Female SI net, in contrast, is only trained on five speakers and goes through half as many epochs of training. The obvious remedy would be to train the Female SI net on more female speakers. In other words, train more SD systems to get a better representation of the sample space. Another possibility is to train each SD net on two or more same-gender speakers that are in the same region of the sample space, creating *quasi*-SD nets and increasing the coverage of the sample space that way. This conjecture was the basis for the main set of experiments.

4 Experiments and Results

In order to get a better representation of the speaker space, we increased the number of training speakers in the next experiment. We used the male speakers' data from the RM SI training set (November 1989), consisting of 49 male speakers, each uttering 40 sentences. Since there is little training data for each speaker, training 49 SD nets was not feasible. Instead, we can divide the speaker space based on some criterion and train one net on each

section of the speaker space. We experimented with two splitting criteria: rate-of-speech, and k-means clustering.

4.1 Splitting the Speakers

First, we used a priori knowledge about the domain and allocated the speakers to groups based on their rate-of-speech, where (inverse) rate-of-speech is measured as average number of seconds per word. In the second method, we use the k-means clustering [e.g., Krishnaiah & Kanal, 1982] algorithm.

4.1.1 Dividing the Space Based on Rate-of-Speech

Two observations motivated us to experiment with this split of the data. In the most recent ARPA WSJ evaluation, researchers reported significantly higher error rates on two fast speakers in the evaluation test set. The second motivation comes from our earlier results (section 3.2). We analyzed the relationship between the rate-of-speech of the female test speakers & the SD system's training data and word error rate. In order to have sufficient training data for each net, we chose to experiment with two and four clusters.

4.1.2 K-means Clustering

For the k-means clustering algorithm, we use a distance measure explained below. Let $X = \{X_1, X_2, ..., X_n, ..., X_N\}$ be the feature vector sequence corresponding to the speech of speaker S_x, where each X_n is a vector, $X_n = (x_{n1}, x_{n2}, ..., x_{nd}, ..., x_{nD})^t$. For each speaker S_k, we calculate a mean vector $\mu_k^j = (\mu_{k1}^j, ..., \mu_{kd}^j, ..., \mu_{kD}^j,)^t$ and a covariance vector $\sigma_k^j = (\sigma_{k1}^j, ..., \sigma_{kd}^j, ..., \sigma_{kD}^j)^t$[4] for each broad phonetic category $j = \{1...J\}$[5]. Define the distance between speaker S_x and speakers S_k as:

$$D(S_x, S_k) = \frac{1}{N} \sum_{n=1}^{N} \min_j \sum_{d=1}^{D} \left[\log \sigma_{kd}^j + \left(\frac{x_{nd} - \mu_{kd}^j}{\sigma_{kd}^j} \right)^2 \right] \tag{3}$$

So, for calculating the distance between two speakers, we use the μ's and σ's of one speaker, and the feature vectors of the other. Except for the distance measure, we follow the standard k-means clustering algorithm.

We can replace the gating network by using this distance measure. In order to determine the weights to use for each of the scaled likelihoods, we measure the distance of the unknown test speaker to the cluster centers and use an estimated probability (computed assuming a Gaussian distribution with a diagonal covariance matrix), the normalized $e^{-distance}$, as weight.

[4] Covariance matrix assumed to be diagonal.

[5] The five broad phonetic categories are based on the phonetic classes in the TIMIT set; they are fricatives, liquids, nasals, stops, and vowels.

Word Recog Percent Error – Male Set				
System	Rate-of-Speech		K-means	
	2 CL	4 CL	2 CL	4 CL
PN, Eqn (2), eql wgts	8.1	7.0	8.8	8.0
PN, Eqn (2), gating, +smth	7.7	8.1	8.6	8.1
PN, Eqn (1), eql wgt	11.0	11.1	13.5	11.5
Best Net	7.6	7.7	8.2	9.8
Avg of Nets	9.4	9.1	10.6	11.9

Table 2: Word Recognition Percent Error for each of the systems tested on RM February 1991 SI male evaluation data. The "Best Net" column represents the error rate of the best single net. The "Avg of Nets" is the average of word recognition error of the nets.

4.2 Results

We used the male RM SI data for training, as mentioned above. Each of the nets in the two-cluster experiments were 512 hidden units each, and in the four-cluster experiments were 256 hidden units, making the number of parameters to be the same across all systems. Each net was trained on a partition of the training data with error back-propagation, started with a learning rate of 0.008, and was cross-validated on male data from February 1989 RM SI evaluation data to determine the stopping point for the training. The same data was also used for development purposes, for example setting the word transition penalty.

In order to perform a fair comparison between the Parallel Net architecture and monolithic net, we trained a 1000 hidden units net on all the male RM SI data. We tested all the systems on the male speakers of the February 1991 RM SI evaluation data. The word recognition error rate of our standard baseline system (which is trained on all genders of RM SI training data) was 8.9% for the males only. In comparison, the error rate of the monolithic all-male system was 7.3%, which is significantly better at $p < .05$ level.

The results of the Parallel Net architecture, presented in Table 2 are similar to that of the all-male monolithic net for the four-cluster cases, and the difference in error rates are not significant (7.0% and 8.0% for the four-clusters versus 7.3% for the male monolithic net). Weighted averaging gives worse results to the equal weighted averaging approach if the weights of the gating network are used directly. By introducing speaker continuity constraint and averaging the weights over a sentence (+smth column), the results of the weighting scheme improves and approaches that of the equal weighting one. The total training time for the monolithic net on our special purpose hardware [Morgan et al, 1992] is 18 hours. However, the total training time for the four-cluster case is 7.5 hours for the rate-of-speech nets and 9 hours for the k-means nets (an average of two hours per net). The total training time for the two-cluster case

the two-cluster case is 18 hours for the rate-of-speech nets, and 11.5 for the k-means nets (average of 7.5 hours per net).

5 Discussion

In this paper, we have proposed a Parallel Net architecture for reducing the training time of the hybrid HMM/MLP system. Each of the experts in the system are trained on one region of the speaker space, hence making each net a *quasi*-speaker-dependent probability estimator. In our initial pilot experiments, we observed a strong gender effect. Also, there was strong evidence of over-tuning to the same category data. These two observations motivated us to restructure our experiments to reduce over-fitting, and to factor in gender effects.

We retrained the experts using same-gender SI cross-validation to avoid over-tuning. Also to further reduce over-fitting to the SD data, we cut the number of parameters by a factor of four and used a smaller learning rate. We experimented with different averaging schemes: weighted vs. equal, and average of scaled-likelihoods vs. sum of posteriors divided by sum of priors. The theory [see also Jacobs & Jordan, 1993] suggest that a non-uniform weighting mechanism is desirable. However, in our experiments, the weighted average was similar, if not worse, than an equal weighted average. This may only mean that we did not develop the correct method for determining the best weights in these examples; but in any event the evidence we have so far does not support computed weights, and equal weights in any event seem to work well enough to support a parallel approach. Also, we consistently got better results from averaging scaled likelihoods (equation 2 vs. 1).

The average error rate of the Parallel Net architecture was better than *both* the best SD system *and* the average error rate of all the female-SD systems, and the four-cluster male systems. Furthermore, the performance of the Parallel Nets was comparable to a single net trained on the aggregate training data. Given the shorter training time and the potential for taking advantage of parallel architectures, we believe that the Parallel Net architecture is the preferable architecture.

6 Acknowledgments

We would like to thank Eric Fosler, Yochai Konig, and Gary Tajchman for their help and advice. We also thank the National Science Foundation for its support through grant MIP-9311980, as well as general support from ICSI.

References

[1] Bourlard, H., & Morgan, N. (1992). "CDNN: A Context Dependent Neural Network for Continuous Speech Recognition", *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. II.349-352, San Francisco, CA.

[2] Bourlard, H., & Morgan, N. (1994). *Connectionist Speech Recognition,* Kluwer Academic Press

[3] Hampshire, J.B., & Waibel, A. (1990). "Connectionist Architectures for Multi-Speaker Phoneme Recognition", *Advances in Neural Information Processing Systems 2*, D. Touretzky (ed.), Morgan Kaufmann, CA.

[4] Hermansky, H. (1990). "Perceptual Linear Predictive (PLP) Analysis of Speech", *Journal of the Acoustical Society of America,* 87:1738-1752.

[5] Jacobs, R., and Jordan M. (1993). "Linear Piecewize Control Strategies in a Modular Neural Network Architecture", *IEEE Trans. on Systems, Man, and Cybernetics,* March/April 1993, vol. 23, nr. 2, pp. 337-345.

[6] Konig, Y., and Morgan, N. (1993). "Supervised and Unsupervised Clustering of the Speaker Space For Connectionist Speech Recognition" *Proc. IEEE Int. Conf. Acoustics, Speech & Signal Processing,*

[7] Krishnaiah, P.R., and Kanal, L.N., eds. (1982). *Classification, Pattern Recognition, and Reduction of Dimensionality.* Handbook of Statistics, vol. 2. Amsterdam: North Holland.

[8] Morgan, N,. Beck, J., Kohn, P., Bilmes, J., Allman, E., and Beer, J., "The Ring Array Processor (RAP): a multiprocessing peripheral for connectionist applications," *Journal of Parallel and Distributed Computing,* Special Issue on Neural Networks, vol. 14, pp.248-259, 1992

[9] Price, P., Fisher, W., Bernstein, J., and Pallett, D. (1988). "A Database for Continuous Speech Recognition in a 1000-Word Domain", *Proc. IEEE Int. Conf. Acoustics, Speech & Signal Processing,* pp. 651.

[10] Robinson, T., Almeida, L., Boite, J.M., Bourlard, H., Fallside, F., Hochberg, M., Kershaw, D., Kohn, P., Konig, Y., Morgan, N., Neto, J.P., Renals, S., Saerens, M., & Wooters, C. (1993). "A Neural Network Based, Speaker Independent, Large Vocabulary, Continuous Speech Recognition System: The WERNICKE Project", Proceedings of *EUROSPEECH'93*, September 21-23, Berlin, Germany.

[11] Schwartz, R. (1993). Oral Presentation, *Speech Research Symposium XIII*, Johns Hopkins

LVQ as a feature transformation for HMMs

Kari Torkkola
IDIAP (Institut Dalle Molle D'Intelligence Artificielle Perceptive)
C.P. 609, CH-1920 Martigny, SWITZERLAND
email: karit@idiap.ch

Abstract. We present a new way to take advantage of the discriminative power of Learning Vector Quantization in combination with continuous density hidden Markov models. This is based on viewing LVQ as a non-linear feature transformation. *Class-wise quantization errors* of LVQ are modeled by *continuous density HMMs*, whereas the practice in the literature regarding LVQ/HMM hybrids is to use LVQ-codebooks as frame labelers and discrete observation HMMs to model a stream of such labels. As decision making at frame level is suboptimal for speech recognition, the presented method is able to preserve more information for the HMM stage. Experiments in both speaker dependent and speaker independent phoneme spotting tasks suggest that significant improvements are attainable over plain continuous density HMMs, or over the hybrid of LVQ and discrete HMMs.

1 Introduction

Hidden Markov models (HMM) are among the most succesfull techniques in automatic speech recognition with well studied and mature training algorithms [14]. These techniques can be roughly divided into two main categories: continuous observation density HMMs (CHMM) and discrete observation HMMs (dHMM) with semi-continuous (tied mixture) HMMs somewhere in between. Either continuous or discrete, the aim of the models is faithful *representation* of the feature vector sequence derived from the speech signal, either directly by mixtures of multivariate Gaussian or other distributions, or through vector quantization (VQ). Both the maximum likelihood training algorithms of the HMMs, and in the discrete case, also the codebook construction algorithms aim at this.

In recent years many ways have been presented to enhance the discrimination capabilities of HMMs. These include, among others, new training criteria [18, 2, 7] and hybrids of discriminative methods with the HMMs. The latter type of systems have been recently dominated by artificial neural networks (ANN). These hybrids can further be grouped into two main clusters: ANNs as probability estimators for HMMs [1, 3], or ANNs as codebooks or labelers [6, 16, 12, 11] for dHMMs.

The present paper is concerned with a novel combination falling into the

0-7803-2026-3/94 $4.00 © 1994 IEEE

latter category. The combination is based on *Learning Vector Quantization (LVQ)* [9, 10]. In the literature, many such LVQ/HMM-hybrids have been presented [5, 6, 8, 12, 16, 20], in which LVQ acts as a phonemic discriminative labeler. The resulting label stream is then modeled by dHMMs. This paper extends the previous work by presenting a way of extracting more information of the LVQ stage followed by *continuous density* HMMs. This is founded on viewing LVQ as a feature transformation.

The structure of the rest of this article is as follows. Section 2 gives an overview of LVQ and reviews its prevalent use together with HMMs. In Sec. 3 we present a way of preserving more information of the LVQ stage, and suggest CHMMs as the tool process that information. In Sec. 4 we describe our experiments in two phoneme spotting tasks, and Sec. 5 is devoted to discussion.

2 LVQ-codebooks in speech recognition

The role of conventional vector quantization algorithms in speech recognition, such as the Linde-Buzo-Gray algorithm, or K-means, is to *represent* speech feature vectors with the smallest possible distortion. This is not the case with the Learning Vector Quantization (LVQ) methods, which try to aim at *discrimination* of pattern classes, whatever they may be [9, 10]. Codebook vectors directly define the class borders in the feature space according to nearest-neighbor rule. LVQ modifies the codebook vectors adaptively so that the borders between classes will approximate Bayes' decision surfaces. Quantization error (distortion) is of secondary interest.

In literature, the customary practice of using LVQ is as a substitute to conventional vector quantization. Short time feature vectors act as the basis for classification, most commonly to phoneme classes. HMMs are then employed to combine local classification decisions by treating them just as VQ-codebook indices are treated with dHMMs. In phoneme-related tasks, training has been done either by single frames [8], or by concatenating several frames together to represent some context [20, 6, 16, 5]. Further, the whole (phoneme) token has been used in training. All of the frames or successive shifts of the concatenated frames have acted as examples. However, this might pose a problem as short-time segments of the speech signal close to transitions between phonemes might be extremely confusing and might resemble very much other parts of other phonemes.

When the task has been speaker dependent and phoneme oriented, significant improvements have been observed due to LVQ when compared against ordinary VQ [6, 16]. Where the task has been word or phrase recognition, the results have been slightly controversial: no improvement

in [20], but significant improvements in [5].

In a recent study it was shown that by taking the local context into account (as much as 220 ms) in the construction of the pattern vectors, performance can be significantly increased compared to using a single frame or a few concatenated frames [12]. Codebooks order of magnitude larger than conventionally used are needed to represent variations in these high dimensional "context vectors". Another LVQ-codebook, whose purpose is to discriminate phoneme centers from transitions, was introduced in parallel with the main codebook.

In addition to enhanced discriminative properties, another advantage of the LVQ over conventional VQ is that the discrete alphabet in phonetically motivated classification, i.e., the number of classes, is smaller than the number of codebook vectors in usual VQ (few tens as opposed to few hundreds). This results in an oder of magnitude smaller amount of output probability parameters to be estimated for the dHMMs. Complicated smoothing schemes are thus usually unnecessary.

3 Class-wise quantization errors

3.1 Extracting more information from the LVQ

All previous work on LVQ-codebooks in speech recognition has concentrated on using the *class label of the closest codebook vector* only. To decode this label stream, discrete observation HMMs have been employed.

However, the normal practice with pattern classifiers, extracting only the final decision of the classification (the class label) is desirable only when that really is the final decision stage of the whole task. This is not the case with the LVQ/HMM hybrids in speech recognition; the final decision is made by the Viterbi search at the HMM-stage. It is suboptimal to resort to too early hard decisions at the frame level.

Remaining with dHMMs, an obvious improvement could be to preserve information about several of the closest classes, possibly weighted in some manner. Some kind of heuristics is required to convert the distance of the current feature vector to the closest classes into usable weights or probabilities. Techniques of fuzzy VQ might be helpful for this stage. Certainly more information would be acquired from the same amount of training data. Alternatively, less training data would result in the same performance, as at each time step, not only the output probability corresponding to the best label, but many or all of them will be gaining information.

However, the idea that we pursue here, is to make use of the actual measure that is the basis of LVQ-classification, the Euclidean distances to the closest codebook vectors of each class. Although these distances could be transformed into weights or probabilities, LVQ is not really

designed keeping probability estimation in mind.

A step in this direction was already taken in [12]. In addition to the classification label stream provided by LVQ, also the VQ error was exploited. VQ error refers now to the conventionally employed scalar quantity: distance to the closest codebook vector, regardless of its label. In contrast to other related work cited in Sec. 2, LVQ was trained by context windows positioned at phoneme centers only. Since the input representation was constructed by taking into account a relatively long duration of context, phoneme borders appear very different from the centers resulting in high quantization errors at transitions between phonemes. The quantization error can thus both give indication of the positions of the phonemes, and of their closeness to the corresponding codebook vector. Multiple input stream HMMs were used to combine the two sources of information, one discrete, another continuous valued.

In this paper, we extend the above work by discarding the label of the closest class. Instead, the distances to codebook vectors of different classes (class-wise quantization errors), can be used directly without resorting to any transformations to weights or interpretations as probabilities. *Continuous observation density* HMMs can then be applied to model a stream of these vectors. Fig. 1 illustrates the computation of the new feature vectors.

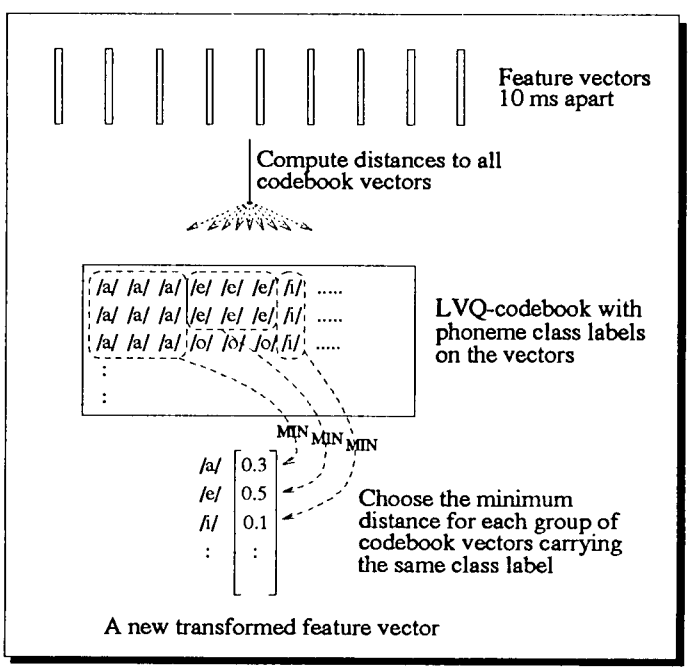


Figure 1: An illustration of the feature transformation. The stream of resulting new feature vectors is then modeled by HMMs.

3.2 LVQ as a feature transformation

In pattern recognition one is always concerned with choosing or computing such a representation of the input data, that enhances class separability while keeping within-class variability as small as possible. Several linear transformations, that are optimal according to criteria usually connected to between-class and within-class scatter matrices can be found in pattern recognition textbooks, and they have been applied succesfully to speech recognition [4].

In the suggested scheme, LVQ can be viewed as a nonlinear, discriminative feature transformation before the CHMM-stage. A simpler version of such a transformation has been presented for example in [15, pp. 85-87], where a new feature vector is formed by computing distances to fixed points in the feature space. These points represent the means of pattern classes, while in case of LVQ, there are several such points per class, and their locations been chosen so as to minimize classification errors through LVQ-training. Other kinds of non-linear feature transformations that are derived on the basis of training by examples have also been used in speech recognition [17].

If the intent is to make a classification decision based on a single feature vector it does not, of course, make any sense to regard LVQ as a feature transformation. LVQ can then just simply do the decision. However, when the feature vectors are parts of a larger entity and when a decision has to be made taking all parts into account, as in speech recognition, this is a useful point of view.

3.3 Relation to semi-continuous HMMs

The presented scheme might also resemble superficially semi-continuous HMMs (SCHMM), but that is not the case. SCHMMs also utilize a "codebook", but it is a codebook of (Gaussian) distributions without any concept of classes. SCHMMs try to *represent* the current feature vector by making use of *a number of the closest codebook entries*, while the presented hybrid uses *for every class the codebook entry closest to the current feature vector.* It would not make sense to take into consideration the second or third closest codebook entries of a class even though they would be closer than the closest entries of other classes, since LVQ is trained in view that only the closest one matters.

4 Experiments

4.1 Speaker dependent phoneme spotting

We will now describe experiments to compare the performance of the suggested method with some established algorithms. A comparison between four architectures is presented in Table 1. The architectures are:

1. conventional CHMMs modeling a stream of cepstral vectors,
2. LVQ/dHMM-hybrids, where LVQ produces a stream of class labels, which is modeled by discrete observation HMMs, and
3. the new approach proposed in this paper, where LVQ produces a stream of class-wise quantization error vectors, which is modeled by CHMMs, and
4. parallel use of the class-wise quantization error with the cepstral vectors.

combi-nation	LVQ Input	HMM Input	stre-ams	mixts/ stream	covar	error rate
1	-	MFCC+Δ	2	5,3	diag	3.2
	-	MFCC+Δ	2	2,1	full	2.9
2	MFCC	LVQ-best label	1	-	-	8.5
	context	LVQ-best label	1	-	-	4.6
3	context	LVQ-qerr+Δ	2	7,7	diag	5.4
	context	LVQ-qerr+Δ	2	2,1	full	2.5
	MFCC	LVQ-qerr+Δ	2	2,1	full	4.9
4	context	MFCC+Δ+LVQ-qerr+Δ	4	2,1,2,1	full	1.8

Table 1: Comparison between CHMMs (1), LVQ/dHMM-hybrids (2), and LVQ/CHMM-hybrids (3, 4) in a speaker-dependent task. "MFCC" refers to 20 component mel-scale cepstral vectors. Δ refers to difference coefficients. "context" denotes the 220 ms context vector described in [12]. "qerr" refers to class-wise quantization errors computed from the LVQ-stage. In all cases the HMMs had 3 emitting states.

The task in this comparison is *speaker dependent phoneme spotting* in the Finnish language [12, 16]. The database contains four repetitions of a set of 311 utterances spoken by three male Finnish speakers. Each set consists of 1737 phonemes. In the original Finnish language, there are only 21 different phoneme classes: 8 vowels and 13 consonants. Four additional phonemes have been adopted with loan words from other languages, but none of those were represented in the database. There were thus 22 phonemic classes for the LVQ to differentiate (21 phonemes and silence), which was also the dimensionality of the class-wise quantization error vectors. Three of the repetitions were used each time for training, and the remaining one for testing. Four independent runs were made for each speaker by leaving one set at a time for testing. Thus all speaker dependent recognition results presented in this paper are averages of 12 test runs, and based on 20844 phoneme spotting scores.

Speech analysis conditions were the following: 12.8 kHz sampling rate, pre-emphasis coefficient 0.95, 25.6 ms Hamming window every 10 ms, and 20-component mel-scale cepstral coefficients (MFCC) computed for every window. Where difference coefficients (Δ) have been used, they were computed from a period of 40 ms. This applies also to the quantization error vectors.

For context vectors as LVQ input, the codebook size was 2000, and for the MFCC, it was 500. The context vectors had a time span of 220

ms [12]. LVQ-training procedure was exactly the same as described in [12]: using only phoneme centers, initialization by K-nearest-neighbor, OLVQ1 for 10000 iterations with $\alpha(0) = 0.3$, and LVQ1 using $\alpha(0) = 0.2$ In the LVQ1 stage, the number of iterations was 100 times the number of codebook vectors. An example of the output of the LVQ stage is presented in Fig. 2. Phoneme center locations can be clearly distinguished.

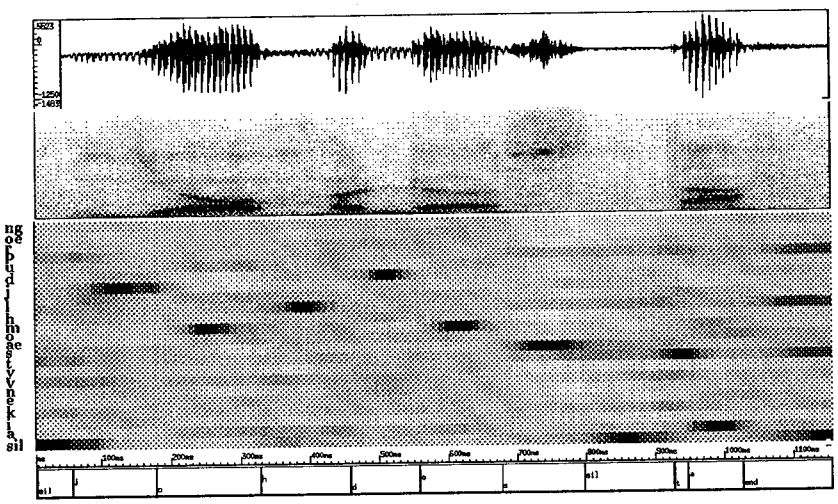


Figure 2: An example of the output of the LVQ-stage. From top to bottom: the speech signal (a Finnsh word */johdosta/*), its spectrogram, class-wise quantization errors with the phoneme labels on the left (darker shade denotes small values), and the labeling of the utterance.

Looking at the results in Table 1, it is obvious that exploiting all class-wise quantization errors instead of using only the class identity of the closest codevector preserves more information for the latter decision stages. This is reflected as the difference between LVQ/dHMM-hybrid and LVQ/CHMM-hybrid. We can also see that the class-wise quantization errors are interdependent, since modeling them by diagonal covariances produces poorer results. We got our best results by using full covariance matrices, and two Gaussian mixtures.

Though the LVQ/CHMM-hybrid is better than plain CHMMs, the difference is barely significant. On the other hand, the LVQ+MFCC/CHMM-hybrid beats plain CHMMs by a clear margin. Due to a relatively large number of tests, the confidence limits (99%) are relatively tight: $\pm 0.23\%$ for the best result (1.8%). This enables us to state that the proposed architecture is significantly better than a LVQ/dHMM-hybrid (4.6%), or a CHMM system (2.9%) in this task. In the case of plain CHMMs, using second order derivatives, increasing the number of mixtures, or using diagonal covariance matrices with larger number of mixtures did not improve our best CHMM result.

In the LVQ/dHMM-hybrid we used only the label sequence produced by LVQ; not any other information, as suggested in [12]. Including another LVQ-codebook for phoneme center/transition classification, or the whole codebook quantization error, would no doubt improve the performance of the LVQ/dHMM-hybrid, as it did in [12].

4.2 Speaker independent phoneme spotting

In this experiment, our aim was to find out whether this LVQ/HMM hybrid is applicable to the speaker-independent case. The database consisted of Swiss-French telephone speech with 56 speakers (about 2 hrs of speech). Half of the speakers were used for training and the other half for testing. The vocabulary also varied across the speakers.

Speech was sampled at 8 kHz, and 12-component mel-scale cepstra were computed each 10 ms. As the input to LVQ, we used slightly narrower context windows whose duration was 140 ms. The LVQ codebook size remained as 2000. 36 context-independent HMM phoneme models were used with 4 emitting states and full covariances throughout this experiment.

combination	HMM input	streams	mixts/stream	correct	accuracy
1	MFCC+Δ	2	2,1	60.9	53.4
3	LVQ-qerr+Δ	2	2,1	61.4	55.8
	LVQ-qerr+Δ+ΔΔ	3	1,1,1	63.5	57.4
4	MFCC+Δ+LVQ-qerr+Δ	4	2,1,2,1	63.9	58.0
	MFCC+Δ+LVQ-qerr+Δ+ΔΔ	5	1,1,1,1,1	65.3	59.4
	MFCC+Δ+LVQ-qerr+Δ+ΔΔ	5	3,1,1,3,1	65.7	60.0
	MFCC+Δ+(LVQ-qerr)+Δ+ΔΔ	4	3,1,(1),3,1	67.7	61.7

Table 2: Comparison between CHMMs (1) and LVQ/CHMM-hybrids (3, 4) in a speaker independent task.

The gain of using LVQ in this (much harder) task is not as dramatic as in the first one, but comparing the baseline CHMM recognizer (the first row) to the result on the last row we can see that the improvement is anyway very significant. In addition to Δ-coefficients we also tried 2nd oder difference coefficients for the quantization error stream, which turned out to be advantageous. In addition, it seems that the actual quantization errors are less important than their difference and 2nd order difference coefficients. In the result of the last row of the table we dropped that altogether out but kept the Δ- and ΔΔ-streams, and the results improved.

It must be noted that throughout these comparisons, exactly the same training conditions and algorithms (embedded Baum-Welch training) have been used. The basic phoneme model structure has also been the same throughout each experiment. The comparison is thus actu-

ally made between different input representations, and not, for example, between different HMM-software packages. In all of the experiments we used a public domain software package LVQ_PAK [10], and a commercial package HTK [19] for the HMMs. A discrete observation version of the HTK was written for dHMM-experiments.

5 Conclusion

We have reviewed ways of employing LVQ-based codebooks with HMMs in speech recognition. We wanted to point out that the prevalent practice in the literature, using LVQ as a frame labeler, and modeling the label stream with dHMMs, unnecessarily makes too early hard decisions. We have demonstrated that modeling class-wise quantization errors by CHMMs leads to significantly better results, and can even be better than the mainstream HMM-techniques. Using VQ-error as an indication of phoneme locations and its use as an input stream to HMMs was introduced in [12], to which this work is a direct extension. Other ways to employ VQ-error with HMMs have been presented, for example, in [13].

Acknowledgement

The author would like to thank Mr. Albert Klaseboer for running experiments with the Swiss-French database.

References

[1] H. Bourlard, N. Morgan, and S. Renals. Neural nets and hidden Markov models: Review and generalizations. *Speech Comm.*, 11:237–246, 1992.

[2] W. Chou, B. H. Juang, and C. H. Lee. Segmental GPD training of HMM based speech recognizer. In *Proc. ICASSP*, pages 473–476, San Francisco, CA, USA, March 23-36 1992.

[3] C. Dugast, L. Devillers, and X. Aubert. Combining TDNN and HMM in a hybrid system for improved continuous-speech recognition. *IEEE Trans. on Speech and Audio Processing*, 2:217–223, 1994.

[4] M. J. Hunt and C. Lefebvre. A comparison of several acoustic representations for speech recognition with degraded and undegraded speech. In *Proc. ICASSP*, pages 262–265, Glasgow, Scotland, May 23-26 1989.

[5] H. Iwamida, S. Katagiri, and E. McDermott. Speaker independent large vocabulary word recognition using an LVQ/HMM hybrid algorithm. In *Proc. ICASSP*, pages 553–556, Toronto, Canada, May14-17 1991.

[6] H. Iwamida, S. Katagiri, E. McDermott, and Y. Tohkura. A hybrid speech recognition system using HMMs with an LVQ-trained codebook. In *Proc. ICASSP*, pages 489–492, Albuquerque, NM, USA, April 3-6 1990.

[7] S. Kapadia, V. Valtchev, and S. J. Young. MMI training for continuous phoneme recognition on the TIMIT database. In *Proc. ICASSP*, pages 491–494, Minneapolis, MN, USA, April 27-30 1993.

[8] D. G. Kimber, M. A. Bush, and G. N. Tajchman. Speaker-independent vowel classification using hidden Markov models and LVQ2. In *Proc. ICASSP*, pages 497–500, Albuquerque, NM, USA, April 3-6 1990.

[9] T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks: Benchmarking studies. In *Proc. ICNN*, pages 61–68, San Diego, July 1988.

[10] T. Kohonen, J. Kangas, J. Laaksonen, and K. Torkkola. LVQ_PAK: A program package for the correct application of Learning Vector Quantization algorithms. In *Proc. IJCNN*, pages 725–730, Baltimore, June 1992.

[11] P. Le Cerf, W. Ma, and D. Van Compernolle. Multilayer perceptrons as labelers for hidden Markov models. *IEEE Trans. on Speech and Audio Processing*, 2:185–193, 1994.

[12] J. Mäntysalo, K. Torkkola, and T. Kohonen. Mapping context dependent acoustic information into context independent form by LVQ. *Speech Comm.*, 14:119–130, 1994.

[13] S. Nakagawa and H. Suzuki. A new speech recognition method based on VQ-distortion measure and HMM. In *Proc. ICASSP*, pages 676–679, Minneapolis, MN, USA, April 27-30 1993.

[14] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proc. IEEE*, 77:257–286, 1989.

[15] C. W. Therrien. *Decision, Estimation, and Classification.* John Wiley and Sons, 1989.

[16] K. Torkkola, J. Kangas, P. Utela, S. Kaski, M. Kokkonen, M. Kurimo, and T. Kohonen. Status report of the Finnish phonetic typewriter project. In *Artificial Neural Networks (Proc. ICANN)*, pages 771–776, Espoo, Finland, June 24-28 1991. North-Holland.

[17] V. Valtchev, S. Kapadia, and S. J. Young. Recurrent input transformations for hidden Markov models. In *Proc. ICASSP*, pages 287–290, Minneapolis, MN, USA, April 27-30 1993.

[18] S. J. Young. Competitive training: a connectionist approach to discriminative training of hidden Markov models. *Proc. IEE*, 138:61–68, 1991.

[19] S. J. Young. *HTK: Hidden Markov model toolkit V1.4 – Reference manual.* Cambridge University Engineering Department, October 1992.

[20] G. Yu, W. Russel, R. Schwartz, and J. Makhoul. Discriminant analysis and supervised vector quantization for continuous speech recognition. In *Proc. ICASSP*, pages 685–688, Albuquerque, NM, USA, April 3-6 1990.

AUTOASSOCIATOR-BASED MODULAR ARCHITECTURE FOR SPEAKER INDEPENDENT PHONEME RECOGNITION

L. Lastrucci[†], G. Bellesi[‡], M. Gori[†], and G. Soda[†]

[†]Dipartimento di Sistemi e Informatica
Università di Firenze
Via di Santa Marta 3 - 50139 Firenze - Italy
Tel. +39 (55) 479.6361 - Fax +39 (55) 479.6363
e-mail : luca@mcculloch.ing.unifi.it

[‡]Softeam Applicazioni di Base
Via P. Carpini 1 - 50127 Firenze - Italy
Tel. +39 (55) 422.1494 - Fax +39 (55) 434.126

Abstract - In this paper, we propose a modular architecture where the interactions among different modules are controled by proper autoassociators. The outputs of these modules are computed by *sigma p-neurons* whose inputs come from both a feedforward network performing classification and an autoassociator. The outputs of the autoassociators are used for performing pattern rejection, thus reducing significantly the problems due to interaction of different modules. The proposed architecture is validated by experiments of speaker independent phoneme recognition on continuous speech with TIMIT data base with very promising results.

INTRODUCTION

In the last few years many researchers have focussed their efforts in specializing neural networks more or less related to Backpropagation learning scheme for phoneme recognition. Unlike the challenging results obtained concerning phoneme discrimination, so far no enough care has been placed to the scaling up of similar solutions. This is certainly an important issue for any practical application. As Jacobs identified [1], there are two problems with monolithic networks, namely spatial and temporal crosstalk, which lead us to believe that modular systems are necessary for training nets on complex problems like phoneme recognition. Spatial crosstalk occurs when different groups of units serve different tasks; in this case hidden units being trained to resolve resid-

0-7803-2026-3/94 $4.00 © 1994 IEEE

ual error will receive conflicting information. Temporal crosstalk occurs when different portions of the training set contain data to separate and conflicting functions; in this case the hidden units being trained will suffer from overfitting with consequent degraded performance. The separation of complex tasks into sub-tasks which are handled by a group of cooperating expert sub-nets allows us to avoid spatial and temporal crosstalk. In fact, each sub-network is trained only on training data for its particular expert task. Moreover, also modular schemes built up with similar architectures as modules (e.g. Waibel's connectionist glue [5]) have a major flaw in the impossibility of guaranteeing that any module, defined for dealing with a limited number of classes, is able to reject effectively patterns of other classes. Rejection criteria based on the error with respect to the target are not very meaningful, because cases can be found where that error is very low, whereas the associated pattern has nothing to do with the classification problem. This happens because the resulting separation surfaces are not closed and do not "envelope" the examples by capturing their probability distribution.

In order to overcome this problem, in this paper we suggest using a modular architecture based on the capabilities of multilayered networks used as autoassociators to offer a reliable criterion for rejecting patterns belonging to classes not used during the learning. The architecture is based on a set of modules based on multilayered networks. Each module is specialized for the recognition of a small phoneme subset. In order to avoid conflicts among modules due to their limited rejection capabilities, the patterns of the same class are autoassociated using a multilayered autoassociator. The outputs of the autassociators are properly processed in order to obtain a single value that is used, together with the outputs coming from the classification network, for feeding the output neurons of the module that act like sigma-p neurons [2]. In so doing, the outputs of the modules are close to the outputs of the classification network only when the autoassociators "enable" the classification. This solution faces the fundamental limitation of feedforward networks to perform pattern classification by open separation surfaces, which limits severely the scaling up.

THE MULTILAYERED AUTOASSOCIATOR

Multilayered networks working as autoassociators are forced to reproduce the input to the output during the training phase. The autoassociators have been suggested mainly for problem of image compression [6]. In this case the compression is performed at the hidden layer. The information represented by the

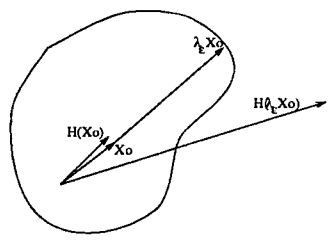


Figure 1: Nonlinear autoassociators perform pattern autoassociation of closed regions of the pattern space.

hidden units can be reproduced subsequently by the computation carried out at the last layer. In this paper, we exploit another nice theoretical property of autoassociators that can easily be understood by analyzing the separation surfaces that are drawn by the learning algorithm. Let us consider introduce the concept of ε-autoassociated patterns. A vector X_0 is ε-autoassociated if

$$\frac{||X_L - X_0||}{||X_L||} < \varepsilon$$

where $X_L = \mathcal{H}(X_0)$, being $\mathcal{H}$ the mapping provided by the autoassociator. If the neurons are based on squashing functions then there exits $\lambda_\varepsilon > 0$ such that

$$\frac{||\mathcal{H}(\lambda X_0) - \lambda X_0||}{||\mathcal{H}(\lambda X_0)||} > \varepsilon \quad \forall \lambda > \lambda_\varepsilon.$$

This is due to the saturation of the hidden units when choosing "high" values of λ. Notice that this nice property does not hold for linear autoassociator. Basically, the nonlinear neurons are responsible of the closed surface depicted in Fig. 1. For this reason, the application of multilayered autoassociators to problems like speech verification seems adequate and successful [3]. For this kind of problems the role of the nonlinearity is very clear, whereas for problems of compression, one may wonder if the use of linear networks can be sufficient in many cases. However, also for problems of compression, the nonlinearity can be desirable since one may expect better interpolation capabilities.

AMA: AUTOASSOCIATOR-BASED MODULAR ARCHITECTURE

The problem of speaker independent phoneme recognition has already been faced with modular architectures by many researchers since it is well known that

supervised neural networks can perform very well on small phoneme sets, but that the results do not scale up very well with the number of phonemes. One of the most successful approaches has been proposed by Waibel *et al.* in [5], where independent classifiers were merged with a sort of *connectionist glue*. The basic idea of using separately trained modules and neurons acting as a connectionist glue is very good, but one basic problem seems to be that the modules use to react significantly also when fed on patterns of different classes. This makes the task of the connectionist glue very hard, since the final optimization step must recover all the module false reactions. The *AMA* (Autoassociator-based Modular Architecture) have been conceived bearing in mind this problem that is faced by the additional introduction of the autoassociators.

In the *AMA* modular architecture, depicted in Fig. 2, each module contains as many autoassociators as classes. The output neurons receive the information from both the module classifier and the autoassociators as follows:

$$a_i(t) = (\frac{x_{c,i}}{s_{c,i}})^{w_{c,i}} * (\frac{x_{a,i}}{s_{a,i}})^{w_{a,i}} \tag{1}$$

being $x_{c,i}$ and $x_{a,i}$, with $i = 1, \ldots, n$ the i-th output of the classifier and autoassociator, respectively. The output of the autoassociator is processed in such a way to give high score when small distance is reported from input and output. A possible choice is simply $x_{a,i} = \frac{1}{\|X_L - X_0\|}$. $w_{c,i}$ and $w_{a,i}$ are the corresponding weights and $s_{c,i}$, $s_{a,i}$ are learneable coefficients used for re-scaling the inputs. The activation is mapped to the output by

$$y_i(t) = f(K * a_i(t) - \theta_i) \tag{2}$$

being f( ) is the squashing function $f(x) = \frac{1}{1+e^{-x}}$ The coefficients used in equation (2) are useful for proper re-scaling of the output neuron activation.

Another possible choice for the output neurons is that of using a thresholding criterion for taking the information of the autoassociators into account:

$$a_i(t) = \begin{cases} x_{ci} & : \quad \text{if } x_{a,i} < T_i \\ 0 & : \quad \text{otherwise} \end{cases} \tag{3}$$

being T_i a threshold related to the $i-th$ autoassociator input/output distance. In so doing, the output neurons act like an ordinary classifier provided that the thresholding criterion is met. When this does not hold, the outputs are set to "0", thus correcting the classifier trend to perform pattern classification no matter what is the input. The difference between the two solutions (1) and (3) is that in the first case the correction of the classifier behavior is gradual, whereas in the second one, it is based strictly on a thresholding criterion.

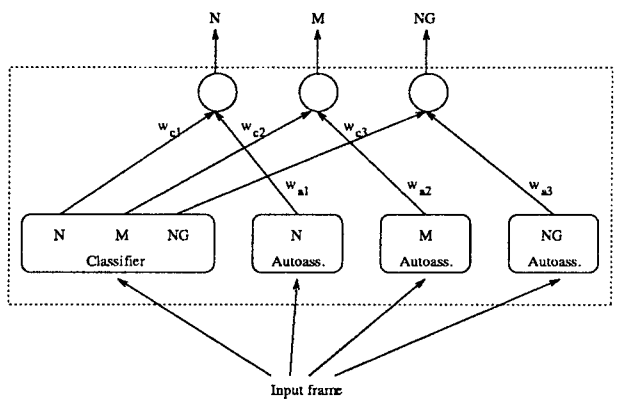


Figure 2: Nasal phoneme module: the classifier and 3 autoassociators, one for each phoneme

The training phase takes place in classifiers and autoassociators independently. In the case of output neurons following equation (1), a subsequent learning step is required for assessing the optimal values of the output neuron parameters. This can easily and quickly be achieved since we need optimizing the parameters of a single layer. For the output neurons of equation (1), one can also perform a final global optimization step for assessing the optimal value of all the module's parameters. The implementation of such a step is not very difficult since, because of the layered architecture, it can be based on Backpropagation for gradient computation.

Feature extraction

The experiments were carried out using RASTA-PLP [4] and Bark-scaled FFT preprocessing schemes.

The PLP speech analysis technique estimates an all-pole autoregressive model of the auditory-like short-term speech spectrum. PLP has been shown to be efficient in suppressing speaker-dependent components in the speech signal. The auditory-like spectrum is obtained by integrating the short-term power spectrum of speech over simulated critical-band auditory masking curves, resampling the integrated spectrum in approximately 1 Bark intervals, modifying the spectral amplitude by a simulated fixed equal-loudness curve, and compressing it through the cubic root nonlinearity to simulate the intensity-loudness power law of hearing. This autoregressive modeling efficiently approximates the spectral peaks in the auditory-like spectrum. The cepstral coefficients of

the PLP all pole model are recursively computed.

An 8th order RASTA-PLP model was used with the same frame length and overlap than FFT preprocessing thus obtaining 9 component vectors (including log power). Delta coefficients were computed for each frame and the 18 components were normalized with respect to their mean value of the training environment.

When using Bark-scaled FFT preprocessing, the spectra were computed by the 256-inputs FFT (frame length=16 ms) with a Hamming window. The network scanned the input parameters every 64 samples (4 ms), thus considering overlapped information among contiguous frames. Each input frame was represented as a 64-component spectral vector, grouping the channels according the Bark scale. The frames were normalized on a temporal window (with 500 ms), extended in the past starting from the last frame.

The experimental results showed that RASTA-PLP needed fewer coefficients than the Bark-scaled FFT for obtaining comparable performance for the nasal classifier.

In the following sections we describe the classifier and auto-associators net architectures and give experimental results obtained with RASTA-PLP.

Nasal Classifier

The experiments were based on a recurrent network classifier having self-loop connections only trained by *BPS* learning algorithm [7]. These architectures are particularly suitable for phoneme recognition since we can guarantee a *forgetting behavior* in advance when choosing the self-loop weights properly [8]. Moreover, their training with *BPS* is more efficient than using general algorithms for recurrent networks since it is local in both time and space.

In our experiments, we tested different architectures using simply "trial and error" to assess the optimal number of hidden units for a given speech preprocessing. The best results were obtained with a 3 layer fully-connected net with 18 inputs, 3 exclusive static output neurons and 35 dynamic hidden neurons with "delay coefficients".

For the training phase, we used nasal phonemes of 36 male speaker each uttering 10 sentence. After preprocessing the signal as previously described, for each phoneme, we placed supervisions only where there was clear phonetic evidence. For this reason we avoided supervising the speech signal around the transition frames. During the learning phase, we weighted the error at the output neurons to balance the different number of frames for different phonemes.

Table 1: EXPERIMENTAL RESULTS FOR THE NASAL CLASSIFIER.

input/output	*m*	*n*	*ng*	*err.*	*n. frames*	*recog. %*
m	**1487**	743	753	1496	2983	49.85
n	476	**1171**	226	702	1873	62.52
ng	80	144	**292**	224	516	56.59
Crosstalk	556	887	979			
Recognition	54.91%					

To test the classifier, we used a database consisting of 25 speakers. The results we obtained are reported in tab. 1.

Nasal autoassociators

We tested different autoassociator architectures. The best results were obtained with a 3 layer feed-forward neural network (20-6-20). We used a 50 male speaker database for training and we created 3 different learning environments, one for each autoassociator (i.e. one for each phoneme). The target on the output units was imposed to be equal to the input.

The autoassociators were tested on 44 American phonetic classes obtained with the Kai-fu-lee table for a 18 male speaker database. Recognition and rejection rate results as a function of the threshold value are reported in Fig. 3.

Experimental results for phoneme recognition

We performed some preliminary experiments of speaker independent phoneme recognition for validating the *AMA* architecture. The output from the classifier and auto-associators were combined into 3 output neurons. We tested the architecture using a model based on both equations (1) and (3).

The experimental results are reported in Fig. 4 for the case of output neurons following the thresholding criterion (3) and equation (1) respectively.

CONCLUSIONS

In this paper, we have proposed a novel modular architecture referred to as AMA. This architecture has been evaluated with preliminary experiments of phoneme recognition. Comparing Fig. 4 and Fig. 5, we can see that the rejection

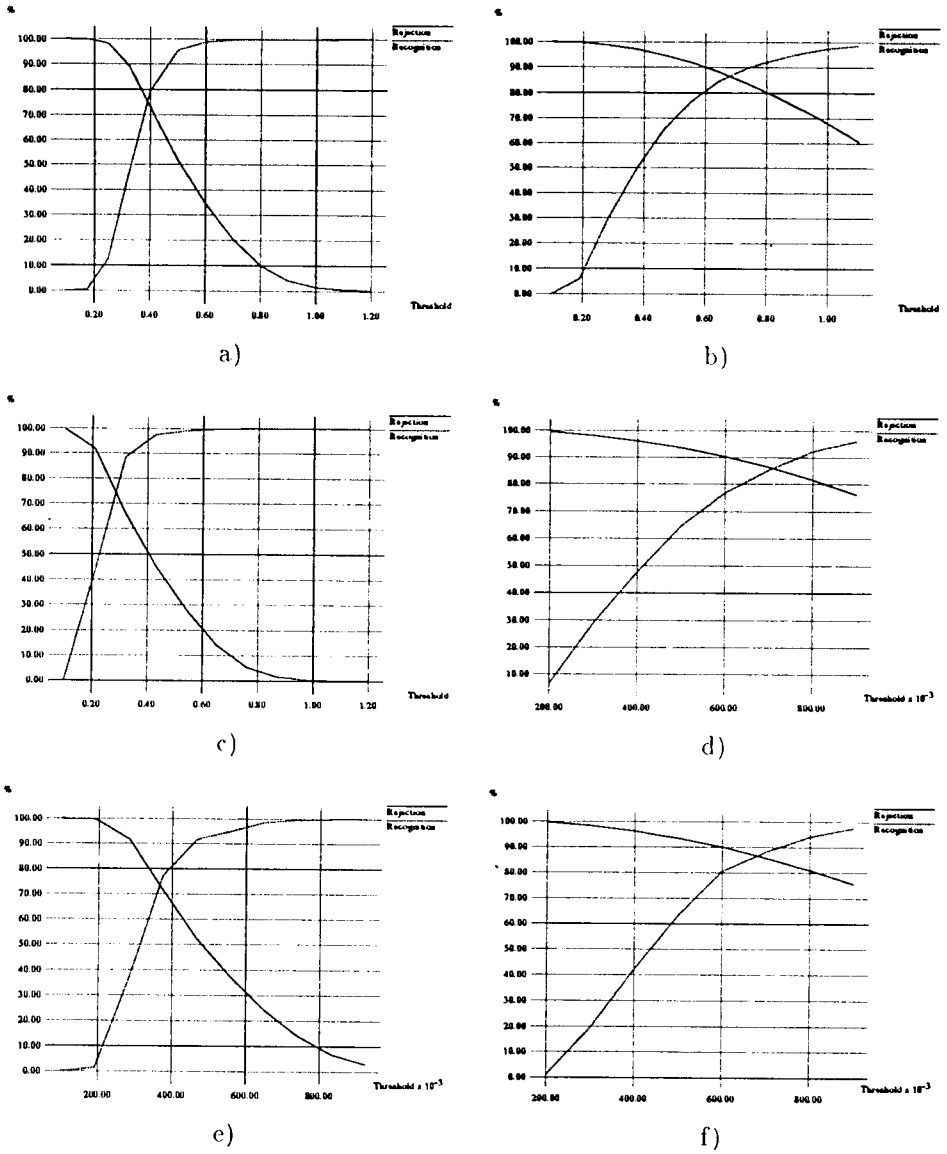


Figure 3: Experimental results on the test set for the autoassociators on 44 phonemes of 18 speakers with DFT and RASTA-PLP: /*n*/ fig. a), b); /*m*/ fig. c), d); /*ng*/ fig. e), f).

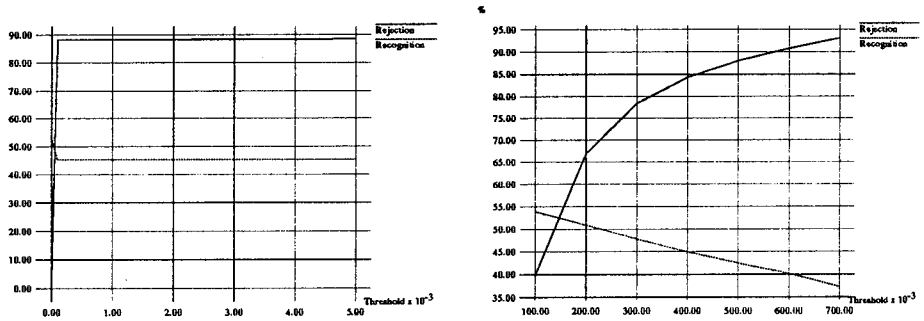


Figure 4: Experimental results on the test set. a) Coder using the thresholding criterion (4); b) Coder using equation (1).

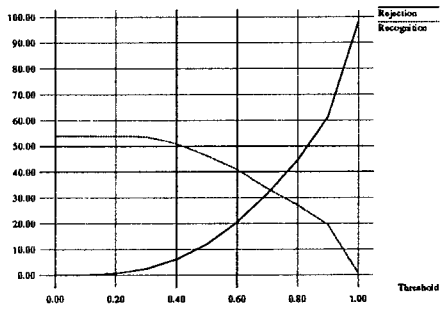


Figure 5: Experimental results of the test for the nasal classifier only, on 44 English phonemes.

rate frame by frame for the nasal phonemes was very high using autoassociators to the prejudice of the recognition rate. The idea on which AMA relies is quite general, and makes it attractive for any problems of pattern recognitions with many classes.

References

[1] R. A. Jacobs, M. I. Jordan, A. G. Barto, "Task Decomposition through competition in a modular connectionis architecture: The what and where tasks", COINS Technical Report 90-27, march 1990.

[2] D. E. Rumelhart, J. L. McClelland, "Parallel Distributed Processing", vol. 1: Foundations, pp. 425-426.

[3] L. Lastrucci, M. Gori, and G. Soda, "Neural Autoassociators for Phoneme-based Speaker Verification", Proc. of Workshop on Automatic Speaker Recognition, Identification and Verification, pp. 158-162, Martigny, Switzerland, April 5-7, 1994.

[4] H. Hermansky and N. Morgan and A. Bayya and P. Kohn, "RASTA-PLP Speech Analysis Technique", Proceedings Int'l Conference on Acoustics Speech and Signal Processing, pp. 121-124, vol. I, San Francisco, California, 1992.

[5] A. Waibel, H. Sawai, K. Shikano, "Modularity and Scaling in Large Phonemic Neural Networks", IEEE Transactions on Acoustics, Speech and Signal Processing, December, 1989.

[6] G.W. Cottrell, and P. Munro and D. Zipser, "Learning Internal Representations from Gray-Scale Images: An Example of Extensional Programming", Ninth Annual Conference of the Cognitive Science Society, pp. 462-473, Lawrence Erlbaum, Hillsdale publisher, Seattle 1987.

[7] Y. Bengio, R. De Mori, and M. Gori, "Learning the Dynamic Nature of Speech with Backpropagation for Sequences", Pattern Recognition Letters, vol. 13, N0. 5, pp. 375-385, may 1992, (special issue on Artificial Neural Networks).

[8] P. Frasconi, M. Gori, and G. Soda, "Local Feedback Multilayered Networks", Neural Computations, vol. 4, No. 1, pp. 120-130, January 1992.

NON-LINEAR SPEECH ANALYSIS USING RECURRENT RADIAL BASIS FUNCTION NETWORKS

Paul A. Moakes Steve W. Beet
University of Sheffield
Department of Electronic and Electrical Engineering
P.O.Box 600, Mappin Street, Sheffield S1 4DU, UK.
Tel: +44 742 825414 Fax: +44 742 726391
e-mail: P.A.Moakes@sheffield.ac.uk

Abstract - This paper presents a recurrent radial basis function network as a one step ahead predictive speech signal filter. The resulting non-linear estimation of the signal state space allows accurate prediction using only three delayed samples of clean speech and in noisy speech six samples allow this performance to be maintained. The prediction residual can be used as a powerful speech pitch detector and the nonlinear network shows significant improvement over conventional auto-regressive filters, allowing post-processors to make more accurate estimations of pitch pulse position, the pitch, and the regions of voiced speech. This represents a new form of preprocessing for pitch tracking of real speech in a noisy environment.

INTRODUCTION

Speech production can be modeled using an auto-regressive (AR) filter with an excitation signal comprising of a series of quasi-periodic pitch pulses during voiced speech and white noise during unvoiced speech. Pitch period and the fundamental frequency estimation are important for speech coding and recognition, however, accurate pitch detection is considered one of the most difficult tasks in speech processing. The variability of speech and speaking environments causes difficulties in pitch determination with low frequency often being masked or lost in noise.

The location of the pitch pulse in voiced speech is important for linear predictive coding (LPC) where reduced sensitivity to the fundamental frequency provides a more accurate estimate of the filter parameters. In frame based speech analysis this leads to pitch synchronous prediction and in an auto-regressive moving average (ARMA) model of speech the pitch is incorporated

0-7803-2026-3/94 $4.00 © 1994 IEEE

into the production model [12]. The pitch pulse can be detected directly from physical measurements of the speaker using an electroglottograph or laryngograph. This results in a two channel speech analysis system where the physical information is used to improve the speech model [6]. This paper proposes a method of pitch extraction based solely on the sampled time series of speech.

The ability of neural networks to estimate non-linear functions and to predict time series [2,10] is well known and the application of neural networks for the identification and interpretation of speech signals is of particular interest due to the non-linear and non-stationary nature of speech [8]. However, neural network applications in speech signal processing have tended to focus on using extracted feature spaces such as LPC coefficients for their inputs [9], mainly as a result of the importance of LPC parameters in vocal tract identification.

Lowe and Webb [7] have trained neural networks to model the dynamics of isolated vowels and fricatives based on the speech time series and Townshend [13] has used LPC prediction residuals for the identification of non-linear speech elements [13]. This paper extends this work to the use of RBFNs as a non-linear AR filter with minimal a *priori* signal information. The filter is implemented for on-line adaptive prediction of speech signals in the time sample domain and the prediction residual is investigated for use as an accurate pitch detector.

RADIAL BASIS FUNCTION NETWORKS

RBFNs [1] are one-hidden-layer neural networks. The hidden layer contains nodes which perform a non-linear transformation of the input data using a parameter vector called a centre and the output layer consists of linear combiners which calculate the weighted sum of hidden layer nodes.

At each iteration the Euclidean distance, $||x - c_j||$, between each node centre, c_j, and the input vector, x, is calculated. The result is passed through a non-linear function, $\Phi(.)$, to generate the node output, h_j

$$h_j = \Phi(||x - c_j||) \tag{1}$$

In this paper the thin-plate spline function, $\Phi(\nu) = \nu^2 \log(\nu)$, is chosen for its non-localised response. This has been shown to achieve excellent approximation ability due to the characteristic $\Phi(x) \rightarrow \infty$ as $x \rightarrow \infty$ [2] and readily accommodates the rapidly changing speech state-space. The network output $\hat{y}$ is given by

$$\hat{y} = \sum_{j=1}^{n_h} \eta_j h_j \tag{2}$$

where η_j are the node weights and n_h is the number of hidden nodes.

An RBFN can approximate arbitrarily well any continuous function on a compact domain if sufficient basis functions are used. The changing complexity of the speech dynamics implies that the number of nodes required will vary depending upon the speech frame, so Kolmogorov's theorem [5] is used to fix

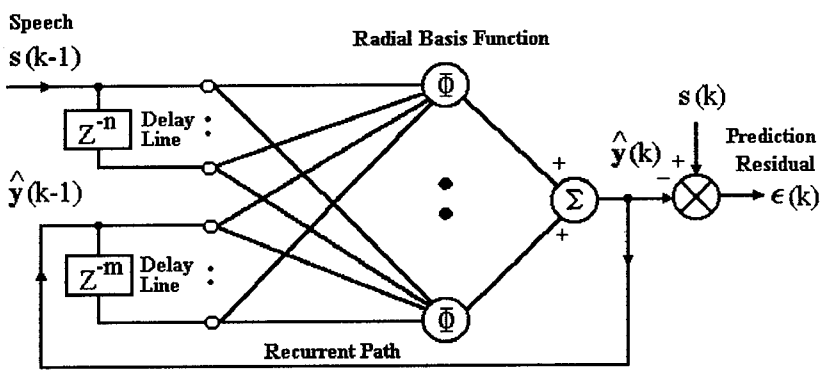


Figure 1: An output feedback recurrent radial basis function network

the number of nodes arbitrarily at $2d+1$, where $d = n + m$ is the dimension of the input vector x.

Recurrent Networks

Recurrent RBFNs (RRBFNs), figure 1, incorporate network parameters in the input vector, in this paper the input vector is augmented with output feedback. The input vector x at sample k is thus

$$x_k = (s_{k-1}, \ldots, s_{k-n}, \hat{y}_{k-1}, \ldots, \hat{y}_{k-m}) \tag{3}$$

where n is the number of lagged speech samples, s, and m is the number of lagged RRBFN predictions, $\hat{y}$.

The speech characteristics do not change rapidly outside of the sample window and the introduction of output feedback adds context to the input vector without increasing the size of the speech window. In noisy speech corrupted samples are augmented with prediction outputs with a reduced noise content, allowing improved recovery of the pitch pulses.

Centre Clustering

Network approximation ability is dependent upon the RRBFN centre locations. The centres are initially selected randomly within the bounds of the speech state-space and clustered using a variation of the Kohonen Self-Organising feature Map (SOM) introduced by Huntsberger and Ajjimarangsee [3] and extended by Zheng and Billings [14]. This fuzzy clustering approach avoids the sensitivity of κ-means clustering to the initial centre positions, preventing false minima being found.

The clustering algorithm used here is

1. Set an initial clustering gain of $\alpha_0 = 0.2$ and an initial number of neighbours $N_c = n_h - 1$.

2. Use a decaying clustering gain of $\alpha = \alpha_0(1 - t/T)$, where T is the total number of iterations during training and t is the current iteration.

3. Select 30 input vectors, x_i, from the speech at random.

4. Calculate the distance of the centres from the input vector x_i.

5. Adapt closest centre c_j and its N_c nearest neighbours, c_{vj}, using

$$c_{vj} = c_{vj} + \alpha\mu_v(x_i - c_{vj}) \tag{4}$$

   where μ_{vj} is the fuzzy membership function.

6. Reduce the number of nearest neighbours, $N_c = N_c - 1$, and repeat from step 2 until $N_c = 0$.

The fuzzy membership function used here is

$$\mu_j = \begin{cases} 1, & if \ \ \|x_i - c_{vj}\| = 0 \\ \left(\sum_{l=1}^{n_h} \frac{\|x_i - c_{vj}\|}{\|x_i - c_l\|}\right)^{-1}, & \forall\|x_i - c_l\| \neq 0 \ \ otherwise \end{cases} \tag{5}$$

During on-line adaptation the centres are adjusted using the κ-means clustering technique to adjust to the changing speech dynamics [4]. It also allows the correction of the recurrent elements of the centres which during initial clustering are assumed to receive an input equal to the speech samples, i.e. the network has no prediction error and $\hat{y}(k) = s(k)$, due to the unavailability of network predictions. At each sample the closest centre to the current input vector, c_j, is updated according to

$$c_j = c_j + \alpha_\kappa(x - c_j) \tag{6}$$

where α_κ is the learning rate. Using a small value of α_κ allows weight adaptation to be considered independently of centre adjustment.

Weight Adaptation

The response of the RRBFN is linear with respect to the node output weights resulting in an output error surface with only one global minimum if the centres are fixed. This allows the weights to be updated using a covariance matrix approach based on the Kalman filter (KF) and results in a rapid convergence

to the optimum weights required to minimise the mean squared prediction error. The KF equations for updating the hidden layer weights [11] are

$$K_k = P_{k-1}\hat{\phi}_k \left(\lambda_k + \hat{\phi}_k^T P_{k-1}\hat{\phi}_k\right)^{-1} \tag{7}$$

$$P_k = \frac{1}{\lambda_k}\left(P_{k-1} - K_k\hat{\phi}_k^T P_{k-1}\right) \tag{8}$$

$$\Theta_k = \Theta_{k-1} + K_k\epsilon_k \tag{9}$$

where K_k is the KF gain and P_k is the RRBFN inverse covariance matrix. Θ_k is the vector of hidden layer weights η_j, $\hat{\phi}_k$ is the vector of node outputs h_j, and ϵ_k is the prediction error $s_k - \hat{y}_k$.

λ_k is a variable forgetting factor (VFF) which allows the KF to estimate time-varying system parameters by exponentially windowing the speech, with an effective memory of $1/(1-\lambda_k)$ samples. λ_k is based on the filter error information content [12], defined as the weighted sum of squares of the residual errors, V_k, and which can be expressed recursively as

$$V_k = \lambda_k V_{k-1} + \epsilon_k^2\left(1 - \hat{\phi}_k^T K_k\right) \tag{10}$$

Applying a constraint of constant error information, $V_k = V_{k-1} = V_1$, allows the VFF to be defined from (10) as

$$\lambda_k = 0.99 - \gamma\epsilon_k^2\left(1 - \hat{\phi}_k^T K_k\right)/V_1 \tag{11}$$

where V_1 can be taken as the average filter error information.

A lower limit of $\lambda_k = max\,[\lambda_k, 0.9]$ and a gain factor of $\gamma = 0.15$ give λ_k the range $0.9 \leq \lambda_k \leq 0.99$ as suggested by Salgado [11] for a compromise of adaptive speed and memory. The error information content is constant for a signal with a Gaussian driving source, but large values occur when the source signal changes, such as at pitch pulses [12]. This results in a small λ_k and the KF estimates are then based on a shorter window of speech allowing rapid adaptation to the changing dynamics. This creates a sharp error at the point of glottal closure which when observed against the average filter error provides a good indication of the onset of the pitch pulse. Simple post-processing techniques can then be used to select the most likely pitch positions from these pulse candidates.

SPEECH PREDICTION

The performance of an RRBFN for the one step ahead prediction of speech was compared with that of a RBFN and a linear AR filter for the noise free utterance of the word "five" sampled at 20kHz. The RRBFN had an input vector of dimensions $n = 3$, $m = 3$, hence a network size of $n_h = 13$ was selected to satisfy Kolmogorov's theorem [5]. In order to obtain a fair comparison the RBFN also had 13 hidden nodes, but used an input vector of $n = 3$, $m = 0$,

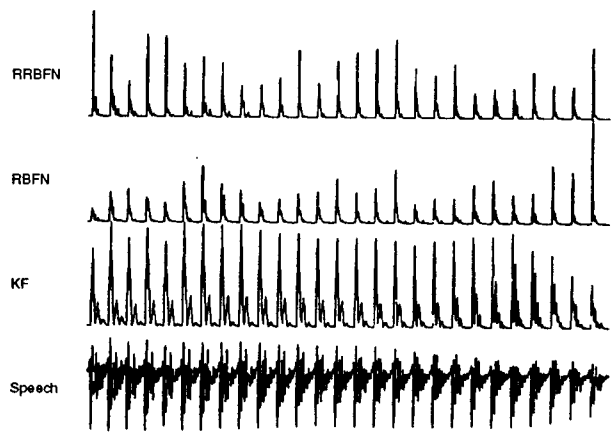


Figure 2: Normalised voice source estimates for the noise free word "five"

and the AR filter used 13 lagged speech samples. The weights in all three networks were updated using the KF approach (7-9) with a VFF defined by (11). The basis function centres were positioned initially using fuzzy clustering and adapted on-line using κ-means clustering (6), with an adaptive rate of $\alpha_\kappa = 0.01$.

Voice Source Estimation

The accumulated squared prediction residual for the filters was calculated over an eight sample window to generate an estimate of the speech voice source. The normalised voice source estimates are shown in figure 2. Even though the KF has a speech window four times wider than the RBFN, the RBFN can detect the change in signal driving function at the pitch pulse with greater precision than the KF and has a lower noise floor between pulses. The increased contextual information inherent in the RRBFN is a marginal improvement over the RBFN, but its most significant contribution occurs when noise is present in the speech signal.

The networks were tested using the word "five" corrupted with additive noise to give a signal to noise ratio (SNR) of 3dB. Due to the poor performance of all networks using only 3 speech samples the speech windows were increased. The RRBFN was extended to $n = 6$, $m = 3$, $n_h = 19$, the RBFN to $n = 6$, $m = 0$, n_h=19, and the AR filter used a 19 sample speech window. Figure 3 shows the resulting voice source estimates. The AR filter residual becomes lost in the noise floor, making pitch prediction difficult even with a large speech window. The RBFN fares better, with distinguishable peaks at pitch pulses and an even noise floor. However, the RRBFN is better still with more distinctive peaks and a lower noise floor.

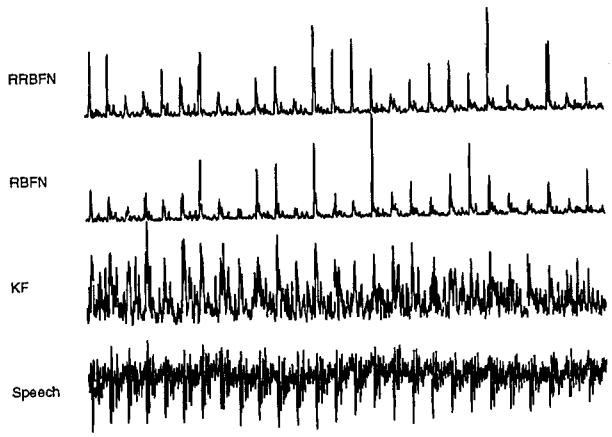


Figure 3: Normalised voice source estimates for the word "five" with an SNR of 3dB

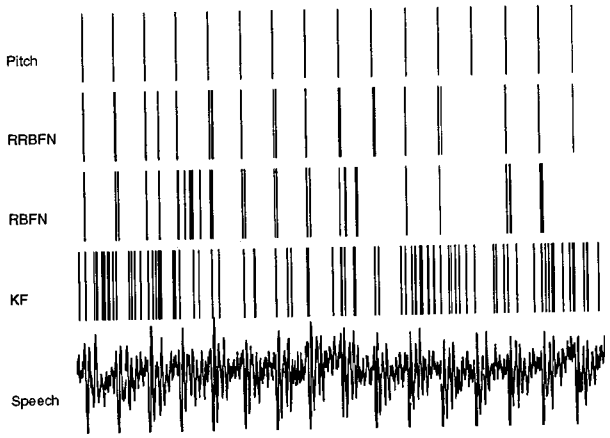


Figure 4: Pitch candidates for the word "five" with an SNR of 3dB

Pitch Detection

Pitch detection was achieve by thresholding 15ms windows of the voice source estimate at twice the standard deviation of the voice source over several several windows. A pitch candidate is observed when the voice source rises above the threshold and figure 4 shows the pitch candidates obtained from the three filters in noisy speech. The improved peaks at pitch events and the reduced noise

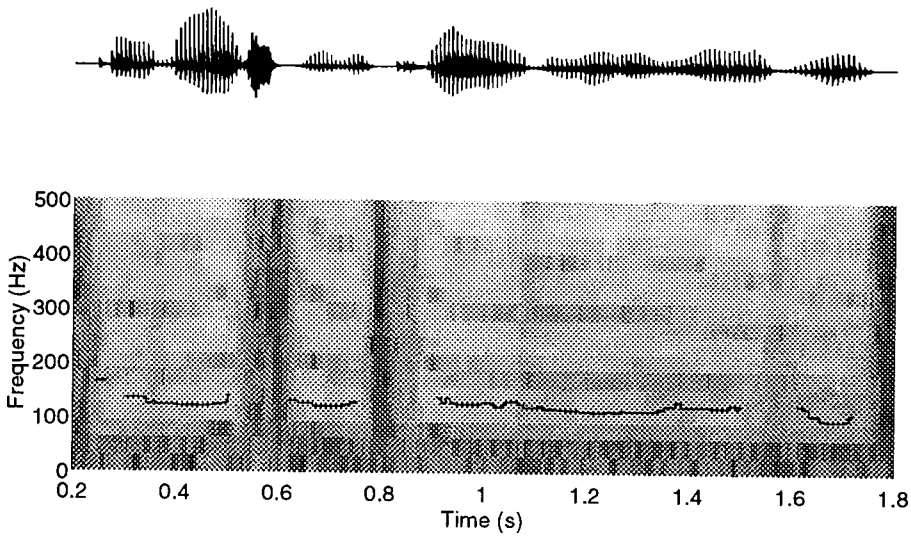


Figure 5: Pitch track and spectrogram for a male TIMIT speaker

floor of the RRBFN ensure that fewer peaks on the voice source estimate are classed as pitch candidates. Therefore the RRBFN is a better filter for voice source estimation with more consistent pitch candidate statistics which allow the easy elimination of false candidates.

Pitch Tracking

An RRBFN pitch detector of the type used for noisy speech was applied to real speech obtained from the DARPA TIMIT continuous speech corpus. Using a sliding window of 15 pitch candidates, the speech was classified as voiced when the median pitch period exceeded the standard deviation of the pitch period within a given window. The median was preferred as it is resilient to extreme pitch estimation errors. Figure 5 shows the pitch frequency tracks of voiced speech obtained when this method is applied to the phrase, "Don't ask me to carry an oily rag like that", spoken by a male.

The tracks have been laid over the FFT derived spectrograph for comparison and can be seen to lie along the fundamental resonance in the spectrogram which suggests the correct determination of the pitch period. The algorithm provides a very clear indication of the areas of voiced speech, with no obvious mis-classification of unvoiced speech as voiced. The largest errors occur in quiet speech where the SNR is lowest and the speech dynamics are changing which could be overcome by incorporating an overall signal power weighting into the algorithm.

DISCUSSION

This paper has demonstrated the ability of RRBFNs to estimate the non-linear system dynamics of both noisy and continuous speech. The prediction residual provides a powerful pitch pulse detector and the improvement compared with a linear AR predictor supports the proposition that a non-linear speech model is more accurate. Signal noise causes significant deterioration of this result in an RBFN filter which is overcome by incorporating output feedback which inherently offers a reduced noise content into the input vector of an RRBFN.

Despite the use of a simple pitch post-processor, the pitch candidates produced by the RRBFN have provided excellent pitch tracks for continuous speech using only the time domain representation of the signal and limited *a priori* information. These pitch candidates are suitable for pitch synchronous estimation, although it is preferable to use the initial voice source estimate as a more accurate guide to the areas of consistent dynamics within speech. This will enable future work to concentrate on the incorporation of the pitch into a non-linear ARMA model of speech which will be adapted on-line.

The prediction of speech using only three sample also provides a firm basis on which to investigate further the underlying non-linear dynamics of speech. The final objective of this work will be the extraction of a minimal feature space for use in speech recognition, encoding, noise reduction, and synthesis.

ACKNOWLEDGEMENTS

The authors wish to thank DRA Malvern, UK, for the CASE Studentship associated with this work.

REFERENCES

[1] D.S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, no. 3, pp. 321–355, 1988.

[2] S. Chen, P.M. Grant, and C.F.N. Cowan, "Orthogonal least squares learning algorithm for training multioutput radial basis function networks," *IEE Proc-F*, vol. 139, no. 6, pp. 378–384, 1992.

[3] T.L. Huntsberger and P. Ajjimarangsee, "Parallel self-organising feature maps for unsupervised pattern recognition," *Int. J. General Systems*, vol. 16, no. 4, pp. 357–372, 1990.

[4] T. Kohonen, "The self-organising map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[5] A.N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," *American Mathematical Society Translation*, vol. 28, pp. 55–59, 1963.

[6] A.P. Lobo and W.A. Ainsworth, "Evaluation of a glottal ARMA model of speech production," in *Proc. IEEE ICASSP 92*, San Francisco, CA, USA, 23-26 March 1992, vol. II, pp. 13–16.

[7] D. Lowe and A. Webb, "Adaptive networks, dynamical systems, and the predictive analysis of time series," in *Proc. First IEE Int. Conf. on Artificial Neural Networks*, London, 16-18 Oct. 1989, pp. 95–99.

[8] S. McLaughlin and A. Lowry, "Nonlinear dynamical systems concepts in speech analysis," in *Proc. EUROSPEECH 93*, Berlin, GERMANY, 21-23 Sept. 1993, pp. 377–380.

[9] S. Moon and J.-N. Hwang, "Coordinated training of noise removing networks," in *Proc. IEEE ICASSP 93*, Minneapolis, USA, 3-5 Aug. 1993, vol. I, pp. 49–54.

[10] M.A.S. Potts and D.S. Broomhead, "Time series prediction with a radial basis function neural network," in *SPIE Proc. Adaptive Signal Processing*, San Diego, CA, USA, 1991.

[11] M.E. Salgado, G.C. Goodwin, and R.H. Middleton, "Modified least squares algorithm including exponential setting and resetting," *Int. J. Control*, vol. 47, no. 2, pp. 477–491, 1988.

[12] Y.T. Ting and D.G. Childers, "Tracking spectral resonances," in *Proc. IEEE 4th ANN Workshop on Spectrum Estimation*, Minneapolis, USA, 3-5 Aug. 1988, pp. 49–54.

[13] B. Townshend, "Nonlinear prediction of speech," in *Proc. IEEE ICASSP 91*, Toronto, Canada, 1991, vol. I, pp. 425–428.

[14] G.L. Zheng and S.A. Billings, "Radial Basis Function Network Training Using a Fuzzy Clustering Scheme," Dept. of Automatic Control, University of Sheffield, Research Report 505, February 1994.

WORD RECOGNITION USING A NEURAL NETWORK AND A PHONETICALLY BASED DTW

Yoshihiro Matsuura*, Hideki Miyazawa*
*Meidensha Corporation
2-1-17 Ohsaki Shinagawa-ku
Tokyo 141 JAPAN

Toby E. Skinner**
**Adaptive Solutions, Inc.
1400 N.W. Compton Drive Suite 340
Beaverton, OR 97006 U.S.A

Abstract: We have developed a speaker-independent, isolated-word recognition system using a neural network to recognize the underlying sequence of phonemes and a DTW technique to time-align the recognized sequence of phonemes with corresponding lexical sequences of phonemes. A significant feature of this system is the ability to easily change the vocabulary, since the lexical entries are simply derived from their phoneme sequences.

INTRODUCTION

Because of its ability to time-align two perceptually-equivalent spoken words, Dynamic Time Warping (DTW) is a technique that is often used in speaker-independent, isolated-word recognition systems [1][2]. As initially formulated, the DTW technique compares the sequence of spectra comprising a spoken word with the corresponding spectral sequences from each word in the lexicon. The most similar lexical entry, as measured by the minimum amount of distortion in both time and frequency, is declared to be the recognized word. However, it is well known that two spectral sequence patterns of the same word spoken on different occasions by the same speaker or by two different speakers may differ appreciably in both duration and frequency aspects. Duration variations are implicitly accounted for by using the DTW alignment technique. Frequency differences are minimized by averaging the component spectral patterns of multiple versions of each word spoken by multiple speakers. Creating the average patterns of spectral sequences for each word in the vocabulary to be recognized requires exemplars of actual speech, including multiple versions of each word spoken by multiple speakers. This requirement makes it very difficult to change the vocabulary of the recognition system, perhaps intractable for large vocabularies. In this paper we will present a word recognition system to solve the difficulty in changing the vocabulary.

DESIGN OF THE WORD RECOGNITION SYSTEM

Outline of Design

To overcome the problem inherent in vocabulary modification, we have developed a speech recognition system (Fig. 1) which uses phoneme recognition as a prelude to word recognition. Phoneme recognition is accomplished using a neural network,

0-7803-2026-3/94 $4.00 © 1994 IEEE

with spectral patterns as input and recognized phonemes as output. Word recognition is achieved by applying a DTW technique, which aligns phoneme sequences rather than spectral sequences of the spoken word with the words in the vocabulary. A simple method is applied to calculate the distance between two phonemes, instead of calculating Euclidean distance between two spectra. The feature extraction, phoneme recognition by a neural network and word recognition by DTW matching are realized on CNAPS, which is a parallel processing system [6].

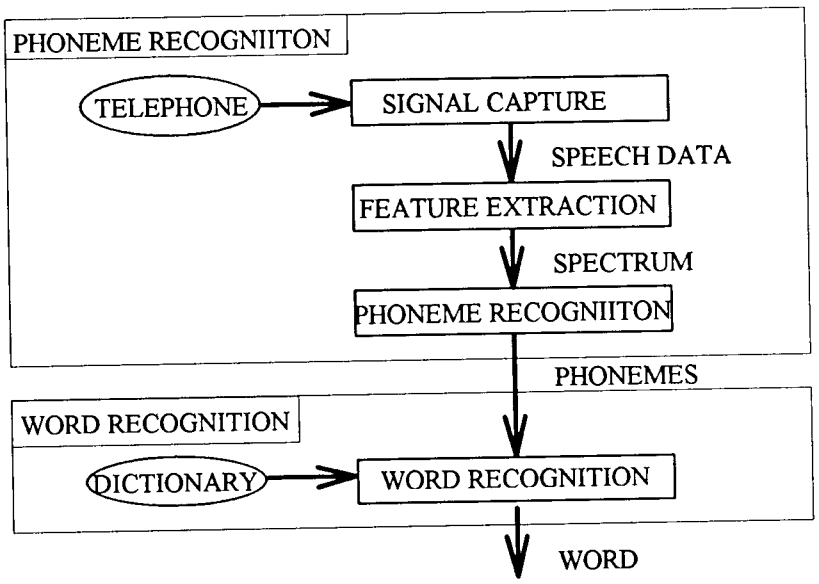


FIGURE 1. SYSTEM ARCHITECTURE

Phoneme Recognition Component

The speech recognition system operates over the telephone. The received analog signal is sampled at 8 kHz, and translated into a sequence of 27-component mel spectral feature vectors by Hanning window for window widths of 16 ms, advanced at 12 ms intervals. Each feature vector, together with the two preceding and two following feature vectors, is fed to a back propagation neural network (NN). The outputs of the NN are the 23 possible phoneme categories, with the category receiving the highest activation being the identity associated with the center input feature vector. The total architecture of the NN is 135 input units (27 x 5), 220 hidden units (as determined by experiment), and 23 output units (Figure 2). The NN was trained using 2 repetitions of each of the 101 words in the vocabulary by 15 different male speakers. The vocabulary is determined so that it covers all possible consonant-vowel, vowel-consonant vowel-vowel phonemic environment in Japanese. During the recognition of an unknown word, the phoneme recognition component uses the trained NN to determine the first and second candidate phoneme identities corresponding to each feature vector, which it passes

on to the word recognition component.

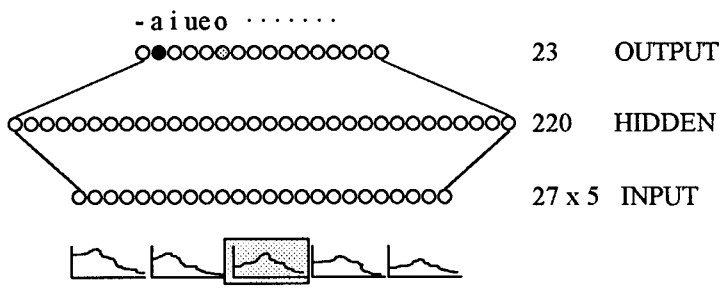


FIGURE 2. NEURAL NET FOR PHONEME RECOGNITION

Word Recognition Component

The word recognition component receives the sequence of first and second best phoneme guesses from the phoneme recognition component. This input sequence is compared with each of the phonemically-encoded templates of the words in the current vocabulary, and the most similar one is declared to be the recognized word. Since the domain of representation of the unknown word and the vocabulary templates is phoneme sequences, a simple method is used to perform the comparison (DTW matching) instead of the Euclidean distance. If the first and second guesses of the i-th frame data of a spoken word are $P_1(i)$ and $P_2(i)$, and the j-th phoneme of the template t is $P_t(j)$, then the distance at (i, j) is decided according to (1).

$$d(i,j) = \text{if} \begin{Bmatrix} P_1(i) = P_t(j) \\ P_1(i) \neq P_t(j) \text{ and } P_2(i) = P_t(j) \\ P_1(i) \neq P_t(j) \text{ and } P_2(i) \neq P_t(j) \end{Bmatrix} \text{then} \begin{Bmatrix} 0 \\ 1 \\ 2 \end{Bmatrix} \tag{1}$$

The DTW path algorithm allows three choices: right-one, right-one-and-up-one, and right-one-and-up-two, all with the same weight. In other words, when calculating the accumulated distance g(i, j) at (i, j), (2) is used.

$$g(i,j) = \min \begin{Bmatrix} g(i-1,j) \\ g(i-1,j-1) \\ g(i-1,j-2) \end{Bmatrix} + d(i,j) \tag{2}$$

Templates

To create the phoneme sequence representation for a word in the recognition vocabulary, the required information is a phonemic transcription, and the duration of each of the component phonemes. The phonemic transcription is a trivial matter especially in Japanese, and the associated duration is defined by a technique

for speech synthesis technology [4]. It determines the duration of the phoneme considering the one preceding phoneme and one following phoneme using a table, which was beforehand obtained from duration information of real spoken words. The duration is calculated with the table and a rule according to the length of the word. The duration $L(P_w^i)$ of the i-th phoneme P_w^i of a word w whose mora is M_w is calculated by (3).

$$L(P_w^i) = k(P_w^{i-1}, P_w^i, P_w^{i+1})\, M_w + b(P_w^{i-1}, P_w^i, P_w^{i+1}), \qquad (3)$$

where k(x, y, z) and b(x, y, z) are coefficients for the phoneme y between the phoneme x and z. Those coefficients were beforehand obtained from duration information of real spoken words.

Finally a phoneme sequence like "zzzeeerroooo" is generated from a transcribed word "zero" and its duration information obtained by the technique.

RECOGNITION EXPERIMENT

Phoneme and Word Recognition Experiment

An experiment was conducted to measure the performance of this speech recognition system. As already mentioned above, the NN for phoneme recognition was trained using 2 repetitions of each of the 101 words by 15 male speakers (training speaker set 1 as seen in Fig. 3). The data for testing consists of 2 repetitions of the 101 words by 5 male speakers excluded from the training of the system (testing speaker set 1 as seen in Fig. 3). Table 1 shows the results of phoneme recognition. Each recognition rate is the average of those of the 23 phoneme categories. Table 2 shows the results of word recognition.

TABLE 1. PHONEME RECOGNITION RESULTS

SPEAKER SET	FIRST GUESS (%)	SECOND GUESS (%)
TRAINING SPEAKER 1	82.63	89.63
TESTING SPEAKER 1	70.10	83.10

TABLE 2. WORD RECOGNITION RESULTS (101 WORDS)

SPEAKER SET	RECOG. RATE (%)
TRAINING SPEAKER 1	98.45
TESTING SPEAKER 1	97.23

Open Word Recognition Experiment

Another word set was also tested as open word recognition test. It has 100 words different from the 101 words used for the above experiment. It consists of 2 repetitions of the 100 words by 6 male speakers. The content of 6 speakers is 3 training speakers out of 15 training speakers and 3 testing speakers out of 5 testing

speakers (training speaker set 2 and testing speaker set 2 as seen in Fig. 3).

TABLE 3. WORD RECOGNITION RESULTS (100 WORDS)

SPEAKER SET	RECOG. RATE (%)
TRAINING SPEAKER 2	98.00
TESTING SPEAKER 2	97.50

Incidentally, the word recognition rate was calculated with the data of the 101 words only by the 6 speakers which are same as those of this experiment. The results are shown in Table 4.

TABLE 4. WORD RECOGNITION RESULTS (101 WORDS)

SPEAKER SET	RECOG. RATE (%)
TRAINING SPEAKER 2	98.35
TESTING SPEAKER 2	97.69

As seen in the tables, over 97% recognition rate is achieved with testing speakers and over 98% for training speakers. Since there are few differences between the results of 101 word-set and those of 100 word-set, it can be said that the system has vocabulary independence.

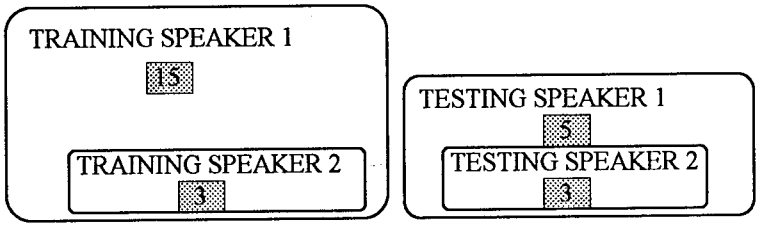


FIGURE 3. SPEAKER SET
SHOWS NUMBER OF SPEAKER

CONCLUSION

A speech recognition system has been developed which achieves 97% isolated-word, speaker-independent recognition for an untrained word vocabulary. A significant feature of this system is that the vocabulary templates are represented as phoneme sequences, which enables the capability to easily change the vocabulary.

REFERENCES

[1] H. Sakoe and S.Chiba, "Dynamic programming algorithm optimization for spoken word recognition.", IEEE Trans. Acoust. Speech, Signal Processing,

ASSP-26, pp.43-49, Feb. 1978.

[2] H. Ney, "The use of a one-stage dynamic programming algorithm for connected word recognition", IEEE Trans. Acounst., Speech & Signal Process., ASSP-32, 2, pp. 263-271, 1984

[3] T. J. Sejnowski and C. R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text", Complex Systems, 1, pp. 145-168, 1987.

[4] A. Waibel et al., "Phoneme recognition using Time-Delay Neural Networks." IEEE Trans. on Acoustics, Speech and Signal Processing. vol.37, No. 3, pp.328-339. March 1989.

[5] N. Suda et al., "Text to Speech Synthesizer Based on Residual Wave Excitation of Acoustic Tube Model", Spring meeting of Acoustical Society of Japan, pp. 215-216, March 1990 (in Japanese).

[6] T. Skinner et al., "Massively parallel DSP technology as applied to speech Recognition.", DSP Applications, pp. 29-36, August 1993.

A MONOLITHIC SPEECH RECOGNIZER BASED ON FULLY RECURRENT NEURAL NETWORKS

Klaus Kasper, Herbert Reininger, Dietrich Wolf and Harald Wüst
Institut für Angewandte Physik, der J.W.Goethe-Universität Frankfurt
Robert-Mayer-Straße 2-4, D-60054 Frankfurt a.M., FRG
Tel.: +49 69 798 3490, Fax: +49 69 798 8510
e-mail:kasper@apx00.physik.uni-frankfurt.de

Abstract. In this contribution we report about investigations concerning the application of fully recurrent neural networks (FRNN) for speaker independent speech recognition. In a phoneme based recognition system separate FRNN are used for feature scoring as well as for compensating variations in time durations of speech segments. A recognizer with a FRNN for feature scoring achieves the same recognition rate as a recognition system where the context information is provided. The performance of the FRNN used for time alignment is comparable to that of a viterbi based alignment with durational constraints. Additionally, a monolithic speech recognizer is realized by FRNN which directly classifies feature sequences. The performance of this FRNN is comparable to that of speech recognition systems which are based on discrete Hidden Markov Models and use a sophisticated durational modeling. Furthermore, simulation experiments revealed that FRNN are able to extract relevant information for speech recognition from noise contaminated speech and thus achieve a robust recognition performance.

1. INTRODUCTION

Speech recognition sytems (SRS) are faced with two basic problems. Exploiting of contextual information between feature vectors during the feature scoring and compensation of variations in time durations of speech segments.

In SRS feature scoring is the process of assigning a feature vector likelihood values characterizing its belonging to the phoneme or word categories. The more contextual information about a feature vector is taken into account in this process the more uniquely the likelihoods indicate a specific category. Speech recognizers based on Hidden Markov Models use time derivatives of feature vectors to incorporate contextual information. SRS with artificial neural networks (ANN), like time-delay neural networks (TDNN) [1, 2], use a sliding window containing several consecutive feature vectors as input and delayed versions of hidden layer activities in order to exploit dependencies of these vectors.

The variations in time durations of speech units are usually compensated during the calculation of word hypotheses by means of a dynamic

0-7803-2026-3/94 $4.00 © 1994 IEEE

programming method which is usually the viterbi algorithm. In the time alignment via viterbi algorithm the feature scores are mapped onto the phoneme sequence defining a word in such a way that the accumulated scores achieve a maximum value.

As a consequence, the different algorithms for feature scoring and the time alignment lead to a heterogeneous structure in realization of SRS. In this contribution we report about investigations to realize SRS with FRNN. FRNN is the most general type of recurrent network because all neurons are connected to all other neurons and to itself. Therefore, the performance of FRNN is not limited due to structural constraints. Two different FRNN based recognition approaches are investigated. Firstly, a phoneme based recognizer in which the feature scoring as well as the time alignment is performed by FRNN. The performance of the FRNN used for feature scoring is compared to that of a TDNN with optimized delay structure in order to evaluate the capability of FRNN to extract contextual information. The performance of the time alignment network FRNN is compared to that of viterbi alignment procedures including different types of phoneme duration modeling. Secondly, a monolithic SRS consisting of a FRNN which directly classifies feature vector sequences and thus combines feature scoring and time alignment is presented. The robustness of this FRNN against additive noise contaminating a speech signal is discussed in comparison to SRS based on discrete Hidden Markov Models.

2. FULLY RECURRENT NEURAL NETWORKS

Time discrete FRNN are neural networks with dynamic behaviour. Because of the fully connected recurrent structure, each connection has a minimum time delay of one time step. With each input pattern $\underline{x}(t)$ the activities of all neurons are updated and an output pattern is emitted.

To distinguish between different types of neurons, the indices of the input neurons are denoted as $\mathcal{I}$, the indices of the internal neurons as $\mathcal{U}$, and the indices of the output neurons as $\mathcal{O}$. The activity of neuron j at time $t+1$ can be calculated as

$$h_j(t+1) = \sum_{i \in \mathcal{U} \cup \mathcal{I}} w_{ij} x_i(t), \tag{1}$$

$$x_j(t+1) = F_j(h_j(t+1)) \quad . \tag{2}$$

with $\mathbf{W} = \{\mathrm{w_{ij}}\}$ denoting the weigth-matrix, F_j a differentiable activation function and $x_i(t)$ the activity of neuron i at time t.

The dynamic behaviour of a FRNN up to time t can be equivalently described by a feedforward multi-layer-perceptron (MLP) with t layers [3]. Therefore, the weights w_{ij} can be calculated using a modified version of the gradient descent algorithm error back-propagation (EBP). In contrast to MLP-networks, the input, the output, and the target patterns

are functions of time t. The time parameter t is equivalent to the layer index l of a MLP. EBP through the layers l can be interpreted as an "error backpropagation through time" (BPTT) [4].

The error-function which has to be minimized is the sum of the quadratic errors of the network over the time period t_a to t_e, i.e.

$$\varepsilon(t_a, t_e) = \sum_{t=t_a+1}^{t_e} E(t) = \sum_{t=t_a+1}^{t_e} \frac{1}{2} \sum_{k \in \mathcal{O}} (z_k(t) - x_k(t))^2, \tag{3}$$

where $z_k(t)$ denotes the desired target function.

In BPTT the weights are changed in direction of the error-gradient

$$\begin{aligned} \Delta w_{ij} &= -\eta \frac{\partial \varepsilon(t_a, t_e)}{\partial w_{ij}} \quad ; \quad \eta > 0 \\ &= -\eta \sum_{t=t_a+1}^{t_e} \frac{\partial E(t)}{\partial w_{ij}} \end{aligned}$$

For $t = t_e$ the gradient can be written as

$$\frac{\partial E(t_e)}{\partial w_{ij}} = -e_j(t_e) F_j'(h_j(t_e)) x_i(t_e - 1) \quad , \tag{4}$$

with

$$e_k(t) = \begin{cases} z_k(t) - x_k(t) & k \in \mathcal{O} \\ 0 & k \notin \mathcal{O} \end{cases}$$

and

$$\delta_j(t_e) = e_j(t_e) F_j'(h_j(t_e)). \tag{5}$$

The weight change arising at time t_e can be written as

$$\Delta w_{ij}(t_e) = \eta \delta_j(t_e) x_i(t_e - 1) \quad . \tag{6}$$

For earlier time steps two forms of errors have to be considered. First,

$$\delta_j^s(t) = \sum_{k \in \mathcal{V}_j} w_{jk} \delta_k(t+1) \cdot F_j'(h_j(t)) \quad , \tag{7}$$

the error which corresponds to the error in the hidden layers of a MLP and second, the error resulting from the output pattern which is given by

$$\delta_j^z(t) = e_j(t) F_j'(h_j(t)) \quad . \tag{8}$$

Denoting the sum of these two error components as

$$\begin{aligned} \delta_j(t) &= \delta_j^s(t) + \delta_j^z(t) \\ &= e_j(t) F_j'(h_j(t)) + \sum_{k \in \mathcal{V}_j} w_{jk} \delta_k(t+1) F_j'(h_j(t)), \end{aligned}$$

an analog weight change formular for times $t_a < t < t_e$

$$\Delta w_{ij}(t) = \eta \delta_j(t) x_i(t-1)$$

results. Starting at time $t = t_e$, the actual weight change can recursively be calculated for all times $t < t_e$. Applying BPTT iteratively leads to a decrease of (3) till a minimum is reached.

For efficient implementation of BPTT the gradient calculation is limited to a time-period t_r and BPTT is initiated only once after every $t_v < t_r$ time steps. In this case, $\delta_j(t)$ has to be calculated according to

$$\delta_j(t) = \begin{cases} e_j(t) F_j'(h_j(t)) & ; t = t_B \\ e_j(t) F_j'(h_j(t)) + \sum_{k \in \mathcal{V}_j} w_{jk} \delta_k(t+1) F_j'(h_j(t)) & ; t - t_v < t < t_B \\ \sum_{k \in \mathcal{V}_j} w_{jk} \delta_k(t+1) F_j'(h_j(t)) & ; t_B - t_r < t \leq t_B - t_v \end{cases}$$

where t_B denotes the time when a BPTT has been initiated [5]. The gradient resulting in this so called truncated BPTT is an approximation to the real gradient. The discrepancy depends on the parameters t_r and t_v. Therefore, t_r and t_v have to be optimized empirically for each task.

3. SPEECH DATA

The system vocabulary consists of the 10 German digits, the word *Zwo*, and 12 telephone command words. The speech signals were limited to telephone bandwidth and sampled with 8kHz. From these signals feature vectors were extracted every 12ms, each consisting of 12 cepstral coefficients derived from LPC parameters.

In the case of phoneme based recognizers for training of the network parameters the hand-labeled feature vectors of 50 utterances of each word, from different male and female speakers, were used. In the case of word based recognizers for the computing of the SRS parameters, feature vectors from 100 utterances of each word spoken from different male and female speakers, were used. In both cases speaker independent recognition rates were measured on the same disjunct set containing 100 utterances of each word from speakers not included in the training set.

4. FRNN FOR FEATURE SCORING AND TIME ALIGNMENT

First, feature vector scoring with FRNN was investigated. Networks with two different input constellations were considered. In the network FRNN1 an input pattern consists of a single feature vector, while in

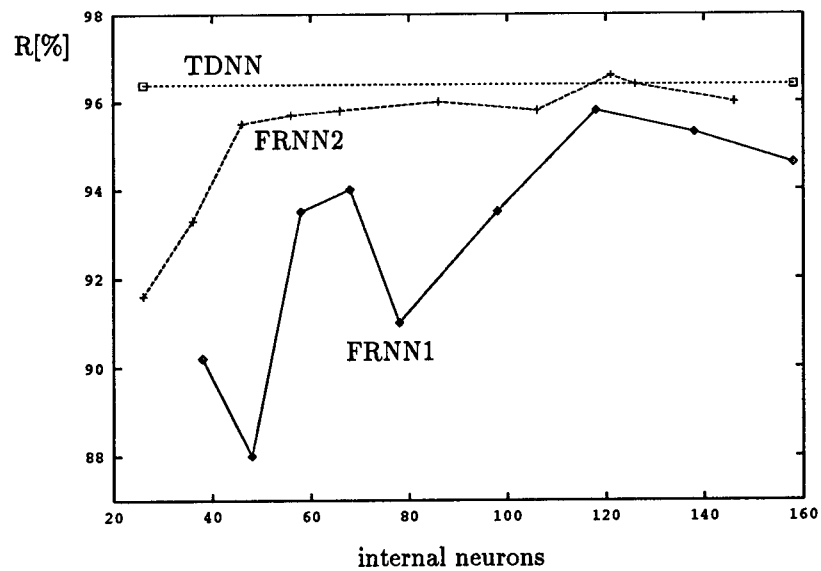


Figure 1: Recognition results for FRNN with different number of internal neurons in comparison to TDNN

the network FRNN2 two consecutive feature vectors are combined to an input pattern. The networks have 24 output neurons, one for each phoneme category. For an input pattern the target output pattern used in the training of the connection weights contains a 1 in the place of the correct phoneme and a 0 in every other place.

To evaluate the capability of FRNN to exploit automatically contextual information, the recognition rates of the FRNN based SRS were compared with the rates of a SRS using TDNN with optimized delay structure. Word recognition rates were measured by feeding the output activities into a viterbi based time alignment including minimum phoneme duration constraints. The TDNN use a window of 3 consecutive feature vectors as input and 40 hidden neurons, which activities together with the delayed activities, were connected to 24 output neurons.

Figure 1 shows the recognition results of SRS with FRNN1 and FRNN2 for different numbers of internal neurons and for comparison the rate obtained with TDNN. As can be seen, using FRNN1 with about 115 neurons for scoring of the feature vectors yields a recognition rate comparable to that achieved with TDNN. For adjusting the 15500 connection weights of FRNN1 150 training epochs were needed. In comparison to the TDNN, for which 2000 training epochs were necessary to optimize the 3400 connection weights, the training time is reduced by the factor

Table 1: Word recognition rates (WR) for different realizations of time alignment

Alignment	WR[%]
TAFRNN	96.9
VIT	96.0
VITMD	96.9
VITPDM	97.8

of 2. With FRNN2 the recognition rates of the SRS with FRNN1 or TDNN for feature scoring are achieved with only 45 neurons. This result indicates that in FRNN1 some neurons are required for storing input information. Increasing the number of neurons in FRNN2 increases the recognition rate only slightly. The highest recognition rate of 96.9% was achieved by a network with 121 neurons after 200 epochs of training. Due to the decimation of the number of input vectors by a factor of 2, the training time of FRNN2 is also reduced by a factor of 2 as compared to FRNN1. These results indicate that both FRNN variants are able to extract contextual information automatically.

In further simulation experiments it was investigated whether a viterbi alignment procedure could be replaced by a FRNN. This time alignment network, further denoted as TAFRNN, was trained to map the feature scores onto word scores. Therefore, the TAFRNN has 24 input neurons in which the phoneme scores of FRNN2 were fed in and 11 output neurons, one for each word of the system vocabulary. Word hypotheses were generated by accumulating the activities of the output units over the duration of an utterance.

In order to evaluate the performance of TAFRNN, viterbi based alignment procedures using different types of phoneme duration modeling were simulated. VIT is a time alignment module consisting of an unconstraint viterbi algorithm. In VITMD minimum phoneme durations are forced by transition constraints applied to the viterbi-path. VITPDM includes a sophisticated phoneme duration modeling. Each phoneme is modeled with a markov chain. The transition probabilities between the states of a markov chain are chosen in accordance with the duration distribution of the corresponding phoneme [6].

As can be seen from Table 1, using TAFRNN with 120 neurons a recognition rate of about 97% is obtained. TAFRNN outperforms VIT and achieves a result comparable to that of VITMD. Only VITPDM achieve with about 98% a higher rate than TAFRNN.

The results of these simulation experiments indicate that the task of

time alignment in an SRS can be accomplished with FRNN. However, further investigations are necessary to see what network sizes have to be considered for larger system vocabularies.

5. SPEECH RECOGNITION ON THE BASIS OF MONOLITHIC FRNN

5.1 Configuration and Performance of a Monolithic Recognizer

In a second set of simulation experiments it was investigated whether FRNN are able to combine feature scoring and time alignment in a single network, further denoted as MFRNN. In these experiments the system vocabulary was extended to 23 words which arise in a telephone task. To evaluate the performance of FRNN, two SRS (D1, D2) based on discrete Hidden Markov Models were realized . While in the case of D1 12 weighted Cepstral coefficients are used as feature vectors, which are quantized using 128 codebook vectors, in the case of D2 12 Delta coefficients, which are also quantized with 128 codebook vectors, are used in addition to the cepstral coefficients. A word model consists in both systems of 5 states each with 5 substates. This allows a sophisticated modeling of state duration and is comparable to VITPDM.

The MFRNN was trained to estimate word probabilities for each input pattern consisting of five consecutive feature vectors. As in TAFRNN word hypotheses were generated by accumulating the word probabilities represented by the activities of the output units over the duration of an utterance. Therefore, MFRNN has 60 input neurons and 23 output neurons. Experiments concerned with the optimization of network size revealed that about 160 internal neurons are sufficient to manage the task. This MFRNN achieves a recognition rate of 96.7% and thus outperforms D1 which achieves only 95.2% on the same task. Obviously, MFRNN is able to exploit automatically information about the dynamic of the feature vectors. This is also confirmed by the fact that D2, which uses DCep for representing dynamical information of the feature vectors, achieves with 97.3% about the same recognition rate as MFRNN.

In Figure 2 the activities of the output neurons representing the digits are shown during recognition of the utterances *Zwei, Eins, Neun, Null, Sieben,* and *Sieben.* This string of digits is arranged in order to investigate the capability of MFRNN for continuous speech recognition. The word *Zwei* and the word *Eins* share the same phonetic unit */aI/* at their boundary as well as the words *Neun* and *Null* share */n/*. As can be seen from Figure 2, at both boundaries the output neuron belonging to the preceding word shows a high activity for a small number of input patterns. Nevertheless, the activity of the output neuron belonging to the right word is dominant. The last word pair consists of two utterances of the word *Sieben.* Obviously, this is a hard task for MFRNN because

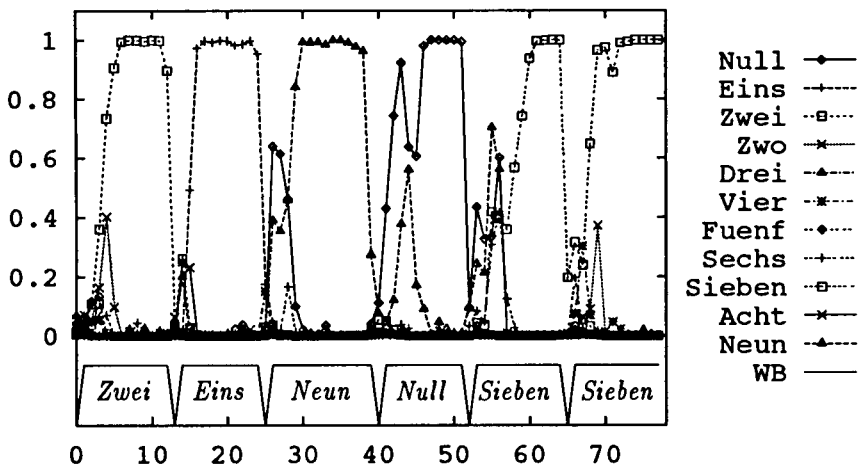


Figure 2: Activities of the output neurons representing the digits during recognition of the utterances *Zwei, Eins, Neun, Null, Sieben, Sieben* with MFRNN

it could be expected that the output neuron corresponding to the word *Sieben* will show an activity near the value 1 during both utterances. As can be seen from Figure 2, the curve representing the activity of the neuron belonging to the word *Sieben* has a minimum at the boundary which is remarkable because MFRNN receives no explicit information about word boundaries. These results indicate that MFRNN is a promising tool for continuous speech recognition.

5.2 Evaluation of the Monolithic FRNN Recognizer in Noisy Environments

In order to assess the robustness of MFRNN utterances distorted by additive white gaussian noise (WGN), office noise (ON) recorded at a big computer fair, and running car noise (RCN) recorded in the cabin of a running car at different speed and driving situations were recognized. The noise was added such that a signal-to-noise ratio (SNR) of 0dB, 10dB, and 20dB for each word resulted. As can be seen from Table 2, the recognition performance of MFRNN break down when confronted with noisy speech while D1 and D2 are more robust. Especially, for speech signals distorted by ON or RCN with a SNR of 20dB D2 achieves recognition rates of about 93%. However, for speech with additive noise at 0dB all sytems reach only very poor results.

In former work [7] it has been shown that SRS based on discrete Hidden Markov Models yield good recognition results if the model parameters were trained with noisy speech. Therefore, noise classification methods have been developed for SNR independent classification of the noise type. Using the noise classifier a robust speech recognizer can be

Table 2: Recognition rates for the speech signals of the test sequence contaminated with WGN, ON, and RCN with SNR values of 0dB, 10dB, and 20dB for non adapted and noise type adapted SRS

	non adapted			noise type adapted		
SNR	D1	D2	MFRNN	D1	D2	MFRNN
WGN						
clean	95.2	97.3	96.9	95.1	96.7	95.7
20dB	58.7	78.7	19.1	88.0	93.3	93.4
10dB	16.7	29.1	6.5	77.3	83.4	86.4
0dB	5.3	6.1	4.4	39.7	48.0	64.6
ON						
clean	95.2	97.3	96.9	94.8	96.5	94.7
20dB	91.2	94.7	34.0	93.5	96.2	93.8
10dB	62.3	73.0	9.5	79.4	87.1	83.2
0dB	18.6	25.0	3.7	31.7	38.9	35.5
RCN						
clean	95.2	97.3	96.9	93.9	96.6	95.2
20dB	91.0	95.1	31.3	93.3	95.8	93.6
10dB	59.7	79.8	6.2	80.0	89.3	84.8
0dB	21.8	34.4	5.2	32.5	46.9	52.0

realized by switching between noise type adapted SRS. Here we investigated the capability of MFRNN to adapt to a noise type by using a mixture of 40% clean speech signals and 60% noisy speech signals, containing 0dB, 10dB and 20dB signals, in the training sequence. It has to be noticed that the paramaters of the noise type adapted SRS has not been enlarged. As can be seen from Table 2, the adapted SRS show significantly improved recognition results for noisy speech while recognition rates for clean speech decrease only slightly. Especially, MFRNN outperforms D1 after adaptation. Moreover, in the case of WGN the recognition rates of MFRNN are substantially higher than those of D2. But in the case of ON or RCN the results of Table 2 show that D2, which receives explicit information about the dynamic structure of the feature vectors, achieves higher rates than MFRNN.

These experiments demonstrate the capability of MFRNN to extract relevant information for speech recognition from noise contaminated speech and thus achieve a robust recognition performance. Further experiments to train MFRNN for recognition of speech signals contaminated with different noise types as well as different SNR levels show promising results.

6. CONCLUSIONS

The investigations show that FRNN are trainable whithin reasonable training time to solve basic problems of speech recognition. It has been revealed that FRNN are able to exploit contextual information of feature vectors automatically as well as to compensate the variations in time durations of speech segments. Furthermore, it has been shown that incorporating feature scoring and time alignment in a single FRNN is a adequate concept for realizing a monolithic SRS for small vocabularies. The monolithic FRNN achieves recognition rates which are comparable to SRS based on discrete Hidden Markov Models using Cep and DCep feature vectors and a sophisticated modeling of state duration. Moreover, monolithic FRNN show a great capability to adapt to a special noise type. These results indicate that FRNN are powerful tools for robust recognition of isolated words and a promising concept for recognition of continuous speech. Especially, the monolithic SRS consisting of a single FRNN is well suited for an efficient hardware implementation.

Current investigations are concentrated on using DCep feature vectors as additional input patterns in FRNN. In further investigations problems important for hardware realization of monolithic FRNN will be considered.

This work is supported by the Deutsche Forschungsgemeinschaft in the research program 'System- und Schaltungstechnik für hochgradige Parallelverarbeitung'.

REFERENCES

[1] Waibel,A., Hanazawa,T., Hinton,G., Shiano,K., and Lang,K., "Phoneme Recognition using Time-Delay Neural Networks", in Proc. IEEE Int. Conf. on ASSP, 1989.

[2] Kasper,K., Reininger,H., and Wolf,D., "Phoneme Based Isolated Word Recognition Using Neural Networks for Prediction and Classification", in Proc. EUSIPCO-92, Brussels, pp.427-430.

[3] Rumelhart,D.E., and McClelland,J.,L., Parallel Distributed Processing, Exploration in the Microstructure of Cognition, vol.1: Foundations, MIT Press,1986.

[4] Werbos,J.P., "Backpropagation Through Time: What It Does and How to Do It", Proc. IEEE, vol.78 no.10, pp.1550-1560, 1990.

[5] Williams,R.J., and Peng,J., "An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories", Neural Computation 2, pp.490-501, 1990.

[6] Nicol,N., et.al., "Improving the Robustness of Automatic Speech Recognizers Using State Duration Information", in Proc. Speech Processing in Adverse Conditions, Cannes 1992, pp. 183-186.

[7] Nicol,N., et.al., "Noise Classification using Vector Quantization", in Proc. EUSIPCO-94, Edinburgh, in print.

FUZZIFICATION OF FORMANT TRAJECTORIES FOR CLASSIFICATION OF CV UTTERANCES USING NEURAL NETWORK MODELS

B.Yegnanarayana, C.Chandra Sekhar and S.R.Prakash
Department of Computer Science and Engineering
Indian Institute of Technology, Madras, India
e-mail : chandra@iitm.ernet.in

1. INTRODUCTION

Fuzzy neural networks are known to be better classifiers than non-fuzzy neural networks in speech recognition [1][2]. In this paper we show that fuzzification of formant data of a sequence of frames in the transition region of a CV utterance improves recognition of CV utterances. Reliable spotting of CV segments in continuous speech can significantly improve the performance of a speech-to text system [3]. CV segments also form the basic speech production units in most languages, and as such carry significant information content about the message in the speech utterance. Formant transitions in the transition region of a CV segment provide important clues for recognition of stop consonant CV segments [4]. Therefore, it is necessary to obtain a suitable parametric representation of speech data in the transition region of a CV segment to be used as input to a classifier. In the next section we discuss the choice of formants as features representing the CV segments and the fuzzy nature of these features. The details of a fuzzy neural network classifier based on the ideas in [1] are discussed in Section 3. We present methods for fuzzification of formant trajectories in Section 4. We present the results of studies on recognition of CV segments using different methods of fuzzification of formant data in Section 5.

2. FUZZY NATURE OF FORMANT FEATURES

In continuous speech the same CV may occur in different contexts. Moreover, there will also be variability in speech production due to different speakers. Therefore there may be variability in the features of the utterance due to variability in speech production as well as due to context. All these factors lead to feature data that can best be described in linguistic terms, such as 'low', 'medium' and 'high', which in turn can best be expressed as values of membership functions of fuzzy sets.

It is necessary to represent the production information in the speech signal in

0-7803-2026-3/94 $4.00 © 1994 IEEE

suitable parameters or features for input to a classifier. Parameters like spectral coefficients, cepstral coefficients, etc., are likely to be influenced by the nature of signal processing as well, besides the natural variations in the production process. Variations due to signal processing operations contribute to distortion and noise, rather than fuzziness. Therefore it is preferable to consider articulatory or related acoustic parameters like formants as features representing the CV segments. Formants are relatively easier to extract compared to the articulatory parameters[5]. Formant features also reflect the dynamics of the vocal tract system in the form of formant trajectories. Therefore the formants were selected as parameters to represent the CV segments in this study.

Speaker variability is caused due to differences in the dimensions of the vocal tract systems. In order to compensate this to some extent, ratios of formants may be considered as features. Since we are considering in this study only data from two speakers, we have decided to consider only the formant values as features. Formant data is collected for successive frames of speech signal data in each CV segment.

Formants are resonances of the vocal tract system, and hence any natural variations in the shape of the vocal tract are reflected in these resonances as well. Since variability due to speech, context and speaker is all reflected in the formant trajectories, the formant data can be assumed fuzzy, and the data is fuzzified before feeding it to a neural network classifier for training and testing.

Fuzzification of formant data involves several issues. For example, one could fuzzify the features individually in the frequency and time domains. But it appears more logical if the fuzzification could be done knowing that the three formants should occur together as a set in each frame. Also the formants in successive frames are not independent. Hence this dependency should also be considered in fuzzifying the input data to the neural network classifier.

It is natural to expect that the class labels will not be crisp either, due to significant overlap of features across the different classes of CV segments. Therefore, for effective classification, it is preferable that the output classes are fuzzy. In the next section we describe a fuzzy neural network classifier that takes fuzzy input data.

3. FUZZY NEURAL NETWORK CLASSIFIER

It was shown in [1] that fuzzification of input data and the output class label data improves the classification performance of a multilayer perceptron network for recognition of vowels using formants as features. The network takes as input the values of fuzzy membership functions for each of the three formants. Each input feature F_j in quantitative form is expressed in terms of membership values to each of the three linguistic properties 'Low', 'Medium' and 'High'. The membership function is used to assign membership values for the input features. The membership function in one-dimensional form, with range [0,1], is defined as given below.

$$\pi(x{:}c, r) = \begin{cases} 2(1-(\,|\,x-c\,|/r))^2, & \text{for } r/2 \le |\,x-c\,| \le r, \\ 1-2\,(\,|\,x-c\,|/r)^2, & \text{for } 0 \le |\,x-c\,| \le r/2, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where x is a pattern point, r is the radius of the π function and c is the central point.

The fuzzy sets for the linguistic properties 'Low', 'Medium' and 'High' for each formant are represented by membership functions π_L, π_M and π_H respectively. The parameters of these membership functions are defined below.

Let F_{jmax} and F_{jmin} be the upper and lower bounds of feature F_j in all pattern points. For the three linguistic property sets, parameters are defined as

$$r_M(F_j) = 0.5\,(F_{jmax} - F_{jmin}) \tag{2a}$$

$$c_M(F_j) = F_{jmin} + r_M(F_j) \tag{2b}$$

$$r_L(F_j) = (c_M(F_j) - F_{jmin})/\text{fdenom} \tag{2c}$$

$$c_L(F_j) = c_M(F_j) - 0.5\,r_L(F_j) \tag{2d}$$

$$r_H(F_j) = (F_{jmax} - c_M(F_j))/\text{fdenom} \tag{2e}$$

$$c_H(F_j) = c_M(F_j) + 0.5\,r_H(F_j) \tag{2f}$$

where 'fdenom' is a parameter controlling the extent of overlapping.

The three π membership functions are defined for each of the three formants and for each of the N frames in the transition region of a CV segment. Thus a CV segment is represented by an Nx9-dimensional matrix of membership values. Such Nx9-dimensional patterns derived from formant feature vectors of CV segments are used as input to a neural network classifier.

During the training phase, the desired output vector is expressed as the desired membership values, lying in the range [0,1]. To obtain these membership values, the distance of a training pattern **F** from the average pattern $\mathbf{O}_k$ for the kth class is defined as

$$z_k = \sqrt{\frac{1}{9*N} \sum_{i=1}^{N} \sum_{j=1}^{9} \left(F(i,j) - O_k(i,j)\right)^2} \tag{3}$$

The membership value for the training pattern **F** to the kth class is defined as

$$d_k(F) = \frac{1}{1+\left(\frac{z_k}{f_d}\right)^{fe}} \tag{4}$$

where the positive constants f_d and f_e control the amount of fuzziness in the class-membership set. The desired output vector for a training pattern is obtained by computing the membership values for the pattern to each of the classes and used in training the multilayer perceptron network.

In fuzzification of the input data, the formant features for each frame are fuzzified independently. But, there is a sequence of frames in each CV segment, and the data in each frame depends to some extent on the adjacent frames. This fact must be used in fuzzification of formant trajectories. Two methods of fuzzification of sequences of formant data are presented in the next section.

4. FUZZIFICATION OF FORMANT TRAJECTORIES

The formant data for one frame is dependent on the adjacent frames. This time-dependency can be incorporated in the fuzzification of the trajectories by reducing the variability allowed for the subsequent frames given the variability of the current frame. The reduction in variability allowed for subsequent frames can be realized by decreasing the radii of the membership functions for fuzzy subsets of features in those frames, and correspondingly modifying the centers of the functions.

The parameters of the membership functions for the features in the first frame are defined as in (2). The parameters of the functions for subsequent frames are obtained from those of the first frame as follows:

$$r_{iM}(F_j) = (1 - \alpha) * r_{(i-1)M}(F_j) \tag{5a}$$

$$c_{iM}(F_j) = F_{jimin} + ß * r_{iM}(F_j) \tag{5b}$$

$$r_{iL}(F_j) = (1 - \alpha) * r_{(i-1)L}(F_j) \tag{5c}$$

$$c_{iL}(F_j) = c_{iM} - 0.5 * ß * r_{iL}(F_j) \tag{5d}$$

$$r_{iH}(F_j) = (1 - \alpha) * r_{(i-1)H}(F_j) \tag{5e}$$

$$c_{iH}(F_j) = c_{iM} + 0.5 * ß * r_{iH}(F_j) \tag{5f}$$

where i is the frame number and $2 \leq i \leq N$. The constants α and ß are chosen such that the distance between the average patterns of the classes is maximum. Typical values for the constants α and ß are 0.075 and 1.30, respectively.

Another way of incorporating the time dependency is to use multi-dimensional membership functions for groups of adjacent frames. The parameters for the multi-dimensional membership functions are obtained from the parameters of the

one-dimensional membership functions of features for individual frames. The definition of one-dimensional function in (1) is extended for an n-dimensional function of a group of n adjacent frames as given below:

$$\pi(\mathbf{x}:\mathbf{c},r) = \begin{cases} 2(1-(\|\mathbf{x}-\mathbf{c}\|/r))^2, \text{ for } r/2 \le \|\mathbf{x}-\mathbf{c}\| \le r, \\ 1-2(\|\mathbf{x}-\mathbf{c}\|/r)^2, \text{ for } 0 \le \|\mathbf{x}-\mathbf{c}\| \le r/2, \\ 0, \text{ otherwise,} \end{cases} \tag{6}$$

where **x** is the vector of values of a feature in n adjacent frames, **c** is the mean vector of x's for all patterns, and r is the radius of the n-dimensional function. The radius of the n-dimensional function is obtained from the radii, r_i, of one-dimensional functions of feature in individual frames.

$$r = \sqrt{\sum_{i=1}^{n} r_i^2} \tag{7}$$

A two-dimensional π function was used in our studies on recognition of CV segments in continuous speech. We present the effects of the methods of fuzzification on the performance of a classifier for CV segments in the next section.

5. STUDIES ON RECOGNITION OF CV UTTERANCES

Speech data for the studies described in this section was collected from utterances of several sentences in Hindi (an Indian language) spoken by two male speakers. From these utterances, occurrences of CV segments are manually excised by visual inspection of the speech signal waveform and by careful listening of the segmented data. Data for the following 9 CV classes have been collected: /ka/, /ke/, /ko/, /ga/, /ta/, /to/, /dha/, /pa/ and /ba/. The choice of these classes was mostly dictated by the availability of sufficient numbers of these segments in the speech data collected for several sentences.

For each CV segment only a fixed 40 msec portion around the vowel onset point was considered. This portion generally reflects the transition of the vocal tract system from the place of articulation corresponding to the consonant position to the shape of the vocal tract corresponding to the following vowel, including some steady vowel part. Formants were extracted using linear prediction analysis for each frame of size 128 samples at 10 kHz sampling rate, with a shift of 32 samples. The formant contours were hand edited and smoothed to remove spurious peaks. From the resulting smooth contours the first three formants were obtained for each of the 10 frames in a CV segment.

The formant data is fuzzified using methods discussed in Sections 3 and 4. Thus for each CV segment, a 90-dimensional vector of membership values is generated. This representation is used as input to the classifier. The desired output data is also

fuzzified as discussed in Section 3 for training the neural network classifier.

The classifier is a multilayer feedforward network trained using back propagation algorithm. Three hidden layers were used in the network. The number nodes in each of the hidden layers was chosen as 50. A total of 150 patterns belonging to 9 CV utterance classes were used for training the network using backpropagation algorithm. A total of 150 patterns were used as test data. The classification performance on test data for different methods of fuzzification is given in Table_1. The performance is given for two cases of deciding the correct class: (1) correct class is the class with the highest output and (2) correct class is amongst the classes with the highest and the second highest outputs.

TABLE_1: COMPARISON OF CLASSIFICATION PERFORMANCE FOR DIFFERENT FUZZIFICATION METHODS

Fuzzification Method	Case1	Case2
Non-fuzzy inputs	29.5	46.3
Fuzzification of individual frames	62.9	82.1
Fuzzification by variability reduction	70.2	84.8
Fuzzification using 2-dimensional function	73.5	85.4

6. CONCLUSIONS

The studies reported in this paper show that fuzzification of input and output data improves the recognition accuracy of CV segments. In particular, fuzzification of input data taking into account the fact that the formant data is for a sequence of frames, improves the recognition of CV segments significantly. In these studies only a simple method was used to implement the dependence of fuzziness on the sequence. But a more sophisticated data dependent approach for determining the fuzzy membership values for data both along frequency and along time may improve the recognition performance still further.

REFERENCES:

1. S.K.Pal and S.Mitra, "Multilayer perceptron, fuzzy sets and classification," IEEE Trans. on Neural Networks, vol.3, no.5, pp.683-697, September 1992.

2. Y.H.Kuo, C.I.Kao and J.J.Chen, "A fuzzy neural network model and its hardware implementation," IEEE Trans. on Fuzzy Systems, vol.1, no.3, pp.171-183, August 1993.

3. H.Sawai, A.Waibel, M.Miyatake and K.Shikano, "Spotting Japanese CV-syllables and phonemes using time-delay neural networks," Proceedings of ICASSP'89, pp.25-28, May 1989.

4. K.N.Stevens, "Models for production and acoustics of stop consonants," Speech Communication, vol.13, nos.3-4, pp.367-375, December 1993.

5. J.Schroeter and M.M. Sondhi, "Techniques for estimating vocal-tract shapes from the speech signal," IEEE Trans. on Speech and Audio Processing, vol.2, no.1, part II, pp.133-150, January 1994.

MINIMUM ERROR CLASSIFICATION OF KEYWORD-SEQUENCES

Takashi KOMORI*and Shigeru KATAGIRI**
* INTEC Systems Laboratory inc.,
3-23 Shimoshin-machi, Toyama 930, Japan
Tel.: +81-764-44-1111, Fax.: +81-764-44-8126
e-mail: komori@isl.intec.co.jp
** ATR Interpreting Telecommunications Research Laboratories,
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan

Abstract — A novel spotter design method, i.e., Minimum Error Classification of Keyword-Sequences (MECK), is proposed. In contrast with conventional approaches, the proposed method directly aims at reducing errors of classifying keyword-sequences (strings of prescribed keyword categories) through a mathematically proven, GPD-based optimization process. Experiments in Japanese keyword spotting tasks clearly demonstrate the utility of a MECK-trained, prototype-based spotter.

1 Introduction

The recognition of natural and spontaneous speech utterances is an important issue for realizing a user-friendly human-machine interface. Since natural speech often contains ill-conditioned phenomena such as hesitations or repetitions, it has been considered that a word-by-word modeling approach to recognition is insufficient (e.g., [1]). Recently, keyword spotting has been studied as an alternative to this conventional approach with increasing vigor [2]–[5].

The goal of spotting is to correctly spot (detect) all of the prescribed keywords included in an input utterance; in other words, to correctly classify the input as one of the possible *keyword-sequences* (strings of prescribed keyword categories). The keyword-sequence itself can be an input to a post-end process such as context modeling or semantic modeling. *Spotter* (spotting system) performance should thus be evaluated by the classification accuracy of the keyword-sequences instead of the accuracy of individual spotting decisions. However, as seen in literature [2]–[5], recent efforts have actually been made in reduction of indi-

This study was conducted while the authors worked at ATR Human Information Processing Research Laboratories.

vidual spotting decision errors and also been entailing no optimality in the sense of minimum error classification of keyword-sequences.

In light of this, we propose in this paper a novel design method for spotters, i.e., Minimum Error Classification of Keyword-sequences (MECK). Key concepts of this method are 1) to formalize the spotting process as a smooth and trainable functional form with the design objective being the keyword-sequence classification accuracy, 2) to formulate an individual keyword spotting as a two-class segmentation/classification by using the *a posteriori* odds-based discriminant function, and 3) to introduce a mathematically proven, GPD-based optimization to achieve the *optimal* (minimum keyword-sequence classification error) status of the spotter.

MECK is quite general and can be applied to any reasonable spotter structure, including artificial neural networks. By way of example, we present a prototype-based spotter whose structure has been widely used in the Learning Vector Quantization (LVQ) application, and evaluate this one in several Japanese keyword spotting tasks.

2 Definition

2.1 Problem formalization

Classification is a simple process to assign one of the possible classes to a given pattern, and it does not include a process to segment (extract) the pattern from its background's wider or larger signal. Similarly, spoken word classification is defined as a process to classify a word segment pre-segmented from a continuous speech utterance as one of the possible word classes. Obviously, for continuous speech recognition, this simple-minded classification is insufficient and an appropriate link of segmentation and classification is required. Spotting can be considered the very framework to achieve this link directly. However, in reality, these two processes are designed separately, entailing no guarantee of the resulting spotting optimality. A main effort in our formalization is therefore to embed this complicated process in a unified functional form that is suited for the use of mathematically proven optimization techniques.

For clarity of presentation, in addition to the term of keyword-sequence classification (*word-sequence classification* for short), we define here the following two terms: 1) *keyword-sequence spotting* (*word-sequence spotting* for short) being used to decide whether a sequence of keywords is included in a given utterance and the location of these keywords, and 2) *keyword-spotting* (*word-spotting* for short) being used to decide whether a keyword exists in a preset segment.

Assume that our task is to classify a given speech utterance X as one of C possible keyword-sequence classes, each consisting of only prescribed keyword names. Each utterance is represented in the form of an acoustic feature vector sequence; $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_I\}$. We denote the entire set of prescribed keywords by $W = \{w_1, \ldots, w_k, \ldots, w_K\}$ and the entire set of possible keyword-sequence classes by $\Omega = \{\Omega_1, \ldots, \Omega_c, \ldots, \Omega_C\}$.

Let us focus on the k-th word w_k's spotting decision in the segment $X_s^e = \{x_s, x_{s+1}, \ldots, x_{e-1}, x_e\}$. We denote this decision for w_k as a_{kse}. The a_{kse} is a kind of indicator function that becomes one (1) for a correct decision and zero (0) for an incorrect decision. This indicator function can be defined in principle for all s, e, and k. Nevertheless, due to several realistic restrictions on combinations of s, e, and k, the functions are defined in a set of limited cases, which is denoted by $\mathcal{G}$.

In this view, one word-sequence classification decision can be considered a sequence of word spotting decisions $\alpha = \{a_{kse}\}$, each are included in a word-sequence class, e.g., Ω_c. Therefore, the goal of the spotter design should be to achieve a state of adjustable spotter parameters, denoted by Λ, that emulates the following Bayesian decision theory-based rule [6]:

$$c(X) = \hat{c} \quad \text{if } \hat{c} = \arg\max_c \Pr(\Omega_c \mid X), \tag{1}$$

where $\Pr(\Omega_c \mid X)$ is the *a posteriori* probability of the word-sequence class Ω_c given X, and $c(X)$ denotes the operation of making a word-sequence classification decision. Note that this $c(X)$ is known to lead to the minimum error classification.

To define our design algorithm in a practical and effective fashion, we embody several operations/concepts in a mathematical form.

First, we approximate the *a posteriori* probability of a word-sequence class $\Pr(\Omega_c \mid X)$ by the *a posteriori* probability of the dominant (most probable) word-sequence spotting in the class:

$$\Pr(\Omega_c \mid X) \approx \max_{\alpha \in \mathcal{A}_c} \Pr(\alpha \mid X), \tag{2}$$

where $\mathcal{A}_c (\subset \mathcal{G})$ is a set of word-sequence spotting decisions, each corresponding to Ω_c, and $\Pr(\alpha \mid X)$ is the *a posteriori* probability of α given X. Accordingly, (1) becomes equivalent to

$$c(X) = \hat{c} \quad \text{if } \arg\max_{\alpha \in \mathcal{A}_{\hat{c}}} \Pr(\alpha \mid X) = \hat{\alpha}, \tag{3}$$

where

$$\hat{\alpha} = \arg\max_{\alpha \in \mathcal{G}} \Pr(\alpha \mid X). \tag{4}$$

This rule more closely represents an actual spotting procedure.

Second, on the assumption that all of the individual *a posteriori* probabilities of w_k's existence in X_s^e, i.e., $\Pr(w_k \mid X_s^e)$'s, are independent of each other, we represent $\Pr(\alpha \mid X)$ as

$$\Pr(\alpha \mid X) = \prod_{k,s,e} \Pr(w_k \mid X_s^e)^{a_{kse}} \{1 - \Pr(w_k \mid X_s^e)\}^{1-a_{kse}}, \tag{5}$$

which is rewritten as

$$\ln \Pr(\alpha \mid X) = \sum_{k,s,e} a_{kse} \ln \frac{\Pr(w_k \mid X_s^e)}{1 - \Pr(w_k \mid X_s^e)} + \sum_{k,s,e} \ln\{1 - \Pr(w_k \mid X_s^e)\}. \tag{6}$$

Now, $\Pr(\alpha \mid X)$ is represented by a set of more elemental *a posteriori* probabilities. However, in practice, even these *a posteriori* probabilities are rarely known and thus we must further replace these probabilities with some proper estimate that is a function of Λ. Note here that the first term on the right-side of (6) includes the *a posteriori* odds

$$O(w_k \mid X_s^e) = \frac{\Pr(w_k \mid X_s^e)}{1 - \Pr(w_k \mid X_s^e)} \tag{7}$$

that have been widely used in artificial intelligence and statistics. Naturally, we then use a *keyword possibility score*, denoted by $\eta_\Lambda(w_k \mid X_s^e)$, as the estimate of the logarithmic *a posteriori* odds, which is a function of Λ. Moreover, to find $\hat{\alpha}$, we can ignore the second term of the right side of (6) that always takes a constant value. Therefore, we can simplify (4) to

$$\hat{\alpha} = \arg\max_{\alpha \in \mathcal{G}} Y_\Lambda(\alpha \mid X), \tag{8}$$

where

$$Y_\Lambda(\alpha \mid X) = \sum_{k,s,e} a_{kse} \eta_\Lambda(w_k \mid X_s^e), \tag{9}$$

and accordingly rewrite (1) to

$$c(X) = \hat{c} \quad \text{if } \hat{c} = \arg\max_c g_\Lambda^c(X), \tag{10}$$

where $g_\Lambda^c(X)$ is defined as a generalized discriminant function for Ω_c, i.e.,

$$g_\Lambda^c(X) = \frac{1}{\xi} \ln \left\{ \frac{1}{|\mathcal{A}_c|} \sum_{\alpha \in \mathcal{A}_c} \exp(\xi Y_\Lambda(\alpha \mid X)) \right\}, \tag{11}$$

with ξ being a positive constant and $Y_\Lambda(\alpha \mid X)$ being referred to as a *word-sequence spotting score* for α. Note that $g_\Lambda^c(X)$ expresses an aggregate possibility that X includes Ω_c.

Consequently, $g_\Lambda^c(X)$ is used in place of $\Pr(\Omega_c \mid X)$; similarly, (10) is used in place of (1). Now it turns out that finding the optimal status of Λ is our design target.

As in conventional classifier designs, there are two main approaches to the design of Λ: 1) the maximum-likelihood design, and 2) the discriminant function method. Taking account of the findings of MCE/GPD studies [7],[8], we chose to use the second approach. Therefore, we next define the loss function

$$\ell(X\,;\Lambda) = 1\left(-g_\Lambda^{c^*}(X) + \max_{c \neq c^*} g_\Lambda^c(X)\right), \tag{12}$$

where

$$1(x) = \begin{cases} 0, & \text{if } x < 0 \\ 0.5, & \text{if } x = 0 \\ 1, & \text{if } x > 0, \end{cases} \tag{13}$$

and c^* is the correct word-sequence class index. This loss is zero (0) for a correct word-sequence classification; one (1) for correct. This loss represents an ideal error count but is discontinuous in Λ and causes mathematical problems in formalization as discussed in [7]. In our approach based on the MCE/GPD concept, we therefore use a smooth loss defined as

$$\ell(X\,;\Lambda) = \tilde{1}\left(-g_\Lambda^{c^*}(X) + \frac{1}{\zeta}\ln\left\{\frac{1}{C-1}\sum_{c\neq c^*}\exp(\zeta g_\Lambda^c(X))\right\}\right), \quad (14)$$

where $\tilde{1}(\cdot)$ is a smooth step function such as $\tilde{1}(x) = (1+\exp(-x/\varsigma))^{-1}$ with ς being a positive constant and ζ a positive constant.

In principle, the loss must be evaluated over all of the possible samples. We thus introduce the expected loss

$$L(\Lambda) = E_X\left[\ell(X\,;\Lambda)\right] \quad (15)$$

as the design objective to be minimized, where E_X is the expectation over the X-space.

2.2 Computation reduction

The design problem is now formalized as the minimization problem of the expected loss. However, full computation of $Y_\Lambda(\alpha\,|\,X)$ is hopelessly time-consuming. There is a clear need for reduction of the computation. A natural way of such attempt is to focus the computation on plausible word-spotting decisions (remove probably incorrect decisions beforehand). In light of this, we introduce a pruning function $\omega_\Lambda(w_k\,|\,X_s^e)$ that indicates zero (0) when the corresponding word-spotting decision should be ignored and one (1) otherwise. Then $Y_\Lambda(\alpha\,|\,X)$ is replaced with the following practical version of the score:

$$\tilde{Y}_\Lambda(\alpha\,|\,X) = \sum_{k,s,e} a_{kse}\left\{\eta_\Lambda(w_k\,|\,X_s^e) + \ln\omega_\Lambda(w_k\,|\,X_s^e)\right\}. \quad (16)$$

Note that $\omega_\Lambda(w_k\,|\,X_s^e)$ being zero makes the above practical score $\tilde{Y}_\Lambda(\alpha\,|\,X)$ negative infinity, which means that the corresponding word-spotting decision does not contribute to computing the sequence-spotting score, and that $\tilde{Y}_\Lambda(\alpha\,|\,X)$ can be substituted for $Y_\Lambda(\alpha\,|\,X)$ in (10) and (11) without any serious mathematical drawback.

The choice of pruning function still affects the computation. If the number of hypotheses pruned by this function is small, computing the scores problem would be still resource-consuming because of combination explosion, especially in a large-vocabulary case. Since an attempt to solve this problem in a separate, heuristic manner, such as the beam-search algorithm, does not allow one to achieve a consistent minimization of the loss, we also consider another attempt of reduction by introducing the loss

$$\tilde{\ell}(X\,;\Lambda) = \sum_{k,s,e}\omega_\Lambda(w_k\,|\,X_s^e) \quad (17)$$

that approximates the number of keyword hypotheses being not pruned. If this number is counted as loss (cost) in the same manner as (12), one can reduce the computation by explicitly attempting to decrease this count. Consequently, incorporating this new loss, the expected loss can also be re-defined as

$$\tilde{L}(\Lambda) = E_X\,[\,\ell(X\,;\Lambda) + \gamma\tilde{\ell}(X\,;\Lambda)\,], \tag{18}$$

where γ is a controllable weighting factor.

2.3 Optimization algorithm

In accordance with the adaptive adjustment rule of GPD, we use

$$\Lambda_{t+1} = \Lambda_t - \epsilon_t U \nabla_\Lambda \ell(X_t\,;\Lambda_t), \tag{19}$$

which has been shown to lead to the optimal state of Λ that corresponds to at least the local minimum of the expected loss, where Λ_t denotes the parameter set at the t-th iteration, ϵ_t is a learning factor that satisfies $\sum_{t=1}^{\infty}\epsilon_t = \infty$ and $\sum_{t=1}^{\infty}\epsilon_t^2 < \infty$, U is a positive definite matrix, ∇_Λ is the gradient symbol with respect to Λ, and X_t denotes the t-th speech utterance given randomly for training.

3 Implementation

MECK can be applied to any reasonable spotter structure such as a prototype-based system or an HMM system. Each keyword can be either directly modeled (represented in spotter parameters) or indirectly modeled by concatenating subword models. Among these many choices, this paper specially presents an implementation example for a prototype-based spotter consisting of subword models; λ_j ($\subset \Lambda$) denotes a class j subword model consisting of a sequence of acoustical feature vectors.

3.1 Log Estimate of A Posteriori Odds

Due to our preference for using subword models, we shall first define a subword-based log *a posteriori* odds estimate. Our prototype-based spotter basically computes the distance $D(X_s^e, \lambda_j)$ between the subword model and a speech segment. We thus need to convert this distance measure to the *a posteriori* odds form. Among many possible ways of doing so, we use the following function form:

$$\eta_\Lambda(\lambda_j \mid X_s^e) = \phi_{j0} + \phi_{j1} D(X_s^e, \lambda_j), \tag{20}$$

where ϕ_{j0} and ϕ_{j1} are constants. The use of this form is motivated by the estimation of $\Pr(\lambda_j \mid X_s^e)$ using a logistic function of $D(X_s^e, \lambda_j)$. Each subword model is represented by a set of reference vectors and the distance $D(X_s^e, \lambda_j)$ is defined following McDermott's formalization [9] and our previous work [5]. A word-level log *a posteriori* odds estimate, $\eta_\Lambda(w_k \mid X_s^e)$, is then defined by accumulating $\eta_\Lambda(\lambda_j \mid X_s^e)$'s considering time warping variations. In both cases, similar to (11), smooth functional form is used. See the detail in [10].

3.2 Pruning Function

Pruning always suffers from the risk of a decrease in accuracy because it often misses some of the correct word-spotting hypotheses. Therefore, the pruning function must be designed carefully. Our actual pruning strategy is summarized as follows; i.e., $\omega_\Lambda(w_k \mid X_s^e)$ is set closer to zero, if one of the following manifolds is correct.

1. $X_{s'}^e$ ($s' \neq s$) is more likely than X_s^e for spotting w_k.
2. $X_{s'}^{e'}$ (s' is arbitrary, $e' \neq e$, and $e' \approx e$) is more likely than X_s^e for spotting w_k.
3. $\eta_\Lambda(w_k \mid X_s^e)$ is less than a preset threshold h_k.

This pruning criterion has actually been used in conventional spotting techniques using starting-end-free dynamic time warping and is not so specific one. Based on our policy of formalizing the process rigorously, we define the pruning function $\omega_\Lambda(w_k \mid X_s^e)$ as the product of three continuous auxiliary pruning functions, $\omega_\Lambda^1(w_k \mid X_s^e)$, $\omega_\Lambda^2(w_k \mid X_s^e)$, and $\omega_\Lambda^3(w_k \mid X_s^e)$, each approximating one of above three conditions in a smooth functional form, e.g.,

$$\omega_\Lambda^1(w_k \mid X_s^e) \approx 1\left(\eta_\Lambda(w_k \mid X_s^e) - \max_{s \in S_k(e)} \eta_\Lambda(w_k \mid X_s^e)\right), \qquad (21)$$

where $1(\cdot)$ is a step function defined in (13) and $S_k(e)$ is the entire set of possible beginning endpoints of keyword w_k, given the ending endpoint e. See [10] for details of $\omega_\Lambda^1(w_k \mid X_s^e)$, $\omega_\Lambda^2(w_k \mid X_s^e)$ and $\omega_\Lambda^3(w_k \mid X_s^e)$. Similar forms to (11) and a smooth step function $\tilde{1}(\cdot)$ are again used in these approximations. Consequently, the resultant pruning function $\omega_\Lambda(w_k \mid X_s^e)$ is also a continuous and differentiable function.

3.3 Simplification for Practical Use

MECK is not sufficiently easy to perform in practice yet. In our experiments described in the next section, we used following simplification techniques: 1) letting the power parameter of all L_p-norm function (e.g.,ξ in (11)) go to infinity, 2) using a piece-wise smooth step function for representing pruning function $\omega_\Lambda(w_k \mid X_s^e)$, 3) tying some parameters, and 4) using a finite, LVQ-like adjustment [11].

4 Experiment

We conducted evaluation experiments for a task of spotting samples of 10 keyword-classes, i.e., *kaigi*, *kokusai*, *denwa*, *kyouto*, *happyou*, *tsuuyaku*, *nihongo* (*nippongo*), *touroku*, *jimukyoku*, and *ronbun*, from 115 different Japanese sentences, each spoken by 10 female and 10 male speakers. Speech was converted to a 112-dimensional acoustic feature vector every 5 msec based on a spectral analysis. The sentences were divided into

Table 1: Characteristic of Speech Corpus

	Set 1	Set 2	Set 3	Set 4	Total
# Sentence	500	500	500	800	2300
# Keyword	480	620	1320	300	2720

Table 2: Initial Spotter Performance

	Set 1	Set 2	Set 3	Set 4	Total
# Correct-K-S	58	93	23	250	424
# Possible-K-S	491	489	476	791	2247
# Spotted-W	20241	15612	23147	17932	76932

4 independent sets; Sets 1–3, each consisting of 25 sentences, and Set 4 consisting of 40 sentences. Each Set included at least one sample for every keyword class. Three Sets were used for design and the remaining Set was used for testing; e.g., Set 1–3 for design and Set 4 for testing.

The reference vector set of subword models was initialized by running k-means clustering over manually selected phoneme segments. The coefficient set $\phi_j = \{\phi_{j0}, \phi_{j1}\}$ was preliminarily initialized based on the sample distribution of each cluster. Also, $\mathcal{G}$ was set so that the unrealistic overlap between adjacent word hypotheses could be eliminated. Neither grammatical nor semantic constraints were used.

Table 1 summarizes the statistics of our tasks. In the table, for each data set, "# Sentence" and "# Keyword" indicate the total number of sentences and the total number of keywords, respectively.

Table 2 shows the testing accuracies of the spotters after above initialization. Note that the accuracies for each set were obtained by a spotter designed using the other three sets of data. In the table, "# Correct-K-S" represents the number of correctly classified keyword-sequences; "# Possible-K-W" represents the number of correct keyword-sequences that remained after pruning the keywords in the spotting stage; and "# Spotted-W" represents the number of keywords that remained after pruning, i.e., keywords actually spotted. The table shows that the spotters spotted an enormous number of keywords, most of which retained the correct keyword-sequences in the beginning of the keyword-sequence classification process, and correctly classified these sequences with an accuracy range of only 5–30%.

After this initialization, we trained the spotters using the method described in 2.3. Following *learning-when-incorrect* strategy, we actually

Table 3: Spotter Accuracies after MECK Training

a) $\gamma = 0.0$

	Set 1	Set 2	Set 3	Set 4	Total
# Correct-K-S	379	361	152	695	1587
# Possible-K-S	500	495	365	800	2160
# Spotted-W	20884	13624	15184	15226	64918

b) $\gamma = 1.0 \times 10^{-3}$

	Set 1	Set 2	Set 3	Set 4	Total
# Correct-K-S	257	273	127	701	1358
# Possible-K-S	356	400	236	797	1789
# Spotted-W	2540	2426	5067	5212	15245

defined the loss $\ell(\cdot)$ in (14) using the following smooth step function:

$$\tilde{1}(x) = \begin{cases} 0, & \text{if } x < 0 \\ \dfrac{1 - \exp(-x/\varsigma)}{1 + \exp(-x/\varsigma)}, & \text{if } x \geq 0. \end{cases} \tag{22}$$

We specially chose two different conditions of simplification: 1) $\gamma = 0.0$ and 2) $\gamma = 1.0 \times 10^{-3}$.

Table 3 shows the accuracies for these two cases. The keyword-sequence classification accuracies were increased to a range of 25–87%, while "# Spotted-W's", i.e., the computation amounts, stayed at a range of 66–103% for $\gamma = 0.0$ and were reduced to a range of 13–29% for $\gamma = 1.0 \times 10^{-3}$. The results clearly proved the high utility of the proposed design method.

5 Summary

We have presented a new spotter design method, called the Minimum Error Classification of Keyword-sequences method (MECK), that is directly linked with continuous speech recognition. The method is characterized by 1) formalizing the spotting-based keyword-sequence classification (continuous speech recognition) in a quite general but rigorous manner, 2) introducing the *a posteriori* odds-based discriminant function that allows one to greatly reduce computation and to easily incorporate a wide range of artificial intelligence techniques in speech recognition, 3) making possible a spotter design that does not use labeled design samples which is necessary in conventional design methods, and 4) incorporating keyword hypothesis pruning process for unified loss minimization.

Experiments in Japanese keyword spotting tasks clearly demonstrated the marked utility of our design method.

Finally, it should be worth addressing that the design method proposed in this paper is a quite general framework useful to solve combi-

nation search problem. The method can be applied to a wide range of inference problem by selecting log *a posteriori* odds estimator functions and pruning functions adequately.

References

[1] W. Ward and S. Young, "Flexible Use of Semantic Constraints in Speech Recognition," IEEE, Proc. of ICASSP-93, Vol. 2, pp. 49–50 (1993).

[2] W. Ward, "Understanding Spontaneous Speech: The Phoenix System," IEEE, Proc. of ICASSP-91, S5.29, pp. 365–367 (1991).

[3] H. Tsuboi and Y. Takebayashi, "A Real-Time Task-Oriented Speech Understanding System Using Keyword-Spotting," IEEE, Proc. of ICASSP-92, Vol. 1, pp. 197–200 (1992).

[4] T. Zeppenfeld and A. H. Waibel, "A Hybrid Neural Network, Dynamic Programming Word Spotter," IEEE, Proc. of ICASSP-92, Vol. 2, pp. 77–80 (1992).

[5] T. Komori and S. Katagiri, "A New Learning Algorithm for Minimizing Spotting Errors," Proc. of 1993 IEEE Workshop on Neural Networks for Signal Processing, pp. 333–342 (1993).

[6] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1973).

[7] B.-H. Juang and S. Katagiri, "Discriminative Learning for Minimum Error Classification," IEEE Trans. on Signal Processing, Vol. 40, No. 12, pp. 3043–3054 (1992).

[8] S. Katagiri, C.-H. Lee, and B.-H. Juang, "New Discriminative Training Algorithms Based on the Generalized Probabilistic Descent Method," Proc. of the 1991 IEEE Workshop on Neural Networks for Signal Processing, pp.299–308 (1991).

[9] E. McDermott and S. Katagiri, "Prototype-based discriminative training for various speech units," IEEE, Proc. of ICASSP-92, Vol. 1, pp. 473–476 (1992).

[10] T. Komori and S. Katagiri, "A Novel Spotter Design Method for Minimum Error Classification of Keyword-Sequences," ATR Tech. Report, TR-H-067 (1994) (partially in Japanese).

[11] E. McDermott and S. Katagiri, "LVQ-Based Shift-Tolerant Phoneme Recognition," IEEE Trans. on Signal Processing, Vol. 39, No. 6, pp. 1398–1411 (1991).

Hybrid training method for tied mixture density hidden Markov models using Learning Vector Quantization and Viterbi estimation

Mikko Kurimo
Helsinki University of Technology
Neural Networks Research Centre
Rakentajanaukio 2 C, FIN-02150, ESPOO, FINLAND
tel: +358 0 451 3266, fax: +358 0 451 3277
email: mikko.kurimo@hut.fi

Abstract. In this work the output density functions of hidden Markov models are phoneme-wise tied mixture Gaussians. For training these tied mixture density HMMs, modified versions of the Viterbi training and LVQ based corrective tuning are described. The initialization of the mean vectors of the mixture Gaussians is performed by first composing small Self-Organizing Maps representing each phoneme and then combining them to a single large codebook to be trained by Learning Vector Quantization (LVQ). The experiments on the proposed training methods are accomplished using a speech recognition system for Finnish phoneme sequences. Comparing to the corresponding continuous density and semi-continuous HMMs in [9] and [8] in the respect of the number of parameters, the recognition time and the average error rate, the performance of the phoneme-wise tied mixture HMMs is superior.

INTRODUCTION

Hidden Markov models are widely used in automatic speech recognition as phoneme models to combine the modeling of stationary stochastic processes producing observable short-time features and the temporal relationships between these processes. The temporal model in HMMs relies on a relatively simple structure of successive states and a probabilistic model of their mutual transitions.

The modeling of the stochastic observation processes associated with the states of HMMs is based on estimation of the probability density function of the short-time observations in each state. Several different approaches have been proposed to represent these output probabilities ranging from the estimation of parameters of multivariate Gaussian density [1] to the construction of multilayer perceptrons [3] or LVQ codebooks [5].

A common model category for the observation densities is the mixture density functions normally accomplished as a linear combination in a set of Gaussian densities. If the mixture densities are tied between all the HMM states the models are often called semi-continuous HMMs [4].

0-7803-2026-3/94 $4.00 © 1994 IEEE

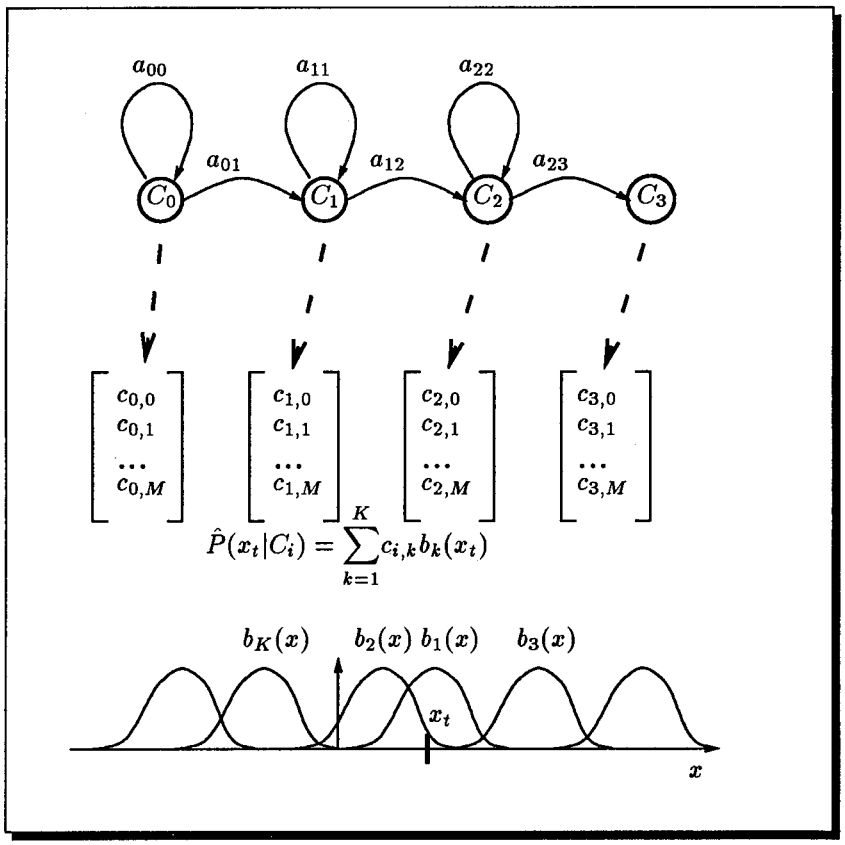


Figure 1: The output probability of state C_i at time t is computed using a tied mixture density function for K nearest mixture densities. The K nearest of all M mixture densities to the current observation x_t are indicated by indexes $k = 1, \cdots, K$. The mixtures are tied for states representing the same phoneme. The HMM is completely defined by the set of transition probabilities a_{ij}, mixture densities $b_k(x)$ and mixture weights $c_{i,k}$.

In this work the tying of the mixture densities is applied in a novel way so that the states belonging to the same HMM (Fig. 1), i. e. representing the same phoneme, use a common codebook of Gaussian densities. Thus there are as many sets of Gaussians as there are HMMs, which is a kind of intermediate for continuous density HMMs (different set for each state) and semi-continuous HMMs (only one large set of Gaussians).

The reason for the phoneme-wise tied Gaussian codebooks is to balance between a vast number of mixture mean vectors and covariances, like in large continuous density HMMs (CDHMMs), and an excessive amount of mixture weights, like in large semi-continuous HMMs (SCHMMs) (Table 1). From the CDHMM point of view, having common Gaussians for states of the same phoneme seems to be an appealing approximation,

since the output densities of successive states are often highly overlapping. From the SCHMM point of view, this means that the normally quite a large set of very small weight values can be reduced, because most of the large weights are often nicely localized around Gaussians resembling to one phoneme [8].

LVQ FOR TIED MIXTURE DENSITIES

The Learning Vector Quantization (LVQ) [6], [7] methods are applied in the present paper to increase the discrimination between the phoneme models. This is a very important objective, since the primary interest in phoneme recognition is to find out the best-matching phonemes instead of estimating the correct probabilities for different phonemes.

In [10] the LVQ methods were applied to provide a discriminative initialization for the Gaussian mean vectors of CDHMMs leading to good recognition results with only a few iterations of the actual HMM parameter estimation performed by Baum-Welch reestimation. In [8] the same ideas were brought to the SCHMMs causing a significant drop in the average recognition error rates.

In the present paper with phoneme-wise tied mixture densities, the LVQ training is similar to the normal SCHMMs [8], except that the initial codebook for LVQ is combined from several smaller codebooks first prepared for each phoneme separately. After LVQ, the codebook is again split among the HMMs. The advantage of this combine-and-split procedure is that in contrast to the labeling method proposed in [8], the satisfactory representation of each phoneme can be guaranteed in addition to the enhanced discrimination between phonemes.

The small codebooks including only representatives of one phoneme can be efficiently trained by Self-Organizing Maps to capture the most essential features of each phoneme and provide a good basis for LVQ. The same initialization procedure could be successfully applied also to the mixture densities in CDHMMs by setting identical mean vectors for the output densities of states of one HMM. A separately trained small codebook of one phoneme is presented in Figure 2 as well as the same codebook after LVQ.

By examining carefully the differences between the corresponding cepstra in the upper and in the lower part of Figure 2 the development of the codebook can be recognized. The upper part is a small SOM (6x4 units) for phoneme /A/ trained in two phases. The first is a short ordering phase with large, but quickly reducing neighborhood radius 5 and learning rate 0.2. The second is a longer specialization phase starting with radius 2 and rate 0.02 reduced to 1 and 0, respectively. The result does not provide very low quantization error, but instead gives a rough representation of the pdf of the corresponding feature space. As an initialization, this presents a smooth two-dimensional surface fitted to the set of training samples with the borders more folded than the center.

The lower part of Figure 2 includes the same codebook vectors, but

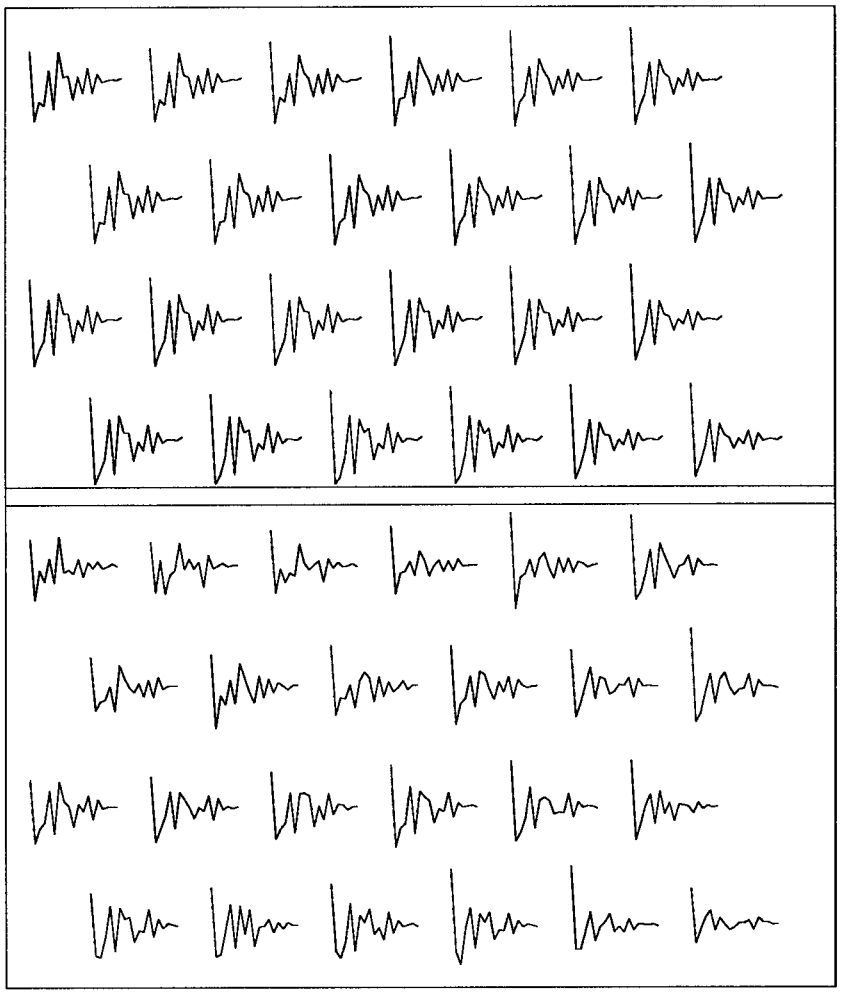


Figure 2: The Gaussian mean vectors of short-time cepstral features for the tied mixture HMM of phoneme /A/ in the different training phases. First (top), after initialization by SOM, and second (bottom) after the LVQ training together with the other phoneme codebooks.

after the combine-and-split procedure and LVQ training. First, all the small SOMs, each representing one phoneme, are concatenated into one large LVQ codebook. The LVQ is started by a short training using the Optimized learning rate LVQ (OLVQ1) [7] followed by a longer training using LVQ3 [6]. The codebook vectors are divided into the same groups as in the original SOMs and each of these groups is converted to the set of mean vectors of the mixture Gaussian densities. Comparing to the upper part of Figure 2, the cepstra are now more distinct, providing a smaller quantization error and, which is more important, a better discrimination between the codebooks of other phonemes. In spite of the changes, some

Training Phase	Phoneme /A/ only			Whole data set	
	Qerror%	MMdist	Cacc%	Cacc%	Rerror%
SOM	17	3	85	61	24
+LVQ	13	13	87	66	8
+Viterbi	13	14	87	66	7
+Tuning	13	13	87	66	7
SOM	17	3	85	61	21
+Viterbi	13	14	87	66	8
+Tuning	15	14	77	61	7
KNN	19	17	83	53	34
+LVQ	16	8	94	49	55
+Viterbi	13	14	88	65	8
+Tuning	15	16	73	60	8
KM	17	10	80	57	18
+Viterbi	14	15	82	65	7
+Tuning	18	20	54	58	7

Table 1: Various statistics in different training phases for one data set (one speaker, 311 words). Qerror% is the average quantization error of the samples. MMdist is the median of distances from each codebook vector to its nearest neighbor. Cacc% is classification accuracy of sample vectors. Rerror% is the phoneme recognition error rate by corresponding tied mixture density HMM. KM is Kmeans clustering and KNN is the method to select the prototypes so that the majority of the K (K=5) nearest data points refer to correct classes.

structure is still visible giving smoothness to the codebook.

In Table 1 some characteristics of codebooks, i.e. the sets of Gaussian mean vectors, are shown numerically. In contrary to the statistically more covering experiments presented at the end of this paper, the values of Table 1 are not average rates but results from single experiments (one speaker, 311 words) and thus insignificant by themselves but only reflecting some general trends.

In Table 1 the characteristics of codebooks during four different training combinations 3–4 distinct phases in each have been analyzed. (The codebooks of phoneme /A/ of the two topmost rows were illustrated in Figure 2.) Kmeans clustering has been used as a substitution of SOM for minimizing quantization error without creating any structure. KNN refers to the basic initialization method for LVQ [7], which creates the codebook by directly selecting valid prototype vectors among the training samples.

The average quantization error reflects, how tightly the codebook is bound to the training samples of that phoneme. Generally, the Viterbi training, being based on the maximum likelihood principle, provides small quantization error. The corrective tuning, while increasing the differentiation between models, tends to increase the quantization error.

The median of the nearest neighbor distances is a measure of the general density of the codebook vectors, i. e. how close they are to each other. This measure shows quantitatively the phenomena observable from Figure 2 that SOM produces tight codebooks avoiding outliers.

The classification accuracy of the training samples simply shows how many of the samples will get the right lable in nearest neighbor classification without any context information. The low total accuracy is due the samples taken from the transition areas between phonemes and from certain plosives that are practically indistinguishable without wider context information.

Because the phoneme recognition error rate is the only measure here that takes account the temporal structure of HMMs, the Viterbi training, by enhancing directly the temporal model, naturally introduces a significant drop of errors. However, the LVQ, if suitably initialized, seems to provide mean vectors that, when combined with simple temporal model, are able to produce comparable error rates. Apparently, the short-time classification accuracy is not directly proportional to the phoneme recognition error rate.

VITERBI TRAINING AND CORRECTIVE TUNING

After the LVQ initialization, the next task in the estimation of the tied mixture density HMMs is the determination of the mixture weights and state transition probabilities. The initial values for the mixture weights can be simply assigned by finding the nearest Gaussians for a set of training vectors and computing the portion of hits for each Gaussian. If no pre-segmentation is available, the training samples can be divided into parts of equal lengths for this initialization. The actual training is accomplished by a version of Viterbi training, i. e. using an initial model to recognize the training words and to segment the sequence of observations into the parts corresponding to each HMM state. Parameter values for each state are then updated using the segmented feature vectors. For example, the resulting mixture weights of the different states representing phoneme /A/ with a tied Gaussian codebook of 24 mixtures are shown in Figure 3.

In Figure 3 the weights in each state are displayed as a surface over the mixtures. It is easy to notice that although some mixtures are used by several states, the main trend is that the center of greatest activation, i. e. largest weights, move from one corner of the codebook to the other.

The Viterbi search for the best path can be constrained on the available information about the phonemes in the training words to produce as correct path as possible [9] to reduce the number of required iterations. That information can include, for example, some pre-segmentation by other models or by hand. The search can be also made more efficiently by approximating the output density probabilities of the tied mixture density functions by using only a couple of the nearest mixtures to the

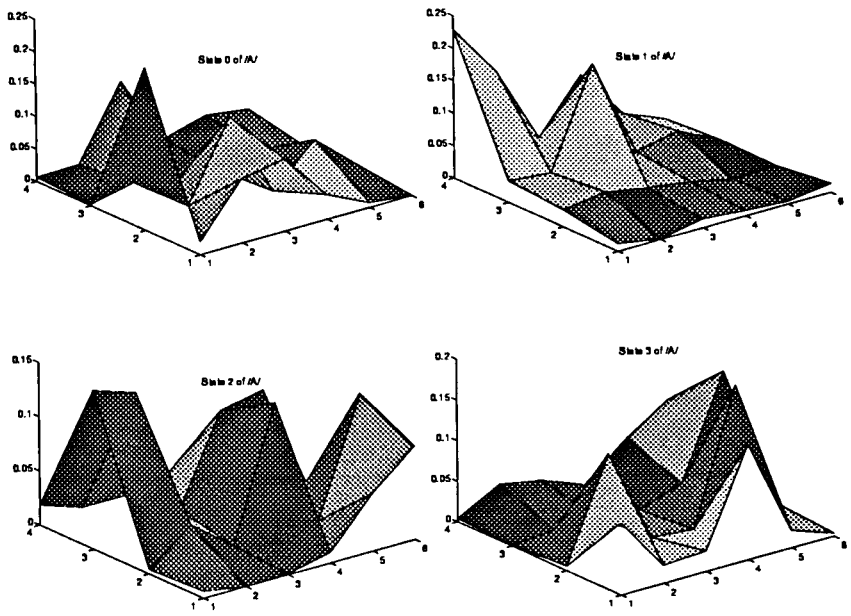


Figure 3: The mixture weights for the Gaussian mean vectors in the four states of tied mixture density HMM of phoneme /A/. The weights are grouped into a 6x4 array corresponding to the dimensions of the Self-Organizing Map used in the initialization (Figure 2).

current observations [2],[9].

To enhance the recognition ability of the models, the LVQ-based corrective tuning methods [9] can be applied to the tied mixture density HMMs of the current paper. These methods are approximative probabilistic descent algorithms tuning the HMMs to produce correct transcriptions by gradually decreasing modifications, if some phonemes in the training words get misclassified. Other methods with some common characteristics are introduced in, e. g. [11] and [12]. The main idea is to use the current models to find misrecognized words and the misrecognized phonemes in them. The incorrect part of the state sequence is then inspected state by state tuning the closest Gaussians that would give the correct result closer and the nearest incorrect ones away [9].

EXPERIMENTS

The experiments involve testing the phoneme recognition accuracy for four sets of 311 Finnish words uttered by three different speakers. By leaving one set at a time for testing totally 12 independent runs are obtained and their average error rates are used in comparison of the

version	error%	iterations
SOM	5.7	1000+10000
SOM+LVQ2	5.7	1000+10000+10000
KM	5.8	200 (rounds)
LVQ3	5.8	5000+50000
SOM*	6.6	1000+10000
SOM+LVQ3	5.7	1000+10000+5000+50000
SOM+LVQ3+Tuning	5.6	+5 times all words

Table 2: Average recognition error rates for variations of training combinations used for the Gaussian mean vectors for 5-state phonetically tied mixture HMMs. KM is the Kmeans algorithm, where 200 is the maximal number of rounds of the all training data for each phoneme. SOM* a reference experiment with SOM of zero neighborhood. SOMs are iterated in two phases (shorter and longer) for each phoneme. LVQ is iterated for all phonemes at the same time and a short OLVQ1 phase precedes the LVQ3 training. The corrective tuning is applied by presenting one word at a time and only misrecognized phonemes cause modifications.

Type of HMM	Number of mixtures	Parameters ($x10^3$) weights	means	Recognition time/word	error%
CDHMM	4	0.4	9	0.5	7.9
PWMHMM	24	3	11	0.7	6.9
SCHMM	494	54	10	1.8	8.1
CDHMM	24	3	55	2.2	5.8
PWMHMM	70	8	32	1.5	5.7

Table 3: Some comparisons between different continuous density HMM structures. The abbreviation PWMHMM refers to the phoneme-wise tied mixture density HMMs proposed in this paper. The CDHMM and SCHMM experiments, reported in earlier papers of the author, use the same speech database and basicly similar training phases as set up here for PWMHMM experiments. No corrective tuning has been used in these experiments.

training methods. The error rates are defined by the sum of missing, changed and extra phonemes divided by the correct sum of phonemes.

The phoneme recognition experiments are performed using the speech recognition system of the Laboratory of Information and Computer Science of Helsinki University of Technology [13]. 20 dimensional cepstral feature vectors concatenated with the energy of the signal are used as the short-time acoustical features. New feature vector is computed every 10 ms using 20 ms signal window.

5 states left-to-right HMMs with no skips are trained by the described

methods for 20 common finnish phonemes and for the silences occurring between the words. The differences of the average error rates in Table 2 are so small that, for example, the Matched Pairs test does not give any significant statistical differences between the various experimented training combinations, except that the experiment of SOM with zero neighborhood is worse than the others.

When comparing the performance of corresponding training combination between different continuous density HMMs the Table 3 reveals that the phoneme-wise tied HMMs provide clearly the most appealing configurations, when the number of parameters, the recognition time and the error rate are compared. The recognition times per word are computed as the average of 311 different finnish words and do not include the preprocessing which is same for each model. The experiments for CDHMMs and SCHMMs were explained in [9] and [8], but the construction of the experiments and the training algorithms are basicly the same and thus the results are comparable.

CONCLUSIONS

A new method to group the Gaussian codebooks for tied mixture density HMMs so that the mixtures are phoneme-wise tied is presented. The aim is to create an intermediate solution between the continuous density and semi-continuous HMMs gathering the best characteristics of both approaches. A hybrid training scheme that tries to combine the discrimination powers of LVQ and the presentation ability of Viterbi training is described for the new models. Some numerical and visual analysis of the created phoneme-wise codebooks of the Gaussian mean vectors are carried out as well as a comparison of different combinations of training algorithms. The performances of the best combinations are not significantly different but when comparing to the previous, corresponding experiments with continuous density and semi-continuous HMMs, significant improvements are achieved in respect of the number of parameters, the recognition time and the average error rates.

REFERENCES

[1] L. Bahl, F. Jelinek, and R. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190, 1983.

[2] Jerome Bellegarda and David Nahamoo. Tied mixture continuous parameter modeling for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(12):2033–2045, 1990.

[3] Herve Bourlard and Christian J. Wellekens. Links between Markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1167–1178, December 1990.

[4] X.D. Huang and M.A. Jack. Semi-continuous hidden Markov models for speech signals. *Computer Speech and Language*, 3, 1989.

[5] H. Iwamida, S. Katagiri, E. McDermott, and Y. Tohkura. A hybrid speech recognition system using HMMs with an LVQ-trained codebook. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 489–492, 1990.

[6] Teuvo Kohonen. The Self-Organizing Map. In *Proceedings of the IEEE*, pages 1464–1480, 1990.

[7] Teuvo Kohonen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. LVQ_PAK: A program package for the correct application of learning vector quantization algorithms. In *Proceedings of the International Joint Conference on Neural networks (IJCNN)*, Baltimore, June 1992.

[8] Mikko Kurimo. Using LVQ to enhance semi-continuous hidden Markov models for phonemes. In *Proceedings of 3rd European Conference on Speech Communication and Technology*, volume 3, pages 1731–1734, Berlin, Germany, September 1993.

[9] Mikko Kurimo. Corrective tuning by applying LVQ for continuous density and semi-continuous markov models. In *Proceedings of International Symposium on Speech, Image Processing and Neural Networks*, volume 2, pages 718–721, Hong Kong, April 1994.

[10] Mikko Kurimo and Kari Torkkola. Training continuous density hidden Markov models in association with self-organizing maps and LVQ. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 174–183, Copenhagen, Denmark, August 1992.

[11] Shinobu Mizuta and Kunio Nakajima. An optimal discriminative training method for continuous mixture density HMMs. In *Proceedings of the International Conference on Spoken Language Processing*, volume 1, pages 245–248, Kobe,Japan, November 1990.

[12] David Rainton and Shigeki Sagayama. Minimum error classification training of HMMs - implementation details and experimental results. *J.Acoust.Soc.Jpn*, 13(6):379–387, 1992.

[13] Kari Torkkola, Jari Kangas, Pekka Utela, Sami Kaski, Mikko Kokkonen, Mikko Kurimo, and Teuvo Kohonen. Status report of the finnish phonetic typewriter project. In *Proceedings of ICANN*, volume 1, pages 771–776, Espoo,Finland, June 1991.

Image Processing

Moving Object Classification in a Domestic Environment using Quadratic Neural Networks

Gek Lim, Michael Alder, Christopher J.S. deSilva
and Yianni Attikiouzel
Centre for Intelligent Information Processing Systems
The University of Western Australia
Nedlands W.A. 6009, AUSTRALIA
Tel: +61-9-3801763, Fax: +61-9-3801011
e-mail: [gek, mike, chris, yianni]@ee.uwa.edu

Abstract-In this paper, we present a moving object recognition system. A description is given of the whole system from the image acquisition through the preprocessing and feature extraction stages to the classification of objects. We use Quadratic Neural Networks (QNN) to model the input data and then extract features from the model which are translation and rotation invariant. We have applied the idea to a practical problem of classifying moving objects in a domestic environment such as a moving heads, curtains blown by the wind and external events such as moving tree branches. Reasonable results are obtained using only the spatial information.

INTRODUCTION

In this paper, we present a moving object recognition system for a domestic environment. In section 2, we briefly describe the system and introduce new techniques for building the system using Quadratic Neural Networks (QNN). We show that a QNN is not only powerful as a classifier [1], it is also capable of other functions such as data modelling. Section 3 discusses the idea of a quadratic neuron and how it can be used in data modelling. A practical case study of a moving object recognition system is presented in Section 4 using QNN and finally conclusions are drawn in Section 5.

A MOVING OBJECT RECOGNITION SYSTEM

The system that we are describing here attempts to recognise only moving objects, in particular, objects that move in a specific environment, for

0-7803-2026-3/94 $4.00 © 1994 IEEE

A quadratic neuron, that is, models a probability density function locally and responds to a particular set of points in $\mathbf{R}^N$. For example, given a data set of points in $\mathbf{R}^2$, we can compute the mean, $\mathbf{m}$, and covariance matrix, C, of the set of points and this gives rise to a gaussian distribution with $Q = C^{-1}$. In our neuron model, the response of the neuron may be treated as the log likelihood of the set evaluated by the associated gaussian distribution. Although our commitment to gaussian distributions is not absolute, they allow us to relate neural models to conventional statistical models to some degree.

Response of the neuron modelling the set of points in $\mathbf{R}^2$ shown in **Figure 1(a)** will be high whereas the response for **Figure 1(b)** will be low as it is a poor fit to the data. Given another set of points as shown in **Figure 2(a)**, in order to achieve the maximum likelihood of the distribution, more than one ellipse (or neuron) is required, **Figure 2(b)**. In this case, we have a mixture of gaussian distributions.

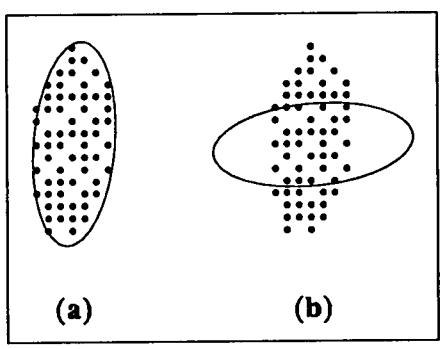


Figure 1: Response of neurons (a) High response, (b) Low response

CASE STUDY

We are interested in building a moving object recognition system that can classify objects in a domestic environment. What one finds in this case is moving human heads, curtains blown by the wind and external events seen through windows such as moving tree branches.

Input Acquistion

Images are collected using a CCD-camera and a frame grabber board. The resolution of the images is 256x256 pixels with 128 grey-levels. **Figure 3(a)** shows two consecutive frames of the three different classes of object.

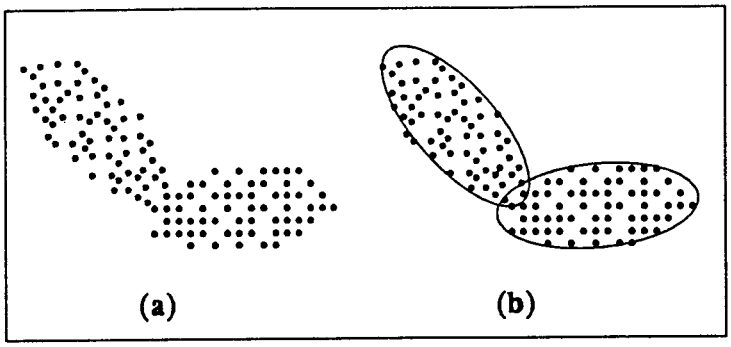


Figure 2: Data Modelling (c) Two clusters of points, (d) Gaussian Mixture Modelling

Preprocessing Stage

First, we detect the motion of the objects by differencing two consecutive frames. The moving parts are shown in white, **Figure 3(b)**.

Next, we apply a border tracing algorithm [2] to find the starting point and the chaincode of all the isolated regions. We remove regions with 'short' chaincode; the border image is shown in **Figure 3(c)**.

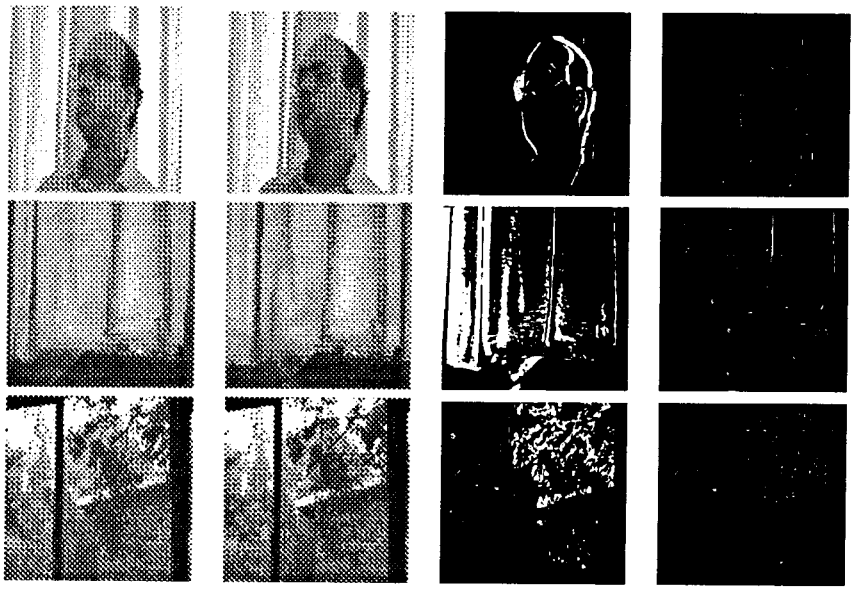
Feature Extraction

We use a QNN to model the border image of the objects. We fit ellipses along the border by taking $\mathcal{L}$ consecutive elements of the chaincode for some suitable $\mathcal{L}$, and compute the mean and covariance matrix for the set. $\mathcal{L}$ must be chosen so that the quadratic forms are not degenerate, but not so large that the structure is distorted.

The resulting image shows the border of the moving parts represented by sequences of ellipses, **Figure 4**. The effect of this is to smooth the noisy boundary contour and compress the representation of the objects. More importantly, the structural information of the objects is not distorted and the ellipse representation can easily be made translation and rotation invariant.

We have assumed that the moving object recognition system knows exactly what it is looking for. In this case, we are looking for a head, curtain or tree. It will be observed that the characteristic of the curtain is that it is made up of mainly vertical edges with some random 'noise', whereas a head composed of more or less curved edges. The tree basically has no structure and has a high entropy. This term makes sense if we regard each ellipse as a predictor of the orientations of its neighbours: recall that the chaincode establishes an ordering on the quadratic forms.

We look at the change of angle between consecutive ellipses and plot the histogram with an intervals of five degrees. **Figure 5** shows histogram plots



(a) (b) (c)

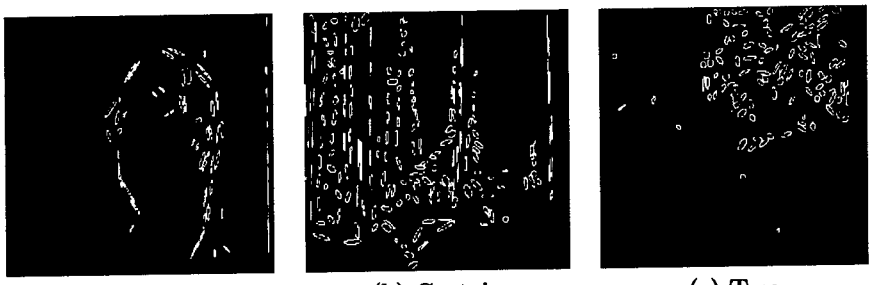
Figure 3: Examples of the three different objects: Faces, Curtains, Tree Branches; (a) Original image, (b) Difference image, (c) Border of the difference image, (d) Border of the difference image with ellipses along it.

for all three objects. The fact that there is a high frequency of small angular change in the curtain histogram reflects that it is made up mainly of vertical edges. The flat distribution of the tree histogram justified our claim that the tree has a complex structure with irregularly placed ellipses. The head is somewhere in between because it has curved edges which are still highly regular, and which leads to a slightly bigger angular change in the mean.

CLASSIFICATION

We classify a new object by computing its histogram and comparing it with the mean histograms for each of the three object classes. There are several possible ways of measuring a distance between histograms; we have a preference for information theoretic methods and therefore use the KullBack-Leibler distance for the classification. Thus we compute the KullBack-Leibler distance of a new histogram from the average distribution for each class using

$$\sum_{i=1}^{N} P(x_i) \log_2 \frac{P(x_i)}{Q(x_i)} \tag{2}$$



(a) Face (b) Curtain (c) Tree

Figure 4: Fitting ellipses along the border of the difference image

where $P(x_i)$ is the average model for class i,
and $Q(x_i)$ is the new distribution, to be classified.

Classification is done by finding the minimum KL distance between the new histogram and the category means. A small constant value is added to each bin in the histogram to avoid the problems of zero probability events.

RESULTS

Using the Kullback-Leibler measure of discrepancy between histograms, the classification rate for the training set is 92% and the testing set is 82%, **Table 1**. Histograms were formed based on the angular change between consecutive ellipses from the ellipse sequences. We compute the mean distribution for the curtain, head and tree classes from 26, 29 and 37 histograms respectively. **Table 1(a)** shows the confusion matrix of the results of the histogram classification. Some of the data was obtained by taking a sequence of time slices of the same scene, others from different scenes.

CONCLUSION

We have shown that using a QNN to model objects with a set of ellipses, we can easily extract features of the objects that preserve the translation and rotation invariance properties. Reasonable results have been obtained using only the spatial information. By this we mean that if we restrict ourselves to consecutive frames only, we still obtain a reasonably high level of correct classification given the inherent complexity of the problem: real images obtained in real time tend to be rather resistant to analysis. Our results showed that there were no significant correlations between the misclassifications for images obtained from differencing slices which are quite close in time, so it

Classes	Curtain	Face	Tree
Curtain	1.0	0.0	0.0
Face	0.0	1.0	0.0
Tree	0.0	0.22	0.78

(a)Confusion matrix for training Data: 26 curtains; 29 faces; 37 trees

Classes	Curtain	Face	Tree
Curtain	0.88	0.06	0.06
Face	0.12	0.82	0.06
Tree	0.0	0.23	0.77

(b)Confusion matrix for testing Data: 17 curtains; 17 faces; 13 trees

Table 1: Classification results using Kullback-Leibler distance between histograms of angular change

would be simple to use temporal information to improve the results further.

REFERENCES

[1] Lim, S.G., Alder, M.D. and Hadingham, P., *Quadratic Neural Nets* Pattern Recognition Letters 13 (May 1992) pp325-329.

[2] Haig, T.D., Attikiouzel, Y. and Alder, M.D. *Border Following: New Definition Gives Improved Border* IEE Proc-I, Vol. 139, No.2, pp. 206-211, April 1992.

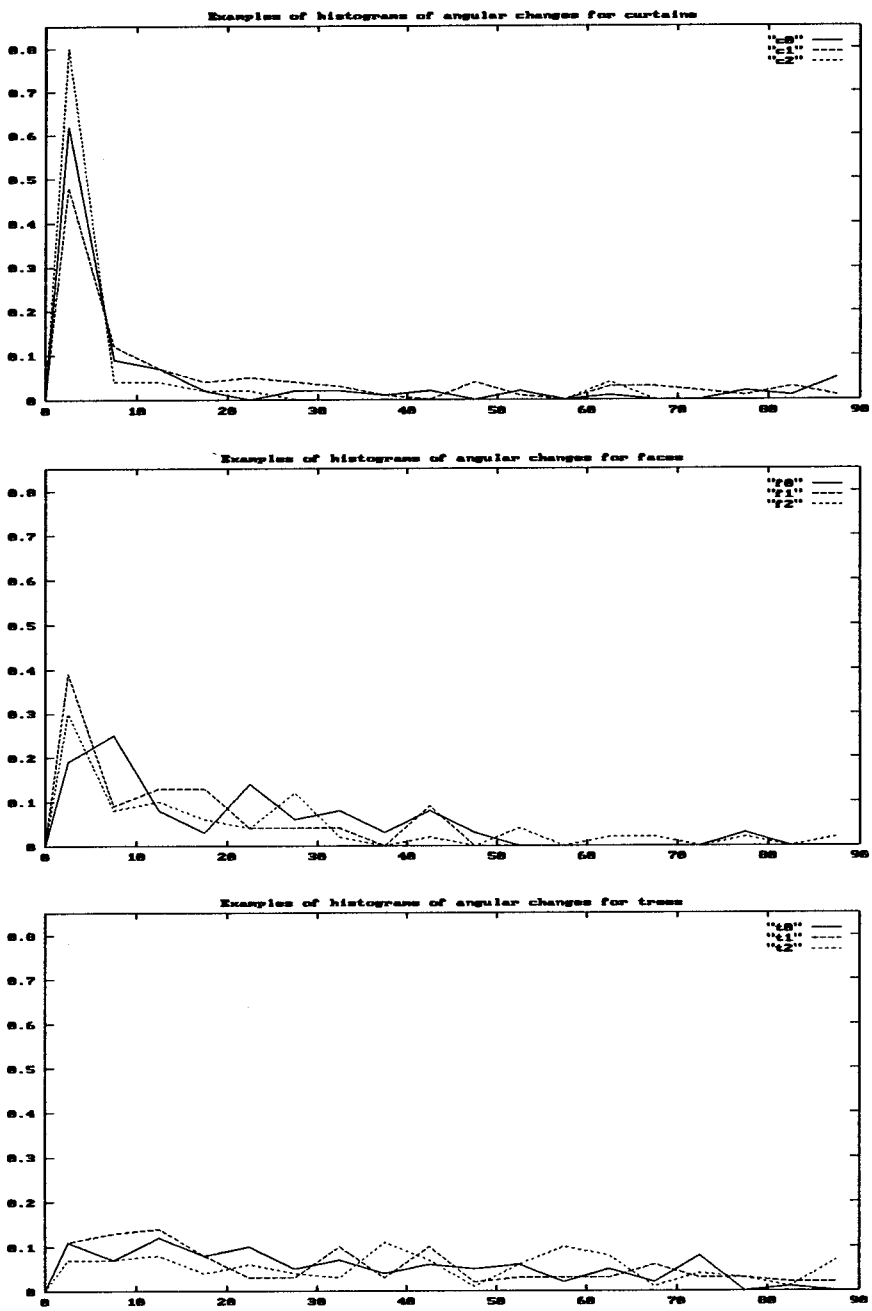


Figure 5: Histograms of angular changes of the objects, (a) curtain, (b) face and (c) tree

Application of the HLVQ Neural Network to Hand-written Digit Recognition

B.Solaiman * and Y.Autret**

* Ecole Nationale Supérieure des Télécommunications de Bretagne
B.P. 832 , 29285 Brest Cedex - FRANCE
(Tél:33 98 00 13 08, Fax: 33 98 00 10 98)

** Université de Bretagne Occidentale- Faculté des Sciences
6, Av le Gorgeu, 29287 Brest Cedex - FRANCE
(Tél:33 98 47 65 39, Fax: 33 98 31 62 13)

Abstract. In this work, the hand-written digit recognition problem is studied. Self organizing Feature Maps (SOFM) are mainly considered. The unsupervised Kohonen as well as the Hybrid Learning Vector Quantization algorithms are applied. The main objective is to obtain a topology preserving map having high recognition rates. This is essentially due to the fact that this kind of maps is very useful in realising results interpretations and in the definition of a rejection strategy during the recognition phase.

INTRODUCTION

Pattern recognition is a challenging problem whose solution is very useful for applications in which large volume of data is processed. The case of hand-written digit recognition is of special interest because it has considerable practical applications.

Hand-written digits suffer not only from scale, location and orientation variations, but also person-dependant deformations which are neither predictable nor mathematically formulated. Therefore, research on hand-written digit recognition has never been easy.

0-7803-2026-3/94 $4.00 © 1994 IEEE

Approaches generally used in solving this problem fall in one of two categories : the global analysis and structural analysis approaches. The use of neural networks offers an alternative and easy method for hand-written digit recognition.

By directly training the network with sufficiently large data set, the recognition rate can be quite high. Multilayer perceptron (MLP) neural network associated with the backpropagation supervised learning algorithm have received more and more attention. This is essentially due to the fact that the recognition rate obtained by this network is very high.

Clustering based neural networks are rarely used in hand-written recognition. The most known clustering neural network is the Self Organizing Feature Map (SOFM) introduced by T.Kohonen [1]. This network will be discussed briefly in the next section. The learning algorithm proposed by T.Kohonen associated with this network, uses the non-supervised paradigm. Obtained results are thus of great interest in terms of obtained cluster centres (also called prototypes) and in terms of topology preserving.

The characteristic of topology preserving is extremely important if one needs to obtain more information concerning the digit to be recognized and if a rejection strategy has to be integrated to the recognition system. Nevertheless, this network suffers greatly of having a very low recognition rate when the trained network is used as a pattern recognition system. This is absolutely normal since the SOFM has never been developed in order to obtain a pattern recognition system.

T.Kohonen has also suggested a supervised algorithm called the Learning Vector Quantization 2 (LVQ2),[2], in order to comply with Bayes making theory. In the LVQ2 algorithm, different neurons are treated independently in the sense that there in no topological relation among them.
Obtained recognition results with this algorithm are comparable to those of the Bayesian classifier.

In this paper, a hybrid learning algorithm called the Hybrid Learning Vector Quantization (HLVQ),[3], is applied in training a SOFM. The main objective of this study is to obtain a SOFM having both characteristics: 1)a topology preserving mapping, and 2)high recognition rates.

The SOFM and the LVQ2 networks

The Self Organizing Feature Map (SOFM) introduced by T.Kohonen, is one of the most successful models in the area of unsupervised learning. This model builds up a mapping from the N-dimensional vector space of real numbers $\Re^N$ to a two dimensional array S of cells. Each cell is given a virtual position in $\Re^N$. This position is given by the synaptic weights connecting this cell to the input vector (figure.1).

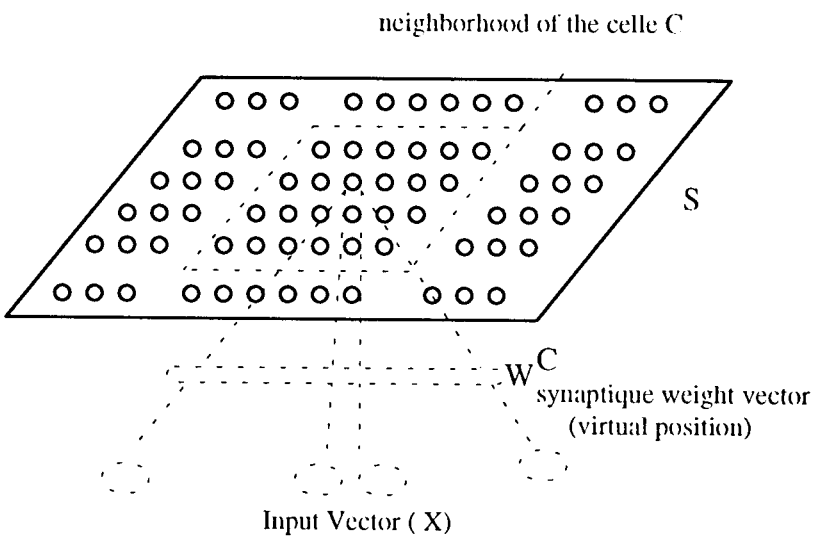


figure.1. The Self Organization Feature Map

The purpose of the self organizing process is to find the position vectors such that the resulting mapping is a topology preserving mapping (adjacent vectors in $\Re^N$ are mapped on adjacent, or identical, cells in the array S). The learning algorithm that forms the feature maps selects the best matching cell according to the minimum distance between its position W^C and the input vector X. This cell is referred to as the winning cell. All position vectors in the neighbourhood of the winning cell are adjusted in order to make them more responsive to the current input. Positions adjustment is given by :

$$\Delta W^C (t+1) = \beta(t) * (X - W^C(t)) \qquad (1)$$

A decreasing step function β (t) and a slowly size decreasing neighbourhood with time, ensure the stability of the process of weights adjustment.

The LVQ2 algorithm has also been suggested by T.Kohonen as a supervised learning algorithm. In this algorithm, different decision neurons have no topological relations among them. The main idea is to adjust the synaptic weights of the best matching and the next best matching cells (noted respectively C* and C"). The learning algorithm is only applied if the three following conditions are verified:

1) the input vector X is miss classified by the best matching cell C*,

2) the next best matching cell C" has the correct class,

3) the input vector is close enough to the decision boundary.

Position vectors W^{C*} and $W^{C''}$ are then adjusted as follows :

$$\Delta W^{C*}(t+1) = -\ \alpha(t) * (X - W^{C*}(t))$$
$$\Delta W^{C''}(t+1) = +\ \alpha(t) * (X - W^{C''}(t)) \qquad (2)$$

where $\alpha(t)$ is a decreasing step function.

The Hybrid Learning Vector Quantization

Learning capability is the most salient feature of neural networks. Learning paradigm largely depends on the neural network structure and the characteristics of the data which the neural network deals with.

The most widely used are the supervised and the non supervised learning algorithms. The combination of the supervised and the non supervised learning paradigms is generally performed sequentially. The non supervised learning is first used in order to extract important features from the learning examples.

Secondly, the supervised learning is used in order to separate the learning patterns using the features already determined by the non supervised approach.

In this study, a new algorithm called Hybrid Learning Vector Quantization(HLVQ), combining the supervised and the non supervised paradigms is presented. The aim of the HLVQ algorithm is to obtain an array of cells, S, realising a topology-preserving mapping. At the same time, those labelled cell positions must be well distributed in $\Re^N$ giving good classification results.

The first version of the HLVQ algorithm consists on the application of the LVQ2 algorithm over a **background** of the unsupervised self-organizing feature map algorithm. This means that the supervised learning paradigm plays the role of **"attention focusing"** applied over the unsupervised background learning.

In fact, after each digit presentation, both best matching and next best matching cells are determined.

First, positions of all cells in the neighbourhood of the best matching cell are adjusted using the unsupervised learning algorithm (1).

The decreasing step function used is given by : $\beta(t) = \eta \ . \ \alpha(t)$, where η is an attenuation factor assuming small values ($\eta \in [0.05,0.2]$).This algorithm can be resumed as follows :

Step-0- Given the labelled data set { (X1,d1), ..., (Xk,dk)}, where Xk $\in \Re^N$, and dk is the corresponding desired output vector representing the class to which the digit Xk belongs.

The dimensions of the array S, the maximum number of learning iterations (Max_Iterations), the decreasing step function $\alpha(t)$ and the attenuation factor η are fixed.

Step-1- All cells of the map are labelled. Labelling affects to each cell $C \in S$, the class for which the cell was the best matching more than other classes.

Step-2-

For iteration=1,2,..., Max_Iterations ,

A) **Learning Iteration:**

for k=1,2,... ,K,

a) Present the digit Xk to the input,

b) Find the best matching C* and the next best matching C" cells,

c)**The background unsupervised learning**:cells in the neighbourhood of the best matching cell are adjusted according to (1).

d)**Attention focusing** :if the three conditions of the application of the LVQ2 algorithm are verified, then cells C* and C" are adjusted according to (2).

next k,

B)**Labelling iteration :** All cells of the map are labelled,

C)Adjust the learning rate,

Next iteration.

The second version of the HLVQ algorithm concerns the attention focusing aspect. In fact, in the first version, the attention focusing consists on the application of the LVQ2 algorithm. Two cases are mainly considered :

1) the best matching and the next best matching cells answer correctly,

2) the best matching and the next best matching cells answer badly.

In the first case, both the best matching and the next best matching cells are very close to the input digit.

Therefor, and in order to optimize the global use of different prototypes, the next best matching cell has to change its position in order to capture other digits information.

In the second case, and this happens generally at the beginning of the learning procedure where different cells are not well assigned to their appropriate classes yet, the best matching and the next best matching cells answer badly. In the HLVQ algorithm a neighbourhood area of the best matching and the next best matching cells is used in order to find a labelled cell corresponding to the class of the input digit.

This neighbourhood area is not necessarily the same as the neighbourhood area of the unsupervised SOFM learning algorithm. Thus the attention focusing used in the second version is resumed as follows :

Attention focusing :

1) if the three conditions of the application of the LVQ2 algorithm are verified, then cells C* and C" are adjusted according to (2).
2) if the best matching and the next best matching cells answer correctly, then the following adjustment rule is applied :

$$\Delta W^{C^*}(t+1) = + \alpha(t) * (X - W^{C^*}(t))$$
$$\Delta W^{C''}(t+1) = - \alpha(t) * (X - W^{C''}(t)) \qquad (3)$$

This rule constitutes a "punishment" to the next best matching cell.

3) if both the best matching and the next best matching cells give bad classification results, a third winning cell C3 from the neighbourhood of the C* or C" corresponding to the class of X is searched for.

If C3 is found, then the updating rule (1) is applied in adjusting the position of this cell.

If no cell corresponding to the class of X is found in the neighbourhood, the algorithm searches a non labelled cell C3 in the same neighbourhood. If such a cell exists, then the updating rule (1) is applied in adjusting the position of this cell.

Simulation results

In this study, the French Postal Service (SRTP) data base containing 6000 hand-written digits was used. This data base has been divided into a learning data base (4000 hand-written digits) and a test data base (2000 hand-written digits).

Each hand-written digit is given as a 16x16 pre-processed image (256 grey levels). Pre-processing consists on scale normalization. All simulations conducted in this study are based on the use of a 10x10 topological map with the same random synaptic weights initialisation. The first simulation consisted on training the SOFM with the non supervised Kohonen algorithm. In figure.2., synaptic weights of trained map are visualised as a 160x160 image.

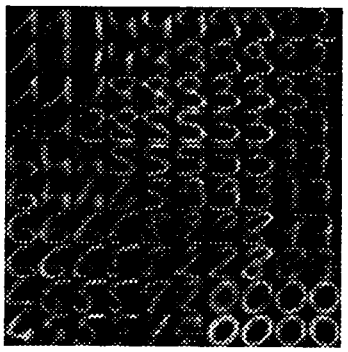


Figure.2. Synaptic weights of the Kohonen SOFM

As can be noticed, the topology preserving characteristic of the Kohonen learning algorithm is obtained in terms of resemblance between adjacent synaptic weights.

Neurons having synaptic weights not clearly distinguishable are simply those having a position very close to decision surfaces. In terms of recognition rate, obtained results are very bad : 75% for the learning data base and 65% for the test data base.

The second simulation has been conducted using the same SOFM trained by the first version of the HLVQ algorithm. In figure.3., the synaptic weights image of the SOFM is shown in different stages of the learning procedure.

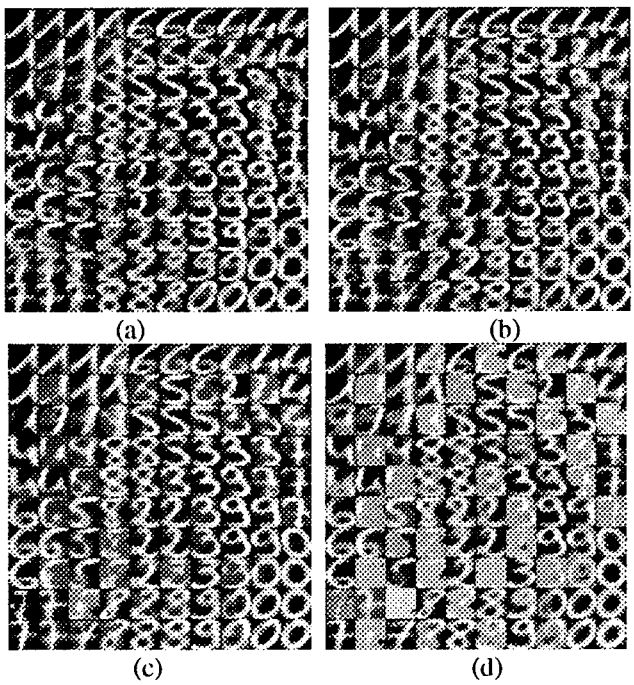


Figure.3. Synaptic weights image in different learning stages: (a) after 10 iterations, (b) after 20 iterations, (c) after 50 iterations, and (d)converged synaptic weights(first HLVQ version),

These results simply mean that the first version of the HLVQ algorithm in the early learning stages produces very similar results to those of the Kohonen non supervised learning algorithm (in terms of topology preserving).

On final learning stages, the topology preserving is lost and results are essentially obtained by the LVQ2 algorithm.Obtained recognition rates are of : 97% concerning the learning data base, and 90% concerning the test data base.As can also be noticed, synaptic weights obtained by this version of the HLVQ algorithm, can be considered as a kind of important visual features that can be treated by other recognition systems.

In the last simulation, the second version of the HLVQ algorithm is applied. Obtained results in terms of recognition rates are nearly the same as those of the first version of this algorithm : 96.8 % concerning the learning data base, and 89.8 % concerning the test data base. Obtained synaptic weights image is given in figure.4.

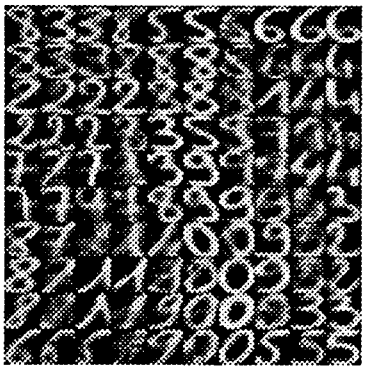


Figure.4. Synaptic weights image obtained by the second version of the HLVQ learning algorithm

As can be noticed, the topology preserving characteristic of the Kohonen non supervised algorithm is maintained by the use of this second version of the HLVQ algorithm.

CONCLUSIONS

In this study, the use of Self organizing Feature Maps (SOFM) in hand-written digit recognition problem is mainly considered. The unsupervised Kohonen as well as the Hybrid Learning Vector Quantization algorithms are applied. The main objective is to obtain a topology preserving map having high recognition rates. This is essentially due to the fact that this kind of maps is very useful in realising results interpretations and in the definition of a rejection strategy during the recognition phase. Obtained results show that the SOFM trained by the second version of the HLVQ algorithm allows to obtain high recognition rates while maintaining the important characteristic of a topology preserving map. In order to show the quality of the obtained results in terms of recognition rate, a multilayer neural network (with one hidden layer) was trained using the Backpropagation learning algorithm. Obtained results are as follows : 96% concerning the learning data base, and 93 % concerning the test data base. Keeping in mind that the SOFM decision strategy (attribution of the class of the best matching unit to a given input) is

piecewice linear decision strategy, and that the MLP is a non linear decision strategy making, it seems that obtained results by the use of the SOFM trained by the HLVQ algorithm are of great interest.

Further work concerning the definition of more sophisticated attention focusing of the HLVQ algorithm and taking into account the decision quality of each neuron of the map, is actually under study.

References

[1]T.Kohonen, "Self-Organization and associative memory", Springer-Verlag,Berlin 1984.

[2]T.Kohonen," Self-Organization and associative Memory",3d ed, 1989, Berlin :Springer-Verlag.

[3]B.Solaiman, M.C.Mouchot and E.Maillard, "A hybrid algorithm, HLVQ, combining unsupervised and supervised learning approaches", International Conferences on Neural Networks, ICNN94, June26-July2, Orlando, USA, 1994.

Ensemble Methods for Automatic Masking of Clouds in AVIRIS Imagery

Charles M. Bachmann *, Eugene E. Clothiaux †, John W. Moore ‡, Keith J. Andreano ◇ and Dong Q. Luong *

* Airborne Radar Branch
Radar Division,Code 5362
Naval Research Laboratory,
Washington, D. C. 20375
e-mail: bachmann@radar.nrl.navy.mil

† Department of Meteorology
Pennsylvania State University
University Park, Penn. 16802
e-mail: cloth@essc.psu.edu

‡ Martin Marietta Services
2231 S. Crystal Park Dr.
Arlington, VA 22202-3735

◇ SWL
1900 Gallows Rd.
Vienna, VA 22182

Abstract- We describe the first-phase of an investigation into techniques for automatic cloud masking in remote sensing data. BCM Projection Pursuit networks are explored as a method of unsupervised feature extraction from AVIRIS images. Search vectors in this method discover directions in the data in which the projected data is skew or multi-modal, by minimizing a projection index which depends on higher moments of the projected data distribution. Ensemble methods are used to fuse information from extracted BCM features and to smooth the mapping of these features to classification of image pixels. Ensemble hierarchies contain multiple levels of networks, combining BCM at the lowest levels with backward propagation algorithms, based on cross-entropy minimization, at higher levels in the ensembles. Predicted cloud masks are compared against cloud masks derived from human interpretation; ensembles achieve better overall classification accuracy than single BP networks.

Introduction

The automatic identification of clouds and cloud type in remote sensing data poses a significant technical challenge to researchers in climate modelling. Human analysis of images is time-consuming, and automatic methods of scene-level and pixel-level identification are needed to cope with large volumes of image data. A number of researchers have investigated multi-class scene- and pixel-level identification of cloud type based on textural and spectral features [6], [7], [31], [29] using AVHRR (Advanced Very High-Resolution Radiometer) [10], [17] images. Examples of such features are moments of gray-level difference vector (GLDV) statistics [32], [4], sum and difference histograms (SADH) [30], [4] and gray-level run length (GLRL) [32]. Neural network techniques based on backward propagation (BP) [26], the probabilistic neural network (PNN), [28] and Kohonen's Learning Vector Quantization (LVQ) [18] have been used successfully to find meaningful information in these features and their inter-relationship for classification [31], [29] ; results

compare favorably to traditional statistical analysis of the same textural features [31].

The present study examines techniques for pixel-level classification in AVIRIS (Airborne Visible and Infra-Red Imaging Spectrometer) imagery. In the first phase of our investigation, we looked only at the raw intensity data without textural and spectral pre-processing steps; a future paper will describe ensemble methods which simultaneously examine both raw data inputs and textural and spectral inputs as found in [7], [31], [29]. In this paper, we describe ensemble techniques which incorporate both unsupervised feature extraction networks, in this case BCM Projection Pursuit, as well as a supervised learning algorithm, BP, for mapping BCM features to a final classification. These results are compared against backward propagation alone. Classification results are generated for low-level cloud masks which only distinguish between pixels containing cloud and those containing no cloud. The problem of identifying cloud-type will be addressed in a future publication.

Unsupervised Feature Extraction Using BCM Projection Pursuit

Recent treatments of the BCM model [3] [5] [27] have shown its relation to the statistical approach known as Projection Pursuit [15] [16]. A Lyapunov function (cost function or projection index) for the modification rule can be defined for BCM; minimization of this function will favor directions where the projection distribution (projection onto the search vector) is statistically skew, i.e bi-modal or multi-modal (Figure 1).The BCM model uses a semi-local learning rule: search vectors are modified based on the information available within a single layer without reference to training labels; in contrast, supervised networks such as BP modify network connections in all layers based on a global error measure in the last layer.

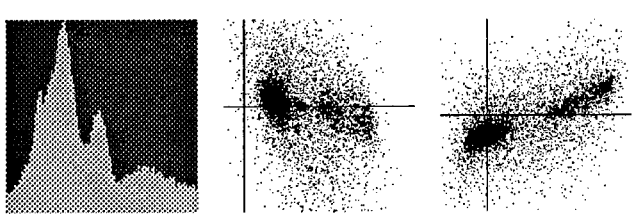


Figure 1: (Left) Multi-modal response histogram from a BCM cell in a network trained with 16x16 pixel patches from AVIRIS Imagery ; (middle and right) two-dimensional scatter-plots from pairs of BCM cells reveal clustering of responses in trained networks.

The ith cell in layer n of a multi-layer BCM network responds according

to: [1] [2]

$$\tilde{c}_i^{(n)} = \sigma(\sum_j L_{ij}^{(n)} c_j^{(n)}) \quad (1)$$

with

$$c_j^{(n)} = \vec{w}_j^{(n-1)} \cdot \vec{\tilde{c}}^{(n-1)} + b_j^{(n)} \quad (2)$$

$$L_{ij}^{(n)} = \begin{pmatrix} 1, for\ i = j \\ -\mu, for\ i \neq j \end{pmatrix} \quad (3)$$

where $L_{ij}^{(n)}$ is the fixed lateral inhibition [3] matrix of weights in layer n, $\vec{w}_j^{(n-1)}$ is the vector of connections to cell j in layer n from the prevous layer, $(n-1)$, and $b_j^{(n)}$ is the bias of cell j in layer n. The Lyapunov function in layer n is a function of higher order moments which will favor projection vectors for which the distribution of cell responses is skew or multi-modal: [4]

$$E[\xi^{(n)}] = -(\sum_i \frac{E[(\tilde{c}_i^{(n)})^3]}{3} - \gamma \frac{E^2[(\tilde{c}_i^{(n)})^2]}{4}) \quad (4)$$

This leads to a learning rule of the form:

$$\Delta \vec{w}_k^{(n-1)} = -\eta \frac{\partial E(\xi^{(n)})}{\partial \vec{w}_k^{(n-1)}} = -\eta \sum_i \frac{\partial E[\xi_i^{(n)}]}{\partial \tilde{c}_i^{(n)}} \frac{\partial \tilde{c}_i^{(n)}}{\partial c_k^{(n)}} \frac{\partial c_k^{(n)}}{\partial \vec{w}_k^{(n-1)}}$$

$$= \eta E\,[\sum_i \phi(\tilde{c}_i^{(n)}, \tilde{\theta}_i^{(n)})\,(\tilde{c}_i^{(n)})'\,L_{ik}^{(n)}\,\vec{\tilde{c}}^{(n-1)}\ ] \quad (5)$$

with:

$$\phi(\tilde{c}_i^{(n)}, \tilde{\theta}_i^{(n)}) = \tilde{c}_i^{(n)}(\tilde{c}_i^{(n)} - \gamma\tilde{\theta}_i^{(n)})\ \ and\ \ \theta_i^{(n)} = E[(c_i^{(n)})^2] \quad (6)$$

$\gamma\theta_i^{(n)}$ is the dynamic modification threshold which separates regions where the ϕ-function yields Hebbian reinforcement and anti-Hebbian weakening in the single-cell theory. For a small and decreasing step-size, Equation 5 can be well approximated by stochastic gradient descent (see [16] for further details).

Ensemble Methods

A number of researchers have explored the use of ensemble methods for the purpose of enhancing overall performance of neural network classifiers [12], [19], [13], [22], [23], [24]. The notion of pooling a set of "experts" is by no

[1] The sigmoidal function is typically of the form: $\tilde{c}(x) = a\tanh(a\lambda x)$, which has the derivative: $\tilde{c}'(x) = \lambda(a - \tilde{c}(x))(a + \tilde{c}(x))$.

[2] Superscripts denoting layer indices appear in parentheses.

[3] The choice of $L_{ij}^{(n)}$ places each cell in a field proportional to the average response of the other cells in the layer. Other choices could be used to establish different fixed influence fields, for instance the Mexican Hat [18].

[4] $E[]$ represents the expectation value.

means confined to research in adaptive neural network algorithms. Whatever the nature of the underlying estimation process, ensemble methods can be employed profitably [11], [14], [22]. We illustrate the general framework for ensembles of estimators in Figure 2.

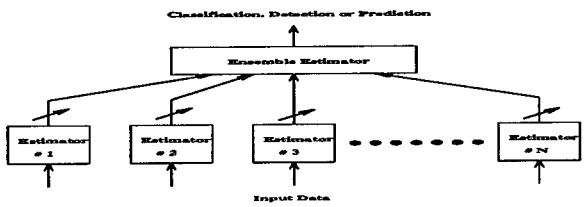


Figure 2: Schematic diagram of an ensemble of estimators. Each individual estimator may have a very large number of parameters as in non-parametric methods, e.g. neural networks, and the input data to each may be identical data sets or different representations of the same problem such as in sensor fusion. The ensemble estimator weights the estimates of members of the ensemble, and the relative weighting may be adapted.

Cloud Masking for AVIRIS Images: Experimental Design (First Phase)

In the initial phase of our research as described in this paper, we investigated ensemble networks. In one version of these ensembles, multiple BCM Projection Pursuit networks performed low-level, unsupervised feature extraction from input patches of AVIRIS images. Inputs to these networks were vectors containing the pixel values as a percentage of the dynamic range over the entire image. In some experiments, we normalized inputs to the dynamic range within the input patch and then renormalized the input vector to a unit vector representation; such a transformation emphasizes the local structure and texture of the input patch by preserving the direction of the high-dimensional vector in input space, at the same time, information about overall intensity is removed by normalizing the vector onto the unit sphere. The ouput of the BCM networks was fed to the input layer of a BP network which performed the mapping to pixel-level predictions. All backward propagation networks described in this paper used a cross-entropy objective function [25]. Error correction in the BP networks was done by comparing network pixel predictions against a ground truth mask generated by human interpretation. The unsupervised BCM networks were trained independently before being attached to the BP networks. As a baseline, results were also obtained for 3-level BP networks operating directly on the input patches. A third experiment consisted of pooling the ensembles in the first set of ensembles to obtain an ensemble of ensembles. This last experiment is closer in spirit to the ensemble concept in [13] and [24], in which the actual output classifications of estimators are pooled. Network configurations for the experiments described here are reported in Table 1. LIBP [1] refers to BP run with fixed lateral inhibition, as

in the feedforward rule for the BCM network; inhibition in the BP networks was found to be important for obtaining reasonable performance with 3-layer networks.

Table 1
Network Configurations: Single & Ensemble Nets

Experiment	Net Type		Configuration
BP1	Single	Level : 1	LIBP: (256-100-16)
		Input:	Band 17
ENSSD1	Ensemble	Level 2:	BP (136-100-16)
		Level 1:	10 BCM Networks: 8 of size: 256-12 ; 2 of size 256-20
		Preprocess:	8 Nets: Pixels as % of Dynamic Range of Whole Image 2 Nets: Pixels as % of Dynamic Range of Input Patch; Renormalized to Unit Sphere
		Input:	Individual Nets Receive Input from a Single Band Either Band 52 or Band 17
ENSSD4	Ensemble	Level 2:	BP (72-100-16)
		Level 1:	6 BCM Networks: All of size 256-12
		Preprocess:	2 Nets: Pixels as % of Dynamic Range of Whole Image 4 Nets: Pixels as % of Dynamic Range of Input Patch; Renormalized to Unit Sphere
		Input:	Individual Nets Receive Input from a Single Band Either Band 52 or Band 17
ENSSD5	Ensemble	Level 2:	BP (48-100-16)
		Level 1:	4 BCM Networks: All of size 256-12
		Preprocess:	2 Nets: Pixels as % of Dynamic Range of Whole Image 2 Nets: Pixels as % of Dynamic Range of Input Patch; Renormalized to Unit Sphere
		Input:	Individual Nets Receive Input from a Single Band Either Band 52 or Band 17
SUPSD1	Ensemble of Ensembles	level 2:	BP (48-100-16)
		level 1:	ENSSD1 ENSSD4 ENSSD5

The AVIRIS data used in this set of experiments were comprised of 10 different images derived from 6 different locations under a variety of weather conditions; they included a variety of terrain, for example a land-sea interface and agricultural areas; a number of different cloud types were also present. For each location, 4 bands were made available to us, three in the visible and one in the near infra-red, although the experiments in this phase only used two bands, Band 52 (near infra-red), and Band 17 (visible). Eight images were used for training and two for testing. More complete statistics using the bootstrap method will be obtained in the future. Note that in general setting a single threshold for the entire image will not suffice since this would lead to unacceptably high levels of false alarms in many of the images (Figure 3). During training, images in the training set were selected at random and from each image, 16x16 patches were sampled at random as input to the networks. A number of different input patch sizes have been explored (8x8, 16x16, 32x32 and 64x64) to examine the qualities of the BCM feature vectors on different scales of area, although in the experiments described here for prediction input patches were all 16x16. An example of BCM feature vectors obtained from experiments with different input patch sizes is shown in Figure 4. The figure shows some particular network solutions which were strong edge detectors. Notice that local structure found in smaller patch vectors appears as a sub-component of the larger patch feature vectors. Feature vector structure in

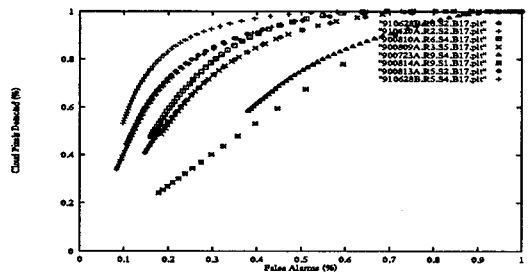


Figure 3: The results of using a global threshold for cloud-masking in AVIRIS Band 17: in many of the images, a good detection rate also means a high false-alarm rate.

BCM takes on a variety of different forms depending on the control parameters η, γ, τ (τ is the temporal width of the sampling window for the sliding threshold $\gamma E(c^2)$ in the BCM rule). Other factors such as dwell-time on each image may also play a role. In experiments with larger values of τ, feature vectors may be banded or speckled (right half of Figure 4) in appearance compared with those in the left half of Figure 4. In the prediction experiments

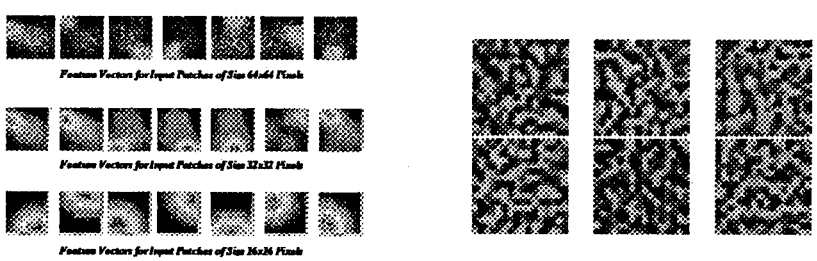


Figure 4: (Left) BCM feature vectors trained with AVIRIS Images (three networks with different patch sizes); vectors are sensitive to edges. (Right) BCM feature vectors sensitive to texture; vectors have been smoothed to emphasize peaks and valleys.

described in the Results section, the final prediction of the network ensembles and single networks was the identity of the pixels in the 4x4 sub-patch in the upper left corner of the input patch. Prediction masks were generated for the training and testing sets by scanning the entire image with the 16x16 pixel input box. The box was moved in 4 pixel increments to obtain a complete prediction mask. This procedure meant that predictions were obtained for all pixels with the exception of a small strip on the bottom and right edges of

the AVIRIS images. These strips comprised only $\sim 2\%$ of the whole image. The results in Table 2 were generated by comparing network prediction masks against masks generated through human interpretation.

Results

BP Performance: Magnitude Sensitivity

The results of the phase 1 experiments are given in Table 2. BP (BPSD1) has a high degree success on the test images because it is fairly similar to those training images for which fairly simple magnitude filtering might obtain fairly good results. However, BP does have some notable failures on the training set for those images for which simple magnitude filtering would not be highly successful, that is, for training images 5, 6 and 7. For these images, BP obtains either unacceptably low rates of classification or high rates of false alarms. Image 5 contains a lot of high magnitude land return which is confused with cloud cover by BP; image 6 has thin cirrus over a land-sea interface, for which backprop fails to detect much of the cloud cover because it has a low return. In both of these cases, the required features vectors need to encode textural information. Similarly, image 7 has bright land as seen through a diffuse layer of smoke from a forest fire. Mistakenly, this image is completely identified as cloud by BPSD1.

BCM-BP Ensemble Performance: Improvements

The ensemble approach allows us to extract features from different preprocessing steps without the problems inherent in constructing a large, single network. Multiple views of data structure from each form of data preprocessing allow improvement in classification perfomance. What is notable about the performance of ensembles incorporating BCM at the lowest levels with BP at the top level is that some of them appear to find solutions which achieve considerably greater classification accuracy than BP alone for images 5, 6 and 7. Indeed all but one of them (ENSSD 4) also achieve a good level of performance on the two testing images.

Ensembles of Ensembles: Most Consistent Performance

Smoothing of the classification estimates can be achieved by learning to pool the estimates of the simple ensembles in the previous section. The ensemble of ensembles run (SUPSD1) which received input from ensembles ENSSD1, ENSSD4 and ENSSD5, achieved the most consistent performance across all of the images comprising the training and testing sets. The classification of cloud location by SUPSD1 for image 6 is compared against the original image and the human interpretation in Figure 5.

Table 2
Pixel-Level Classification of AVIRIS Images

Training Images	% Cloud Pixels	BPSD1	ENSSD1	ENSSD4	ENSSD5	ENSSD6	SUPSD1
		% Cloud Pixels Correct					
		% Non-Cloud Pixels Correct					
910628B.R6.S2	12%	89.8 %	94.3 %	91.2 %	89.6 %	94.8 %	96.0 %
		97.8 %	95.5 %	94.7 %	97.0 %	94.3 %	93.8 %
910620A.R2.S2	39 %	75.7 %	81.4 %	82.8 %	78.6 %	85.1 %	85.1 %
		98.7 %	97.3 %	94.8 %	97.2 %	96.4 %	96.0 %
900810A.R6.S4	33.8 %	98.0 %	97.6 %	91.4 %	95.8 %	99.4 %	98.9 %
		95.0 %	94.9 %	84.1 %	96.2 %	84.9 %	92.3 %
900809A.R3.S5	21.4 %	95.2 %	93.0 %	89.3 %	91.7 %	96.6 %	96.0 %
		94.9 %	97.7 %	94.4 %	98.1 %	94.0 %	96.1 %
900723A.R9.S4	4.9 %	100.0 %	91.4 %	82.9 %	90.9 %	92.1 %	91.8 %
		2.3 %	92.6 %	92.7 %	88.7 %	80.1 %	90.3 %
900814A.R9.S1	51.0 %	31.6 %	86.1 %	62.5 %	35.9 %	63.7 %	88.3 %
		97.8 %	78.5 %	92.5 %	98.2 %	93.4 %	74.6 %
900813A.R9.S3	0.0 %	—	—	—	—	—	—
		0.0 %	68.0 %	98.5 %	70.6 %	56.3 %	83.6 %
900813A.R5.S2	53.6 %	78.3 %	88.9 %	77.9 %	79.5 %	87.1 %	91.6 %
		99.3 %	96.9 %	97.3 %	98.6 %	97.9 %	96.0 %
Testing Images							
910628B.R5.S4	23.2 %	92.7 %	72.4 %	94.2 %	94.5 %	97.8 %	93.3 %
		98.0 %	96.7 %	90.1 %	95.6 %	91.7 %	94.9 %
900814B.R12.S2	100.0 %	100.0 %	98.7 %	18.2 %	81.7 %	79.9 %	84.1 %
		—	—	—	—	—	—

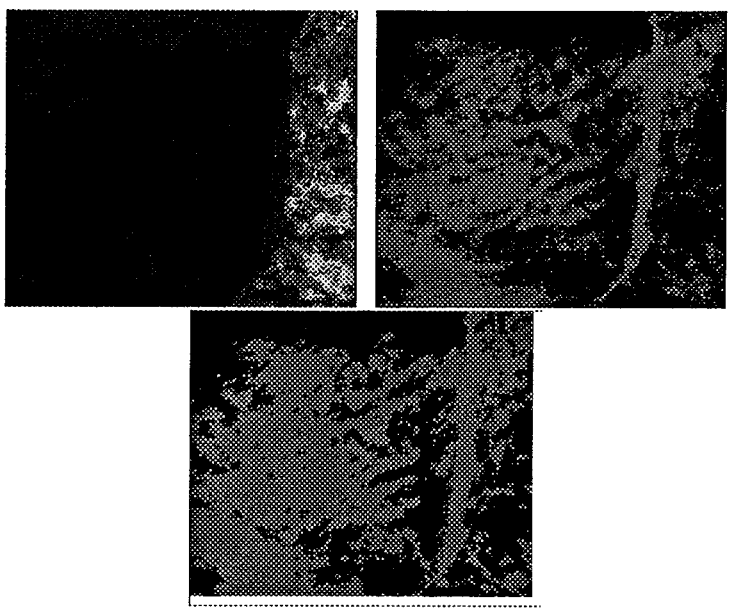


Figure 5: (Upper left) AVIRIS image of cirrus clouds over a land-sea interface (band 52); (upper right) human interpretation of location of clouds ; (bottom) identification of clouds by an ensemble of ensembles (SUPSD1).

Conclusions and Future Directions

BCM Projection Pursuit in an ensemble configuration is capable of discovering textural features which may be useful in separating bright land from cloud, as well as detecting cirrus over a land-sea interface. The ensemble of ensembles run SUPSD1 achieved the most consistent performance across all of the images. BP solutions tended to be very sensitive to overall intensity of return and had notable failures on the training set for these difficult cases. Further experimentation with ensemble configurations of BCM-BP and other hybrid networks such as BCM with radial basis functions [21] [20] to optimize classification performance will be explored in a future publication; we will also study the performance of ensembles receiving inputs from statistical measures derived from GLDV, SADH and GLRL distributions as well as the raw data.

Acknowledgement C. Bachmann gratefully acknowledges the support of Dr. Thomas McKenna of the Office of Naval Research (ONR) through ONR Grant # N0001494WX23060 and also of Fred Lee, AEW Section Head, Airborne Radar Branch, Naval Research Laboratory (NRL), for internal funding under ONR Grant #N0001494WX35052. This work was also supported by NRL core funds (53-1501-04). E. E. Clothiaux is supported by an appointment to the Global Change Distinguished Postdoctoral Fellowships sponsored by the U.S. Department of Energy, Office of Health and Environmental Research, and administered by the Oak Ridge Institute for Science and Education.

References

[1] C. M. Bachmann, S. Musman, D. Luong, and A. Schultz, Unsupervised BCM Projection Pursuit Algorithms for Classification of Simulated Radar Presentations, Neural Networks, Vol. 7, No. 4, pp. 709-728, 1994.

[2] C. M. Bachmann, S. A. Musman, A. Schultz, Classification of Simulated Radar Imagery Using Lateral Inhibition Neural Networks, in Neural Networks for Signal Processing II - Proceedings of the 1992 IEEE Workshop, Aug. 31 - Sept. 2, 1992, Copenhagen, Denmark, pp. 279 - 288.

[3] E. L. Bienenstock, L. N. Cooper, P. W. Munro, Theory for the Development of Neuron Selectivity: Orientation Specificity and Binocular Interaction in Visual Cortex, The Journal of Neuroscience, Vol. 2, No. 1, pp. 32-48, 1982.

[4] D. W. Chen, S.K. Sengupta, and R.M. Welch, Cloud Field Classification Based upon High Spatial Resolution Textural Features: 2. Simplified Vector Approaches, J. Geophys. Res., 94, No. D12, 14749-14765, 1989.

[5] E. E. Clothiaux, M. F. Bear, L. N. Cooper, Synaptic Plasticity in Visual Cortex: Comparison of Theory with Experiment, Journal of Neurophysiology, Vol. 66, No. 5, pp. 1785 - 1804, 1991.

[6] E. Ebert, A Pattern Recognition Technique for Distinguishing Surface and Cloud Types in the Polar Regions, J. Climate Appl. Meteor., 26, 1412-1427, 1987.

[7] E. E. Ebert, Analysis of Polar Clouds from Satellite Imagery Using Pattern Recognition and a Statisitical Cloud Analysis Scheme, Journal of Applied Meteorology, Vol. 28, pp. 382 - 399, 1989.

[8] J. H. Friedman, J. W. Tukey, A Projection Pursuit Algorithm for Exploratory Data Analysis, IEEE Transactions on Computers, Vol. c-23, No. 9, pp. 881 - 890, 1974.

[9] J. H. Friedman,Exploratory Projection Pursuit, Journal of the American Statistical Association, Vol. 82, No. 397, Theory and Methods, pp. 249 - 266, March 1987.

[10] L. Garand, Automated Recognition of Oceanic Cloud Patterns. Part I: Methodology and Application to Cloud Climatology. J. Climate, 1, 20-39, 1988.

[11] C. Genest, J. V. Zidek, Combining Probability Distributions: A Critique and Annotated Bibliography, Statistical Science, Vol. 1, No. 1, pp. 114 - 148, 1986.

[12] L. K. Hansen, P. Salamon, Neural Network Ensembles, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 10, pp. 993 - 1001, October, 1990.

[13] L. K. Hansen, C. Liisberg, P. Salamon, Ensemble Methods for Handwritten Digit Recognition, in Neural Networks for Signal Processing II - Proceedings of the 1992 IEEE Workshop, Aug. 31 - Sept. 2, 1992, Copenhagen, Denmark, pp. 333 - 342.

[14] T. K. Ho, J. J. Hull, S. N. Srihari, On Multiple Classifier Systems for Pattern Recognition, Proceedings of the 11th IAPR International Conference on Pattern Recognition, Vol. II., Conference B: Pattern Recognition Methodology and Systems, pp. 84 -87, 1992.

[15] N. I. Intrator, (1990). Feature Extraction Using an Unsupervised Neural Network, in Proceedings of the 1990 Connectionist Models Summer School, D. S. Touretzky, J. L. Ellman, T. J. Sejnowski (eds.), Morgan Kaufmann, San Mateo, CA, 1990.

[16] N. Intrator, L. N. Cooper, Objective Function Formulation of the BCM Theory of Visual Cortical Plasticity: Statistical Connections, Stability Conditions, Neural Networks, Vol. 5, pp. 3-17, 1992.

[17] J. Key, Cloud Cover Analysis with Arctic Advanced Very High Resolution Radiometer Data. 2. Classification with Spectral and Textural Measures, 95, No. D6, 7661-7675, 1990.

[18] T. Kohonen, Self-Organization and Associative Memory, Springer-Verlag, Berlin, 1984, pp. 128 - 133.

[19] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, Adaptive Mixtures of Local Experts, Neural Computation, Vol 3., pp 79-87, 1991.

[20] J. Moody, C. Darken, Fast Learning in Networks of Locally Tuned Processing Units, Neural Computation, 1(2): pp. 281-294, 1989.

[21] A. J. Robinson, M. Niranjan, F. Fallside, Generalising the Nodes of the Error Propagation Network, Technical Report, CUED/F-INFENG/TR.25, Cambridge University Engineering Department, November 1, 1988.

[22] M. P. Perrone, Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization, Ph.D. Dissertation, Department of Physics, Brown University, May, 1993.

[23] M. P. Perrone, L. N. Cooper, Learning from What's Been Learned: Supervised Learning in Multi-Neural Network Systems, to appear in: Proceedings of the World Congress on Neural Networks, Portland, OR, July 11 - 15, 1993.

[24] M. P. Perrone, L. N. Cooper, When Networks Disagree: Ensemble Methods for Hybrid Neural Networks, to appear in **Artificial Neural Networks for Speech and Vision**, R. J. Mammone, ed., Chapman-Hall, 1993.

[25] M. D. Richard, R. P. Lippman, Neural Network Classifiers Estimate Bayesian a posteriori Probabilities, Neural Computation, 3, 461-483, 1991.

[26] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning Internal Representations by Error Propagation, in Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol. 1, Rumelhart, D. E., McClelland, J. L. (eds.), MIT Press, Cambridge, MA, 1986, pp. 318-362.

[27] B. S. Seebach, Evidence for the Development of Phonetic Property Detectors in a Modified BCM Neural Network without Innate Knowledge of Linguistic Structure, Ph. D. Dissertation, Brown University, Program in Neural Science, 1990.

[28] D. F. Specht, Probabilistic Neural Networks, Neural Networks, Vol. 3, pp. 109-118, 1990.

[29] V. R. Tovinkere, M. Penaloza, A. Logar, An Intercomparison of Artificial Intelligence Approaches for Polar Scene Identification, Journal of Geophysical Research, Vol. 98, No. D3, pp. 5001-5016, March, 20, 1993.

[30] M. Unser, Sum and Difference Histograms for Texture classification. IEEE Trans. Pattern Anal. Mach. Intell., PAMI-8, 118-125, 1986

[31] R. M. Welch, S. K. Sengupta, A. K. Goroch, P. Rabindra, N. Rangaraj, M. S. Navar, Polar Cloud and Surface Classification Using AVHRR Imagery: An Intercomparison of Methods, Journal of Applied Meteorology, Vol. 31, pp. 405-420, May, 1992.

[32] J. S. Weszka, C.R. Dyer, and A. Rosenfeld, A comparative Study of Texture Measures for Terrain Classification, IEEE Trans. Syst., Man, Cybern., SMC-6, 269-285, 1976.

Saddle-node Dynamics for Edge Detection

Yiu-fai Wong*
Institute for Scientific Computing Research, L-416
Lawrence Livermore National Laboratory
Livermore, CA 94550
wong@redhook.llnl.gov

Abstract

We demonstrate how the formulation of a nonlinear scale-space filter can be used for edge detection and junction analysis. By casting edge-preserving filtering in terms of maximizing information content subject to an average cost function, the computed cost at each pixel location becomes a local measure of edgeness. This computation depends on a single scale parameter and the given image data. Unlike previous approaches which require careful tuning of the filter kernels for various types of edges, our scheme is general enough to be able to handle different edges, such as lines, step edges, corners and junctions. Anisotropy in the data is handled automatically by the nonlinear dynamics.

1 Introduction

Edge detection is a basic operation for many image analysis and machine vision systems. It is not surprising that numerous schemes have been invented for various purposes. The earlier schemes such as Roberts and Sobel operators are gradient-based [1]. Over the years, ideas from many different fields have been applied to this problem. For example, one approach is based on surface fitting [2, 3, 4]. Fitting functions resembling edge profiles to the data, one can extract the edges based on the residuals or gradient strengths. Mathematical morphology has also been applied to detect edges [5]. The most popular approaches, however, are based on convolving the images with Gaussian-like kernels. Typically, peaks or ridges in the filtered output are identified as the edges, with some proper thresholding. For example, zero crossings of the Laplacian of Gaussians were considered to be edges in [6]. In his seminal work [7], Canny derived Gaussian-like filters that maximize the SNR and minimize the localization error. Recently, energy filters have been proposed [8]

*Work was performed at the Institute for Scientific Computing Research and was supported by Lawrence Livermore National Laboratory through DOE contract No. W-7405-ENG-48.

in which the squared responses of a pair of quadrature filters are computed and the local maxima were defined to be the edges [8]. To maximally tune the responses of the filters to the edges, it is advantageous to use a set of oriented filters. This idea was implemented in the Binford-Horn linefinder [9].

Corners and junctions are very important for object description and recognition. Since their descriptions are even more complicated than edges, their extraction are harder. There are two types of approaches for corner and junction detection: 1) extracting the edges and then look for points with maxima curvature [10, 11]; and 2) working directly on the grey-scale images [12, 13, 14, 15, 16].

Since edges can be present in multiple orientations and scales, one major drawback of these linear-filter-based techniques is that careful choice of the shapes, orientations and scales of the kernels [17, 18] is required to extract meaningful edges. This becomes much more complicated if accurate detection of corners and junctions is needed.

Recently, a clustering filter [19, 20] that can remove noise, preserve edges and smooth data was derived, all conditioned upon a scale parameter. In this work, we discuss another aspect of the filter. It is shown that the filter contains a mechanism suitable for edge detection and junction analysis.

For completeness, let us review the essential ideas of the filter.

2 Clustering Filter

Let $\mathbf{x}$ be the coordinate of a pixel in an image[1] and y its gray level, or really any real-valued attribute. It is well-known that the pixels are highly redundant due to spatial correlation. Thus, given a scale, we can estimate the new pixel value y at position $\mathbf{x}$ by its neighboring pixels. Common sense tells us that the data points near $(\mathbf{x}, y)$[2] should give more information while those far away should give less. This can be implemented by having each data point contributing to the estimate y pay a cost. The cost function should be small for nearby data points but larger for those further away. To make this estimate robust, the information should be spread among the neighboring data. If we treat the contributions to the determination of y from the neighboring data as a probability distribution, then this probability distribution should be chosen such that its entropy is maximized subject to linear cost constraints [21].

The above reasoning implies that we can estimate each pixel independently. Say we are given a neighborhood of data points $S = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, N\}$. Let P_i denote the contribution of $(\mathbf{x}_i, y_i)$ to $(\mathbf{x}, y)$, or equivalently, the probability that $(\mathbf{x}, y)$ is influenced by $(\mathbf{x}_i, y_i)$. To specify the cost function, we note that the input and output domains should be treated differently because the former is fixed and known but the latter is really environment-

[1] $\mathbf{x} = (i, j)$ for an image with rectangular grids. We choose this notation for simplicity.
[2] We use $(\mathbf{x}, y)$ to denote the joint image plane and intensity space.

dependent. Thus, let the cost function have two components $e_x(\mathbf{x}_i)$ and $e_y(y_i)$. Our criterion seeks to maximize the entropy $S = -\sum_i P_i \log P_i$, subject to the linear constraints

$$\sum_i P_i e_x(\mathbf{x}_i) = C(x), \sum_i P_i e_y(y_i) = E(x) \tag{1}$$

which are obtained by averaging the costs in the neighborhood.

Using Lagrangian multipliers, the contribution of ith pixel to the determination of the filter output y at pixel location x was found to be

$$P_i = e^{-\alpha e_x(\mathbf{x}_i) - \beta e_y(y_i)}/Z \tag{2}$$

where $Z = \sum_i e^{-\alpha e_x(\mathbf{x}_i) - \beta e_y(y_i)}$. Using an analogy with statistical physics, we can define a free energy $F = \frac{-1}{\beta} \log Z$.

If one uses squared distance for the cost functions e_x and e_y, one further obtained that, by minimizing F, the output y at x is given by

$$y = \sum_i \frac{y_i w_i e^{-\beta(y_i - y)^2}}{\sum_j w_j e^{-\beta(y_j - y)^2}}, \tag{3}$$

the weighted mean of the data and $w_i = e^{-\alpha \|\mathbf{x}_i - \mathbf{x}\|_2^2}$.

Let us now explain what α means. Clearly, a large α implies that only the pixels very close to $\mathbf{x}$ have significantly non-zero w_i's. Thus, only a few data points can influence the output. Conversely, a small α implies that more neighbors of $\mathbf{x}$ can contribute. Hence, α is a measure of scale in the input space.

Once the scale α in the input space is selected, it is clear that the particular estimate one obtains depends on β and initial y. A simple procedure was used in [20]. Let us compute the mean $\bar{y} = \sum_i y_i w_i / \sum_i w_i$ and variance $\sigma_y^2 = \sum_i (y_i - \bar{y})^2 w_i / \sum_i w_i$. One then sets $\beta = (2\sigma_y^2)^{-1}$. To compute the filter output, one simply iterates (3), with initial $y = \bar{y}$, until it converges.

It has been observed that this filter can accomplish three tasks [19, 20]:

- removing impulsive noise;
- improved smoothing of nonimpulsive noise and
- preserving edges.

3 Approach for Edge and Junction Analysis

The clustering filter uses a new mechanism, namely, saddle-node dynamics, for edge-preserving filtering. Can we use this new mechanism for edge detection? A key observation here is that the energy (cost) function

$$E(x) = \sum_i (y - y_i)^2 P_i \tag{4}$$

is a measure of edgyness at x in the image data. The energy is small for smooth areas and large for areas containing "edges." The reasoning is as follows: Over a smooth area, the pixels in a neighboring area are highly correlated. Thus, the cost of having a smooth estimate is low. At an edge, the usual notion of spatial redundancy breaks down and strong nonlinear action is needed to preserve an edge. Thus, the cost is high.

Furthermore, the energy is invariant to the orientation of an edge. Imagine that one rotates an edge profile within a circular region. The contributions P_i's are rotated too. Thus, both $E(x)$ and y will remain unchanged. Our formulation gives a response which is orientation invariant. It is also unnecessary to tune the shape of the neighborhood used in the nonlinear filtering.

We now illustrate this observation with some synthetic and real images. For visualization purposes, the square root of the energy image E is shown, unless other specified. When the outputs are generated for different αs, no scaling is performed. This allows one to compare the magnitudes of the outputs. White means small and dark means large in the figures.

3.1 Energy at Lines

Figure 1a is an image with a horizontal line and a pair of crossed lines. Figures 1b and 1c are the energy images for $\alpha = 1/2, 1/8$ respectively. One can make three observations:

- The energies are highly localized along the lines and their terminals.
- At the corners and junctions, the energies become local maxima. This can serve as a scheme to corner and junction detection.
- The ridges in the energy image are the edges.

3.2 Energy at Step Edges and Corners

Figure 2a shows an image with step edges and corners. Figures 2b and 2c show the energy images for $\alpha = 1/2, 1/8$ respectively. Again, one can see that the energies are concentrated at the boundaries which can clearly be identified by the ridges. At the corners, the energies are distinctly local maxima. At a larger scale, two local maxima are generated at the corners. They can be merged by noting that there cannot be two adjacent corners at a large scale.

3.3 Energy at Junctions

Junctions pose great difficulty for Canny-like edge detectors. The behavior of the edges obtained by linear filters is very complicated at junctions because an ideal junction can be characterized by four parameters. To demonstrate that the energy image can be useful for detecting such junctions, Figure 3a shows a corner and Figure 3b is the filtered image. Figures 3c and 3d show the energies at $\alpha = 1/2$ and $1/8$ respectively. Indeed, the ridges are the edges and they meet at the junction, the energy of which is a local maxima.

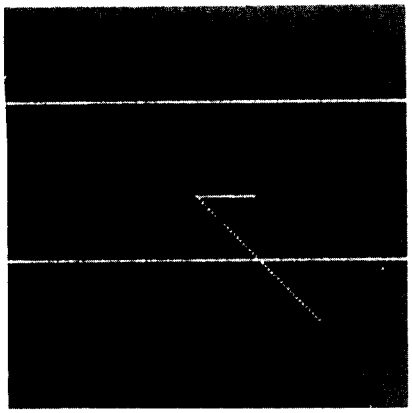


Figure 1a. Line image.

Figure 1b. Energy image, a=1/2.

Figure 1c. Energy image, a=1/8.

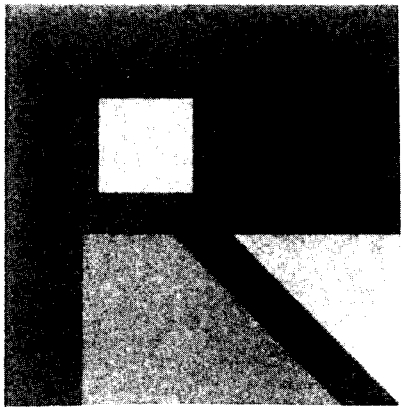


Figure 2a. Step edges and Corners.

Figure 2b. Energy image, a=1/2.

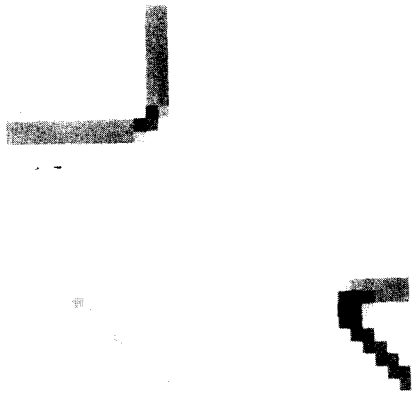


Figure 2c. Energy image, a=1/8.

3.4 Results using real data

Edges in real data differ from the synthetic ones considerably due to sampling. A step edge has nonzero components at all frequencies. For two-dimensional images, depending on the sampling periods in the x and y directions and the edge's orientation, sampled edges can appear jagged. Thus, care must be taken in using our formulation because the filter is nonlinear. Edges are detected as follows: For each pixel, it is determined if the energy is a local maximum in any of the four directions. If the energy is a local maximum in one or more directions, the maximum gradient among these directions is computed. Otherwise the gradient is set to zero. One then thresholds this gradient image to get the edges. Figure 4a shows a girl image. Figure 4b shows the edges extracted by our method. Figure 5a shows a MR image of a human hand. Figure 5b shows the edges extracted.

4 Summary

We have shown how the formulation of a nonlinear scale-space filter can be utilized for edge and junction analysis. Using the nonlinear filter, an energy value can be calculated at each pixel. Energy is large at an edge and small over smooth areas. Moreover, it is invariant to the orientation of an edge. It was observed that the ridges correspond to edges and the local maxima correspond to the corners and junctions. Our main contribution is that the use of saddle-node dynamics allows us to perform tasks quite effortlessly which would require careful tuning of the shapes and orientation of the filter kernels in conventional methods. Future work would include more detailed experiments and integration of edges over scales to generate a complete and robust edge detection scheme.

References

[1] R.C. Gonzalez and P. Wintz, *Digital Image Processing*, Addition-Wesley, Menlo Park, CA, 1987.

[2] R.M. Haralick, "Digital edge steps from zero crossings of second directional derivatives," *IEEE Trans. Pattern Anal. Mach. Intell.*, 6, 58-68, 1984.

[3] V.S. Nalwa and T.O. Binford, "On detecting edges," *IEEE Trans. Pattern Anal. Mach. Intell.*, 8, 699-714, 1986.

[4] D. Lee, "Coping with discontinuities in computer vision: their detection, classification and measurement," *IEEE Trans. Pattern Anal. Mach. Intell.*, 12, 321-344, 1990.

[5] R.M. Haralick, J.S.J. Lee and L.G. Shapiro, "Morphologic edge detection," *IEEE J. Rob. Automat.*, RA-3, 1987.

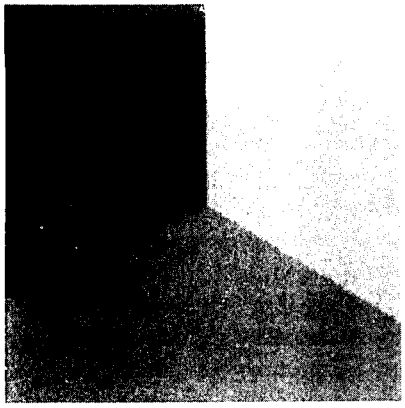


Figure 3a. Junction image.

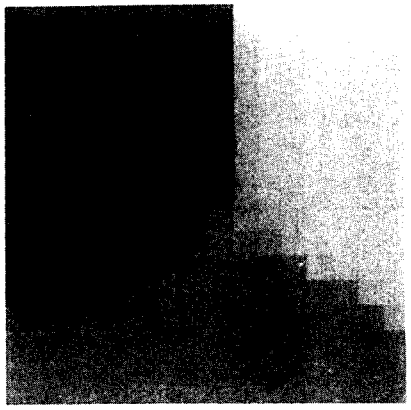


Figure 3b. Filtered image, a=1/2.

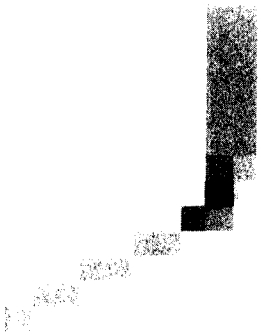


Figure 3c. Energy image, a=1/2.

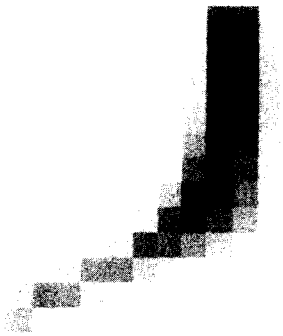


Figure 3d. Energy image, a=1/8.

[6] D.C. Marr and E.C. Hildreth, "Theory of Edge Detection," *Proc. Royal Soc. London B*, 207, 187-207, 1980.

[7] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Patt. Anal. and Mach. Intell.*, 6, 679-698, 1986.

[8] M.C. Morrone and R.A. Owens, "Feature detection from local energy," *Pattern Recognition Letters*, 6, 303-313, 1987.

[9] T. Binford, "Inferring surfaces from images," *Artificial Intelligence*, 17, 205-244, 1981.

[10] H. Asada and M. Brady, "The curvature primal sketch," *IEEE Trans. Pattern Anal. Mach. Intell.*, 8, 2-14, 1986.

[11] R. Deriche and O.D. Faugeras, "2-D curve matching using high curvature points: application to stereo vision," *Proc. 10th Intl. Conf. Patt. Recog.*, Atlantic City, 240-242, 1990.

[12] P.R. Beaudet, "Rotational invariant image operators," *4th Intl. Conf. Patt. Recog.*, Tokyo, 579-583, 1978.

[13] L. Kitchen and A. Rosenfeld, "Gray-level corner detection," *Patt. Recog. Lett.*, 1, 95-102, 1982.

[14] L. Dreschler and H.H. nagel, "On the selection of critical points and local curvature extrema of region boundaries for interframe matching," *Int. Conf. Patt. Recog.*, 542-544, 1982.

[15] O.A. Zuniga and R.M. Haralick, "Corner detection using the facet model," *Proc. Conf. Patt. Recog. Image Process.*, 30-37, 1983.

[16] R. Deriche and G. Giraudon, "A computational approach for corner and vertex detection," *Intl. Journal Computer Vision,*, 10, 101-124, 1993.

[17] W. Freeman and E. Adelson, "The design and use of steerable filters for image analysis, enhancement and multi-scale representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 1991.

[18] P. Perona, "Steerable-scalable kernels for edge detection and junction analysis," *Proc. ECCV*, 1-17, 1992.

[19] Yiu-fai Wong, "A Nonlinear Scale-space Filter by Physical Computation," *Proc. IEEE Workshop on Neural Networks for Signal Processing*, September, 1993.

[20] Yiu-fai Wong, "Nonlinear Scale-space Filtering and Multiresolution Systems," to appear in *IEEE Trans. Image Processing.*

[21] A. Papoulis, , *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, Ch. 15, 1984.

Figure 4a. Girl image.

Figure 4b. Edges from Fig. 4a.

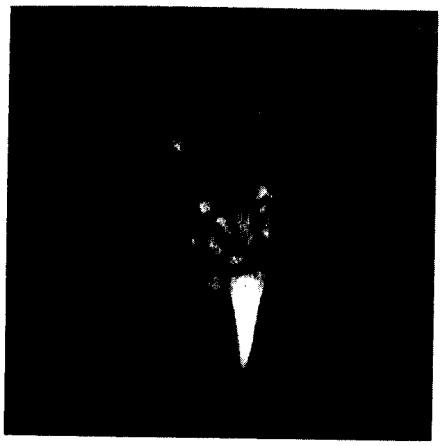


Figure 5a. MR Hand image.

Figure 5b. Edges from Fig. 5a.

Application of SVD Networks to Multi-Object Motion-Shape Analysis

S.Y. Kung
Princeton University

J.S. Taur M.Y. Chiu
Siemens Corporate Research

1 Introduction

Singular Value Decomposition (SVD) has been a well-known technique for signal/image processing. Recently, an SVD approach to the so-called *structure-from-motion* problem was proposed by Tomasi and Kanade [3], which has attracted a lot of renewed attention. For the single object case, Kanade and colleagues have devised a sequential algorithm so that it would be able to recover the scene in real time as the video images are taken. So this is called a *motion-shape estimation* (MSE) problem. The main thrust of this paper is to further evolve the single object MSE to multi-object MSE problem. *Given a sequence of 2-D video images of multiple moving objects, the problem is to track the 3-D motion of the objects and reconstruct their 3-D shapes.* After selection of initial *feature points* (FPs), the SVD may be applied to a measurement matrix formed by the FPs sequentially tracked by a video system. The distribution of singular values would first reveal the information about the number of objects at hand. Then, using an algebraic-based *subspace clustering method*, the FPs may be mapped onto their corresponding objects. Thereafter, the motion and shape may be estimated from a matrix factorization.

Our method hinges upon the numerical effectiveness and stability of the SVD factorization. In addition, for potential real-time application, we shall stress the importance of recursively extraction of the *principal components* by e.g. adaptive neural architectures. A parallel processing *APEX* neural model, for example, may provide a very attractive implementation[1] .

2 SVD Analysis for Moving Objects

With reference to Figure 1, where the coordinate of a video imaging system is sketched. Here we assume an *orthographic projection* of a 3-D object point onto an FP on the image plane. [1] For convenience of notation, we also assume that the rotation center coincides with the (local) coordinate system pertaining to an object A. Let $\mathbf{a}(p)$, $p = 1, \ldots, P$, denote the 3-D position vector of any feature point on the object.

[1]Note that the rank theorem, as originally proposed by Tomasi and Kanade, relies on the assumption of orthographic projection[3]. however, the result was later extended and verified to be valid also for the para-perspective projection model[2].

0-7803-2026-3/94 $4.00 © 1994 IEEE

Given any time (or frame) f, where $f = 1, \ldots, F$, this 3-D location may be projected onto the 2-D image plane of the camera. The x-coordinate's value of this feature point (FP) is

$$w^i(f,p) = \mathbf{i}R_a(f)\mathbf{a}(p) + \mathbf{i}t_a(f) \tag{1}$$

where the vectors $\mathbf{i} = [1\ 0\ 0]$ and $R_a(f)$, $t_a(f)$ respectively denote rotational matrix and the translational vector. Likewise, for the y-coordinate in the image plane,

$$w^j(f,p) = \mathbf{j}R_a(f)\mathbf{a}(p) + \mathbf{j}t_a(f) \tag{2}$$

where the vector $\mathbf{j} = [0\ 1\ 0]$. Now we construct an expanded matrix (with $\mathbf{E}_a = [1\ 1\ 1\ 1\ 1 \cdots 1\ 1\ ]$)

$$\mathbf{W}_a^i \quad = \quad \{w^i(f,p)\} = \mathbf{R}_a^i\mathbf{S}_a + \mathbf{T}_a^i\mathbf{E}_a \tag{3}$$

and the motion matrices

$$\mathbf{R}_a^i = [\mathbf{i}R_a(1)^T|\mathbf{i}R_a(2)^T|\cdots|\mathbf{i}R_a(F)^T]^T \quad \text{and} \quad \mathbf{T}_a^i = [\mathbf{i}t_a(1)^T|\mathbf{i}t_a(2)^T|\cdots|\mathbf{i}t_a(F)^T]^T$$

and the shape matrix $\mathbf{S}_a = [a(1)|a(2)|\cdots|a(P)]$ is formed from all the feature vectors of object A.

For the y-axis, a similar matrix is formed:

$$\mathbf{W}_a^j = \mathbf{R}_a^j\mathbf{S}_a + \mathbf{T}_a^j\mathbf{E}_a \tag{4}$$

Stacking the x- and y- image measurement matrices, we obtain

$$\mathbf{W}_a = \left[\frac{\mathbf{W}_a^i}{\mathbf{W}_a^j}\right] = \mathbf{R}_a\mathbf{S}_a + \mathbf{T}_a\mathbf{E}_a \tag{5}$$

where

$$\mathbf{R}_a = \left[\frac{\mathbf{R}_a^i}{\mathbf{R}_a^j}\right] \quad \text{and} \quad \mathbf{T}_a = \left[\frac{\mathbf{T}_a^i}{\mathbf{T}_a^j}\right]$$

The dimensions of the matrices $\mathbf{W}_a$ and $\mathbf{R}_a$ are 2F x P and 2F x 3, respectively.

Let us further define $\mathbf{M} = [\mathbf{T}_a|\mathbf{R}_a]$ and $\mathbf{S} = \left[\frac{\mathbf{E}_a}{\mathbf{S}_a}\right]$, then the measurement matrix (for a single object) can be expressed as

$$\mathbf{W} = \mathbf{W}_a \quad = \quad \mathbf{MS} \tag{6}$$

The dimensions of the matrices $\mathbf{W}$, $\mathbf{M}$ and $\mathbf{S}$ are 2Fx P, 2Fx4, and 4xP respectively. Eq. 6 suggests an important rank property stated below[3]:

Theorem 1 (Rank Theorem for Single-Object Case) *(a) The matrix* $\mathbf{W}$ *given in Eq. 6 has a generic rank of 4. (b) Subtracting the average (over all the FPs) of the measurement matrix* $\mathbf{W}$*, then the resulting matrix* $\tilde{\mathbf{W}}$ *(given in Eq. 7) will have a generic rank of 3.*

■

As to Part (a), the total rank is obviously 4 with $\mathbf{S}_a$ contributing rank 3 and E_a rank 1. Part (b) is closely related to an earlier obervation by Ullman[4], which suggested that *three pictures of four points of a rigid body determine its structure and motion.* The rank is reduced because translational component may be effectively removed by subtracting the average of the measurement matrix:

$$\tilde{\mathbf{W}} = \mathbf{R}_a\mathbf{S}_a + \mathbf{T}_a\mathbf{E}_a - \mathbf{R}_a\overline{\mathbf{S}}_a - \mathbf{T}_a\overline{E}_a = \mathbf{R}_a\left[\mathbf{S}_a - \overline{\mathbf{S}}_a\right] \tag{7}$$

where $\overline{\mathbf{S}}_a = 0$. The elements of $\mathbf{E}_a$ are identically 1, therefore, so are $\overline{\mathbf{E}}_a$.

SVD for Multi-Object MSE Problem For the multi-object MSE problem, a new challenge arises in having to distinguish the FPs of adjacent objects so that they may be correctly classified into their corresponding objects. Let us for simplicity concentrate on an example with three objects, A, B, C, whose shapes are denoted by $\mathbf{S}_a$, $\mathbf{S}_b$, and $\mathbf{S}_c$. Just like the single-object case, the measurement matrices for A, B, and C, are respectively:

$$\mathbf{W}_a = \mathbf{R}_a\mathbf{S}_a + \mathbf{T}_a\mathbf{E}_a$$

$$\mathbf{W}_b = \mathbf{R}_b\mathbf{S}_b + \mathbf{T}_b\mathbf{E}_b$$

$$\mathbf{W}_c = \mathbf{R}_c\mathbf{S}_c + \mathbf{T}_c\mathbf{E}_c$$

The row vectors $\mathbf{E}_a$, $\mathbf{E}_b$, $\mathbf{E}_c$ have dimensions P, Q, and R, respectively. Concatenating these matrices, we have

$$\mathbf{W} = [\mathbf{W}_a|\mathbf{W}_b|\mathbf{W}_c] \quad = \quad [\mathbf{T}_a|\mathbf{R}_a|\mathbf{T}_b|\mathbf{R}_b|\mathbf{T}_c|\mathbf{R}_c] \left[\begin{array}{c|c|c} \mathbf{E}_a & 0 & 0 \\ \hline \mathbf{S}_a & 0 & 0 \\ \hline 0 & \mathbf{E}_b & 0 \\ \hline 0 & \mathbf{S}_b & 0 \\ \hline 0 & 0 & \mathbf{E}_c \\ \hline 0 & 0 & \mathbf{S}_c \end{array}\right] \equiv \mathbf{MS} \tag{8}$$

Main Theorem for Reclustering of Feature Points Two points are noteworthy: (1) The dimensions of the matrices $\mathbf{W}$, $\mathbf{M}$, and $\mathbf{S}$ are 2F x (P + Q + R) , 2F x $4k$, and $4k$ x (P + Q + R) respectively. Here k denotes the number of the objects. In this case, $k = 3$, so that rank of a noise-free measurement matrix W should be exactly $4k = 12$. (2) The above representation is deceptively simplified, because it assumed that the columns from each of the objects are pre-aligned in correct clusters (as shown in Eq. 8). In reality, this is seldomly the case. Assuming now that the FPs are not in a correct order, so a very first task is to recluster the FPs such that FPs corresponding to the same object may be regrouped. (The more complete collection of the FPs would mean a more complete 3-D display of the object.) *The main theme of this paper is to demonstrate that, under noisy environment, SVD again offers an effective reclustering technique.* Mathematically, the task is to identify the correct mapping from the FPs onto the matching objects. Assume that the measurement matrix $\mathbf{W}$ turns into full rank due to noise corruption and its SVD is

$$\mathbf{W} = \overline{U}\ \overline{\Sigma}\ \overline{V} = U\Sigma V + U'\Sigma' V'$$

we first remove the "noise" singular values Σ'. Approximately

$$\mathbf{W} \approx U\Sigma V = U\Sigma^{1/2}\Sigma^{1/2}V \tag{9}$$

Let us assume that the SNR is sufficiently large (ideally noise-free) and that each (rigid-body) object comprises at least 4 or more *linear independent* FPs and has a total freedom of 3-D (rotational and translational) motion. Then the following theorem is valid:

Theorem 2 (Subspace Rank Property) *Compute the SVD of* $\mathbf{W}$ *and obtain* U, Σ, V *as given in Eq. 8.*

(1) Total Rank: The total number of (numerically) nonzero singular values in Σ *(i.e. those attributed to the objects) will be* $4k$, *where* k *is the number of objects. Here an object is by definition a rigid body.*

(2) Inclusive Rank Property: If the column vectors of the matrix V *(or, equivalently,* $\Sigma^{1/2}V$*) is correctly grouped into* k *clusters, each corresponding to one object, and the correctly permuted matrix is rewritten as*

$$V \equiv [V_a|V_b|V_c] \tag{10}$$

then each submatrix V_a, V_b, *and* V_c *has (generically) rank 4.*

(3) Exclusive Rank Property: Due to a mutual orthogonality property, any mixture of column vectors from different objects would generically cause the submatrix (comprising of columns from more than one objects) to exceed rank 4. In other words, no column of V_a *may fall in the span of the submatrix of* V_b, *and vice versa. Generically, any mixture of (5 or more) columns from* V_a *and* V_b *would cause the rank to exceed 4. Careful observation of this property could prevent over-subscribing alien or unwanted columns into an object.*

(4) Uniqueness Property: The inclusive and exclusive rank properties together guarantee the uniqueness of the solution.

■

Proof: The proof is largely by inspecting Eq. 8. In particular, we note that $\Sigma^{1/2}V = Q\mathbf{S}$ for some nonsingular matrix Q. Therefore, $V_a = Q\left[\mathbf{E}_a^T|\mathbf{S}_a^T|0|0|0|0\right]^T$. Since Q is nonsingular, thus V_a must have rank 4. Eq. 8 also indicates the mutual rank independency between V_a and V_b, thus verify part (3). Part (4) follows naturally Parts (2) and (3).

Extraction of Motion-Shape Factorization Once an object (say V_a) is properly aggregated, then the next step is reconstruct its 3-D motion/shape. This is done by applying the QR transfformation (X) on the matrix $V_a[I - \frac{1}{P}\mathbf{E}_a^T\mathbf{E}_a]$. According to Theorem 1, there will be three nonzero rows (the first three) which define the matrix $\mathbf{S}'_a$ as shown below:

$$XV_a\left[I - \frac{1}{P}\mathbf{E}_a^T\mathbf{E}_a\right] = \left[\frac{\mathbf{S}'_a}{0}\right]$$

Apply an inverse transformation X^{-1} to $U\Sigma^{1/2}$, resulting in

$$U\Sigma^{1/2}X^{-1} = [\mathbf{R}'_a|Y]$$

By duality, the matrix $\mathbf{R}'_a$ should also be formed from the first three columns. Thus a rotation-shape factorizaton can be obtained as $\mathbf{R}'_a\mathbf{S}'_a$, which may be further transformed to an exact factorization: $\mathbf{R}_a\mathbf{S}_a$[3].

3 Subspace Clustering Problem

The rank theorem prescribes the common bound shared by FPs from the same (rigid) object. This motivates a general algebraic framework formulated in a so-called a *subspace clustering problem.* This formulation has potential applications including, but not limited to, the MSE problem.

Definition 1 (Subspace Clustering Problem) *Given a set of feature vectors $V = \{ v_i \}$, the problem is to find all the (rank-r) objects in V by identifying their corresponding subsets of feature vectors. Here, a rank-r object is defined as a subset of V which forms a rank-r subspace.*

For example, $r = 4$ in the multi-object MSE application. If there are three moving objects, then the number of objects is $k = 3$.

Algorithm 1 (Subspace Clustering Method) *For the noise-free case, the following steps may be adopted:*

1. *Determine a pool of basis vectors S as a maximally linearly independent subset of V. Generically, S should contain exactly $k \times r$ basis vectors.*

2. *A subset of r basis vectors in S will be incorporated into a partnership if there exists at least one vector in V, but not in S, which falls on the span of the subset. The justification of forming such a partnership is that, due to the exclusive rank property, if $r+1$ vectors fall in a span of rank-r subspace, then they could not possibly be from a mixture of two objects, i.e they belong to the same object. (For notational convenience, the r basis vectors shall be called major members in the partnership.)*

3. *Attract other minor members to join the partnership. By the inclusive rank property, a vector is elected to membership if and only if it falls on the span of the r basis vectors (i.e. major members).*

4. *Continue the process until all the membership for the k objects (i.e. partnerships) are identified.*

Example 1 (Noise-Free Case) *Here for simplicity the object rank is set to be $r = 2$. Given a set of vectors $A_1, B_1, B_2, C_1, A_2, A_3, C_2, B_3, A_4, B_4, C_3, \cdots = \{v_i, i = 1, 2, \cdots\}$, from objects A, B, C, then the clustering process can be illustrated by the following:*

```
Vector Dependence     Basis Pool          Partnership

1      No             1,                  -
2      No             1,2,                -
3      No             1,2,3,              -
4      No             1,2,3,4,            -
5      No             1,2,3,4,5           -
6      Yes            (The 6-th vector is excluded from the pool.)
7      No             1,5 ,2,3,4,7      -
-- This completes basis pool and the membership drive begins here:
6      Yes(on 1,5)    (1,5),2,3,4,7       (1,5 | 6): partnership induced by 6
8      Yes(on 2,3)    (1,5),(2,3),4,7     (2,3 | 8): partnership induced by 8
9      Yes(on 1,5)    (1,5),(2,3),4,7     (1,5 | 6,9)
10     Yes(on 2,3)    (1,5),(2,3),4,7     (2,3 | 8,10)
11     Yes(on 4,7)    (1,5),(2,3),(4,7)   (4,7 |11): partnership induced by 11
```

The final clustering result is that the vectors (1,5, 6,9,...) form one object (say, A), (2,3, 8,10, ...) form another object (B), and (4,7, 11, ...) yet the third (C).

Numerical Consideration to Account Noise Effect

1. To improve numerical behavior, the basis vectors should be numerically as nonsingular as possible. Here the "numerical nonsingularity" is measured by the smallest singular value associated with the basis vectors. This would result in a more stable linear dependency check.

2. It is not necessary to identify all the objects in on shot, it may be done sequentially. This is important, since the smallest singular value associated with the basis vectors usually decrease (rapidly) with increasing number of basis vectors. So when the number of objects is too large, it may be difficult to form a complete set of basis vectors with a decent smallest singular value. It is then advisable to use only a partial basis set which offers a better and more comfortable "numerical nonsingularity". As long as the partial basis set contains the r basis vectors needed for at least one object, then all the (minor) members of that object may be identified afterward. The members of the first object may be removed from the set V, before the search process for the second object is started.

3. A confidence measure on linear dependency may be estimated and used. This concept is elaborated further in this subsection. Under noisy situation, a *total least square solution*, based on SVD, can be used to determine a confidence level of numerical linear dependency. The membership check should take into account the confidence.

A Confidence Measure Under the practical and noisy situation, it is more meaningful to ask "is there an approximate linear dependency, and if so, how close?" The answer to this question is no longer straightforward. The complexity hinges upon the criterion adopted. One popular approach is to have A perturbed by a perturbation matrix Δ, such that linear dependency comes to existence. Suppose that the SVD of $A = U\Sigma V^T = \sum_{i=1}^{m+1} u_i\sigma_i v_i^T$. Then, by setting $\Delta = -u_{m+1}\sigma_{m+1}v_{m+1}^T$ it would make

$$[A + \Delta] = \sum_{i=1}^{m} u_i\sigma_i v_i^T \tag{11}$$

to have rank deficiency. It further implies that $[A + \Delta]$ is the closest approximation of A with rank at most m. By Eq. 11, we note that $[A + \Delta]v_{m+1} = 0$, so the "best" normalized null-space solution is simply $x = v_{m+1}$. In summary,

1. The last singular vector of A,v_{m+1}, provides a critical information on the most likely dependency existing in A.

2. The last singular value σ_{m+1} gives a quantitative measure on the confidence of such a linear dependency. (The smaller σ_{m+1} is the higher the confidence, since it is closer to linear dependency.)

4 Simulation Results

Example 2 (Four Moving Objects) *The objects considered in the simulation consist of two cylinders, one block and one pyramid. There are 20 feature points on the cylinders and the block and 10 points on the pyramid. The order of the feature points is randomly permuted. In the duration of 50 frames, all objects are rotating independently. One frame of the orthographic projection of the four objects is shown in Figure 2, which in the appearance is not easily separable, at least not by conventional clustering algorithms. is depicted in Figure 3 (b), which indicates a substantial drop on the 17th singular value. Therefore, the rank is 16 and the number of objects is 4, just as predicted. After applying subspace clustering method, we can obtain four different groups of column vectors* $[V_aV_bV_cV_d]$ *for the feature points. Then the translation-rotation decomposition is used to obtain the shape information. The result is shown in Figure 4.*

Example 3 (Two Moving Objects with Noisy Measurements) *Experiments on two moving objects with noisy measurement have been conducted. Preliminary study shows that the subspace clustering depends very much on the numerical behaviour. Under noisy situation, a* **total least square solution** *is found to be effective in determining a good threshold for checking (numerical) linear dependency. This however incurs the use of the computationally more demanding SVD technique for the dependency check. Nevertheless, preliminary study shows that the total-least-square based subspace clustering method can cope with 1-5% noise tolerance on the FP measurements.*

5 Multi-Camera Multi-Object Analysis

In many application domain, images via multiple cameras can offer vital information. A prominent example is that of shape reconstruction of an occluded object. Suppose that during the period of video recording, an object is only partially seen by the first camera, but its incomplete viewing angle can be compensated by a second camera. Under this scenario, it is essential that the information from the two cameras be "fused" for the reconstruction of a complete 3-D shape. On the other hand, one must also prevent FPs from different objects being mixed.

To this end, the **subspace clustering method** offers a simple solution.

Without loss of generality, let us assume that the first camera is (like before) located at the origin [0,0,0] with the direction of the imaging plane defined by its normal vector [0,0,1]. A second camera, located at a new location at $\mathbf{m} = [m_i, m_j, m_k]$, has its own image plane defined by a new normal vector $[k_1, k_2, k_3]$. Since $\mathbf{m}$ is known before hand and remains constant, so its shift effect can be removed by first pre-shifting the FPs recorded on the second camera. Therefore, without loss of generality, we shall simply pre-align the FPs of the second camera so that in the following derivation it may be considered in effect $\mathbf{m} = 0$.

There exists a common viewing angles from the two cameras, since two image planes (assuming non-parallel) must intersect on one line, denoted by $\mathbf{l}$, which is orthogonal to both the normal vectors.

$$\mathbf{l}^T[0,\ 0,\ 1] = 0 \quad \text{and} \quad \mathbf{l}^T[k_1, k_2, k_3] = 0$$

This yields a solution

$$\mathbf{l} = [k_2, -k_1, 0]$$

Just like Eqns. 1 and 2, along the line $\mathbf{l}$, the FP is recorded as

$$w^l(f,p) = \mathbf{l}R_a(f)\mathbf{a}(p) + \mathbf{l}t_a(f) \tag{12}$$

Based on this we construct the measurement matrix for the first camera

$$\mathbf{W}_a^l \quad = \quad \{w^l(f,p)\} = \mathbf{R}_a^l\mathbf{S}_a + \mathbf{T}_a^l\mathbf{E}_a \tag{13}$$

Similarly for the second camera, we have another matrix

$$\overline{\mathbf{W}}_a^l = \mathbf{R}_a^l\overline{\mathbf{S}}_a + \mathbf{T}_a^l\overline{\mathbf{E}}_a \tag{14}$$

Assuming two objects (A and B), then the toal measurement matrix becomes

$$[\mathbf{W}_a^l|\overline{\mathbf{W}}_a^l|\mathbf{W}_b^l|\overline{\mathbf{W}}_b^l] \quad = \quad [\mathbf{T}_a^l|\mathbf{R}_a^l|\mathbf{T}_b^l|\mathbf{R}_b^l] \left[\begin{array}{c|c} \mathbf{E}_a|\overline{\mathbf{E}}_a & 0 \\ \hline \mathbf{S}_a|\overline{\mathbf{S}}_a & 0 \\ \hline 0 & \mathbf{E}_b|\overline{\mathbf{E}}_b \\ \hline 0 & \mathbf{S}_b|\overline{\mathbf{S}}_b \end{array}\right] \tag{15}$$

By inspection, it should be clear that all the multiple-object rank properties in Theorem 2 and the same *Subspace Clustering Method* remain applicable.

Conclusion Even though the derivation in this paper is based on the orthographic projection, it can be shown that the SVD analysis for multiple moving objects is also applicable for the case of para-perspective projection[2]. As a matter of fact, for a single object A, Eq. 1 becomes

$$w^i(f,p) = \frac{l}{t_a^k(f,p)}[(\mathbf{i}R_a(f,p) - \frac{R_a^k(f,p)t_a^i(f,p)}{t_a^k(f,p)})\mathbf{a} + t_a^i(f,p)] \qquad (16)$$

for the para-perspective projection. Here l is the focal length of the camera. From Eq. 16 all major rank theorems described above are valid except that $\mathbf{i}R_a(f,p)$ is replaced by $\mathbf{i}R_a(f,p) - \frac{R_a^k(f,p)t_a^i(f,p)}{t_a^k(f,p)}$. Most importantly, Eq. 8 remains the same for multiple objects, which means the main theorem for feature points clustering is valid. In the experiments conducted by Poelman and Kanade[2] for a single moving object, the para-perspective method in general performs significantly better than the orthographic factorization method. Based on the simulational study at hand, we are convinced that the SVD factorization method will lead to a robust solution to multi-object motion-shape analysis. We plan to test the proposed multi-object factorization method with sequences of laboratory-calibrated and real outdoor digital video images. The experimental results will be reported in a future publication.

References

[1] S. Y. Kung and K. I. Diamantaras. A neural network learning algorithm for adaptive principal component extraction (APEX). In *Proceedings, ICAASP*, pages 861–864, April 1990.

[2] C. J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. Technical Report CMU-CS-92-208, CMU, 1992.

[3] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *Internation Journal of Computer Vision*, 9(2):330–334, 1992.

[4] S. Ullman. *The Interpretation of Visual Motion.* MIT Press, 1979.

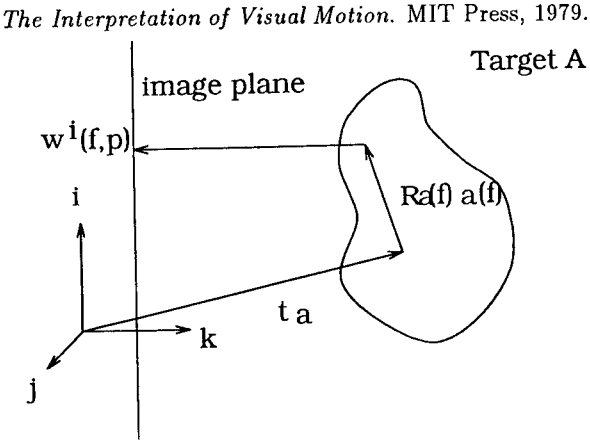


Figure 1: Coordinate system of moving objects w.r.t. camera image plane. Shown here is one of P feature points of Object A.

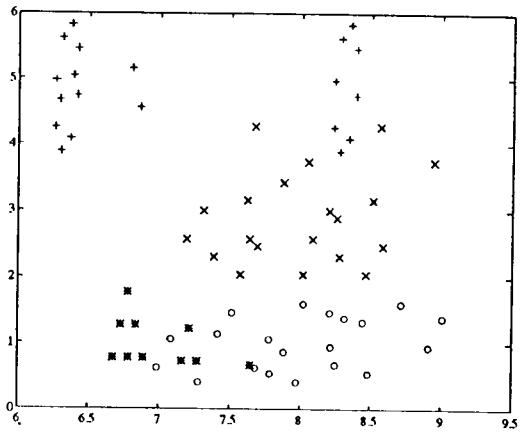


Figure 2: This figure shows the feature points on four targets used in the simulation. Points on different targets are denoted as different symbols.

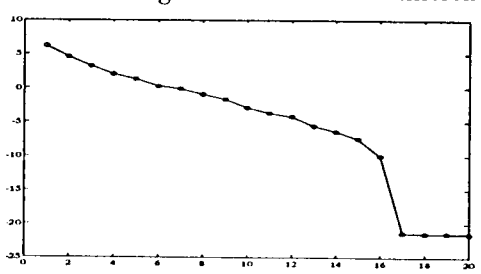


Figure 3: The log-scale of the singular values in the simulation.

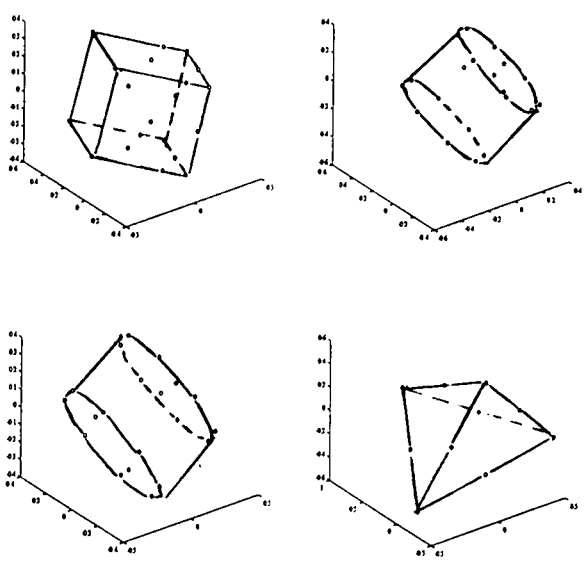


Figure 4: This figure shows the shapes of the four targets in the simulation.

NEURAL NETWORKS FOR ROBUST IMAGE FEATURE CLASSIFICATION: A COMPARATIVE STUDY

Sharma V R Madiraju and Chih-Chiang Liu
Department of Computer Science
The University of Melbourne
Parkville, VIC 3052, Australia
madiraju, zliu @cs.mu.oz.au
Phone: 61-3-287-9124
Fax : 61-3-348-1184

Abstract In this paper, we propose a simple and powerful feature extractor using neural networks. This feature extractor is trained to detect features such as lines, corners, junctions in images. Different feature models are generated based on discontinuity in intensity values and the orientation of the boundary in the pixel neighborhood. Locating feature points in the image is carried out in two steps by considering an $n \times n$ window as a processing unit. At the first step, a covariance technique is used to calculate rotation-invariant descriptors, which represent discontinuities for edge types. At the second step, a multilayer feedforward neural network, trained with the invariant feature descriptors, is used to classify the centre pixel into one of the possible features. Experimental results using the proposed method are compared with Marr-Hildreth edge operator results to show the effectiveness of the proposed method.

INTRODUCTION

Feature extraction is a process to obtain relevant features from an image depending on a given task. Generally feature extraction [1, 5, 8, 9] can be divided into three categories in image processing applications namely, region based, where areas of images with homogeneous properties are found in terms of boundaries; edge based, where the local discontinuities are detected first and then connected to form longer lines; and pixel based, which classifies pixels based on gray levels.

0-7803-2026-3/94 $4.00 © 1994 IEEE

Recently, artificial neural networks (ANN), have found successful applications in such diverse areas as medicine, biology, control systems, manufacturing, etc. ANNs also have been applied to image segmentation. ANNs have the advantage over parametric statistical classification techniques, in that they do not require priori knowledge about the data distribution. Moreover, they are generally characterized by intrinsic parallelism and fault tolerant characteristics. In [3] a neural network system for edge detection is proposed. In this paper, simulated annealing, and mean field annealing are implemented and tested on synthetic images. Recently, a method for image segmentation has been developed using neural networks [10]. The image segmentation is cast as a constraint satisfaction problem.

We previously proposed that [7], instead of training a neural network with patterns in all possible orientations, it is more attractive and efficient to extract rotation-invariant features from a given set of feature models. This allows us to train a neural network with a small set of patterns.

In this paper, a method for extracting such features as edges, corners, lines, roofs, and ramps, using neural networks is developed. The pixel classification is performed in two stages: In the first stage, models are generated using the covariance techniques for different features. These feature models represent discontinuities in the neighborhood of the pixel. In the second stage, a feedforward neural network is trained by the set of feature models obtained from the previous stage. The neural network subsequently assigns a feature index to each pixel.

We have carried out extensive experiments on both synthetic and natural images and obtained dramatically better feature representations as compared to more traditional methods. The simulation results on natural scenes are also compared with the results of Marr-Hildreth edge operator. These show considerable visually correct and accurate edges as opposed to those obtained by the popular Marr-Hildreth edge detector. In the following we describe our method and present some experiment results.

PREPROCESSING

Prior to feature extraction, the image undergoes an initial nonlinear smoothing which preserves abrupt changes in the pixel neighborhood. This prevents erroneous and spurious features from being extracted by the neural network for the given model. We apply an averaging technique developed previously [8, 6] to the image, which, in effect, is a low pass filtering process. However, in this process, The center pixel p_o of an $n \times n$ window is replaced by the average of that window. If the difference between the values of p_o and its immediate

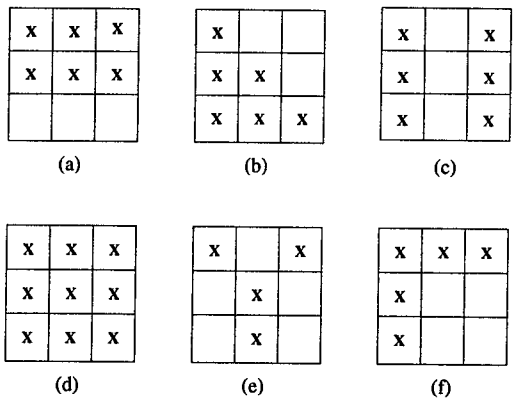


Figure 1: Feature models representing lines (a), diagonal lines (b), roofs (c), flat (d), junctions (e), and corners (f) in the windows. Each feature model represent four possible models in different orientations.

neighbor is greater than a certain value T, then that neighbor is not included in the computation of the average.

FEATURE MODELS

Feature extraction methods are typically based on local properties such as edges, lines, curves, etc. However, it is difficult to explicitly define what constitute different features in an image. The perception of features by the human visual system is an extremely complex process, that is strongly influenced by prior knowledge [4]. For our purposes, we will define feature models in a general sense to include a wide variety of edge types.

In our experiment each feature model represents a 3×3 window, i.e, $n = 3$ where the centre pixel is of interest. Different models are generated as shown in Figure 1. These models represent features, horizontal lines (a), vertical lines, diagonal lines (b), roofs (c), flat (d), junctions (e), and corners (f) in the windows.

FEATURE EXTRACTION

To extract invariant feature descriptors, we have used the eigenspace [7] of the covariance matrix.

Covariance Technique Let $W = [\vec{w}_1^T, \vec{w}_2^T, ..., \vec{w}_N^T]$ be a $3 \times N$ matrix, and $N = n^2$ where $n \times n$ is the size of pixel neighborhood. $\vec{w}_i = [x_i, y_i, z_i]$, whose x_i, y_i are the locations of the ith pixel in horizontal, vertical directions and z_i is the intensity value of ith pixel.

The covariance matrix is given by

$$C = \frac{1}{N} \sum_{i=1}^{N} (\vec{w}_i - \vec{w}_m)(\vec{w}_i - \vec{w}_m)^T$$

and

$$\vec{w}_m = \frac{1}{N} \sum_{i=1}^{N} \vec{w}_i$$

The Covariance matrix is symmetric, $C \in \Re^{3\times3}$, and has a set of three orthonormal vectors whose corresponding eigenvalues characterize the variances of the data set in the directions specified by their eigenvectors.

It is observed that the eigenvector corresponding to the dominant eigenvalue is set in the direction of maximum variance in the data: typically in the direction of the variance of intensity and the eigenvector corresponding to the smallest eigenvalue lies in the direction of the orientation of the window. It should also be noted that second-order covariances and associated eigenvalues can also be determined from these first-order eigenvalues or eigenvectors over neighborhoods of a given pixel (see Berkmann and Caelli [2]). However, in this application we have restricted our attention to only the first-order covariances. The third eigenvalue simply indexes the sampling density and we have excluded it from further analysis - though, in some cases, it would set as a silent feature. Figure 2 shows the discriminating ability of the eigenvalues for different features lines, diagonal lines, corners, roofs in the image in one of our experiments.

CLASSIFICATION

The previous stage finds an approximate solution to the classification problem by constraining the shape of the classification regions. From the nature of the eigenvalues found in the previous stage, the clustering properties are more clear between logarthmic values of dominant eigenvalue and absolute values of the remaining eigenvalues.

The feedforward neural network is one of the most popular ANN because of its structural simplicity and the ease in which it can be utilized in various applications. A multilayer feedforward neural network was trained by backpropagation algorithm to assign one of the possible feature classes (lines, ramps, corners, etc.) to each pixel.

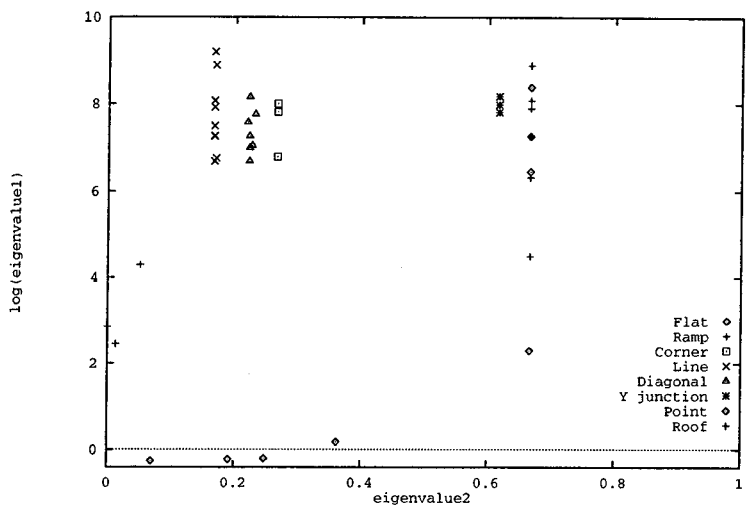


Figure 2: Features in eigenvalue space

EXPERIMENTAL RESULTS

We have conducted extensive experiments to verify the performance of the proposed method. All the images used in the experiments were 256×256 in size and the pixel intensity range was from 0 to 255.

In our experiments we used a 3×3 window as a model. We generated feature models representing lines, corners, points, roofs as defined in section 3. Ten models for each feature class were selected. A training set of 80 models was used.

To generate the training set for the neural network, rotation invariant eigenvalue descriptors were calculated using the covariance technique. The architecture of the neural network adopted is 3-3-3-1 net, with 3 neurons in the input layer, 3 neurons each in two hidden layers, and one neuron in the output layer. The neural network is trained using the delta rule. The training parameters α and η were 0.1 and 0.9 respectively. The convergence was achieved after 2000 iterations.

In Figures 3 and 4 we show some of our experimental results. Figure 3(a) shows an original synthetic image. Figure 3(b) is the image after feature extraction using the proposed algorithm. Figure 4(a) is a natural scene image. Figure 4(b) is the feature image, and show that the feature detector was able to pickup different types of edges, corners, junctions in the image while preserving the characteristics like thinness, and continuity.

We have implemented Marr-Hildreth edge operator based on the zero crossings of Laplacian. In effect, this is a non-directional second derivative zero crossing operator. The mask is generated using the formula [5]

$$\nabla^2 G(r) = \frac{-1}{\pi\sigma^4}(1 - \frac{r^2}{2\sigma^2})exp(\frac{-r^2}{2\sigma^2}),$$

where $r^2 = x^2 + y^2$ and x and y are the position of the row and column of the mask. We have selected $2\sigma = w = 5$ where w is the size of the mask. Figures 3(c) and 4(c) are the edge detected images by applying the above mentioned operator on the images shown in Figures 3(a) and 4(a) respectively.

CONCLUSION

In this paper, we proposed a method to perform feature extraction using a 3 layer neural network. The sensitivity of the feature extraction is adjustable and the training set of the neural network can be changed to increase the reliability of the results. By using the covariance technique, a small set of training patterns can cover a large number of models because of their rotation-invariant property. Furthermore, the neural network can be trained very fast. Finally, the covariance technique constrains the shape of the classification regions, which improves the classification accuracy. We have also compared the results using this method with the results obtained by zero-crossings of the Laplacian of Gaussian filter. The proposed approach is also being used in other pattern recognition problems.

References

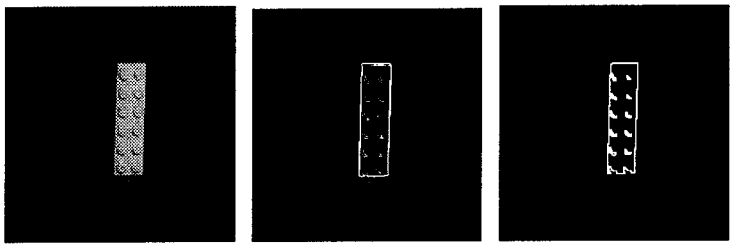
[1] A.Rosenfeld and A.C.Kak. *Digital Picture Processing.* Acadamic Press, New York, 1982.

[2] Jens Berkmann and Terry Caelli. Computation of surface geometry and segmentation using covariance techniques. Technical report, Collaborative Information Technology Research Institute, Melbourne, Australia, 1992.

[3] C.Cortes and J.A.Hertz. A network system for image segmentation. *Proc. Int.Joint.Conf. on Neural Networks*, 1:121 - 125, 1989.

[4] D.Marr. *Vision.* W H Freeman, New York, 1982.

[5] D.Marr and E.Hildreth. Theory of edge detection. *Proc. Royal Society of London*, 207:187 - 217, 1980.



(a) (b) (c)

Figure 3: (a) Synthetic image. (b) Feature detected image using neural network. (c) Edge detected image using Marr-Hildreth edge operator.

(a) (b) (c)

Figure 4: (a) Original image. (b) Feature detected image using neural network. (c) Edge detected image using Marr-Hildreth edge operator.

[6] J.M.Prager. Extracting and labelling boundary segments in natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(1):16–27, 1 1980.

[7] Sharma Madiraju, Terry Caelli, and Z.Q.Liu. On the covariance technique for robust and rotation invariant texture processing. *Asian Conference on Computer Vision -93*, pages 171 – 174, Nov 1993.

[8] R.C.Gonzalez and Paul Wintz. *Digital Image Processing.* Addison -Wesley, Massachusetts, 1987.

[9] V.Torre and T.A. Poggio. On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):147 – 163, 3 1986.

[10] Wei-Chung, Lin Eric Chen-Kuo Tso, and Chin-Tu Chen. Constraint satisfaction neural networks for image segmentation. *IEEE Transactions on Systems, Man, and Cybernetics*, 12(7):679–693, 1992.

MEDICAL IMAGING WITH NEURAL NETWORKS

C.S. Pattichis[1] and A.G. Constantinides[2]

[1]Department of Computer Science, University of Cyprus,
Kallipoleos 75, P.O. Box 537, Nicosia, Cyprus
Tel:357 2 360589, Fax: 357 2 360881, email: pattichi@jupiter.cca.ucy.cy

[2]Department of Electrical Engineering, Imperial College of Science,
Technology and Medicine, London SW7 2BT, England
Tel:071 225 8506, Fax:071 581 3441, email: tony@sig.ee.ic.ac.uk

Abstract - The objective of this paper is to provide an overview of the recent developments in the use of artificial neural networks in medical imaging. The areas of medical imaging that are covered include: ultrasound, magnetic resonance, nuclear medicine and radiological (including computerized tomography).

INTRODUCTION

In everyday medical practice, the physician has to evaluate a number of complementary diagnostic imaging modalities such as ultrasound, magnetic resonance, nuclear medicine, and radiological. These images are usually evaluated qualitatively by visual examination. However, the need for quantitative analysis is becoming of increasing importance in the clinical environment. This allows measurements to be standardised, to be more accurate and save diagnostic time. The advantages of quantitative analysis in medical imaging can be summarised as follows: i) *Standardisation.* Diagnoses obtained from different laboratories using similar criteria can be verified. ii) *Sensitivity.* Findings on a particular subject may be compared with a database of normal values and/or a decision can be made by an automated imaging diagnostic system deciding whether or not an abnormality exists. iii) *Specificity.* Findings may be compared with databases for various diseases and/or a decision can be made by an automated imaging diagnostic system with respect to the type of abnormality. iv) *Equivalence.* Results from a series of examinations of the same patient may be compared in order to decide whether there is evidence of disease progression or of response to treatment. In addition, the findings of different automated imaging diagnostic methods can be compared to determine which are more sensitive and specific. v) *Efficacy.* The results of different treatments can be more properly evaluated.

Different approaches have been used to address the problem of quantitative analysis in medical imaging. Classical methods range from simple thresholding to more advanced multidimensional data classification techniques. The use of artificial neural networks (ANN) in image analysis has recently been proposed. The advantages of artificial neural networks that make them so attractive to investigate as an alternative are the following: i) exhibit adaptation or learning, ii) pursue multiple hypothesis in parallel, iii) may be fault tolerant, iv) may process degraded or incomplete data, v) make no assumptions about underlying data probability density functions and iv) seek answers by carrying out transformations.

This work was carried out through a European Community International Scientific Cooperation Initiative, Marie Curie Fellowship No. 930180, awarded to Dr. C.S. Pattichis.

0-7803-2026-3/94 $4.00 © 1994 IEEE

This paper provides a survey of the different applications of neural network technology in medical imaging and in particular in the fields of ultrasound, magnetic resonance, nuclear medicine, and radiology. The scope of the paper is quite wide and although our literature review is thorough it is by no means totally complete. Publications reviewing the use of ANN in medical diagnostic systems [ANM1]-[ANM4], as well as in medical signal and image processing [ANM5], [ANM6] may provide additional information.

For each imaging field discussed, a table summarizes the profile of the different ANN studies. The following table entries are given: name of first investigator, [reference], problem under consideration, organ or part of the body investigated, imaging modality, imaging operation, and neural network learning method. For each imaging modality selected papers are discussed more extensively.

LIST OF ACRONYMS

ALOPEX	optimization procedure to train ANN	HOP	Hopfield network
ANN	artificial neural network	LOGN	logical neurons
ART	adaptive resonance theory	MLC	maximum likelihood classsifier
AUASS	autoassociative learning paradigm	MRI	magnetic resonance imaging
BP	back propagation	PET	positron emission tomography
CLASS	classification	PCA	principal component analysis
COMP	compression	PNN	probabilistic neural network
CT	computerized tomography	RECO	reconstraction
ENHA	enhancement	ROI	region of interest
FEAE	feature extraction	SEGM	segmentation
FFCC	fast forward cascade correlation algorithm	SPECT	single photon emission computer tomography
FMMANN	fuzzy Min-Max ANN	TEXCL	texture classification

ULTRASOUND

Neural network models for ultrasound imaging have been developed for cardiology [ULT1]-[ULT5], liver tissue identification [ULT6]-[ULT8], and ophthalmology [ULT9]. Table 1 lists several representative examples of ultrasound imaging with NN, and we briefly present one of these below.

Detection of Myocardial Infarction [ULT1]

Echocardiographic images from 11 normal, 7 hypertrophic cardiomyopathy, and 11 myocardial infarction subjects were digitised into a 256x256 pixel matrix with 256 gray levels [ULT1]. The regions of interest (ROI) were predetermined by a cardiologist, avoiding the endocardium echo, epicardium, and valves and consisted of a 10x10 pixel matrix. These gray

levels were subsequently normalised between 0 and 1. The back propagation (BP) NN algorithm [ANN9] was used with two types of images at the input i) 10x10 pixel matrix and ii) 5x5 pixel matrix with an overlap factor of 4x4. Results of this study demonstrated that the former network was more sensitive in classifying the data than the latter one. The paper concludes that the BP ANN is capable of recognizing the slight differences between normal and abnormal diseases of myocardial tissue. However, no quantitative measure was given to support this.

TABLE 1 SUMMARY OF ULTRASOUND IMAGING ANN STUDIES

Investigator	[Ref.]	Problem	Organ	Operation	ANN Method
Cios	[ULT1]	Myocardial infarction	heart	TEXCL	BP
Tzanakou	[ULT2]	Myocardial infarction	heart	TEXCL	BP-ALOPEX
Yi	[ULT3]	Myocardial infarction	heart	TEXCL	BP-ALOPEX
Brotherton	[ULT4]	Structure and tissue	heart	TEXCL	FMMANN
Karkhanis	[ULT5]	Ejection fraction	heart	FEAE	BP
Kim	[ULT6]	Liver diagnosis	liver	TEXCL	BP
Daponte	[ULT7]	Liver diagnosis	liver	TEXCL	BP
Botros	[ULT8]	Liver diagnosis	liver phantom	CLASS	BP
Silverman	[ULT9]	Tumour detection	eye	CLASS	BP
Nikoonahad	[ULT10]	Wave velocity correction		ENHA	BP

MAGNETIC RESONANCE

Examples of the application of neural network technology in magnetic resonance imaging (MRI) are given in Table 2. Most of these applications have been developed for segmentation of MRI images [MRI1]-[MRI9]. Some of these studies were demonstrated to perform as well as or better than classical statistical analysis using for example the maximum likelihood classifier (MLC). The usefulness of ANN models in blood vessel identification, and invariant aorta segmentation was also investigated by [MRI15] and [MRI16] that gave promising results.

Segmentation of brain images [MRI1]

Segmentation of medical images obtained from magnetic resonance imaging is a very important operation in the visualization of soft tissues in the human body. MRI is inherently multidimensional as it provides information about three tissue dependent parameters: spin-lattice relaxation time, T1, the spin-spin relaxation time, T2, and the proton density, PD. Neural network models supplied with T1, T2, and PD, weighted intensity values, and in addition the X-ray CT intensity value of images of the human brain were trained with the

back propagation algorithm to classify the following six tissue types: background, cerebrospinal fluid, white matter, gray matter, skull and fat, and bone and bone marrow [MRI1]. Results reported in this work support the use of neural networks as a promising method for the classification of multi-modality medical images. One of the major advantage of ANN over classical statistical pattern recognition techniques, like the MLC is their relative insensitivity to the selection of the training sets. In the case of single slice MRI classification, it has been demonstrated that neural networks are able to segment the images, although training points did not appropriately sample the image spatially, a task the maximum likelihood classifier was not able to perform well. Also, in the case of multiple slice classification, the characteristics of the class boundaries obtained by the neural network models have permitted the successful development of an adaptive 3-D classification scheme [MRI1].

TABLE 2 SUMMARY OF MAGNETIC RESONANCE IMAGING ANN STUDIES

Investigator	[Ref.]	Problem	Organ	Operation	ANN Method
Ozkan	[MRI1]		brain	SEGM	BP
Amartur	[MRI2]		brain	SEGM	HOPF
Hall	[MRI3]		brain	SEGM	FFCC
Cagnoni	[MRI4]		brain	SEGM	BP
Piraino	[MRI5]		brain	SEGM	BP
Dawant	[MRI6]		brain	SEGM	BP
Schellenberg	[MRI7]		brain	SEGM	BP
Toulson	[MRI8]		brain	SEGM	BP
Morrison	[MRI9]		brain	SEGM	PNN
Raff	[MRI10]	Lesion detection in Multiple Sclerosis	brain	FEAE	AUASS
Lehar	[MRI11]	Boundary contour identification	brain	FEAE/ ENHA	ART
Manduca	[MRI12]	Diagnosis of Avascular Necrosis	brain	CLASS	BP
Yan	[MRI13]	Artifact rejection	brain	RECO	BP
Ohhashi	[MRI14]	Gray level adjustment	brain	ENHA	BP
Gronovist	[MRI15]	Vessel identification	brood vessel	FEAE	BP
Katz	[MRI16]	Translation invariant segmentation	aorta	SEGM	BP

NUCLEAR MEDICINE

Nuclear medicine imaging analysis using neural networks includes positron emission

tomography (PET), and single photon emission computer tomography (SPECT). These studies are summarized in Table 3.

Diagnosis of Alzheimer's Disease through ANN analysis of PET images [NM1] [NM2]
The back propagation neural network algorithm was applied for the analysis of cerebral function as demonstrated in positron emission tomography (PET). Data was obtained from PET scans of 22 patients with Alzheimer's Disease (AD), and 30 aged-matched normal subjects. Data describing each subject consisted of eight values, representing cerebral glucose metabolism in the eight lobes of the brain (left and right): frontal, parietal, temporal, and occipital. The network was trained with data from 26 subjects (15 normal, 11 AD), representing one half of the above subject group. Subsequently, the network's performance was tested on the remaining half. The trained network's classification agreed with the clinical diagnosis in 24 of the 26 cases, giving a 92% correct classifications score. Neural networks performed better than standard statistical methods like discriminant analysis.

TABLE 3 SUMMARY OF NUCLEAR MEDICINE IMAGING ANN STUDIES

Investi-gator	[Ref.]	Problem	Organ	Moda-lity	Opera-tion	ANN Method
Kippenhan	[NM1]	Alzheimer's disease	brain	PET	CLASS	BP
Kippenhan	[NM2]	Alzheimer's disease	brain	PET	CLASS	BP
Miller	[NM3]	Parameter identification	brain	PET	FEAE	BP
Tourassi	[NM4]	Lesion detection	brain	SPECT	CLASS	BP
Mason	[NM5]		brain	SPECT	CLASS	BP/LOGN
Floyd	[NM6]			SPECT	RECO	BP
Anthony	[NM7]	Thallium 201 scintigrams	lung	SPECT	CLASS	BP
Anthony	[NM8]	Thallium 201 scintigrams	lung	SPECT	CLASS	BP

RADIOLOGY

Neural networks in radiology have been applied in cineangiography, digital subtraction angiography, mammography and X-ray CT as shown in Table 4.

Coronary artery angiography [XR1]
The classification of digital angiograms using NN was investigated by NeKorei and Sun [XR1]. The network consists of an 11x11 pixel input mask, 17 hidden nodes, and two output nodes. The mask is applied to the whole 256x256x8 bit angiograms, with the network output classifying the center pixel of the input mask as either vessel or background. Results of this study suggested that a suitable network can achieve an acceptable vessel detection rate. Two types of coronary angiograms were investigated i) cineangiogram, and ii) digital subtraction angiogram. For the cineangiography vessel detection rate was 96, and 93% for the training and test sets respectively, and for the digital subtraction angiography vessel detection rate was 96, and 82% for the training and test sets respectively. The performance of the NN

approach was compared with traditional pattern recognition techniques, the maximum likelihood classifier. MLC vessel detection rate for cineanography was 38% and for the digital subtraction angiography 78%.

TABLE 4 SUMMARY OF X-RAY IMAGING ANN STUDIES

Investi-gator	[Ref.]	Problem	Organ	Modality	Opera-tion	ANN Method
Nekevei	[XR1]	Identification	coronary artery	Angio-graphy	SEGM	BP
Nekevei	[XR2]	Identification	coronary artery	Cineangio-graphy	SEGM	BP
Dhawan	[XR3]	Microcalsi-fication classification	breast	Mammo-graphy	CLASS	BP
Chitre	[XR4]	Microcalsi-fication classification	breast	Mammo-graphy	CLASS	BP
Stathaki	[XR5]	Microcalsi-fication classification	breast	Mammo-graphy	SEGM	BP
Pinho	[XR6]	Edge detection	brain	X-ray CT	FEAE	BP
Gan	[XR7]			X-ray CT	ENHA	HOPF

CONCLUDING REMARKS

A review of the various applications of neural network's technology in medical imaging was given. The concluding remarks that were drawn based on these studies are summarised as follows:

- Almost all of the studies used the supervised learning training BP algorithm to train multi-layer perception feed-forward nets. In a very few studies the supervised learning Hopfield net was also used.

- Studies that compared neural network results with classical statistical analysis like the maximum likelihood classifier, and discriminant analysis reported similar or better performance [ULT7] [ULT9] [MRI1] [NM1] [NM2] [XR1] [XR4].

- In some of the papers reviewed, the amount of data available for training and testing the neural network models were limited, thus affecting the results obtained. It should be emphasized that training data must form a representative sample set of all possible inputs if the network is to perform correctly.

- Data preprocessing significantly affects the NN performance not only regarding classification score, but also the size of architecture, and training time (coupled with the number of epochs to achieve learning).

- The BP neural network learning algorithm has a number of limitations; heavy computational and memory requirements, as well as the non existence of design

methodologies for determining the values of the learning coefficient, λ, and the momentum coefficient, μ, number of hidden layers, and architecture size.

- New learning algorithms are currently investigated to address the above problem. It has recently been demonstrated by Charalambous [ANN3] that the conjugate gradient back propagation algorithm (CGBP) eliminates the selection of λ and μ. This algorithm is the same as the BP algorithm, but with adjustable values of λ_k and μ_k at each iteration. In addition the CGBP algorithm does not exhibit any oscillatory behaviour during learning, like the BP algorithm.

- Further to the search of new learning algorithms, it is anticipated that the development of neural network hardware will allow a cost-performance-effective implementation of neural networks in image processing.

We hope that neural network technology will help the physician in reaching a more accurate diagnosis.

REFERENCES

Artificial Neural Networks

[ANN1] J.A. Anderson and E. Rosenfeld (Eds), "Neurocomputing Foundation of Research," The MIT Press, 1988.

[ANN2] G.A. Carpenter and S. Grossberg, "Neural Dynamics of Category Learning and Recognition: Attention, Memory Consolidation, and Amnesia," in J. Davis, R. Newburgh and E. Wegman (Eds.) Brain Structure, Learning, and Memory, AAAS Symposium Series, 1986.

[ANN3] C. Charalambous, "Conjugate Gradient Algorithm for Efficient Training of Artificial Neural Networks," IEE Proc., Vol. 139, No. 3, June 1992.

[ANN4] R. Hecht-Nielsen, "Counter-Propagation Networks," Proc. 1st Int. Conf. on Neural Networks, June, San Diego, California, USA, Part 2, pp. 19-32, 1987.

[ANN5] G.E. Hinton, "Connectionist Learning Procedures," Artif. Intell., Vol. 40, pp. 185-234, 1989.

[ANN6] J.J. Hopfield and D.W. Tank, " Computing with neural circuits: a model," Science, Vol. 223, pp. 625-633, 1986.

[ANN7] B. Kosko, "Neural Networks for Signal Processing," Prentice-Hall Inc. 1992.

[ANN8] R.P. Lippman, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp. 4-22, April 1987.

[ANN9] D.E. Rumelhart, G.E. Hinton, and J.L. McClelland, "A General Framework for Parallel Distributed Processing," In Parallel distributed processing: explorations in the microstructure of cognition Volume 1 Foundations. D.E. Rumelhart, J.L. McClelland, and The PDP Research Group (Eds.), MIT Press, Cambridge, USA, pp. 45-76, 1986.

[ANN10] P.K. Simpson, "Fuzzy Min-Max Neural Networks - Part 1: Classification," IEEE Trans. on Neural Networks, Vol. 3, No. 5, Sept. 1992.

[ANN11] P.K. Simpson, "Fuzzy Min-Max Neural Networks - Part 2: Clustering," IEEE Trans. on Fuzzy Systems, Vol. 1 No. 1, Feb. 1993.

[ANN12] E. Tzanakou, R. Michalak, E. Harth, "The ALOPEX Process: Visual Receptive Fields with Response Feedback," Biol. Cybernetics, Vol. 35, pp. 161-174, 1979.

Artificial Neural Networks in Medicine: Review Studies

[ANM1] J.M. Zurada, "Introduction to Artificial Neural Systems," West publishing company, St. Paul, M.N., 1992.

[ANM2] D. Jones, "Neural Networks for Medical Diagnosis," in: Handbook of Neural Computing Applications, Academic Press, New York, pp. 309-317, 1990.
[ANM3] J.A. Reggia, "Neural Computation in Medicine," Artificial Intelligence in Medicine, 5, pp. 143-157, 1993.
[ANM4] Artificial Intelligence in Medicine, Special issue on Neural computing in medicine, Vol. 6, 1994, to be published.
[ANM5] A.S. Miller, B.H. Blott and T.K. Hames, "Review of Neural Network Applications in Medical Imaging and Signal Processing," Med. Biol. Eng. Comp., Vol. 30, No. 5, pp. 449-464, 1992.
[ANM6] IEEE Engineering in Medicine and Biology, Special issue on the applications of neural networks, Vol. 9, No. 3, September 1990.

Ultrasound Imaging

[ULT1] K.J. Cios, K. Chen and R.A. Langenderfer, "Use of Neural Networks in Detecting Cardiac Diseases From Echocardiographic Images," IEEE Eng. in Med. and Biol., Vol. 9, No. 3, pp. 58-60, 1990.
[ULT2] E.M. Tzanakou, "Biomedical Applications of Parallel Processing and Artificial Neural Networks," Proc. of the 1992 International Biomedical Engineering Days, Istanbul, pp. 10-17, Aug. 18-20 1992.
[ULT3] C. Yi, E. Micheli-Tzanakou, D.M. Shindler and J.B. Kostis, "Study of Echocardiogram for Myogardial Infarction Using Neural Networks," Proc. of the 15th Annual Int. Conf. of the IEEE Eng. in Medicine and Biology Scociety, San Diego, California, pp. 255-256, Oct. 1993.
[ULT4] T. Brotherton, T. Polland, K. Haines and A. DeMaria, "Echocardiogram Structure and Tissue Classification Using Hierarchical Neural Networks," Proc. of the 15th Annual Int. Conf. of the IEEE Eng. in Medicine and Biology Scociety, San Diego, California, pp. 290-291, Oct. 1993.
[ULT5] P.A. Karkhanis, J.Y. Cheung and S.M. Teague, "Using a PC Based Neural Network to Estimate the Ejection Fracion of a Human Heart," Microcomputer Appl. 9 (3), pp. 99-107, 1990.
[ULT6] S.I. Kim, K.C. Choi and D.S. Lee, "Texture Classification Using Run Difference Matrix," IEEE 1991 Ultrasonics Symposium Proceedings, Orlando, FL, Vol. 2, pp. 1097-1000, Dec. 1991.
[ULT7] J.S. DaPonte and P. Sherman, "Classification of Ultrasonic Image Texture by Statistical Discrimination Analysis of Neural Networks," Computer Med. Imaging Graph., 15, (1), pp. 3-9, 1991.
[ULT8] N.M. Botros, "A PC-Based Tissue Classification System Using Artificial Neural Networks," IEEE Transactions on Instrumentation and Measurement, Vol. 41, No. 5, pp. 633-638, 1992.
[ULT9] R.H. Silverman and A.S. Noetzel, "Image Processing and Pattern Recognition in Ultrasonograms by Back-Probagation," Neural Networks, 3, pp. 593-603, 1990.
[ULT10] M. Nikoonahad and D.C. Liu, "Medical Ultrasound Imaging Using Neural Networks," Electron. Lett., 26, pp. 545-546, 1990.

Magnetic Resonance Imaging

[MRI1] M. Ozkan, B.M. Dawant and R.J. Maciumas, "Neural-Network-Based Segmentation of Multi-Modal Medical Images: A Comparative and Prospective Study," IEEE Transactions on Medical Imaging, Vol. 12, No. 3, pp. 534-544, 1993.
[MRI2] S.C. Amartur, D. Piraino and Y. Takefuji, "Optimization Neural Networks for the Segmentation of Magnetic Resonance Images," IEEE Trans. on Medical Imaging, Vol. 11, No. 2, pp. 215-220, 1992.

[MRI3] L.O. Hall, A.M. Bensaid, L.P. Clarke, R.P. Velthuizen, M.S. Silbiger and J.C. Bezdek, "A Comparison of Neural Network and Fuzzy Clustering Techniques in Segmenting Magnetic Resonance Images of the Brain," IEEE Trans. on Neural Networks, Vol. 3, No. 5. pp. 672-682, 1992.

[MRI4] S. Cagnoni, G. Coppini, M. Rucci, D. Caramella and G. Valli, "Neural Network Segmentation of Magnetic Resonance Spin Echo Images of the Brain," J. Biomed. Eng., Vol. 15, pp. 355-362, 1993.

[MRI5] D. Piraino, S. Sundar, B. Richmond, J. Schils and J. Thome, "Segmentation of Magnetic Resonance Images Using a Back Propagation Artificial Neural Network," Proc. of the Ann. Int. Conf. of the IEEE Eng. in Med. and Biol. Soc., Vol. 13, No. 3, pp. 1466-1467, 1991.

[MRI6] B.M. Dawant, M. Ozkan, H.G. Sprenkels, H. Aramata, K. Kawamura and R.A. Margolin, "A Neural Network Approach to Magnetic Sesonance Imaging Tissue Characterisation," Proc. Bilkent Int. Conf. on New Trends in Comm. Control & Sig. Proc., Arikan E. (Ed.), Elsevier, Amsterdam, part 2, pp. 1803-1809, July 1990.

[MRI7] J.D. Schellenberg, W.C. Naylor and L.P. Clarke, "Application of Artificial Neural Networks for Tissue Classification from Multispectral Magnetic Resonance Images of the Head," Proc. 3rd Ann. IEEE Symp. on Computer-Based Medical Systems, Chapel Hill, North Carolina, USA, pp. 350-357, June 1990.

[MRI8] D.L. Toulson and J.F. Boyce, "Segmentation of MR Images Using Neural Nets," IEE Colloq. on Image Proc. in Med., IEE Coll. Dig. 1991/84, London, UK, Paper 5, April 1991.

[MRI9] M. Morrison and Y. Attikiouzel, "A Probabilistic Neural Network Based Image Segmentation Network for Magnetic Resonance Images," IJCNN Int. Joint Conf. on Neural Networks, Baltimore, MD, USA, Vol. 3, pp. 60-65, June 1992.

[MRI10] U. Raff, and F.D. Newman, "Lesion Detection in Radiologic Images Using an Autoassociative Paradigm: Peliminary Results," Med. Phys., (US), 17, 926-928, 1990.

[MRI11] S.M. Lehar, A.J. Worth and D.M. Kennedy, "Application of the Boundary Contour/Feature Contour System to Magnetic Resonance Brain Scan Imagery," Proc. 4th Int. Joint Conf. on Neural Networks, pp. 435-440, 1990.

[MRI12] A. Manduca, P. Christy and R. Ehman, "Neural Network Diagnosis of Avascular Necrosis from Magnetic Resonance Images," Proc. of the Ann. Inter. Conf. of the IEEE Eng. in Med. and Biol. Soc., Vol. 13, No. 3, pp. 1429-1431, 1991.

[MRI13] H. Yan and J. Mao, "Data Truncation Artifact Reduction in MR Imaging Using a Multilayer Neural Network," IEEE Trans. on Med. Imaging, Vol. 12, No. 1, pp. 73-77, 1993.

[MRI14] A. Ohhashi, S. Yamada, K. Haruki, H. Hatano, K. Nishimura, Y. Fujii, K. Yamaguchi and H. Ogata, "Application of a Neural Network to Automatic Gray-level Adjustment for Medical Images," Int. Conf. on Neural Networks, Westin Stamford and Westin Plaza, Singapore, Vol. 2, pp. 974-980, Nov. 1991.

[MRI15] A. Gronqvist and R. Lenz, "Detection of Blood Vessels in 3-D MR-Images," Proc. Int. Joint Conf. on Neural Networks, Washington DC, USA, Part 1, pp. 145-149, June 1989.

[MRI16] W.T. Katz and M.B. Merickel, "Translation Invariant Aorta Segmentation from Magnetic Resonance Images," Proc. Int. Joint Conf. on Neural Networks, Washington DC, USA, Part 1, pp. 327-333, June 1989.

Nuclear Medicine Imaging

[NM1] J.S. Kippenhan and J.H. Nagel, "Diagnosis and Modelling of Alzheimer's Disease Through Neural Network Analysis of PET Studies," Proc. 12th Ann. Conf. IEEE Eng. in Med. & Biol. Soc., 1st-4th Nov., Philadelphia, Pennsylvania, USA, Vol. 12, pp. 1449-1450, 1990.

[NM2] J.S. Kippenhan, W.W. Barker, S. Pascal, R. Duare and J. Nagel, "Optimization and Evaluation of Neural-Network Classifier for PET Scans of Memory-Disorder Subjects," Proc. of the 13th Ann. Inter. Conf. of the IEEE Eng. in Med. and Biol. Soc., Vol. 13, No. 3, pp. 1472-1473, 1991.

[NM3] L.F. Miller, G.T. Smith, Y. Wu and R.E. Uhrig, "Evaluation of Neural Networks for Parameter Identification from Positron Emission Tomography Scans," Trans. Am. Nuclear Soc., 62, pp. 5-6, 1990.

[NM4] G.D. Tourassi, C.E. Floyd, M.T. Munley, J.E. Bowsher and R.E. Coleman, "Application of Neural Networks to Lesion Detection in SPECT," IEEE Nuclear Science Symposium and Medical Imaging Conference, Santa Fe, NM, USA, Vol. 3, pp. 2179-2183, Nov. 1991.

[NM5] J. Mason, S. Sheppard, E. Hines, D. Taylor and J. Barham, "Application of Logical Neural Networks to the Analysis of Single Photon Emission Tomography Images," IEE Colloquium on Neural Networks for Image processing Applications, London UK, Vol. 9, pp. 1-5, Oct. 1992.

[NM6] C.E. Jr Floyd, J.E. Bowsher, M.T. Munley, G.D. Tourassi, S. Garg, A.H. Baydush, J.Y. Lo and R.E. Coleman, "Artificial Neural Networks for SPECT Image Reconstruction with Optimized Weighted Backprojection," IEEE Nuclear Science Symposium and Medical Imaging Conf. Santa Fe, NM, USA, Vol. 3, pp. 2184-2188, Nov. 1991.

[NM7] D.M. Anthony, E.L. Hines, J. Barham and D. Taylor, "The Use of Neural Networks in Classifying Lung Scintigrams," INNC 90 PARIS Proc of Int. Neural Nerwork Conf., Paris, France, Vol., 1, pp. 71-74, July 1990.

[NM8] D.M. Anthony, "The Use of Artificial Neural Networks in Classifying Lung Scientigrams," PhD thesis, University of Warwick, 1991.

X-Ray Imaging

[XR1] R. Nekovei and Y. Sun, "Classification of Digital Angiograms Using Artificial Neural Networks," Proc. of the Ann. Intern. Conf. of the IEEE Engin. in Med. and Biol. Soc., Vol. 13, pp. 1440-1441, 1991.

[XR2] R. Nekovei and Y. Sun, "An Adaptive Algorithm for Coronary Artery Identification in Cineangiograms," Proc. of the Twelfth Ann. Intern. Conf. of the IEEE Engin. in Med. and Biol. Soc., Philadelphia, Pensylvania, USA, pp. 1459-1460, November 1-4 1990.

[XR3] A.P. Dhawan, Y.S. Chitre, M. Moskowitz, and E. Gruenstein, "Classification of Mammographic Microcalcification and Structural Features Using An Artificial Neural Network," Proc. of the Ann. Intern. Conf. of the IEEE Eng. in Med. and Biol. Soc., Vol. 13, No. 3, pp. 1105-1106, 1991.

[XR4] Y. Chitre, A.P. Dhawan, and M. Moskowitz M., "Artificial Neural Network Based Classification of Mammographic Microclacifications Using Image Structure Features," Proc. of the 15th Ann. Int. Conf. of the IEEE Eng. in Med. and Biol. Soc., Vol. 15, pp. 50-51, 1993.

[XR5] P.T. Stathaki, and A.G. Constantinides, "Higher Order Spectral Estimation Techniques in Mammography", Proc. of the Int. Conf. on Digital Signal Processing and II Int. Conf. on Computer Applications to Engineering Systems, Nicosia, Cyprus, pp. 276-280, July 14-16, 1993.

[XR6] A.J. Pinho, "Modeling Non-Linear Edge Detectors Using Artificial Neural Networks," Proc. of the 15th Ann. Int. Conf. of the IEEE Eng. in Med. and Biol. Soc., Vol. 15, pp. 306-307, 1993.

[XR7] W.S. Gan, "Application of Neural Networks to the Processing of Medical Images," Int. Joint Conference on Neural Networks, Westin Stamford and Westin Plaza, Singapore, pp. 300-306, November 18-21 1991.

HIGH RESOLUTION IMAGE RECONSTRUCTION USING MEAN FIELD ANNEALING

Thanachart Numnonda and Mark Andrews
Department of Electrical & Electronic Engineering
School of Engineering, University of Auckland
Ph: +64 9 373 7599 ext.8104, Fax: +64 9 373 7461
e-mail: thanon@ccu1.auckland.ac.nz

Abstract: A high resolution image can be reconstructed from a sequence of lower resolution frames of the same scene where each frame taken by the camera is offset by a subpixel displacement. In this paper, it is shown that such a reconstruction task can be cast as an optimisation problem, and that a reconstruction can be found using the mean field annealing algorithm. The proposed technique has the added advantage over existing techniques of not requiring the registration of the displacement of each low resolution frame. In addition, the proposed technique greatly reduces the required computation as compared to a simulated annealing approach.

INTRODUCTION

Many image processing applications, such as satellite remote sensing, industrial quality control and scientific or medical imaging, require a high resolution image in which the use of commercial video camera seems rather limiting. Increasing the resolution requires an increase in the sampling rate, and thus its implementation by sensor modification is usually undesirable. Therefore, attention has turned to obtaining higher resolution images using signal processing techniques instead. One promising approach is to reconstruct a high resolution still-frame image from a sequence of lower resolution frames of the same scene where each frame taken by the camera is offset by a subpixel displacement. This reconstruction problem has been addressed by several researchers, and various reconstruction techniques have also been proposed [1–7].

In spite of their apparent variety, the existing techniques have a common structure and contemporary reconstruction procedures usually consist of two main parts; the registration phase and the reconstruction phase. In practice, the best possible reconstruction quality is, however, unlikely to be obtained due to the limitation of currently available registration and reconstruction methods. Indeed, undersampled images include aliased frequency components

0-7803-2026-3/94 $4.00 © 1994 IEEE

which cause errors in the registration phase. On the other hand, the accuracy of this estimation influences the reconstruction quality, since most existing reconstruction methods are based on the assumption that the displacements are correctly estimated [2,5,6].

In an effort to resolve these difficulties, the authors demonstrated [8] that the high resolution image reconstruction task can be recast as an optimisation problem in which the registration and reconstruction phases can be performed simultaneously, and that a solution can be found using the simulated annealing algorithm. Nevertheless, this algorithm is still not suitable for real-world images due to its large computational effort.

In this paper, the authors have applied the mean field annealing algorithm [9,10] to the high resolution image reconstruction problem. By using this new approach, the reconstruction can still be achieved without requiring a separate registration phase, but with much less computational effort compared with the simulated annealing algorithm.

AN OPTIMISATION APPROACH FOR HIGH RESOLUTION IMAGE RECONSTRUCTION

The concepts involved in reconstructing a high resolution image from multiple low resolution images may be elucidated by considering the process of obtaining a low resolution image, $g(m,n)$, from a higher resolution image, $f(k,l)$, as illustrated in Fig. 1, in which the relationship between the image pixels $g(m,n)$ and $f(k,l)$ can be expressed as [4]:

$$g(m,n) = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} f(k,l) h(m,n;k,l), \tag{1}$$

where

$$h(m,n;k,l) = \frac{\mathcal{A}\{S_h(k,l) \cap S_l(m,n)\}}{\mathcal{A}\{S_l(m,n)\}}, \tag{2}$$

denotes the point spread function (PSF), $\mathcal{A}$ denotes the area of its argument, and $S_h(.,.)$ and $S_l(.,.)$ denote the support of the high resolution and low resolution sensors centred around the pixel $(.,.)$, respectively.

Supposing that the PSF of the imaging system is known, one can obtain a number of low resolution images, $\hat{g}_i(m,n)$, from an estimated high resolution image, $\hat{f}(k,l)$, using such an imaging process. If $\hat{f}(k,l)$ is identical to the correct high resolution image, then the estimated image $\hat{g}(m,n)$ should be identical to the given image $g(m,n)$. This hypothesis can also be applied to the case of multiple low resolution images of the same scene where each frame taken by the camera is shifted by a subpixel displacement. In the latter case, each low resolution frame has different PSF, $h_i(m,n;k,l)$, which can be determined from its corresponding displacement. In addition, if these displacements are unknown, then the estimated image $\hat{g}(m,n)$ will be identical

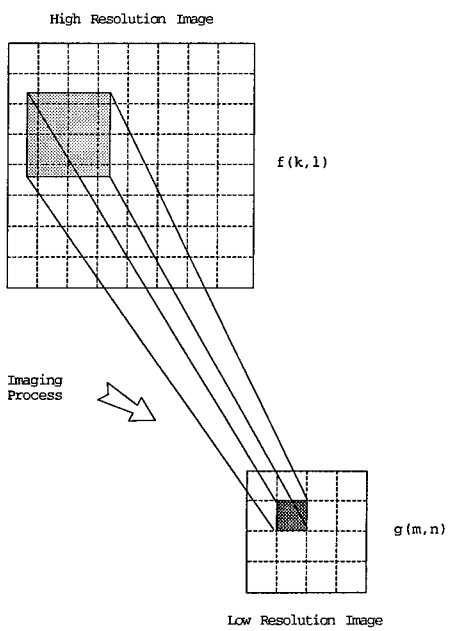


Figure 1: A schematic diagram simulating a process of obtaining a low resolution image.

to the given image $g(m,n)$ only if both the high resolution image and the displacements are correctly estimated.

Therefore, the high resolution image reconstruction task can be cast so as to find the *maximum a posterior* (MAP) estimate of the high resolution image and the displacements when a sequence of low resolution images is given. That is

Estimate $\boldsymbol{f}$ and $\boldsymbol{d}$ such that $P(\boldsymbol{f}, \boldsymbol{d} \mid \boldsymbol{G})$ is maximized

where $\boldsymbol{f}$ denotes a high resolution image, $\boldsymbol{G}$ denotes a sequence of low resolution images, and $\boldsymbol{d}$ denotes a sequence of displacements corresponding to each low resolution image, that is

$$\boldsymbol{d} = \{\delta_{xi}, \delta_{yi};\ i = 1, \ldots, P\},$$

where P denotes the number of available images, and δ_{xi} and δ_{yi} are the displacements of the ith low resolution image along the x and y direction respectively.

In general, an image can be modeled as a Markov random field (MRF), and the posterior probability can be described by the Gibbs distribution as follows [11]:

$$P(\boldsymbol{f}) = \frac{1}{Z} \exp\left[-\frac{U(\boldsymbol{f})}{T}\right], \tag{3}$$

where Z is the normalization constant (also called partition function), and $U(\boldsymbol{f})$ is the energy function of the form

$$U(\boldsymbol{f}) = \sum_{c} V_c(\boldsymbol{f}), \tag{4}$$

c being the set of cliques associated with the neighbourhood. Thus it can be shown that [12] the MAP can be given as:

$$P(\boldsymbol{f}, \boldsymbol{d} \mid \boldsymbol{G}) = \frac{1}{Z} \exp\left[-\frac{U(\boldsymbol{f}, \boldsymbol{d} \mid \boldsymbol{G})}{T}\right]. \tag{5}$$

Moreover, the energy function is given as:

$$U(\boldsymbol{f}, \boldsymbol{d} \mid \boldsymbol{G}) = \begin{aligned} & c \cdot \left[\sum_{i=1}^{P} \sum_{m} \sum_{n} \left| g_i(m,n) - \sum_{k} \sum_{l} \hat{f}(k,l) \hat{h}_i(m,n;k,l) \right|^2 + \right. \\ & \left. \lambda \sum_{k} \sum_{l} \left\{ \sum_{u \in N_k} \sum_{v \in N_l} \left| \hat{f}(u,v) - \hat{f}(k,l) \right|^2 \right\} \right], \end{aligned} \tag{6}$$

where $\hat{h}_i(m,n;k,l)$ is the estimated PSF of the ith low resolution image, c is a constant and λ is the regularising parameter, and the partition function is given as

$$Z = \sum_{\{\boldsymbol{f}, \boldsymbol{d}\}} \exp\left[-\frac{U(\boldsymbol{f}, \boldsymbol{d} \mid \boldsymbol{G})}{T}\right], \tag{7}$$

where $\sum_{\{\boldsymbol{f}, \boldsymbol{d}\}}$ means the sum over all the possible configurations $\{\boldsymbol{f}, \boldsymbol{d}\}$.

In summary, the reconstruction of a high resolution image is the solution that maximises $P(\boldsymbol{f}, \boldsymbol{d} \mid \boldsymbol{G})$ which coincides with minimising the cost function given in (6). The authors have demonstrated that the solution of this reconstruction problem can be successfully obtained using the simulated annealing algorithm [8]. Using this reconstruction approach, it has the advantage of not requiring a separate registration phase. It is therefore possible to obtain higher resolution images even if the displacements of the low resolution images are unknown.

MEAN FIELD ANNEALING

It is well known that the major disadvantage of simulated annealing is its large computational effort. To avoid this computational burden, this section proposes the mean field annealing algorithm [13] to solve the high resolution image reconstruction problem. Mean field annealing is an optimisation technique which can be derived from two different perspectives: statistical mechanics [9] and information theory [13], and it has been shown to provide good results much faster than simulated annealing [10, 14]. In the field of

image processing, mean field annealing has been employed in several applications, notably image restoration [10, 14, 15], motion estimation [16], image segmentation [17], and etc. [18, 19].

Mean field annealing uses a deterministic approach to find the mean of the Gibbs distribution which is an approximation of the thermal equilibrium distribution of the temperature T. In terms of the high resolution image reconstruction problem, the mean of image pixel $f(k,l)$ can be given as:

$$\begin{aligned}\langle f(k,l)\rangle &= \sum_{\{\boldsymbol{f},\boldsymbol{d}\}} f(k,l)P(\boldsymbol{f},\boldsymbol{d}\mid \boldsymbol{G}) \\ &= \frac{1}{Z}\sum_{\{\boldsymbol{f},\boldsymbol{d}\}} f(k,l)\exp\left[-\frac{U(\boldsymbol{f},\boldsymbol{d}\mid \boldsymbol{G})}{T}\right]. \end{aligned} \tag{8}$$

It can be seen that the calculation of the above equation is not possible, or at least infeasible, since it involves interaction between all the possible configurations. The mean field theory suggests an approximation of (8) by the assumption that the mean of the field $f(k,l)$ can be updated by the mean values of its neighbours, and the mean value can also be approximated by its local energy, that is

$$\langle f(k,l)\rangle = \frac{1}{Z_{kl}}\sum_{f(k,l)\in R_D} f(k,l)\exp\left[-\frac{U(f_{kl},\langle \boldsymbol{d}\rangle \mid \boldsymbol{G})}{T}\right]. \tag{9}$$

The terms $U(f_{kl},\langle \boldsymbol{d}\rangle \mid \boldsymbol{G})$ and Z_{kl} are called the mean field local energy and local partition function at pixel (k,l), respectively. These can be written as

$$\begin{aligned} &U(f_{kl},\langle \boldsymbol{d}\rangle \mid \boldsymbol{G}) \\ &= c\cdot\left[\sum_{i=1}^{P}\sum_{m\in Y_k}\sum_{n\in Y_l}\left|g_i(m,n)-\sum_u\sum_v\langle\hat{f}(u,v)\rangle\langle\hat{h}_i(m,n;u,v)\rangle\right|^2 + \right. \\ &\qquad \left.\lambda\left\{\sum_{u\in N_k}\sum_{v\in N_l}\left|\hat{f}(u,v)-\hat{f}(k,l)\right|^2\right\}\right], \end{aligned} \tag{10}$$

and

$$Z_{kl} = \sum_{f(k,l)\in R_D}\exp\left[-\frac{U(f_{kl},\langle \boldsymbol{d}\rangle \mid \boldsymbol{G})}{T}\right]. \tag{11}$$

Note that, $m \in Y_k$ and $n \in Y_l$ mean a low resolution pixel (m,n) which is influenced by a high resolution pixel (k,l).

The mean of the displacement δ_{xi} can be approximated by:

$$\langle \delta_{xi}\rangle = \frac{1}{Z_d}\sum_{\delta_{xi}\in R_D}\delta_{xi}\exp\left[-\frac{U(\langle \boldsymbol{f}\rangle,\delta_{xi}\mid \boldsymbol{G})}{T}\right], \tag{12}$$

where the mean field local energy and local partition function are defined as

$$U(\langle \boldsymbol{f} \rangle, \delta_{xi} \mid \boldsymbol{G}) = c \cdot \left[\sum_m \sum_n \left| g_i(m,n) - \sum_u \sum_v \langle \hat{f}(u,v) \rangle \langle \hat{h}_i(m,n;u,v) \rangle \right|^2 \right], \quad (13)$$

and

$$Z_d = \sum_{\delta_{xi} \in R_D} \delta_{xi} \exp \left[-\frac{U(\langle \boldsymbol{f} \rangle, \delta_{xi} \mid \boldsymbol{G})}{T} \right], \quad (14)$$

respectively. The mean of the displacement δ_{yi} can be approximated in a similar manner.

In addition, the image intensities and the displacements are continuous values which implies that the summations $\sum_{f(k,l) \in R_D}$, $\sum_{\delta_{xi} \in R_D}$ and $\sum_{\delta_{yi} \in R_D}$ may be replaced by integral equivalents.

By using the above approximations, the equilibrium state at each temperature T can be obtained through the mean field. As the temperature $T \rightarrow 0$, this approximation will coincide with the exact distribution, and the image $\boldsymbol{f}$ will be equal to $\langle \boldsymbol{f} \rangle$. In summary, the mean field annealing algorithm for high resolution image reconstruction can be stated as:

```
T ⇐ initial temperature
while (T > T_min)
    do until (a steady state is reached)
        for all image pixels
            Calculate U(f_kl, ⟨d⟩ | G)
            Calculate the mean ⟨f(k,l)⟩.
        end
        for all image displacements
            Calculate U(⟨f⟩, δ_xi | G) and U(⟨f⟩, δ_yi | G)
            Calculate the mean ⟨δ_xi⟩, and alternately ⟨δ_yi⟩.
        end
    end
    Decrease T
end
```

EXPERIMENTAL RESULTS

In order to evaluate the performance of the mean field annealing algorithm, a picture of characters with different sizes was imaged by a *charge-coupled device* (CCD) camera. The pixel size in each digital image was measured to be 3.06 and 2.19 mm along the x and y directions, respectively. The picture was placed on a mechanical device which can be shifted on both directions with an accuracy of 0.1 mm. By shifting this mechanical device, 16 low resolution

images of size 64 × 64 pixels were taken with different displacements, and Fig. 2a illustrates one of the low resolution images.

Mean field annealing was applied to the low resolution images in order to improve the resolution. The algorithm was implemented with $T_o = 150$, $c = 0.001$, $\lambda = 0.0001$, and the temperature was decreased using an exponential rule $T_n = 0.95 \times T_{n-1}$. The initial estimated image was defined as the constant pattern. In addition, the algorithm was terminated when the temperature was lower than 0.01.

Two higher resolution images were reconstructed, and are shown in Fig. 2c and 2e with resolution increase of two-fold and four–fold, respectively. From the results, it can be seen that invisible detail in the low resolution images becomes clearly apparent in both reconstructions. In terms of the frequency domain, Fig. 2b illustrates the spatial frequency spectrum of the low resolution image shown in Fig. 2a. Whereas Fig. 2d and 2f illustrate the spatial spectrum of the corresponding high resolution images. From these illustrations, it is obvious that some of the distorted high frequency spectrum can be recovered when applied this reconstruction algorithm.

CONCLUSIONS

This paper has considered the problem of increasing image resolution from multiple low resolution images. To avoid the limitations of existing methods, this paper has demonstrated that the high resolution image reconstruction task can be formulated as an optimisation problem, and that a solution can be found using the mean field annealing algorithm. This new technique has the advantage over existing techniques in that it does not require a separate registration phase. It is therefore possible to obtain higher resolution images even if the displacements of the low resolution images are unknown. In addition, the experimental results have demonstrated the success of this new algorithm for real–world images.

In summary, the contributions of this paper are: (i) to present an optimisation approach for reconstructing a high resolution image, and (ii) to show that the proposed technique can be successfully applied to real–world images.

ACKNOWLEDGMENT

The authors would like to thank Dr. Ramakrishna Kakarala for useful discussions.

REFERENCES

[1] R. Y. Tsai and T. S. Huang, "Multiframe image restoration and registration," in Advances in Computer Vision and Image Processing: Vol.1 (T. S. Huang, ed.), pp. 317–339, JAI Press, 1984.

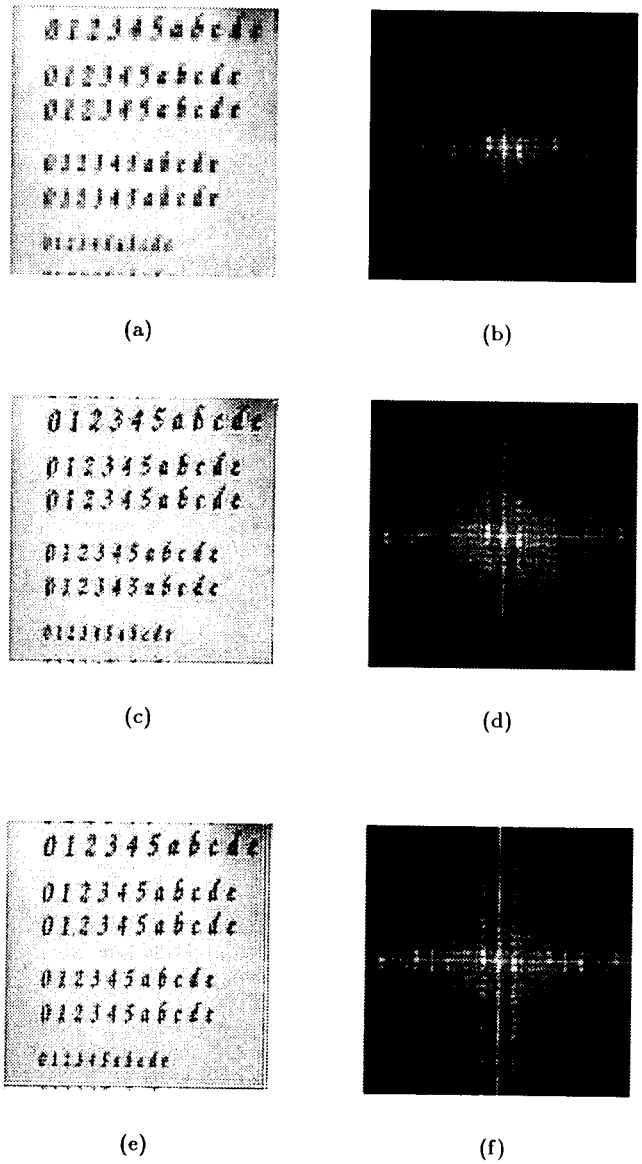


Figure 2: (a) A real–world low resolution image of size 64 × 64 pixels, (b) its corresponding frequency spectrum, (c) a reconstruction of size 128 × 128 pixels, (d) its corresponding frequency spectrum, (e) a reconstruction of size 256 × 256 pixels, (f) its corresponding frequency spectrum.

[2] S. Kim, N. Bose, and H.M.Valenzuela, "Recursive reconstruction of high resolution image from noisy undersampled multiframes," IEEE Trans. Acoust., Speech, Signal Processing, vol. 38, no. 6, pp. 1013–1026, 1990.

[3] H.Ur and D.Gross, "Improved resolution from subpixel shifted pictures," CVGIP: Graphical Models and Image Processing, vol. 54, no. 2, pp. 181–186, 1992.

[4] A.M.Tekalp, M.K.Ozkan, and M.I.Sezan, "High-resolution image reconstruction from lower-resolution image sequences and space-varying image restoration.," in Proc. IEEE Int. Conf. Acoust. Speech, Signal Processing, 1992, pp. III169–III172.

[5] T. Komatsu, K. Aizawa, T. Igarashi, and T. Saito, "Signal-processing based method for acquiring very high resolution images with multiple cameras and its theoretical analysis," IEE Part I, vol. 140, no. 1, pp. 19–25, 1993.

[6] M. Irani and S. Peleg, "Improving resolution by image registration," CVGIP: Graphical Models and Image Processing, vol. 53, no. 3, pp. 231–239, 1991.

[7] H. Stark and P. Oskoui, "High-resolution image recovery from image-plane arrays, using convex projections," J. Opt. Soc. Amer. A, vol. 6, no. 11, pp. 1715–1726, 1989.

[8] T. Numnonda, M. Andrews, and R. Kakarala, "High resolution image reconstruction by simulated annealing," Optics Communications, vol. 108, no. 2, 1994.

[9] G.L.Bibro, R.Mann, T.K.Miller, W.E.Snyder, D.E.Van den Bout, and M.White, "Optimization by mean field annealing," in Advances in Neural Network Information Processing Systems (D.S.Touretzky, ed.), pp. 91–98, Morgan-Kaufmann, 1989.

[10] H.P.Hiriyannaiah, G.L.Bilbro, W.E.Snyder, and R.C.Mann, "Restoration of piecewise-constant images by mean-field annealing," J. Opt. Soc. Am. A, vol. 6, no. 12, pp. 1901–1912, 1989.

[11] S.Geman and D.Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," IEEE Trans. Patt. Anal. Machine Intel., vol. 6, no. 6, pp. 721–741, 1984.

[12] T. Numnonda. PhD thesis (In preparation), Department of Electrical and Electronic Engineering, University of Auckland, New Zealand.

[13] G.L.Bilbro, W.E.Snyder, S.J.Garnier, and J.W.Gault, "Mean field annealing: A formalism for constructing GNC-like algorithms," IEEE Trans. Neural Networks, vol. 3, no. 1, pp. 131–138, 1992.

[14] D. Geiger and F. Girosi, "Parallel and deterministic algorithms from M-RF's: Surface reconstruction," IEEE Trans. Patt. Anal. Machine Intell., vol. 13, no. 5, pp. 401–412, 1991.

[15] J.Zhang, "The mean field theory in em procedures for blind Markov random field image restoration," IEEE Tran. Image Processing, vol. 2, no. 1, pp. 27–40, 1993.

[16] J.Zhang and J.Hanauer, "The mean field theory for image motion estimation," in Proc. IEEE Int. Conf. Acoust. Speech., Signal Processing, 1993, pp. V197–V200.

[17] W.Snyder, A.Logenthiran, P.Santago, K.Link, G.Bilbro, and S.Rajala, "Segmentation of magnetic resonance images using mean field annealing," in Proceedings of Int. Conf. on Information Processing in Medical Imaging, (United Kingdom), 1991, pp. 218–226.

[18] J.Zerubia and R.Chellappa, "Mean field approximation using compound Gauss–Markov field models for edge detection and image estimation," IEEE Trans. Neural Network, vol. 4, no. 4, pp. 703–709, 1993.

[19] L.Herault and R.Horaud, "Finger-ground discrimination: A combinational optimization approach," IEEE Trnas. Patt. Anal. Machine Intell., vol. 15, no. 9, pp. 899–914, 1993.

HARDWARE NEURAL NETWORK IMPLEMENTATION OF TRACKING SYSTEM

George G. Lendaris[1], Robert M. Pap, Richard E. Saeks & Chas. R. Thomas
ACCURATE AUTOMATION CORPORATION
7001 Shallowford Road, Chattanooga, TN 37421
(615) 894-4646 FAX (615) 894-4645
[1] Portland State University, P.O. Box 751, Portland, OR 97207
(503) 725-4988 e-mail: lendaris@sysc.pdx.edu
Richard M. Akita
Naval Command Control and Ocean Surveillance Center, RDT&E Division
San Diego, CA 92152 (619) 553-5611

Abstract: A neural network (NN) filter/target-tracking system has been developed as reported in [6]. The design accepts and inputs signal data to a noise/target classifier which uses spectral estimation techniques to distinguish noise from real targets. In that design, the NN is used to calculate the coefficients of an auto regressive linear predictive filter. The current evolution of that design invokes the use of Lagrange Multiplier methods to incorporate known characteristics of the noise vs. signal. A (linear) Hopfield NN is used to perform the constrained optimization to solve for the filter coefficients. This algorithm has been demonstrated on real stochastic data. The filter resulting from this process succeeds in reducing the noise, whose structure was learned by the NN. Not only did this approach reduce structured noise without target attenuation or the addition of a 'ghost' signal, but it also lowered the base level of the resultant signal significantly. The overall concept has been tested and validated using real data on a workstation and the nardware NN implementation has been validated. This concept has been tested on the AAC Multiple Instruction Multiple Data (MIMD) Neural NetworkProcessor (NNP) hardware. Each processor runs at 140 million connections/sec with 8K neurons. An expanded version of the system performs a total of a billion plus connections/sec. Unlike classical SIMD NN architectures, which are really general purpose array processors, this MIMD system architecture was custom designed for NN applications.

INTRODUCTION

A neural network filter and target-tracking system has been developed for the Navy by AAC [6] which uses spectral estimation techniques to distinguish targets from background and noise (the combination is here called structured noise) . The

0-7803-2026-3/94 $4.00 © 1994 IEEE

evolution of that design is described here. The improvements are based on using Lagrange Multiplier methods to incorporate characteristics of the structured-noise vs. signal that are known in the tracking context. A specially designed linear Hopfield network is used to solve for the coefficients of the auto-regressive linear predictive filter

The basic layout is shown in Figure (1). A sequence X_k containing both signal and structured noise (**str-noise**) is input to the filter component whose task it is to filter out the signal and pass through only an estimate of the str-noise, which is then combined with the input sequence to yield just an estimate of the target-related signal. [Note: because the words signal and str-noise both start with s, a different subscript for one of the terms was chosen. The letter 'c' is used for the str-noise -- motivated by the word 'contamination.'] Then, the estimate/prediction of the str-noise (contamination) at time k, $\hat{X}_{ck}$, is subtracted from the input sequence X_k to yield an estimate of the signal, $\hat{X}_{sk}$ at time k:

$$\hat{X}_{sk} = X_k - \hat{X}_{ck}.$$

A selected neural network structure [3, Chapt. 14]is used to determine the coefficients of the equation in the "box" that predicts the contamination at time k in terms of the incoming sequence (via the well known autoregressive formula):

$$\hat{X}_{ck} = \sum_{i=1}^{n} a_i X_{k-i}, \qquad k \geq n+1 \qquad (1)$$

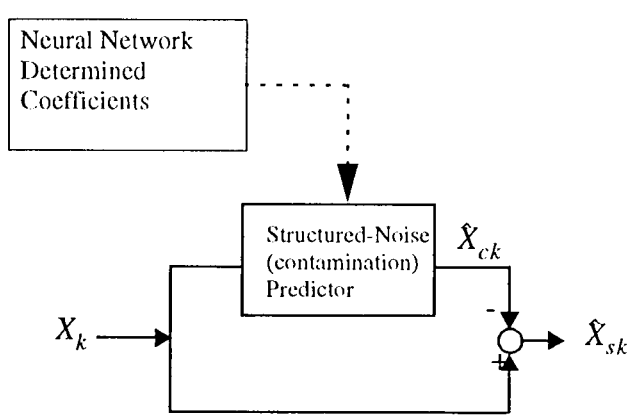


Figure 1: Layout of Structured-Noise Filter.

where $X_k = I_k + iQ_k$ is a complex sequence containing both in-phase and quadrature components and the a_i are arbitrary complex coefficients.

In the earlier work [6], the a_i coefficients were selected based only on information about the structured noise, without taking into account any *a-priori* information that might be known about the signals representing the targets being tracked. The **improved approach described here** injects constraints into the process of selecting the a_i's so as to incorporate such *a priori* information.

CONSTRAINED OPTIMIZATION

Based on the assumption that the str-noise (structured noise) of interest could be modeled by a relatively low order filter [2], a least squares approach to determining the autoregressive filter coefficients was initially adopted. The Linear Hopfield network was used to minimize

$$J(a_i) = \sum_{k=n+1}^{M} \left\| X_k - \hat{X}_{ck} \right\|^2 = \sum_{k=n+1}^{M} \left\| X_k - \sum_{i=1}^{n} a_i X_{k-i} \right\|^2 \qquad (2)$$

over all possible complex coefficient vectors $\underline{a}$ was used. This approach yielded an excellent predictor for the str-noise, but it often also contained a good prediction of the *signal* as well. So after the subtraction indicated in Figure 1 took place, there was not significant improvement in the signal-to-noise ratio.

In effect, by choosing the $\underline{a}$ vector in the minimization process without information about the expected signal, a filter is obtained which optimally predicts the contamination (str-noise) but is oblivious to the signal. Indeed, without use of a priori information about the signal, the manner in which the resulting predictor deals with the signal is not at all under control; it may ignore it or predict it perfectly.

To address this problem, **a constraint is added** to the process which, in effect, says "predict the structured noise as well as possible, being mindful of what is known about the expected signal(s)." To this end, it is here assumed that the signal from a target has *constant magnitude*, and a *phase that changes linearly* over the extent of the target. This (assumed) *a priori* information is captured in the model

$$\hat{X}_{sk} = ce^{a+ibk}$$

Although not precisely true, the constant ("dc") value of the magnitude (c in the above equation) and the linear (affine) component of the phase typicaly dominate the return from a target. Moreover, as a first approximation, even though it changes linearly over the extent of the target, it is reasonable to assume that the phase is constant over the relatively short n-sample interval seen by the filter (i.e., b is "small

enough," in which case the signal representing the target may be approximated by

$$\hat{X}_{sk} = C = ce^{a}.$$

Accordingly, a pure signal from a target would yield:

$$\sum_{i=1}^{n} a_i X_{k-i} = \sum_{i=1}^{n} a_i C = C\left(\sum_{i=1}^{n} a_i\right)$$

and hence if the selected $\underline{a}$ vector were to have the additional property that

$$\sum_{i=1}^{n} a_i = 0 \tag{3}$$

the str-noise predictor would not pass through the signal part of the incoming sequence. To accomplish this, the coefficients of the str-noise predicition are to be determined as follows: **minimize equation (2) subject to the constraint that equation (3) holds**.

With the aid of the Lagrange multiplier theorem, this constrained optimization problem can be converted into an equivalent unconstrained optimization that minimizes $J(a_i) + \lambda\left(\sum_{i=1}^{n} a_i\right)$ over all $\underline{a}$ and λ.

A side effect of the constraint imposed above is for the resulting filter to suppress the dc component of the str-noise as well as that of the signal. To accomodate this potential difficulty, the dc component of the str-noise is subtracted off before computing the autoregressive coefficients for the filter and is then added back into the prediciton model. The data used for developing the str-noise (contamination) predictor is generated via:

$$\tilde{X}_k = X_k - X_{cdc}; \text{ where } X_{cdc} = \left(\frac{1}{M}\right)\sum_{k=1}^{M} X_{ck}$$

and X_{ck} is a sequence containing str-noise (contamination) data only.

In effect, the dc component of the str-noise is included in the predictor via analytical means, thereby eliminating the conflict between minimizing $J(a_i)$ and satisfying the "no dc" constraint.

Finally, it would be possible to incorporate the additional (assumed) *a priori* information that the signal phase is constant over the n-sample interval used by the filter by adding a second linear constraint to the optimization problem. To explore this, a process using $\ln(X_k)$ rather than X_k itself was investigated. By doing so, the

signal representing the target takes on the form of a constant term plus a linear term $\ln(X_k) = \ln(c) + a + bk$. With this formulation, the str-noise predictor would possibly be superior in ignoring the signal if both of the linear constraints

$$\sum_{i=1}^{n} a_i = 0 \quad \text{and} \quad \sum_{i=1}^{n} a_i i = 0$$

were satisfied. In practice, however, the benefit gained by incorporating the "constant phase" assumption was not sufficient to justify the added implementation complexity, and therefore, the single constraint formulation is being pursued.

SOLUTION VIA LINEAR HOPFIELD NETWORKS

Given a complex time-series such as X_k defined earlier, a modified Hopfield network was given in [6] that computes the complex autoregressive coefficients a_i, $i = 1, \ldots n$ which best fit Equation (1) in the sense of minimizing the performance function $J(a_i)$ defined in Equation (2). The network uses continuous-valued data as opposed to binary, and operates in discrete time steps. The neural-element transfer function (from the summed inputs of a neural element to its output) is linear. The output of the i-th neural element is the solution for a_i. Nonlinearity is introduced only in the calculation of the network's inputs and weights (see equations for I and T below). In [6] it is shown that the time series performance function $J(a_i)$ satisfies the definition of a computational "energy function" with respect to the complex Hopfield states a_i, $i = 1, \ldots n$. It is shown therein that the update formula for the modified, complex Hopfield network is essentially identical to the real case. The complex case differs in that only the real part of the input excitation, Re $[I]$, is used to update Re $[a_i]$, and only the imaginary part (with a sign change) to update Im $[a_i]$.

The "input excitation" for each element of the Hopfield network is constructed from the input data string as follows:

$$I_j = \sum_{k=n+1}^{M} \left(\bar{X}_k X_{k-j} \right) \qquad j = 1, 2, \ldots, n,$$

($\bar{X}$ is the complex conjugate of X) and the weight matrix for the Hopfield network is constructed as follows:

known in the numerical analysis field that there are computationally superior methods for solving such equations, such as the various Gauss elimination methods. So why the iterative Linear-Hopfield Neural-Network approach? The proposed answer lies in the **hardware implementation** of the computing engine. Accurate Automation has developed a hardware neural network processor that is optimized for neural-network type computations, and runs in real time on an multiple-instruction, multiple-data (MIMD) processor, which promises to win the competition, even using less efficient computational algorithms. This processor is implemented on a card that can be run in a standard PC type computer.

The underlying philosophy in the design of the Accurate Automation Corp. MIMD **Neural Network Processor** module (AAC NNP) has been to have it run in real time and to achieve maximum computational efficiency in both a single processor and multiprocessor environment by optimizing the design to compute neuron values - and *nothing but neuron values* [7]. Indeed, this is ideally suited to a *neural network application* and stands in stark contrast to previously proposed processors which are typically based on classical SIMD (single instruction multiple data) matrix/vector multiplication architectures. Rather, the design fully *exploits the intrinsic characteristics* (sparse, local, random) *of the neural network topology.* Moreover, by using an *MIMD parallel processing architecture* one can update multiple neurons in parallel with efficiency approaching 100% as the size of the neural network increases.

To achieve the desired efficiency, Accurate Automation's design:

Uses an *instruction set which is optimized for neural network processing* allowing one to compute a neuron activation without arranging the weight matrix into linear arrays and/or inserting "artificial zero weighted connections",

Uses an *MIMD (multiple instruction multiple data) parallel processing architecture* to permit neurons with totally different input topologies to be updated simultaneously without loss of efficiency, and

Uses *dual neuron memories* to virtually *eliminate memory contention* and *maintain absolute memory coherence.*

This architecture allows AAC to implement a relatively simple single processor NNP module and then string together multiple NNP modules along a dedicated Interprocessor Bus with *computational power (and cost) increasing "almost" linearly with the number of modules* [7].

The AAC MIMD Neural Network Processor:

Is designed to *implement multiple interconnected neural networks of differing architecture simultaneously* using *16-bit twos-complement binary fixed-point arithmetic* with up to *8k total neurons* and *32k connection weights* per module,

Is capable of running at *140,000,000 connections* (byte wide multiply/additions) per second per module for a total of **one billion plus connections per second in an 8 processor array,**

Supports two I/O buses, an *Interprocessor Bus* which can also be used for on-

line I/O in parallel with the computational process, and a *Memory I/O Bus* through which the various processor memories may be mapped into the memory space of a supporting microprocessor or DSP for downloading programs, connection weights, etc, and

Each processor in an NNP array is controlled by a separate program written in a *"RISC-like" instruction set* supported by an *NNP Module Simulator*, an *Assembler*, a *Neural Network Compiler*, and the *Accurate Automation Neural Network Toolbox* [1].

Unlike the classical SIMD neural network architectures which are really general purpose array processors which invert matrices and do Fourier transforms as readily as they do neural networks, the NNP architecture is custom designed for neural network applications, such as the one discussed in this paper.

See [7] for a functional description of the AAC Sparse MIMD Neural Network Processor.

CONCLUSION

Neural networks can be used in a process that adaptively adjusts itself for the task of removing structured-noise out of signal+structured-noise, in real time. The adaptive process can learn a broad range of structured-noise, on line, and perform the filtering task without compromising the valid information. The hardware Neural Network Processor described allows for a real-world implementation using a PC type computer with an ISA bus. Various types of signals have been processed -- both real data as well as simulated. The concept developed for the processing needed to accomplish the signal filtering task allowed us to come up with a generic hardware implementation for neual network processes that is faster than done with previous generations of neural network hardware. The resulting technology has a broad range of real-world applications.

ACKNOWLEDGEMENTS

This work was sponsored by the US Navy Small Business Innovation Research program and administered by the Naval Command, Control and Ocean Surveillance Center, RDT&E Division, Code 451 under a (SBIR) Phase II contract number N66001-90-C-7021. The authors wish to express their appreciation to the Naval Air Systems Command, Codes PMA-213, AIR 251 and the Office of Naval Research code 362 and 342 CN for their support of this work.

REFERENCES

[1] Accurate Automation, **Neural Network Toolbox Manual**, Accurate Automa-

tion Corp., 7001 Shallowford Rd., Chattanooga, TN 37421, 1993.

[2] Haykin, S., W. Stehwien, C. Deng, P. Weber & r. Mann, "Classification of Radar Clutter in an Air Traffic Control Environment," **Proceedings of the IEEE**, Vol. 79, No. 6, pp. 742-772, 1991

[3] Maren, A.J., C.T. Harsten, & R.M. Pap, **Handbook of Neural Computing Applications**, Academic Press, 1990.

[4] Mathia,K., G. Lendaris & R. Sacks, "The Linear Hopfield Network and its Applications," submitted, 1994.

[5] Mathia,K. & R. Sacks, "Inverse Kinematics via Linear Dynamic Networks," in **Proceedings of World Congress on Neural Networks (WCNN-94)**, San Diego, CA, Earlbaum Assoc., NJ, 1994.

[6] Pap,R., R. Sacks, C.Thomas & R.Akita, "Neural Network Implementation of Stochastic Filters for Radar Tracking," in **Proceedings of IEEE Workshop on Neural Networks for Signal Processing**, Helsingoer, Denmark, 1992.

[7] Sacks, R., Priddy, K., Pap, R., and S. Stowell, "On the Design of the MIMD Neural Network Processor," in **Proceedings of SPIE Symposium on Neural Networks V**, Orlando, FL, 1994.

FAST IMAGE ANALYSIS USING KOHONEN MAPS

D. Willett, C. Busch, F. Seibert,
Visual Computing Group
Darmstadt Computer Graphics Center
Wilhelminenstraße 7, D 64283 Darmstadt
Tel: +49 6151 155 255, Fax: +49 6151 155 299
E–mail: busch@igd.fhg.de

Abstract – The following paper considers image analysis with Kohonen Feature Maps. These types of neural networks have proven their usefulness for pattern recognition in the field of signal processing in various applications. The paper reviews a classification approach, used in medical applications, in order to segment anatomical objects such as brain tumors from magnetic resonance imaging (MRI) data. The same approach can be used for environmental purposes, to derive land–use classifications from satellite image data. These applications require tremendous processing time when pixel–oriented approaches are chosen. Therefore the paper describes implementation aspects which result in a stunning speed–up for classification purposes. Most of them are based on geometric relations in the feature–space.

The proposed modifications were tested on the mentioned applications. Impressive speed–up times could be reached independent of specific hardware.

1 INTRODUCTION

Image classification is a crucial step in the image processing pipeline. Typical applications for image classification are the interpretation of medical data or remote sensing data. In medical applications modern image acquisition techniques like magnetic resonance imaging (MRI) supply 3D data sets of high resolution and quality. 3D data need to be classified in order to separate different tissue types such as brain tumors in the image data. Further processing will use the classified data as input for 3D reconstruction algorithms of the volume. Advanced volume renderers, as in [5],[9] or [11] require opacity–maps, as well as 3D surface reconstruction methods like marching cubes [10] or Delaunay triangulation [12] require a description of the surface, which can be easily derived from the classified data.

Environmental applications use remote sensing data in order to achieve semantic ground information. Remote sensing data stemming from satellite–based sensors like SPOT, Landsat–TM or ERS1 can serve as multispectral data input for environmental control systems. Robust image analysis of the data delivers a land–use–classification.

The following paper reviews in Section 2 a pixel–oriented discrimination method for image data based on topological mappings of Kohonen[7],[8]. Application stud-

0-7803-2026-3/94 $4.00 © 1994 IEEE

ies as in [4] and [1] have shown that this method can either perform feature extraction, clustering and classification in a unique approach or – in a more traditional manner –separate the single steps and calculate the independent components of the classification pipeline [13],[2].

Corresponding to the immense amount of data that is analyzed in the above mentioned applications, the use of the Kohonen Feature Map can be extremely expensive, especially when long feature–vectors are considered. Thus Section 3 proposes several modifications of Kohonen's algorithm. They were inspired by the insight, that most of the computational time is wasted in calculations of Euclidean distances in the feature–space in order to determine the neuron which bears the closest weight–vector to a presented input–vector. This task corresponds to the nearest neighbor search in a multidimensional space. It can be shown that most of the weight–vectors could be excluded from consideration.

In addition basic constraints can be used, like the similarity of consecutive input–vectors as they appear in the classification of consecutive pixels stemming from a homogenous region in an image. Those modifications can be used for clustering or classification tasks. For classification purposes further optimization is possible, since only a neuron's class assignment has to be determined and not the closest neuron itself.

The modifications were implemented and tested on the described applications. Section 4 reports some results in terms of speed–up times. The times are not based on any specific hardware, but nevertheless computing the image classification on powerful hardware like vector– or parallel–processors can realize further speed–up.

2 CLUSTERING AND CLASSIFICATION

2.1 General Remarks

In general, a robust classification pipeline for the automatic recognition, classification and visualization of the data can be divided into the following three tasks:

I) feature extraction
II) cluster analysis
III) supervised classification

Those steps can be solved separately, or in a single approach. Kohonen Feature Maps are capable of solving that task in a unique paradigm, since they allow subspace mapping, visualization of a multidimensional texture feature–space and supervised classification. This is explained in detail in [1] and [3].

2.2 Kohonen Mapping

The Kohonen Map as introduced in [7] or [8], is a self–organizing network which is basically trained without supervision. It organizes a set of input patterns in a topological structure represented by neurons, where the relations between different patterns are preserved.

To use the Kohonen map for cluster analysis, the Kohonen Map can be configured with a 3D output–layer as shown in Figure 1. The neurons in the input–layer pick up the data from the feature–extractor or directly from the image. The weights associated with each connection of an output–layer's neuron are adjusted during training where only one single neuron can be active at a time. A time–dependent neighborhood implies an update in the neuron's environment as well. After the self–organizing training–procedure, each neuron in 3D represents a cluster in the multi-dimensional feature–space. Therefore the network can be used for cluster analysis and dimensionality reduction as described in [3].

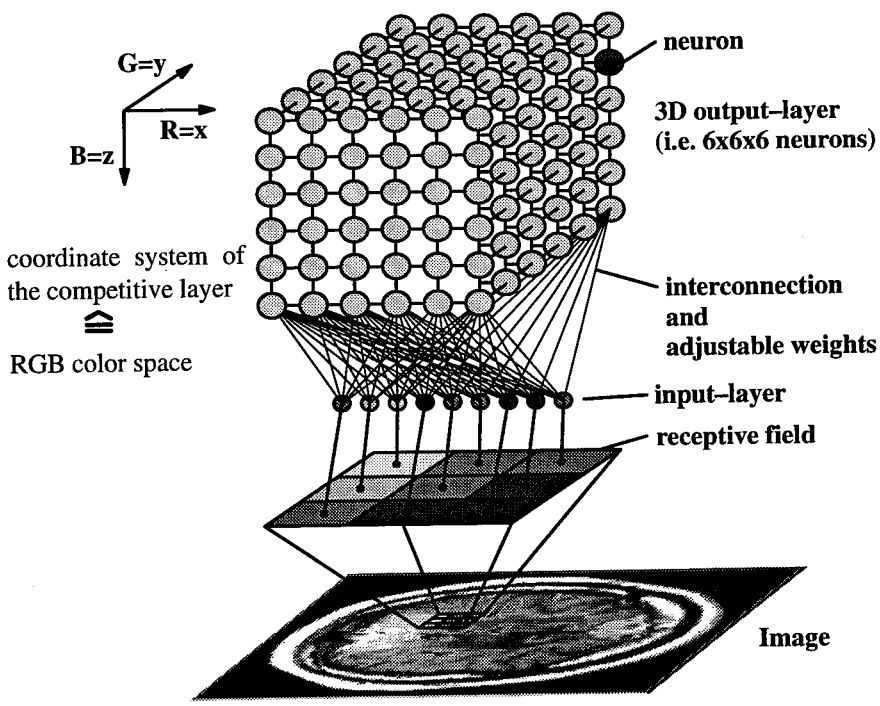


Fig. 1: Topology of the 3D–Kohonen map.

Essential for training as well as for the work procedure of the network is the spatial distance in the feature–space, since it decides which neuron in the output–layer is activated. For a presented data–vector **x** the distance is calculated to all weight–vectors $\mathbf{m}_i$ representing the connections between the input–layer and the competitive layer. If N is the dimension of the data, the Euclidean distance d_i between **x** and $\mathbf{m}_i$ is defined as

$$d_i = \| \boldsymbol{x} - \boldsymbol{m}_i \| = \sqrt{\sum_{j=1}^{N} (x_j - m_{ij})^2} \qquad (1)$$

The neuron c with the minimum distance is activated, where

$$d_c = \min_i \{d_i\} \qquad (2)$$

The activated neuron is of fundamental importance. On one hand in its environment the weight–vectors will be updated according to the training rules [7]. On the other hand its coordinate will be taken as RGB–information for clustering tasks or its class assignment will be taken as class information for a classification task. Fast and effective calculation of a winning neuron will be the subject of the next Section.

In order to use a self–organized Kohonen Feature Map for supervised classification a class assignment for each neuron is required. User defined training areas can set up a training set, where a class–assignment exists for each sample, i.e. a feature–vector. Sequentially presentation of labeled input–vectors and subsequent majority voting can lead to class–assignment of each output neuron. For optimal description of the Bayes decision boundary, additional postprocessing with learning vector quantization (LVQ) is recommended [7].

3 SPEED–UP METHODS

3.1 Remarks

The most time consuming part of the classification is the determination of the neuron in the output–layer which is activated by a given input. It is the neuron whose weight–vector is the closest to the input–vector in the feature–space.

According to equation (1) and (2) the activated neuron is determined by calculating the exact Euclidian distances of each weight–vector to the input–vector and selecting the one with the least distance. Slight changes of this process can bring the first improvements.

Avoiding the square–root–function. In order to calculate the Euclidean distance of two vectors, the square–root–function is used. For the given purpose only the relation between the Euclidean distances is of importance. Thus we simply change equation (1) and (2) and calculate the squared distances

$$d_i^2 = \| \boldsymbol{x} - \boldsymbol{m}_i \|^2 = \sum_{j=1}^{N} (x_j - m_{ij})^2 \qquad (3)$$

and determine the activated neuron c with

$$d_c^2 = \min_i \left\{ d_i^2 \right\} \qquad (4)$$

So in the following text the *distance* in general refers to the square of an Euclidian distance.

Furthermore the term *best vector* will be used for the temporary closest vector found. A *good vector* in general will be any weight vector close to the input–vector.

Threshold Summation. During the calculation of the distances, the *best* distance can be used as a threshold [6]. If, while performing the summation in formula (3) we reach a sum greater than the threshold, even if $j < N$, then we can stop and disregard this vector.

Usage of Correlated Input. The threshold summation and other optimization given in this text are most effective whenever a *good* vector is found early during the cal-

culation, so that the determination of the exact distance of many other vectors can be avoided. In a lot of applications the input presented in one step correlates to the input presented in the previous step. If such a correlation is obvious, as in the pixel-based image–segmentation, the activated neuron of the previous step should be considered first in the activation–algorithm.

3.2 Immediate Activation

Once a map has been trained, its weight–vectors remain static. The distances between the weight–vectors themselves can be calculated in advance in order to be used to optimize the calculations. A first approach is illustrated in the figure below which refers to a 2–dimensional feature–space. The circle around each weight–vector has a radius r_i of half the minimum distance to all other vectors.

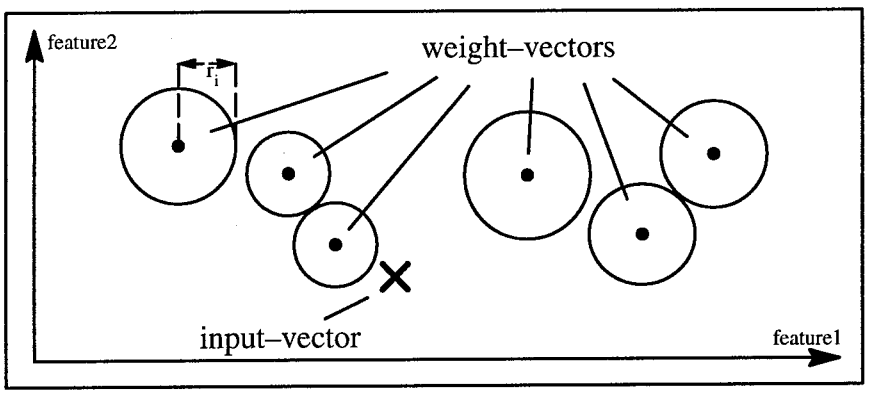


Fig. 2: Areas of immediate activation around the weight–vectors in the feature–space

Once an input–vector turns out to be situated inside of an activation area,

$$d_i^2 \leq r_i^2 \quad \Rightarrow \quad d_i \leq r_i \quad \Rightarrow \quad d_i = \min_j \{d_j\} \qquad (5)$$

the closest weight–vector is the one in the center of that area. Thus the activated neuron is found.

When using labeled neurons, like for classification purposes, only the label of the activated neuron is relevant. Areas of neurons with the same label may overlap. In Figure 3 the labels of the neurons are represented by different shadings of the areas.

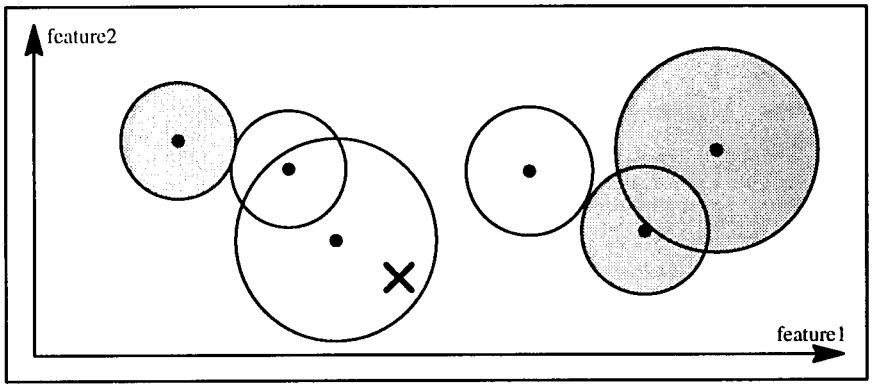


Fig. 3: Areas of immediate classification (identical shading refers to identical classes)

The immediate activation does not give an idea of how to find the weight–vector in whose circle the input might be situated. In the worst case it is the last vector that is being considered and the immediate activation does not save time at all. For this reason Section 3.3 and 3.4 present two relations which can be used in addition to the immediate activation to avoid the determination of several distances.

3.3 Triangle Relation

From a mathematical point of view we consider the feature–space as a metric space. In a metric space the triangle–relation (6) is valid, where d is the distance between two points.

$$d(A, C) \leq d(A, B) + d(B, C) \tag{6}$$

Looking at any two weight–vectors $\mathbf{m_1}$ and $\mathbf{m_2}$ and an input–vector $\mathbf{x}$, where the vectors express the coordinates of points in the feature–space, the relation can be transposed to (7).

$$d(\boldsymbol{x}, \boldsymbol{m}_2) \geq d(\boldsymbol{m}_1, \boldsymbol{m}_2) - d(\boldsymbol{x}, \boldsymbol{m}_1) \tag{7}$$

Assuming that the distance between $\mathbf{m_1}$ and $\mathbf{m_2}$ is at least twice as high as the distance between $\mathbf{m_1}$ and $\mathbf{x}$, it can be concluded that in no case $\mathbf{m_2}$ can be closer to $\mathbf{x}$ than $\mathbf{m_1}$.

$$d(\boldsymbol{m}_1, \boldsymbol{m}_2) \geq 2\, d(\boldsymbol{x}, \boldsymbol{m}_1) \Rightarrow d(\boldsymbol{x}, \boldsymbol{m}_2) \geq d(\boldsymbol{x}, \boldsymbol{m}_1) \tag{8}$$

The activation–algorithm can make use of this circumstance by not considering those vectors whose distance to the actual *best* vector is at least twice that high as the distance of the *best* vector to the input–vector. For the case that the squared distances are determined, equation (8) can be transposed to:

$$d^2(\boldsymbol{m}_1, \boldsymbol{m}_2) \geq 4\, d^2(\boldsymbol{x}, \boldsymbol{m}_1) \Rightarrow d(\boldsymbol{x}, \boldsymbol{m}_2) \geq d(\boldsymbol{x}, \boldsymbol{m}_1) \tag{9}$$

where $d(\mathbf{x},\mathbf{m_i})$ corresponds to d_i in equation (3).

3.4 Minimum Distances Derived from the Vector–Sums

In Section 3.3 we used the triangle relation to derive a minimum value for a specific distance. Once this minimum value turns out to be greater than the *best* value to that point of the calculation, the exact distance does not need to be determined anymore.

Another possibility which allows the determination of such a minimum value with only a few calculations is based on the absolute sum–values of the vectors. Using the relation

$$d^2(\boldsymbol{m}_1, \boldsymbol{m}_2) \geq \frac{\left(\sum \boldsymbol{m}_1 - \sum \boldsymbol{m}_2\right)^2}{N} \qquad (10)$$

where $\sum \boldsymbol{m}_i := \sum_{j=1}^{N} m_{ij}$ and $\boldsymbol{m}_i := (m_{i1}, m_{i2}, \ldots, m_{iN})$

one gets another lower boundary for the distance of two vectors, which can be derived from the sum–values of the vectors. The correctness of relation (10) is fairly easy to prove.

3.5 Maximum Likelihood–Search

The immediate activation presented in Section 3.2 offers a fast way to determine the closest vector. The time needed to find out that an input is inside an area of immediate activation mainly depends on the order in which the vectors are looked at. In the best case the closest vector is considered first and the immediate activation prevents the determination of all other distances. In the worst case the closest vector is being looked at after all other vectors, so that the immediate activation does not help at all.

The main aspect of the Maximum Likelihood–Search is the usage of the minimum values presented in Sections 3.3 and 3.4 to control the order in which the vectors are considered. The idea is that the vector with the least minimum value has the highest probability to be the vector that is being searched for.

3.6 Approximate Determination of a *Good* Vector

As mentioned in Section 3.1 the success of most of the presented methods largely depend on how fast a *good* vector is found. If there is no correlation between the consecutive input–vectors, or if the correlation is weak, another method can be applied. The activation is split up into a two–pass algorithm. During the first pass an approximation is applied to determine a *good* vector as an approximate solution. The second pass should be handled like a normal (optimized) activation–algorithm, starting with the *good* vector found in the first pass.

A method to determine a *good* vector shall now be suggested. It allows the fast determination of a vector which is at most n times farther away from the input–vector than the accurate closest vector. The parameter n can be set to any real value greater than or equal to one.

The basic idea of the approximation is the application of the triangle–relation (3.3) and the vector–sums (3.4) to mark those vectors that certainly cannot be more than n times closer to the input than the *best* vector found so far. The marked vectors do

not need to be considered during the further approximate pass of the algorithm. Once all vectors are marked or have been considered, the *best* vector found to that point is at most n times farther away from the input than the accurate closest vector.

3.7 N–Tree Based Search

Another example of a fast nearest neighbor search is the organization of the given weight–vectors in an n–dimensional Quadtree. Following this approach the weight–vectors are ordered according to their positions in hierarchically refined hypercubes, where each cube contains up to 2^n cubes of half of its edge lengths. This subdivision is done until a fixed number of vectors is inside the cube. In this way the depth of the subdivision is controlled by the density of the vectors in the feature–space.

The regular box oriented structure allows a reduced calculation of distances by a privileged search for neighbors in related cubes. Additionally the methods described in Sections 3.1 and 3.4 can be used for a further reduction of time consuming calculations.

As shown in Section 4 the usability of N–Trees for next neighbor search depends on the number of output neurons and the dimension of the input data. High dimensional input data accompanied by only a small number of output neurons causes a complex and sparse internal structure which results in low performance. The reverse case of lower data dimension and a large number of vectors shows the advantage of N–Trees based search in comparison to methods described in previous Sections.

4 APPLICATION AND RESULTS

4.1 General Remarks

The methods outlined in Section 3 were tested on two different applications where both deal with multidimensional image data. Both applications face an 8–class problem.

The medical application aims at segmentation of brain tumors in MRI–Data which is required in order to gain knowledge about localization and extension of the tumor in the skull. That knowledge can be used as input for 3D–renderers of 3D–surface reconstruction algorithms. Section 4.2 refers to a volume data set of 5 slices, where each slice is recorded as a two–channel image of size 256x256 pixels. The environmental application uses satellite image data, which was recorded from Landsat–TM. For environmental control the six–channel image data was analyzed with 1000x100 pixels in each channel. The classified image data outlines the actual land–use and illustrates the impact of pollution sources in the environment.

4.2 Measured Times of Calculation

The success of the different methods given in Section 3 widely depends on the data that the Kohonen–Map is used for. Furthermore it depends on the size of the feature–vector. In order to give an idea of the amount of possible optimization, the following table shows the measured times of calculation for different kinds of image–segmentations on a HP–Workstation 720 and a DEC–Workstation 5000/240.

Method applied \ Image data	Clustering 5 slices of 256x256 Pixels with a map of 54 input–neurons and a 6x6x6 output–layer on a HP 720	Classifying 5 slices of 256x256 Pixels with a map of 54 input–neurons and a 6x6x6 output–layer on a HP 720	Classifying a Picture of 1000x100pixels with a map of 54 input–neurons and a 6x6x6 output–layer on a DEC 5000/240	Classifying a Picture of 1000x100 pixels with a map of 6 input–neurons and a 9x12x18 outputlayer on a DEC 5000/240
Determination of all distances while avoiding the square–root–function (see 3.1)	**57.10** minutes	**57.10** minutes	**20.97** minutes	**26.22** minutes
All aspects from 3.1, Immediate Activation (see 3.2), Triangle Relation (see 3.3)	**4.32** minutes	**3.33** minutes	**11.15** minutes	**15.15** minutes
As above and additionally use of Relation (10) (see 3.4)	**4.11** minutes	**3.03** minutes	**5.72** minutes	**10.11** minutes
Maximum Likelihood–Search (see 3.5)	**4.39** minutes	**4.24** minutes		
2–pass determination with an approximating first pass (see 3.6)	**6.53** minutes	**4.00** minutes		
N–Tree–Search (see 3.7)			**31.88** minutes	**4.81** minutes

Fig. 4: Measured times according to different modifications and methods

5 CONCLUSION

We conclude that image segmentation based on Kohonen Feature Maps is an excellent tool to realize pixel–oriented analysis of images. Unfortunately the implementation of the straightforward algorithm leads to enormous computation times. In order to make the image analysis acceptable for applications optimizations of the algorithm are required. The proposed modifications fulfill this requirement since our results demonstrate that a reduction to approx. 5% of the standard implementation could be reached.

The time consuming part of the Kohonen Feature Map reduced by our modifications is the determination of the weight vector next neighbored to the input–vector. So the proposed optimizations realize a fast and effective next neighbor search which can be directly transposed to other applications in the field of classification and computational geometry.

The presented results demonstrate that image analysis can be computed in a acceptable time even without parallelization on special and expensive hardware. Nevertheless a subset of the aspects in this paper can be applied on a vector architecture.

6 ACKNOWLEDGMENTS

The authors would like to thank the research division of the German Telekom for the financial support for this work within the project KAMEDIN. Furthermore

many thanks to the GAF (Munich) and the Department of Biophysics and Medical Radiation Physics of the German Cancer Research Center (Heidelberg) for kindly providing the image data.

7 REFERENCES

[1] C. Busch, M. Gross: Interactive Neural Network Texture Analysis and Visualization for Surface Reconstruction in Medical Imaging. Computer Graphics forum, Vol.12, No.3,(EUROGRAPHICS'93), pp.C49–C60, (1993)

[2] M. Gross, R. Koch, L. Lippert, A. Dreger: Segmentierung und Klassifikation von Texturen mittels Wavelets und neuronalen Netzen, DAGM–Proceedings, to be published (1994)

[3] M. Gross, F. Seibert: Visualization of Multidimensional Data Sets using a Neural Network. The Visual Computer, Vol.10, No.3, pp.145–159, (1993)

[4] M. Gross, F. Seibert: Neural network image analysis for environmental protection. In Grützner (Edts.): Visualisierung von Umweltdaten 1991, GI, Berlin – Heidelberg – New York: Springer (1991)

[5] K.H. Höhne, M. Bomas, A. Pommert, M. Riemer, C. Schiers, U. Tiede, G. Wiebecke: 3D Visualization of Tomographic Volume Data using the Generalized Voxel Model, The Visual Computer, Vol.6, No.1, pp.28–36, (1990)

[6] R. Koch: Entwicklung eines 2D und 3D Texturanalysesystems basierend auf einer Merkmalextraktion mit Wavelets, Diplom–thesis, Computer Science Department, Technische Hochschule Darmstadt, (1994)

[7] T. Kohonen: The Self–Organizing Map. Proceedings of the IEEE, Vol. 78, No. 9, pp. 1464–1480, (1990)

[8] T. Kohonen: Self–Organization and Associative Memory, Berlin – Heidelberg – New York: Springer (1984)

[9] M. Levoy: Display of Surfaces from Volume Data, IEEE CG&A, Vol. 8, No. 5, pp. 29–37, (1988)

[10] W.E. Lorensen, H.E. Cline: Marching cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics, Vol.21, No.4, pp.163–169, (1987)

[11] G.M. Nielson: Visualization in the Scientific Discovery Loop, EUROGRAPHICS'93 tutorial notes, (1993)

[12] F. Preparata, M. Shamos: Computational Geometry. An Introduction, New York: Springer Publishing Company, (1985)

[13] F. Sauerbier, D. Scheppelmann, H.P. Meinzer: Segmentierung biologischer Objekte aus CT– und MR– Schnittserien ohne Vorwissen, DAGM–Proceedings, pp.289–293, Springer, (1989)

Medical Applications

mammogram is often very low (ii) features in mammograms indicative of breast disease are often very small [2].

The aim of this paper is to describe methods based on third order spectral estimation techniques with artificial neural networks, for modelling and segmentation of mammograms.

MATHEMATICAL REPRESENTATION OF THE MAMMOGRAM

We represent the mammogram with a 2-D random field $x[m,n]$, where $x[m,n]$ denotes the value of the random field at the point $[m,n]$ which is defined theoretically over the integers $-\infty < m,n < \infty$ in the 2-D plane. A typical finite extent for a realistic sequence is the measured data array, which usually has a region of support $0 \le m,n < N$, where N may be for example 256 or 512. For the purposes of spectral estimation and modelling we represent $x[m,n]$ as the output of a two dimensional linear shift invariant (LSI) system driven by white noise [3]:

$$x[m,n] = -\sum_i \sum_j a_{ij} x[m-i,n-j] + w[m,n], \ [i,j] \neq [0,0]$$

where a_{ij} with $a_{00} = 1$ are the parameters of the autoregressive model.

It is assumed that the noise $w[m,n]$ is non-Gaussian, zero mean, at least sixth order weakly stationary and white [6] that is to say,

$$r_{ww}[m,n] = \sigma_w^2 \delta[m,n]$$

where $r_{ww}[m,n]$ and σ_w^2 are the input autocorrelation and variance respectively and

$$C_{3w}([m_1,n_1],[m_2,n_2]) = \gamma_w \delta[m_1,n_1]\delta[m_2,n_2]$$

where $C_{3w}([m_1,n_1],[m_2,n_2])$ and $\gamma_w = E\{w^3[m,n]\} \neq 0$ denote the third order input cumulant and skewness respectively; and $\delta[m,n]$ stands for the two dimensional Kronecker delta. In general the random field is represented as:

$$x[m,n] = \sum_i \sum_j h[m-i,n-j] w[i,j]$$

where $h[m,n]$ is the impulse response of the linear shift requirement. As indicated already the range of summation has been purposely left unspecified. The system function for the stable 2-D AR model of impulse response $h[m,n]$ is given by:

$$H(z_1,z_2) = \frac{1}{\sum_i \sum_j a_{ij} z_1^{-i} z_2^{-j}} \text{ where } \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} |h[m,n]| < \infty.$$

Stability in two dimensions is far more difficult to test than stability in one dimension, because 2-D polynomials in general cannot be factored, due to a lack of a fundamental theorem of algebra in 2-D. For the purpose of

causality the region of support used here is confined to the quarter plane (QP). Then the two dimensional autoregressive model is given by the difference equation:

$$\sum_{i=0}^{p_1}\sum_{i=0}^{p_2} a_{ij} x[m-i,n-j] = w[m,n]$$

where $a_{00} = 1$. The values of p_1 and p_2 define the order of the model. If the region of support for the AR parameters is the quarter plane, the output may be recursively computed as a function of the "past" outputs and the "past" and "present" inputs [3],[4].

Because we know nothing about the location and the size of the tumour in the mammogram, our method consists of representing each pixel in the image by an autoregressive model whose parameters are estimated by using an appropriate neighbourhood for the pixel. This is in effect a small compared to the whole image quadrangular window around the appropriate pixel. Then the parameters of the model are used as features for classification and segmentation. We make the assumption, that all pixels in the small window belong to the same class, because image pixels which are spatially close are likely to be of the same texture. After estimating the AR parameters for the pixel $[m,n]$ we replace the value $x[m,n]$ of that pixel with the vector $\underline{\mathbf{a}}[m,n]$, which contains its corresponding parameters. This repeated for the entire image, and thus we create a set of P new images, where $\mathrm{P} = \left[(p_1+1)(p_2+1)-1\right]$ is the number of the AR parameters for the particular model. The results of the segmentation will be influenced significantly by the size of the window over which AR parameters are extracted.

AR PARAMETER ESTIMATION USING AUTOCORRELATIONS

Let the region of support for the AR parameters for the purposes of this paper, be the truncated quarter plane (TQP). The order of the model is $p_1 \times p_2$, and hence the two dimensional field is:

$$\sum_{i=0}^{p_1}\sum_{j=0}^{p_2} a_{ij} \mathrm{y}[m-i,n-j] = w[m,n], \qquad a_{00} = 1$$

The extended Yule-Walker equations are given by[5]:

$$\sum_{i=0}^{p_1}\sum_{j=0}^{p_2} a_{ij} r_{yy}[i-k,j-l] = \begin{cases} \sigma_w^2 + \sigma_v^2 & k = l = 0 \\ \sigma_v^2 \cdot a_{ij} & k \in [0,p_1] \cap l \in [0,p_2] - \{[k,l] = [0,0]\} \\ 0 & \text{elsewhere} \end{cases}$$

The variance of the Gaussian noise σ_v^2 is usually unknown and hence the above equations cannot be solved directly. However, for high signal to noise ratios, σ_v^2 (which is the power of noise) is small compared to the power of

the signal $x[m,n]$ and hence it can be ignored, to produce the noiseless Yule-Walker equations. These equations give a good AR parameter estimation in high SNR's. However, the error increases significantly for large σ_v^2. The situation is even worse when the Gaussian noise $v[m,n]$ is coloured as the correlation properties of $v[m,n]$ are now needed.

AR PARAMETER ESTIMATION USING HIGHER ORDER STATISTICS

Higher order spectra defined in terms of higher order moments of the process contain information regarding the deviation of the process from a Gaussian form. It is known that only for zero mean Gaussian processes only all polyspectra of order greater than two are identically zero [6]. Thus a non zero higher order spectrum indicates deviation from normality. For a given zero mean stationary real random process (X_{mn}), non zero skewness $E\{X_{mn}^3\} \neq 0$ indicates the existence of its bispectrum. Hence, in an environment where the derived signal is a non-Gaussian stationary process and the additive noise process is stationary Gaussian there are certain advantages in estimating signal parameters through third order spectrum techniques. To date in the open literature almost all random field models and their associated processing procedures have been based on the assumption that the signal is corrupted by Gaussian noise (white or coloured). Second order techniques usually require knowledge about spatial correlations of the Gaussian noise which are unknown, while third order techniques have the advantage that they are blind to such noise.

We suppose that the signal $x[m,n]$ is corrupted by white Gaussian noise $v[m,n]$ and hence in practice we observe the noisy signal:

$$y[m,n] = x[m,n] + v[m,n]$$

The equations that relate the AR parameters with the cumulant function samples of the signal $y[m,n]$, have the following form [6],[7]:

$$\sum_{i=0}^{p_1}\sum_{j=0}^{p_2} a_{ij} C_{3y}([i-k_1, j-l_1],[i-k_2, j-l_2]) = \begin{cases} \gamma_w & k_1 = l_1 = k_2 = l_2 = 0 \\ 0 & \text{elsewhere} \end{cases}$$

where $\gamma_w = E\{w^3[m,n]\}$, $a_{00} = 1$ and $k_i, l_i \geq 0$, $i = 1,2$. Thus if we use the above equations, it is not necessary to know the statistical properties of the Gaussian noise, as they disappear from the equation. In the above equation we need a total of $(p_1+1)\cdot(p_2+1)$, equations in order to determine the unknown parameters a_{ij} and the skewness of the driving noise γ_w. However if we are not interested in estimating γ_w, we rewrite the above systems as follows:

$$\sum_{i=0}^{p_1}\sum_{j=0}^{p_2} a_{ij} C_{3y}([i-k_1, j-l_1],[i-k_2, j-l_2]) = 0\ ,\ k_1+l_1+k_2+l_2 \neq 0$$

In this form we need $[(p_1+1)\cdot(p_2+1)-1]$ equations to determine the a_{ij} parameters.

Comments on the Choice of Slices

An implicit and additional degree of freedom is connected with the specific direction chosen for the cumulants to be used in the AR model. Such a direction is referred to as a slice in the cumulant plane we have found that it has profound implication on the effectiveness of the AR modelling.

Let us consider the particular case: $(k_2, l_2) = (k_1 + c_1, l_1 + c_2)$ where c_1, c_2 are constants, then

$$C_{3y}([i-k_1, j-l_1],[i-k_2, j-l_2]) = C_{3y}([i-k_1, j-l_1],[i-k_1-c_1, j-l_1-c_2]).$$

From the symmetry properties of cumulants we know that: $C_{3y}([k_1, l_1],[k_2, l_2]) = C_{3y}([-k_2, -l_2],[[k_1, l_1]-[k_2, l_2]])$ so

$$C_{3y}([i-k_1, j-l_1],[i-k_2, j-l_2]) = C_{3y}([c_1+k_1-i, c_2+l_1-j],[c_1, c_2]).$$

We write $c_1 + k_1 = k$ and $c_2 + l_1 = l$ and hence the equations above take the form:

$$\sum_{i=0}^{p_1}\sum_{j=0}^{p_2} a_{ij} C_{3y}([k-i, l-j],[c_1, c_2]) = 0,\ k_1+l_1+k_2+l_2 \neq 0.$$

In this form they are shown to be explicitly dependent on the choice of c_1 and c_2.

Remarks

The choice of slices affects significantly the estimation of AR parameters. In this paper we choose $c_1 = c_2 = 0$, so

$$\sum_{i=0}^{p_1}\sum_{j=0}^{p_2} a_{ij} C_{3y}([k-i, l-j],[0,0]) = 0,\ k \in [0, p_1] \cap l \in [0, p_2] - (k, l) = (0,0)$$

In other work [10] we have concentrated in the choise of slices to provide additional information.

NEURAL NETWORKS

For the purpose of classification we employ the (a_{ij}) parameters as inputs to a multilayer perceptron. The examples contained in this paper are based on a ***three layer perceptron*** neural network which is employed with the AR parameters as the inputs and the different classes as the outputs. The network in the example below is a $K(8,10,2)$ Kuratowski graph trained by

the Charalambous approach [8]. This approach relies on conjugate gradient techniques for the training of the multilayer perceptrons. The conjugate gradient methods belong to a class of unconstrained optimisation algorithms that automatically adjust their parameters to meet local optimisation objectives. With a fairly accurate "line search" algorithm that form part of the procedure such methods are guaranteed to converge to a minimum with a fast convergence rate [8].

RESULTS

In this work we use an autoregressive model of order 2×2, that is to say $p_1 = p_2 = 2$. As we mentioned before, the segmentation results will be influenced greatly by the size of the window over which image features (in this case AR parameters) will be extracted. In general, a large window leads to a good segmentation in the inner regions of each class. Whereas in the regions around boundaries of classes the window covers two or more different classes and the features extracted are the mixed features of these classes, so the parts of the image which are around boundaries cannot be correctly segmented. Therefore, the window in inner regions should be large enough to separate different classes and it should become smaller as it nears to the boundaries. We use first a window of size 32×32 to segment the mammogram. Then we take another window around each pixel which must be relatively smaller than the first one and we choose that to be of size 16×16. We *define* a pixel ***misclassified*** if the pixels in the new window around it do not belong to the same class according to the first classification. For all misclassified pixels we estimate again the AR parameters but using now the 16×16 window and we do the same process for image classification and segmentation. Results shown in figures 1-10 below.

REFERENCES

[1] "Breast Cancer Screening", Report to the Health Ministers of England, Wales ,Scotland & Northern Ireland by a working group chaired by Prof. Sir Patrick Forrest.

[2] W.M. Morrow and R.B. Paranjape, "Region-based contrast enhancement of mammograms", IEEE Trans. on Medical Imaging, *Vol.11, No 3, September 1992.*

[3] Steven Kay, "Modern Spectral Estimation: Theory and Application", Prentice Hall 1988.

[4] Stephen P. Banks, "Signal Processing,Image Processing and Pattern Recognition", Prentice Hall 1990.

[5] Bart Kosko, "Neural Networks for Signal Processing", Prentice Hall 1992.

[6] C.L. Nikias and M. Raghuveer, "Bispectrum Estimation: A digital signal processing framework", IEEE Trans. on ASSP ,vol. 75, pp. 869-891, 1987.

[7] A. Swami and J.M. Mendel, "ARMA parameter estimation using only output cumulants", IEEE Trans. on ASSP, vol. 38, no. 7, July 1990.

[8] C.Charalambous, "A conjugate gradient algorithm for the efficient training of artificial neural networks", CRST Technical Report 90/06 May 1990.

[9] T. Stathaki and A.G. Constantinides, "Higher order spectral estimation techniques in mammography", presented on the IEEE Int. Conf. on DSP, July 1993, Nicosia, Cyprus.

FIGURES

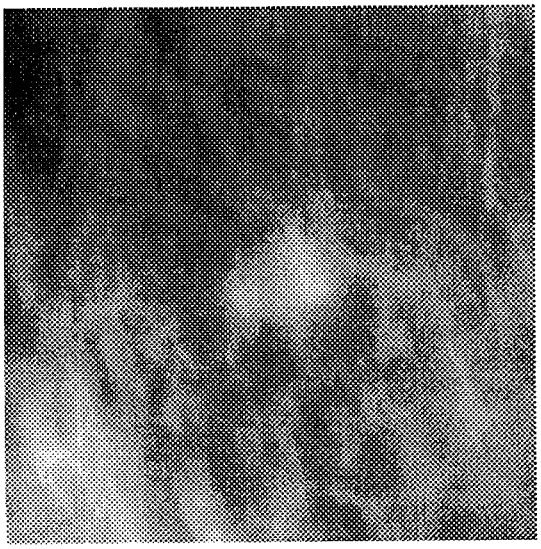


Fig. 1. mammogram

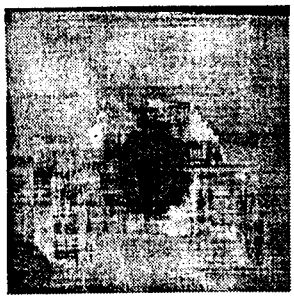


Fig. 2: a_{01}

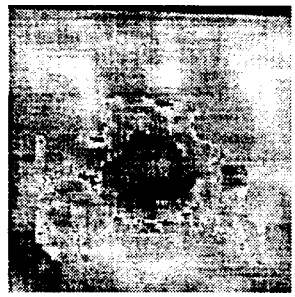


Fig. 3: a_{02}

Fig. 4: a_{10}

Fig. 5: a_{11}

Fig. 6: a_{12}

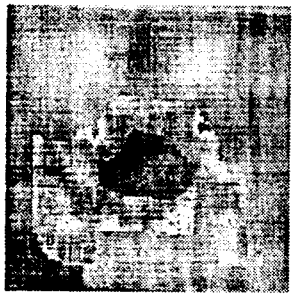


Fig. 7: a_{20}

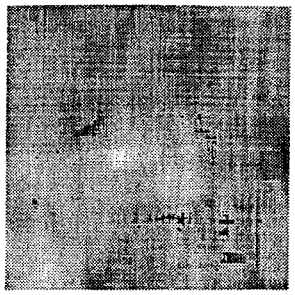


Fig. 8: a_{21}

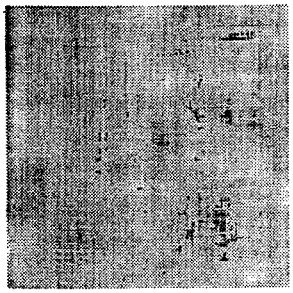


Fig. 9: a_{22}

As we mentioned above a three layer perceptron neural network is employed with the eight AR parameters as the inputs and the two texture classes (tumour and healthy breast tissue) as the outputs. The segmented mammogram is shown in the following figure.

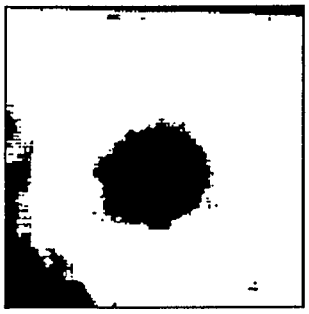


Fig. 10: segmented image

Medical Diagnosis and Artificial Neural Networks: A Medical Expert System applied to Pulmonary Diseases

G. - P. K. Economou*, C. Spiropoulos**, N. M. Economopoulos*, N. Charokopos**, D. Lymberopoulos*, M. Spiliopoulou**, E. Haralambopulu**, and C. E. Goutis*

* Department of Electrical Engineering
** Pulmonary Department, Regional University Hospital of Patras
University of Patras, GR 261 10, Patras Greece

Abstract-An original Medical Expert System (MES) in the field of Pulmonary Diseases (PDs), is the topic of this article. This MES covers the full spectrum of PDs, being the first that attempts to treat a whole category of distressed body organs. It is based on a first presented composition of powerful Artificial Neural Networks (ANNs) that have been taught by means of real patients' clinical data,. The proposed MES exhibits an overall performance of at least 85% in its generalization results.

Keywords-Medical Expert Systems, Artificial Neural Network, Pulmonary Diseases.

INTRODUCTION

A great deal of Research and Development activities have recently highlighted into building and evaluating systems that could decide based on human thinking concepts and expertise: Expert Systems (ES - [6], [8]). Medical ES aim to supply for tools to assist doctors of medicine (MD), guide trainee students and encourage medical experts in their diagnoses ([2]). In addition, MDs who serve distantly from Medical Centres, can utilize MESs in order to judge more accurately upon a particular, not familiar disease. An MES is a powerful tool of induction best tuned when used in conjunction to a human, rather than as a stand-alone authority.

This article proposes the new and creative MES that was developed by a team of medical and technical experts in the University of Patras. Although it has focused on PDs, PDs dealt with as a whole category, it is structured in such a way to easily being adapted to generalize in other domains of experience, too. Real-world clinical data were used to instruct its layers and preliminary and more detailed experiments showed its great capabilities of making correct classification of symptoms and PDs (approximately 85% out of 150 possible new PDs cases).

A mighty composition of ANNs, a prominent section of Artificial Intelligence (AI), is the core of the presented MES. AI techniques were most recently used either to accomplish the inference engine or as the means to implement both a knowledge base and the induction rule of ES ([3], [5], [14]). The expansive utilization of MESs in hospitals world-wide, has begun to show their competence, whereas new methodologies are altogether posed to give the necessary evaluation criteria to compare AI-based MES to more algorithmic-based methods ([3], [4]). However, Artificial Neural Network (ANN) architectures, seem to he a head off.

The implementation of ANNs towards the formation of MESs, currently experiences a vigorous growth ([1], [3], [5], [10], [11], [12]). Valuable assets AI and medical experts seek in MESs, can be found in ANNs as a part of their very structure. Parallel searching, dynamic data storage, robustness, generalization virtues and the amazing working speed factor are inherent abilities of ANNs. ANNs may be implemented purely in software, on general-purpose platforms or on microprocessors, or be made of more or less custom hardware and VLSI chips.

Besides, ANNs vast application domain and general tasks' accomplishment, furnish a sound ground of exploit. To enhance this end, a large number of ANNs were structured and simulated on an 386SX@33 PC and were taught by means of a variety of learning algorithms to favour the best-suit ones. The most known and the most severe Pulmonary Diseases' (PDs) symptoms were integrated into their structure so that the proposed environment is able to deal with the big majority of them.

ORGANIZATION OF MEDICAL DATA

The building of knowledge-based environments for assisting MDs on diseases' diagnosing, is a complex task due to the particular importance of all the medical data and the interpretation that different doctors give to them. Thus, the construction of an MES based on ANNs, have been forwarded to provide for the categorization and generalization of the medical data into new patients' cases. Furthermore, this MES follows step by step the Clinical Differential Diagnosis (CDD) methodology, due to the nature ANNs treat expertise. A mapping of patient's symptoms exhibition to the classes of possible PDs, is therefore achieved.

Clinical experts in PDs, established the boundaries of the project. A definite number of inputs were set, the same questions that MDs ask when examining patients. They contain related findings of each one of the PDs' symptoms, i.e. Cough, Sputum, Haemoptysis, Fever, Dyspnea, Wheezing and Chest Pain and historical as well as data obtained from physical examinations. Consequently, those data, were fed to a large number of ANNs ([7], [9], [13], [15]) and evenly distributed to both a sum of thirty-five (35) PDs and they related twelve (12) major PDs' classes (Table I).

Data were fed by introducing their existence or non-existence in possible PDs' exhibition. Major influences, such as the gravity of certain symptoms or findings to determine certain PDs, multiple PDs' interference in a diagnosis and resulted PDs' ordering on a higher-fitness basis, were left to the ANNs to learn. Still, lethal PDs a patient could suffer, were made certain to the highest degree to be excluded or confirmed by the proposed MES, through using suitable input patterns.

TABLE I: CUMULATIVE FINDINGS FOR THE "COUGH" SYMPTOM

	Val	Rcn	Chr	Prd	Non Prd	Prx	Exr	Day	Mrn	Evg	Ssn	Anx	Swt	Wg-	Wg+	Vmt	Slp
C.O.P.D.	****	*	*	*			*	*	*			*	*	*	*		*
Tuberculosis	****	*	*		*	*		*				**	**	*		*	
Interstitial PDs	****	*	*		*	*		*			*						
Abnormalities of the Diaphragm																	
Cancer of the Lungs	****	***	***	*	*	*		*				*		*			
Disorders of the Mediastinum	**	*			*	*		*	*	*		*		*			
Infection Diseases of the Lungs	****	*		*	*	*		*				*					
Disorders of Pleura	*	*			*			*		*		*		*			
Bronchial Asthma	**	*	*	*	*	*	***	*		*	***						
Disorders of the Pulmonary Circulation	*	*		*	*	*		*				*					
Occupational Disorders of the Lungs	***		*	*	*	***		*	***	*							
Non PDs																	

Table I depicts the cumulative patterns with which the ANN dealing with the **Cough**'s symptom was taught. Column 1, denotes PDs' *Classes* (including a Non-PDs one); Column 2, the particular *Value* that this symptom has to a specific PD class; next Columns, whether the Cough's findings could be *Recent*, *Chronical*, *Productive*, *Non-Productive*, or/and *Paroxismic*, after *Exercision*, all *Day* long, only in the *Morning* or/and in the *Evening*, *Seasonary* , followed by *Anorexia*, excessive *Sweating*, *Weight Loss* or *Increase*, *Vomiting* or/and *Sleepiness*.

PROPOSED COMPOSITION OF ANNS

Conducted preliminary experiments proved the feed-forward ANN, to be the most bright one and was elected as the basis of this MES suggested composition. This MES is arranged in a three layers form, following data's time propagation sequence (Figure 1). Two different learning algorithms were forwarded to teach its ANNs: back-propagation ([9]) and Kalman filtering of back-propagation equations ([13]). The latter, however, utterly swayed; it performed better with less initialization steps and better learning speed, accuracy, convergence and data handling. Typical learning parameters include ANNs with binary inputs, float arithmetic processing, 30-44 nodes, 250-300 input patterns, 3 slabs each and a requested learning accuracy of 5%. Learning times spread between 20'-30', i.e. 2000-3000 learning cycles.

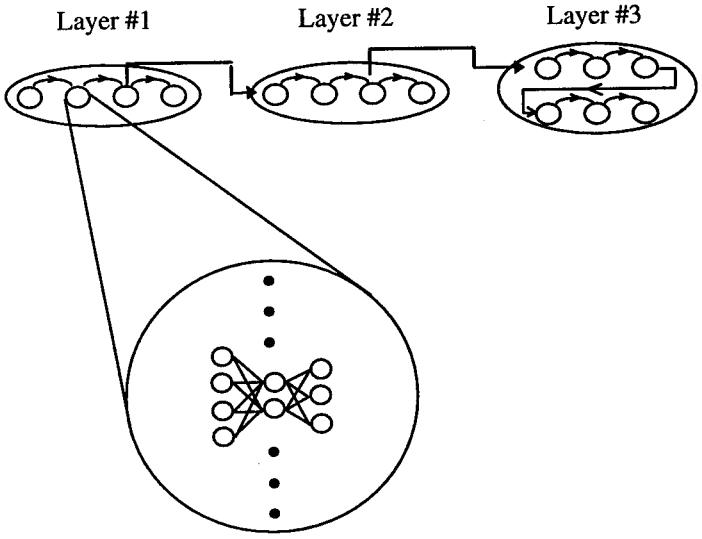


Figure 1. Layers: Composition of the ANNs

First Layer

A four levels ANNs' formation, three-slab structured, was used. Inputs to the first level was fed separately for each one of the major symptoms (subjective medical data) and in random order. Moreover, two other identical-structured ANNs were added, to treat historical and physical exams data (objective medical data). The outputs of each of those ANNs, are the general classes of the possible PDs expressed as percentages of similarity to their learnt patterns. Figure 2 depicts first Layer.

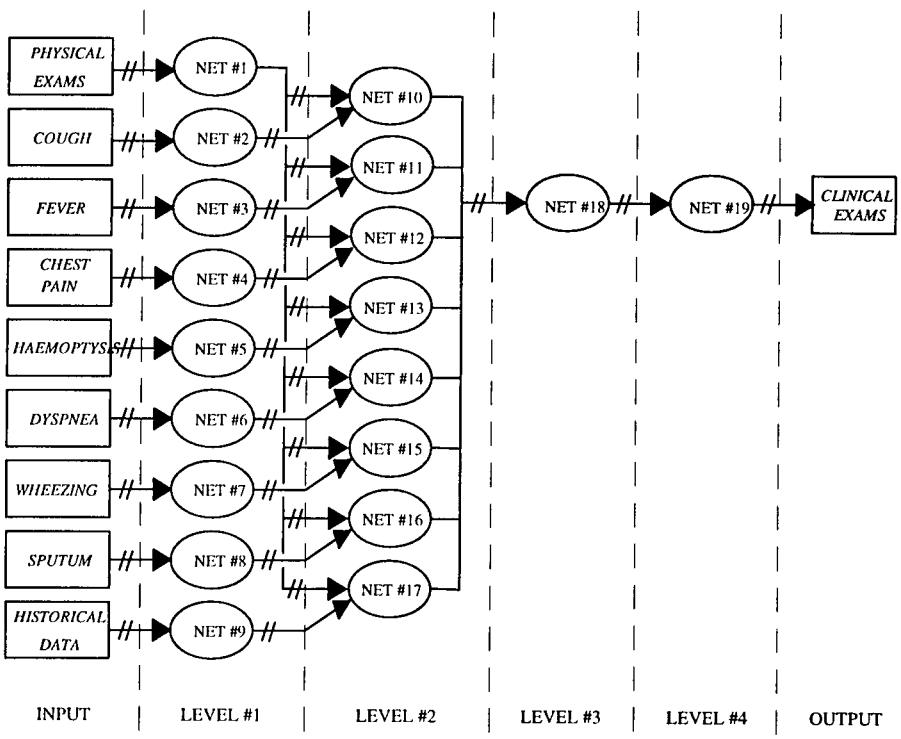


Figure 2. Layer #1: Structuring

These outputs are then weighed by eight three-slab ANNs in the second level of the first layer, thus forming pairs of symptoms or historical data, and physical examinations handling, as CDD methodology imposes. Again, outputs will be the classes of PDs with a percentage of fitness to the stored weights in the ANNs. On the third level, all the outputs of the second one are inserted in another three-slab ANN which outputs PDs' classes, too, and combines all the available information.

Up to this point, all medical data given are pondered independently and only finally summed-up, following step by step the CDD methodology. Hence, the end results show the general tendency of the possible PDs a patient could suffer of. In the fourth level, though, suggested clinical exams outcome by means of a two-slab ANN.

This induction methodology scheme is so far submitted, due to its transparency to the intermediate results. An expert is able at every level of process to intervene and select the most crucial diagnosis' facets, according to his own opinion. The ANNs, altogether, do not inhibit but, instead, offer percentages of possible PDs existence, letting the expert not only to make the final decision but also to combine his own selected symptoms' responses. However, this MES can be let to prune out not possible PDs, too, following already instructed induction threads.

Second Layer

This scheme given, the use of another four-levels, three-slabs ANNs' formation was advanced. These ANNs are identical to and operate exactly as in the first layer, but they handle PDs in all their outputs, intermediate and final. In addition, some new inputs are added: those that relate the final PDs' percentages of fitness the third level of the first layer have already computed. This way, a strong positive feedback will be exercised in the second layer's ANNs to enhance and promote the final diagnosis. Preliminary and more elaborate results have shown a great increase in the final (layer #3) outputs' accuracy than ever (15%-25%).

Third Layer:

Two four-levels ANNs' formations are also planed. These will handle all data the former ones did, plus clinical examinations' results and new data from the PDs' eventual progress. Findings of those data as being fed to both ANNs handling patients' symptoms before the clinical examinations as well as after those, will contribute to the final PD diagnose. A voting process between the two is arranged, to be performed by another ANN. The final MES's output, however, is scheduled to be the necessary medication potions as well as their dosage and time-schedule.

Discussion

As the reader may note, this novel architecture is being built in such a way that assures friendliness of utilization, transparency on all its levels and efficacy on its results. As for the latter quality, medical data and clinical experts' interaction help to make the system achieve a good performance. So far results have shown an overall performance of nearly 85% successfully promoting the correct PD as the already taught ANNs were left to generalize into their newly fed inputs (150 total PDs' cases).

The special value of the proposed MES, is that the percentage of exactness achieved does not imply that in the 15% of the cases left, the PDs are classified incorrectly out of patients' symptoms. In this presented MES, the correct PD should be one out of the next five (and only five!) less fitted results (in descendant order).

Moreover, since the first priority of MDs, when examining patients, focuses on being able to suggest the correct clinical examinations, this MES performs accordingly. It prunes these examinations and waits to consider their results for the final diagnoses outcome, too. Therefore, a safety margin is thus structured , such that will surely enhance the overall precision of the system, at least to MDs' levels.

Additionally, the proposed architecture for the MES composition, can very easily be retargeted to fit in other domains of interest. Already the aforementioned team is working towards the expansion of this MES to other fields of Medicine. MDs could offer their judgement about patients' cases considering the whole perspective of the human body. Of course, the problem of associating the results of all the intermediate MESs, will be posed and will remain to be solved. Still, applications can be found on every terrain treated by human expertise and not only Medicine.

CONCLUSIONS

The structuring of an efficient MES to assist MDs was the target of this research. The precursory along to the ultimate results are very stimulating. However, the intensifying of this MES through the augmentation of its data, is a necessary next step. It passes through the integration of medical theoretic knowledge and the interference with other pulmonary teams' knowledge as well as to the learning by new algorithms and additional data. The presentation of a general-purpose MES to be the basis of other medical diseases induction diagnosis is the limit.

REFERENCES

[1] D. G. Bounds, P. J. Lloyd, B. Matthew, and G. Waddell, "A Multi Layer Perceptron Network for the Diagnosis of Low Back Pain", in Proc. of the Int. Conf. on Neural Networks, San Diego, CA, vol 2, pp. 481-489, 1988.

[2] A. P. Dhawan, "An Expert System for the Early Detection of Melanoma Using Knowledge-Based Image Analysis", Anal., Quant. Cyt. and Hist., vol 10, no 6, 1988.

[3] A. Durg, W. V. Stoecker, J. P. Cookson, S. E. Umbaugh, and R. H. Moss, "Identification of Variegating Coloring in Skin Tumors: Neural Network vs. Rule-Based Induction Methods", IEEE Eng. in Med. and Biol., vol 12, no 3, pp. 71-74 & 98, 1993.

[4] A. Hart and J. Wyatt, "Evaluating Black-Boxes as Medical Decision Aids: Issues Arising from a Study of Neural Networks", Med. Inf., vol 15, no 3, pp. 229-236, 1990.

[5] K. Henson-Mack, and H. - C. Chen, "Integrating Probabilistic and Rule-Based Systems for Clinical Differential Diagnosis", IEEE Proc. SOUTHEASTCON '92, Birmingham, AL, USA, pp. 699-702, 1992.

[6] W. C. House, "Decision Support Systems: A Data-Based, Model-Oriented, User-Development Discipline", Petrocelli Books Inc., Mc Graw Hill, 1991.

[7] D. R. Hush, and B. G. Horne, "Progress in Supervised Neural Networks", IEEE Signal Processing Magazine, vol 10, no 1, pp. 8-39, 1993.

[8] R. J. K. Jacob, J. N. Froscher, "A Software Engineering Methodology for Rule-Based Systems", IEEE Trans. on Knowledge and Data Engineering, vol 2, no 2, 1990.

[9] R. P. Lippmann, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp. 4-22, 1987.

[10] B. H. Mulsant, "A Neural Network as an Approach to Clinical Diagnosis", M. D. Computing, vol 7, no 1, pp. 25-36, 1990.

[11] T. J. O' Leary, U. V. Mikel, and R. L. Becker, "Computer-Assisted Image Interpretation: Use of a Neural Network to Differentiate Tubular Carcinoma from Sclerosing Adenosis", Modern Pathology, vol 5, no 4, pp. 402-405, 1992.

[12] R. Poli, S. Cagnoni, R. Livi, G. Coppini, and G. Valli, "An NN Expert System for Diagnosing and Treating Hypertension", IEEE Comp., vol 24, no 3, pp. 64-71, 1991.

[13] R. S. Scalero, and N. Tepedelenlioglu, "A Fast New Algorithm for Training Feedforward NN", IEEE Trans. on Sig. Proc., vol 40, no 1, pp. 202-210, 1992.

[14] S. E. Unbaugh, R. H. Moss, and W. V. Stoecker, "Applying Artificial Intelligence to the Identification of Variegated Coloring in Skin Tumors", IEEE Engineering in Medicine and Biology, vol 12, no 1, 1991.

[15] B. Widrow, and M. A. Lehr, "30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", Proc. of the IEEE, vol 78, pp. 1415-1442, 1990.

MODELING OF GLAUCOMA INDUCED CHANGES IN THE RETINA AND NEURAL NET ASSISTED DIAGNOSIS

Simon von Spreckelsen
Hvidovre Hospital
Kettegårds Alle 30
DK-2650 Hvidovre, Denmark
and
Peter Grumstup, Jon Johnsen, and Lars Kai Hansen*
CONNECT, Electronics Institute, build. 349
Technical University of Denmark, DK-2800 Lyngby, Denmark
*Request for reprints: lkhansen@eileen.ei.dth.dk

Abstract. A system for modeling and early detection of Glaucoma induced chances in the human retina is described. The system includes a modeling tool for design of semi-realistic retinal pictures, that may either be used for educational purposes or (as here) as a laboratory for controlled signal processing experiments. The detection system includes preprocessing algorithms for elimination of intensity variations and other artefacts. The final segmentation step is based on a cellular neural network.

INTRODUCTION

Vision is the most important of the human senses. The human eye can be considered a spherical structure with a radius of about $12mm$. The inside is filled up by a transparent substance *corpus vitreum*. The *retina* covers the inside surface and holds the optically sensitive nerve ends. Furthermore the retina holds nerve fibers collecting the optical signals. These fibers leave the eye through the optical disc (the so-called blind spot). If the internal pressure of the eye increases or if the blood supply to the optic disc or retina is decreased nerve fibers can degenerate and defects in the visual field appear. Since the human visual system can partly compensate for the lack of visual field, the increased pressure may go undetected beyond the limit where the induced changes are reversible. The pathological changes are collectively called *Primary Open Angled Glaucoma* (POAG). Early diagnosis is essential for preserving good visual function. If untreated POAG will lead to blindness within five to ten years. The diagnosis is quite common, in Denmark involving

0-7803-2026-3/94 $4.00 © 1994 IEEE

about 2% of the population, mostly elderly [10]. About 5% of the diagnosed eventually end up blind.

In this project the objective is to develop tools for early diagnosis of POAG based on so-called *fundus* pictures, i.e., images of the retina taken though the pupilla with a dedicated camera system. First, a model has been developed for modeling of the changes in the retina induced by Glaucoma. This model may be used for training of physicians, and as in this study, a laboratory for controlled experiments with pattern recognition devices. Secondly, we have developed a preprocessing scheme and cellular neural networks for detection of the nerve fiber pattern as seen in fundus pictures, for more details see [10, 7].

An image analysis study of Glaucoma detection was carried out by Yamazaki et al. [11]. In this investigation diagnosis is based on a analysis of a single intensity profile approximately orthogonal to the line connecting the optical disc and the fixation point. In our experiment this scheme has shown not to be robust to the variations in image quality of standard fundus pictures.

MODELING OF FUNDUS PICTURES AND GLAUCOMA

In order to create a workbench for comparison of various computer algorithms for enhancement and diagnosis, we have created a simple parametric model of fundus pictures. The model is based on analysis of original fundus pictures c.f. figure 1. The prominent features of the scene are *blood vessels* here seen as dark structures emanating from the periphery of the optical disc. The black spot is *Macula* (the fixation point), where the optical sensor density is maximal. Close visual inspection reveals that the nerve fibers form a multi-layered quite noisy line-pattern texture. For a discussion of textures see e.g. [6]. A close-up is shown in the right panel of figure 1. Glaucoma induced nerve fiber loss takes two different forms: a uniform reduction of fiber density or a characteristic regional (wedge) disappearance of fibers. An example of the latter is seen in figure 2.

ARTIFICIAL FUNDUS PICTURES

The image model consist of two modules that add features to a matrix with predefined optical disc and fixation point locations: *Blood vessel design* is a mouse based drawing module in which a semi-realistic design can be created with varying vessel diameter and vessel edge smooting. The *nerve fiber generator* draws line patterns of parametric second order curves. The parameters can be defined separately in 36 sectors surrounding the optical disc. The orientation of these sectors is defined by the optical disc and the fixation point. Noise is added with controlable signal to noise ratio to provide a realistic local appearence of the texture.

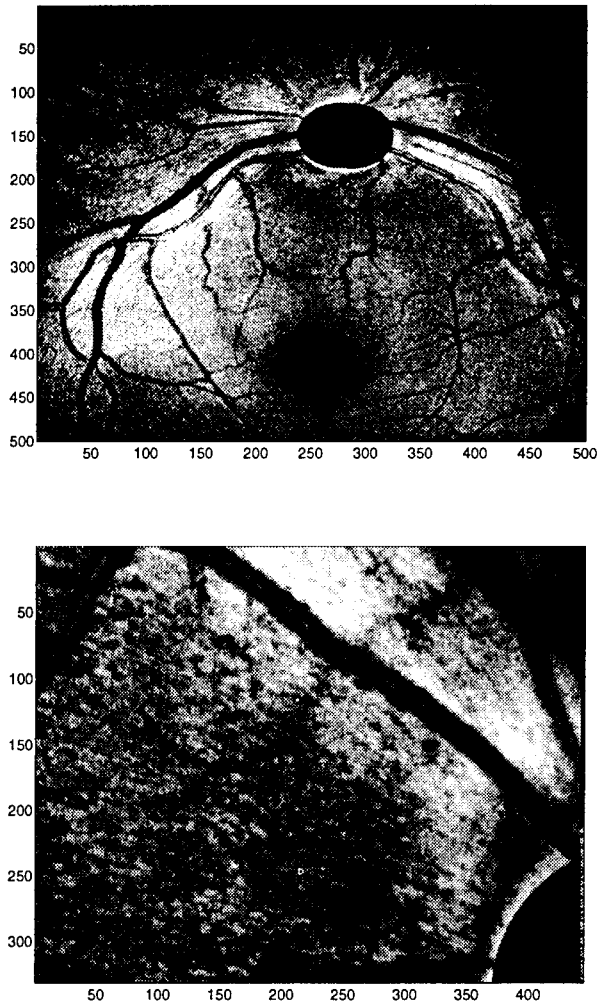


Figure 1: A socalled *fundus* picture showing the retina. Note the blood vessels emanating from the blind spot and *Macula*, the fixation point, where optical sensitivity is maximal. In the lower panel we show a close-up with the characteristic texture due to nerve fibers.

To emulate the multi-layered structure found in real fundus pictures several families of curves can be added. The curve parameters are slightly modified from one family to the next. In the images generated for the present study we have used four such layers. Two layers were added before addition of the blood vessel system and two layers added after, reflecting the layering in the real photograph.

PREPROCESSING

The first steps in the preprocessing of a fundus picture concern elimination of illumination artifacts and of the blood vessel system. These steps involve band-pass filtering and simple thresholding to segment the (dark) blood vessels. The location of the blood vessel system is kept in a separate mask.

TEXTURE PREPROCESSING

Texture detection algorithms are legio. Many such algorithms are based on a two-level design in which a basic filter detect local features followed by a merging algorithm that forms a global segmentation in regions of textures see e.g. [6] for a recent example. We follow a similar approach here. Our preprocessor consist of simple adaptive, quadratic, local discriminant trained by example. The segmentation step is carried out by a cellular neural network based on mean field annealing.

QUADRATIC DISCRIMINANT

The preprocessor classifies individual pixels based on the statistics of its neighborhood (a square window of MxM pixels). A simple maximum likelihood discriminator was found sufficient. The two populations of MxM windows (nerve fiber texture presence/absence) are modeled as two Gaussian distributions with individual means and covariance structures. The mean and covariances are estimated from a sub-image of $350x350$ pixels generated by the Glaucoma model. Since the line textures show the characteristic directions of flow c.f. figure 2 individual populations are trained for eight regions determined by the location of the optical disc and the fixation point. To illustrate the performance of the quadratic discriminant we show in the left panel of figure 2 a synthetic image with a characteristic regional defect (note also that the blood vessels have been eliminated leaving areas of uniform grey value). In the right panel of figure 2 we show the output (difference in log likelihoods) provided by the discriminant (window size $M = 13$). To segment the noisy result of the preprocessor into regions that quantifies the presence of nerve fiber a global segmentation algorithm is needed.

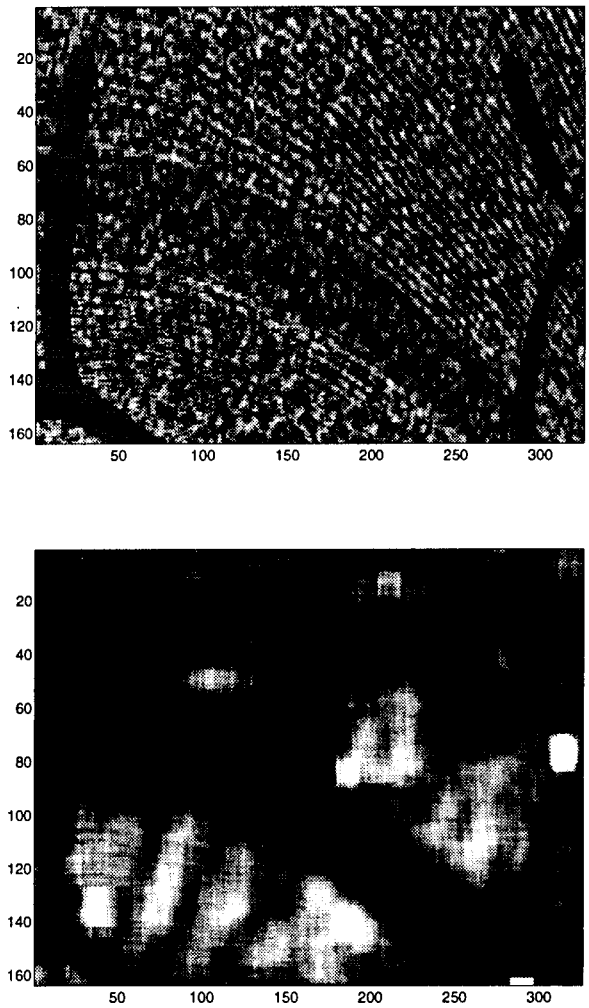


Figure 2: Upper panel: Synthetic fundus picture with a characteristic wedge regional absence of nerve fibers. Lower panel: Output of likelihood based discriminant.

CELLULAR NETWORK DESIGN

The cellular network concept was developed by Chua and Yang [2]. A cellular neural net is a locally connected network of simple processing units, typically operating as feed-back processes converging to a fixed point. The Bayesian or *Maximum Posterior* approach is a very successful device for signal processing (see [9] for a review), a particular attraction is that it leads to algorithms that map well onto cellular neural networks. The basic idea of the Bayes approach is to consider both the source (un-degraded) signal and the degradation as stochastic processes. Bayes' formula can then be used to construct the distribution of the reconstructed signal, conditioned on the observed degraded signal. Segmentation is an important step in many computer vision systems. Here we use the Bayes scheme to derive a simple cost-function that can be minimized by a cellular neural network. The resulting cost-function is identical to the one used by Carnevali *et al.* [1]. The target signal is a "smooth" binarization of a grey-scale image d_j, in terms of two-valued pixels $S_j \in \{-1, +1\}$. The prior distribution is designed to emphasize smoothness

$$P[S] \propto \exp\left(-\sum_{j=1}^{N}\sum_{j'=1}^{N} M(j,j')(S_j - S_{j'})^2\right) \tag{1}$$

$M(j,j')$ defines the connectivity, hence the unit cell of the cellular network. Here we just connect the nearest neighbors with strenght w_{M}.

We furthermore assume the signal degradation to consist in addition of white Gaussian noise. This degradation process leads to the following conditional distribution:

$$P[d|S] \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{N}(S_j - d_j)^2\right) \tag{2}$$

We use Bayes' formula to obtain the posterior distribution: $P[S|d] \propto P[d|S] * P[S]$. Clearly the posterior distribution is of the Gibbs form[1], with a cost-function given by the negative logarithm of the posterior distribution: $-\log P[S|d]$. We also note that the state dependent part of the cost-function is linear in the parameters w_{M} and $w_{\mathrm{d}} \equiv 1/\sigma^2$. This makes it suitable for Boltzmann machine learning [3, 4]. In this communication, however, we apply the cellular network with fixed parameters.

The Mean Field annealing method for estimation of averages over Gibbs distributions is well documented in the litterature see e.g. Hertz *et al.* [5]. The cellular neural network minimizes the Mean Field *free energy* and can be implemented either in analog mode or in discrete time mode:

[1] A distribution of the form $P(x) = Z^{-1} exp(-E(x)/T)$, where $E(x)$ is a cost-function, bounded from below, and T is a parameter

$$\langle S_j^{t+1} \rangle = (1 - \frac{\Delta}{\tau})\langle S_j^t \rangle + \frac{\Delta}{\tau} \tanh \left(\beta^t \left(\sum_{j'=1}^{N} M(j,j')\langle S_{j'}^t \rangle + w_d d_j \right) \right) \quad (3)$$

where the time-scale Δ/τ can be used to control the stability of the iteration process in digital implementation[8]. β^t quantifies the annealing schedule, in this work we use the simple schedule: $\beta^t = \beta^1 + (t/t_{max})(\beta^2 - \beta^1)$.

We illustrate the performance on the image produced by our synthetic fundus picture system. In particular we use the Mean Field scheme for segmentation of the output of the quadratic texture discriminant (figure 2). We set the parameters to $w_M = 1.0$, $w_d = 0.01$. The resulting segmentation after 20 iterations of the deterministic equation (3) is presented in figure 3.

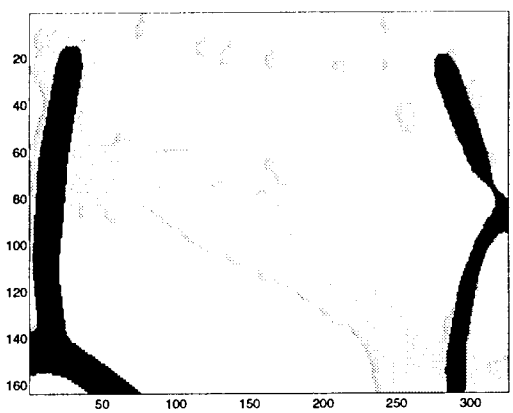


Figure 3: Output of the cellular segmentation network. The black areas represent the blood vessels that have been identified independently. Grey color signify areas of nerve fiber loss. The characteristic regional defect is assigned correctly. Close visual inspection of the synthetic input corresponding to the smaller grey spots indeed shows that the noise has degraded the texture significantly and it is unclear what the "true" classification should be.

Note that the segmentation network eliminates local noise. The resulting segmented image can be used either for rutine inspections to produce early warnings or for quantitation of the progress of the desease state and response to medication.

CONCLUSION

We have presented a modeling tool for design of synthetic fundus pictures. The system may be used either for educational purposes or as here for use as a laboratory for experiments with pattern recognition devices aimed at diagnosis of Glaucoma. We have designed and presented results of a simple and fast system for detection of nerve fibers in fundus pictures. Quantitative nerve fiber detection will be a key component in future automatic systems for rutine screening against Glaucoma. Current work concerns further test of the system and experiments on real fundus pictures.

ACKNOWLEDGMENT

This research is supported by the Danish Research Councils for the Natural and Technical Sciences through the Danish Computational Neural Network Center.

REFERENCES

[1] P. Carnevali, L. Coletti and S. Paternello: "Image processing by simulated annealing". IBM Journal of Research and Development **29**, 569-579, (1985).

[2] L.O. Chua and L. Yang: "Cellular Neural Networks: Theory". IEEE Transactions on Circuits and Systems **35**, 1257-1272, (1988).

[3] L.K. Hansen: "Boltzmann Learning of Parameters in Bayes Visual Reconstruction". Proceedings of the First Danish Conference on Pattern Recognition and Image Analysis. Ed.: S.I.Olsen. Department of Computer Science, University of Copenhagen, 92/8, (1992).

[4] L.K. Hansen: "Boltzmann Learning of Parameters in Cellular Neural Networks". Proceedings of Second Int. Workshop on Cellular Neural Networks and Applications CNNA'92, Munich (1992). IEEE Service Center, Piscataway NJ, 62-67, (1992).

[5] J. Hertz, A. Krogh and R.G. Palmer: "Introduction to the Theory of Neural Computation". Addison Wesley, New York (1991).

[6] M.M. Van Hulle and T. Tollenaere: "A Modular Artificial Neural Network for Texture Processing" Neural Networks **6**, 7-32 (1993).

[7] Jon Johnsen and Peter Grumstrup: "Modeling of Glaucoma induced changes in the retina and neural net assisted diagnosis (in Danish)". Master thesis (in Danish). Electronics Institute 1994.

[8] C. Peterson and J.R. Anderson: "A Mean Field Learning Algorithm for Neural Networks". Complex Systems **1** 995-1019, (1987).

[9] M.W. Roth: "Survey of Neural Network Technology for Automatic Target Recognition". IEEE Transactions on Neural Networks **1**, 28-43, (1990).

[10] M.B. Shields (Ed.): "Textbook of Glaucoma 3'rd Ed. " Lippencot, New York (1993).

[11] Y. Yamazaki, T. Miyazawa, and H. Yamada: "Retinal Nerve fiber analysis by a computerized digital image analysis system". Japanese Journal of Ophthalmol **34**, 174-180 (1990).

TOWARD IMPROVING EXERCISE ECG FOR DETECTING ISCHEMIC HEART DISEASE WITH RECURRENT AND FEEDFORWARD NEURAL NETS

Georg Dorffner
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria
georg@ai.univie.ac.at

Ernst Leitgeb, Heinz Koller M.D.
GTZ, A-3101 St. Pölten, Austria

Abstract. This paper reports about a study evaluating the usefulness of neural networks for the early detection of heart disease based on ECG and other measurements during exercise testing [10]. Data from 350 persons who underwent stress tests consisted of patient demographic data and fifteen time frames of measurements during stress and rest. Three different neural networks, two recurrent and one feedforward using background knowledge for preprocessing, were trained and compared to the performance of skilled cardiologists. It could be shown that the best neural networks can compete with experts in classifying tests as CAD (coronary artery disease) or normal. What concerns an index value expressing the likelihood of disease, to be used for monitoring the success of treatments, the neural networks outperformed classical statistical techniques published previously. This study has thus shown large evidence in favor of using neural nets to improve the exercise ECG as a non-invasive technique for detecting heart diseases.

THE APPLICATION

The electrocardiogram (ECG) is the recording of voltage changes transmitted to the body surface by electrical events in the heart muscle, providing direct evidence of cardiac rhythm and conduction and indirect evidence of certain aspects of myocardial anatomy, blood supply and func-

0-7803-2026-3/94 $4.00 © 1994 IEEE

tion. Electrocardiography has been used for many years as a key non-invasive method in the diagnosis and early detection of ischemic heart disease (coronary artery disease, or CAD), which is the leading cause of mortality in Western countries [5,6].

To improve the accuracy of the electrocardiogram and obtain more information on the dynamic state of the heart, exercise testing was introduced [5,11]. During stress testing not only the electrogardiogram is continously registered but also other physiological parameters are monitored (blood pressure, physical symptoms and angina pectoris). According to different established protocols, the workload is increased step by step and the changes of parameters during stress and recovery are recorded and analysed. Skilled cardiologists achieve 65–75% specificity (correctly classified normals) and 75–85% sensitivity (correctly classified CAD cases) in detecting CAD based on the resulting data [5,6].

In patients with suspected angina pectoris, exercise testing may confirm the diagnosis of ischemic heart disease and indicate the severity and prognostic importance of coronary artery lesions. In patients with definite ischemic heart disease, the exercise test is used to follow the progression or regression of the disease and the effect of therapy including drugs, invasive cardiology (e.g. angioplasty, atherectomy,..) or coronary artery surgery. Following myocardial infarction, exercise testing is performed to allow risk stratification, patients identified as being at low risk for death or re-infarction can be reassured and those at high risk can be managed appropriately [6].

If contra-indications (e.g. in the presence of acute, severe illness) are strictly observed, stress testing is a safe, cheap and non-invasive method, and is widely used in hospitals, by cardiologists, and general practitioners in primary care and health care centers. The success of the test is widely determined by the skill of the observer (cardiologist, general practitioner,..) and the patients themselves. Several efforts have been made to minimize these effects [7]. The following list shows a short summary of how automatic methods of CAD detection could improve the value of ECG and stress testing as indicator for heart diseases:

- automatic methods could minimize inter- and intraobserver variability on the test
- they could generally improve the detection of diseases like CAD
- they could contribute to improved monitoring of different therapies
- they could select continously new information on a given data set
- they could improve the acurracy of unskilled observers

Previous approaches to such improvements, such as [2,4,7,9], concentrated on classical statistical techniques and yielded results of up to 79 %

sensitivity and 76 % specificity. In this paper we report about studying artificial neural networks with respect to their ability for such improvements. In particular, if neural networks prove to be able to (objectively) classify cases comparably to (partially subjective) expert performance, and if they can provide tools for monitoring therapies, they can be viewed as valuable tools for future diagnostic systems in this domain. As the results below show, neural networks indeed prove to be able to do so.

THE DATA

The data used in this study consisted of patient-demographic parameters and fifteen frames of measurements from stress testing. The former included the person's sex, age, weight, and size, an indication whether a prior infarction is known, the workload that was reached by the person, the duration of the phase of the highest workload, and the expected heart rate, as well as workload to be achieved, computed according to [12,13]. The latter consisted of the above-mentioned measurements — namely heart rate, systolic and diastolic blood pressure, physical symptoms, angina pectoris, and features extracted from the ECG such as ST-segment depression and rhythmic anomalities. These measurements were taken during 11 stress phases (from 0 to 250 W, incremented by 25 W at each phase) and 4 subsequent rest phases (immediately after stress, and after 1, 3, as well as 5 minutes).

Data from 350 persons was available, including 107 normals and 243 with coronary artery disease, ranging from single to three vessel diseases. Among the 107 normals, data from 31 athletes were included. As compared to the other normals, these constitute "ideal normals," since all other persons undergoing stress testing were at least suspected of CAD and thus had a non-negligible prior probability for the disease. This is a well-known problem in using techniques like neural networks that rely on available data material. In many cases, normals are too similar to the pathologicals to permit clean separation. Non-evasive stress testing, on the other hand, can without risk be applied to persons with a negligible prior probability for the disease. The following table depicts the distribution of all cases, including a distinction according to the persons' sex:

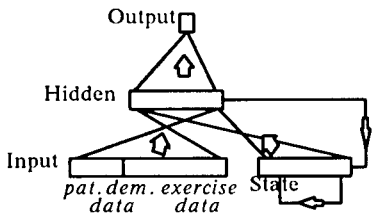


Figure 1

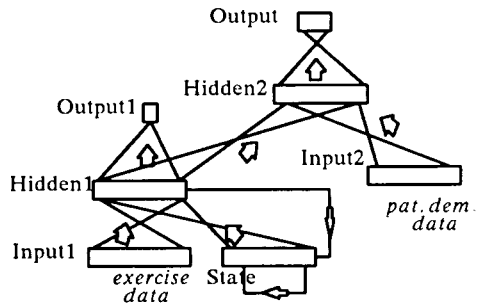


Figure 2

	total	females	males
athletes	31	2	29
other normals	76	33	54
1 vessel CAD	60	16	44
2 vessel CAD	80	13	67
3 vessel CAD	103	14	89

THE NEURAL NETWORKS USED

The task of this study was to evaluate the ability of neural networks to indicate coronary artery disease based on the data described above. Three types of neural network were used. The first and the second network were recurrent (roughly Elman-type) networks to account for the fact that the fifteen frames of stress test measurements form a time series with temporal evolution of all parameters. The third was a multilayer perceptron applied to preprocessed data (using knowledge about the domain).

Neural network 1: The first attempt of applying a neural network to the data was a somewhat "blind" training using only little background knowledge about the domain. An input layer of 19 units was used encoding patient demographic and stress test data for each time frame. This layer fed a recurrent network, somewhat similar to [3], as depicted in figure 1. At each update step through the network, the hidden layer activations were fed back to a state layer of the same size with weighted but fixed one-on-one connections. Each unit in this state layer was connected onto itself with a fixed weight. State and input layer together formed the input for the hidden layer, which in turn spread activation to an output layer of one unit.

Aside from the feedback via the recurrent connections the network was considered as a multilayer perceptron and thus trained by backpropagation at each of the fifteen time frames. The target for the output unit was chosen 1 for pathological cases and 0 for normal ones. To account for the temporal evolution during the fifteen time frames, the target for pathological cases was continuously raised from 0 to 1 between the start of the sequence and the last stress phase reached by the patient (i.e. the highest workload successfully passed). After that, for the remaining time frames, it was clamped at 1. A similar method for classifying sequences has been suggested elsewhere (e.g. [1]).

The hidden layer size was varied between 8 and 20. The weights on the one-on-one connections between hidden and state layers were kept fixed at 1, the weights of the state layer units onto themselves were all set at 0.5 (thus preserving 50 % of the previous history at each time frame in the sequence — compare [16]). Roughly half of the 350 cases (173) were chosen as a training set that was kept fixed for all training runs reported in this paper. It was chosen such that the distribution of athletes vs. other normals vs. pathologicals was roughly the same for training and test set, and such that no significant difference in the distribution of the patient-demographic parameters occured between training and test sets. Other than that, the selection was random. Each training run consisted of between 60,000 and 100,000 presentations of single cases (each consisting of the full fifteen time frames), picked randomly from the training set, with a learning rate of 0.01, and of between 60,000 and 100,000 further presentations with a learning rate of 0.001. This simple schedule of lowering the learning rate had proved sufficient for reaching convergence in several preliminary training runs, and was also fixed for all runs reported here. In addition, a momentum term (according to [15]) with scaling factor 0.9 was used.

One problem with this blind application of a recurrent network might be the over-representation of patient demographic data, which did not change during the temporal sequence. Thus, in several variations of this network scheme, the input units corresponding to this part of the data were activated only either at the beginning time frame, the final two time frames, or at both such ends of the sequence, while being clamped at 0 for the other time frames.

Neural network 2: To solve that possible problem of over-representation of patient demographic data in a more elaborate way, a second network architecture was devised and tested. It consists of two modules explicitly separating the data changing over time from the time-independent data, depicted in figure 2. The first module is another recurrent network as described in the previous section, but which was only fed with the time-chang-

ing stress test data. The second module is a multilayer perceptron with two input layers — a layer encoding the patient demographic data similar to above, and the hidden layer of the recurrent network after complete update cycles through the sequence. Training consisted of two phases — first of training the recurrent network as above, and secondly, of training the multilayer perceptron by backpropagation.

In addition, three output units instead of one were used encoding the more detailed cases of normal (all units 0), one, two or three vessel disease (first, first and second, or all three units active at 1). For evaluation, still only the distinction between normal and CAD was considered. The expected effect of the two additional units was improved discriminability through the extra information in the target (this was reported previously as improving network performance, e.g. [14]). Both hidden layer sizes were varied between 10 and 20.

Since many patients could not finish the stress test up to the highest workload (which itself is a certain indicator for CAD), many time frames consisted of zero measurements. Thus, in a further extension, the sequential update of the recurrent network was adjusted such as to skip those null frames, making the length of each sequence variable.

Neural network 3: The third attempt at a neural network solution involved an additional amount of background knowledge, which was mainly used to preprocess the data. The major difference to above was that no longer a recurrent network, but instead a multilayer perceptron with three input layers was used. The information in the time sequence was explicitly encoded by making use of previous methods of arriving at an indicator for CAD from the same kind of data [8]. There, each time frame was evaluated separately, and the contributions (basically a weighting of several factors considered as possible single indicators for CAD) of all time frames were summed. For the computation of a final index, which can be shown to highly correlate with CAD (see also below) only those sums were used. In addition, an explicit distinction between stress and rest phases was made.

According to these expert decisions, the third neural network was fed with the sums of the following indicators (taken from [5,6]; as in [8]):

- a deviation of the change in heart rate from a given tolerance interval
- a decrease in systolic blood pressure
- the presence one of several critical phsysical symptoms
- the presence of angina pectoris
- ST-segment depression
- the presence one of several critical rhythmic anomalities

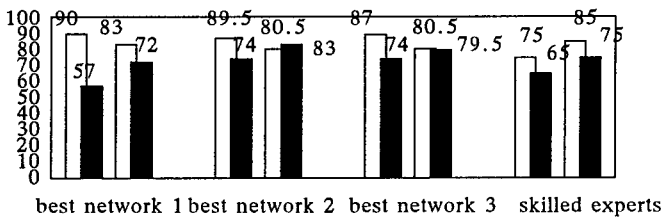


Figure 3

In distinction to [8], the first two were included as scaled values, instead of binary decisions about their presence. Furthermore, the following informations were also included [5,6]:

- a decrease in diastolic blood pressure
- pathological systolic blood pressure (larger than 140)
- pathological diastolic blood pressure (larger than 90)

again as scaled values. This was done separately for the stress and rest phases, leading to the activations of two of the three input layers. The third input layer encoded the patient demographic data as above. While many of these indicators were also used for network 2, here they were specifically tuned according to literature and, above all, explicitly summed up (rather than accumulated in the recurrent network).

Each training step consisted of one presentation of input patterns and one learning cycle with backpropagation. The hidden layer size was varied between 10 and 20. Again three output units were used.

THE RESULTS

In this study the neural networks were evaluated against two criteria:

(1) their ability to correctly classify cases into CAD and normal.

(2) their ability to produce an index expressing the likelihood of disease, which can be used to monitor the success of treatments (a decreasing index after treatment would indicate less likelihood of CAD and thus success of treatment).

Figure 3 shows an overview of the results concerning criterion (1). It depicts the best performances of the three networks, drawn as sensitivity (correct positives — white bars) and specificity (correct normals — black bars) in percentages. Since through varying the decision threshold at the

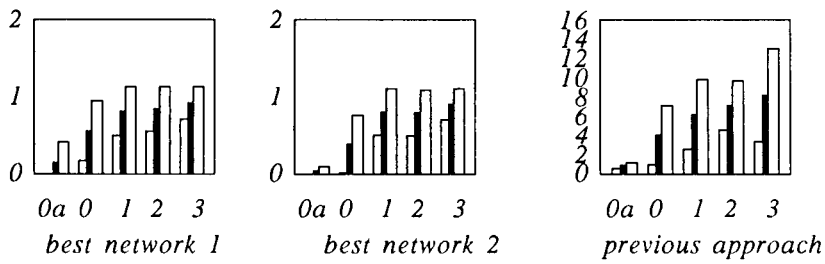


Figure 4

output unit these two values can be changed, for all results two pairs of values are depicted — one with relatively high sensitivity (always the result at the default threshold 0.5), the other with relatively high specificity (such that sensitivity stays above 80 %). For comparison, the range of the best performances of skilled cardiologists in interpreting the same data is shown (the two pairs of values corresponding to worst and best performance, i.e. the 75/65 % and 85/75 % mentioned above).

Concerning criterion (2), the original output value (which is simply compared to a threshold for the former criterion) appears to be usable as an index expressing the likelihood of disease. To demonstrate this, in figure 4 the mean (black bars) and standard deviations (white bars) of the output value for the five classes *athletes (0a)*, *other normals(0)*, *one*, *two*, and *three vessel disease* are shown. In the case of three output units the activation values of all units was averaged. This depiction shows a significant correlation between the index produced by the network and the extent of the disease. For comparison, the same five ranges (although on a different scale) are shown for a previously published statistical method for computing such an index [8].

DISCUSSION

The results show that neural networks can reach the upper ranges of expert performance, in some cases they can even perform slightly better. The second recurrent network using less background knowledge than the feedforward network but with the ability to exploit the time series based on the training data achieved best performance, although closely followed by the feedforward network. Neural networks 1 and 2 could also outperform previous non-neural approaches [2,9].

With respect to an index for monitoring the success of treatment, neural networks appear superior to traditional statistical methods. Standard devia-

tions are smaller and the separation between normals and pathologicals involves fewer overlap.

CONCLUSION

In this paper we have demonstrated the usefulness of neural networks in early detection of heart disease based on measurements during exercise testing. Recurrent networks which can exploit temporal dependencies appear as the best solution at the moment. Future research will investigate the combination of the recurrent approach with the type of background knowledge used in the feedforward case (e.g. through initialization), and the use of neural networks in hybrid neural/rule-based diagnostic systems. The results so far show great promise for significant contributions to making non-invasive ECG measurements during stress testing a prominent method for detecting one of today's most fatal diseases.

ACKNOWLEDGMENTS

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Science and Research. We thank the Economic Chamber of Lower Austria for supporting this research, designed to lead to improved software systems.

REFERENCES

[1] Anderson S., Merrill J., Port R.: Dynamic Speech Categorization with Recurrent Networks, Indiana University, Computer Science Dept., Techn. Report No. 258, 1988.

[2] Deedwania P.C., Joshi B., Cabajal E.V.: Analysis of Temporal Electrical Heterogeneity by a New 22 Lead ECG System Accurately Identifies Patients with Coronary Artery Disease, suppl. to *Circulation* 84, no. 4, 1991.

[3] Elman J.L.: Finding Structure in Time, *Cognitive Science* 14(2)179-212, 1990.

[4] Greenberg P.S., Cangiano H., Leamy L., Ellestad M.H.: Use of the multivariate approach to enhance the diagnostic accuracy of the treadmill test, *J. Electrocardiology* 63, pp.987-1000.

[5] Hurst W.: *The Heart*, 7th ed., New York: McGraw Hill, 1990.

[6] Julian D.G., Camm A.J., Fox K.M., Hall R.J.C., Poole-Wilson P.A.: *Dieseases of the Heart,* London: Bailliere Tindall, 1989.

[7] Kligfield P., Ameisen O., Okin P.: Heart rate adjustment of ST-segment depression for improved detection of coronary artery diesease, *Circulation* 79, pp. 245-255, 1989.

[8] Koller H., Leitgeb E.: Ist eine Verbesserung der Aussagekraft und Reproduzierbarkeit des Belastungs-EKG erzielbar?, *Wiener Medizinische Wochenschrift* 143(5), pp.110-117, 1993.

[9] Mark D.B., et al.: Prognostic Value of a Treadmill Exercise Score in Outpatients with Suspected Coronary Artery Disease, *New England Journal of Medicine* 325 (12), pp.849-853, 1991.

[10] McNeer J.F., Margolis J.R., Lee K.L., et al.: The role of exercise test in the evaluation of patients for ischemic heart disease, *Circulation* 57, pp.64-70, 1978.

[11] Mellerowicz H., Maidorn K., Matzdorff F., Nowacki P., Rittel H.F., Schmutzler H., Schön F.A., Stoboy H., Waterloh E., Zapfe H.: *Ergometrie*, München/Wien/Baltimore: Urban & Schwarzenberg, 1979.

[12] Niederberger M.: Grundlagen der Ergometrie, *Österreichische Ärztezeitung* 4., pp. 21-36, 1978.

[13] Niederberger M.: Belastungsuntersuchungen in der Kardiologie, *Herz* 7, pp.1-19, 1982.

[14] Prem E., Mackinger M., Dorffner G., Porenta G., Sochor H.: Concept Support as a Method for Programming Neural Networks with Symbolic Knowledge, in Ohlbach H.J.(ed.), *GWAI-92: Advances in Artificial Intelligence*, Berlin: Springer, Lecture Notes in AI, Vol.671, 1993.

[15] Rumelhart D.E., Hinton G.E., Williams R.J.: Learning Internal Representations by Error Propagation, in Rumelhart D.E., McClelland J.L.(eds.), *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, Vol 1: Foundations, Cambridge, MA: MIT Press, 1986.

[16] Ulbricht: Multi-reccurent networks for traffic prediction, to appear in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Seattle, 1994.

TOWARDS SEMEN QUALITY ASSESSMENT USING NEURAL NETWORKS

Chr. Linneberg, P. Salamon, C. Svarer, and L.K. Hansen
CONNECT, Electronics Institute, build. 349
Technical University of Denmark,
DK-2800 Lyngby, Denmark
email: linne,salamon,csvarer,lkhansen@eileen.ei.dth.dk

and

J. Meyrowitsch
DynaVision, Copenhagen Science Park, SYMBION
DK-2100 Copenhagen Ø, Denmark

Abstract. The paper presents the methodology and results from a neural net based classification of human sperm head morphology. The methodology uses a preprocessing scheme in which invariant Fourier descriptors are lumped into "energy" bands. The resulting networks are pruned using *Optimal Brain Damage*. Performance comparable to the error rate for human technicians is obtained.

1 INTRODUCTION

Semen quality assessment is important to fertility studies and standards have been introduced by the World Health Organization[1]. Recent research has demonstrated that semen quality has decreased by 50% during the past 50 years in the western world, possibly as result of the increasing exposure to pollutants and changes in diet [4, 5]. The Atlas of Sperm Morphology[2] defines 19 classes of sperm cells based on morphology. The characteristic abnormalities involve shape modifications, including multiple heads and tails. Furthermore, sperm cell motility is also important for definition of abnormality. Since sperm cells show great variety, the morphological classification problem represents a complex task. Furthermore, the appearence of the specimen depends on several factors that are only partially controllable, such as reduced imaging quality, caused by co-fixation of precipitates and presence of non-sperm cells. Hence flexible and robust classification tools are necessary.

0-7803-2026-3/94 $4.00 © 1994 IEEE

In this communication we analyse a neural net approach to automatic cell shape classification. Neural net learning makes it possible to compensate for the problems of specific imaging devices, and the user is not forced to produce an algorithmic description of the discriminant. Rather the same general software can adapt to a given setting based on examples produced by a skilled technician. Robustness is achieved by the networks' ability to generalize. The prime objective in machine learning is the ability to discriminate appropriately for *test* cases, i.e., examples that are different from the examples used for training of the neural system. Additional robustness can be obtained by careful *preprocessing* by which we mean model-based information processing, such as extraction of known salient features.

As a first step towards automatic quality assessment we have developed preprocessing algorithms for locating sperm cells and for extraction of salient features of the "head" and the "tail". In this presentation we will discuss the neural classifier for head shape classification. A convenient preprocessing strategy for shape discrimination has been proposed by [18, 11] and is based on Fourier analysis. A polygon in the image plane is isomorphic to a periodic sequence in the complex plane. For reviews and further analysis see also [11, 19, 16, 17, 14, 8]. Fourier descriptors are robust to the number of points used in sampling the shape and readily provide features which are invariant to changes in position, orientation, and starting point [18, 11].

Neural networks in conjunction with Fourier descriptors have been applied for shape discrimination previously in [13] and more recently in [8]. The objective of [13] was to recognize tools from a mechanical toolbox. The Fourier coefficients were crudely preprocessed by keeping only a few manually selected amplitudes. The results were promising; good performance was obtained with very few examples using a fully connected feed forward neural net trained by standard Backpropagation. However for sperm cell classification we have found that the inherent variability forces us to invoke optimized classifiers and more sophisticated preprocessing schemes. Our neural classifier approach was first described in [6], and in [8] it was tested on two sets of artificial cell shape data. Here we apply the system to real world data. In the context of cell discrimination the approach involves a new feature: the neural networks used have *adaptive* architectures. In particular, the networks are *pruned* to obtain the optimal connectivity. The advantage of the pruned networks is that they perform better on test data, ie. they do not simply memorize the training data but are able to *generalize* better than fully connected architectures. They are also able to select out the relevant inputs and thereby can mitigate the results of using too many input descriptors. More details on the simulator for the design of application specific architectures can be found in [7, 6]. Based on statistical theories of generalization, the best generalization is expected from the least complex networks. In our simulator we search for the best network with a pruning scheme based on the Optimal Brain Damage technique of [10]. We first train a large network that can easily implement the training set. Subsequently we compute the *saliency* of the

weights of the network and delete a fraction of the weights with the lowest saliency. The network is re-trained and the procedure repeated as long as the pruned network is able to implement the training set. For noisy problems, of course, one would tolerate a certain amount of error on the training set in order not to *overfit*[1]. Our results indicate that pruning improves performance but, however, should be terminated well before reaching the smallest network that is able to do the training set.

2 SHAPE PREPROCESSING

As stated in the introduction, we used Fourier *shape* descriptors, i.e. the inputs to our neural networks were computed from the Fourier coefficients of the sequence of (x, y) coordinate pairs viewed as complex numbers: $z = x+iy$. The complex representation of a sequence of N points becomes periodic with the definition: $z_{N+k} = z_k$. The Fourier representation has the advantage that the description is rather robust to sampling the actual cell shape with different points and is rather insensitive to the number of points in the sample [3]. It also allows for finding descriptors which are invariant to translation, rotation, and the choice of a starting point in our description of the curve [18, 11]. The invariances are important because they effectively increase the size of the training set. Using invariant inputs is roughly equivalent to enlarging the training set by adding rotated, renumbered etc. versions of each training sample.

The invariances are achieved by transforming the shape to "standard form". For example, the centroid of the shape can be moved to the origin by setting the zeroth Fourier coefficients equal to zero. The standard form inherent in the popular "elliptic Fourier descriptors" [11] is based on the elliptic approximation for the shape which is obtained by truncating the Fourier series to terms with frequency one. This so-called *primary ellipse* then serves through its major axis to provide a natural starting point and a natural coordinate system which takes care of rotation and scale. The resulting transformed Fourier coefficients are invariants which provide a complete description of the shape and are called the elliptic Fourier coefficients.

To further reduce the complexity of the input representation, hence the network, we lumped the component by summing the energy in certain frequency bands.

3 NEURAL CLASSIFIER

Our simulation tool for design of adaptive neural architectures was described in [7, 6]. The initial architecture is an ordinary feed-forward network with one hidden layer. The standard output coding scheme for multi-class problems is used, hence, each class is represented by a specific output neuron. The

[1]For an implementation of this within time series prediction see [7].

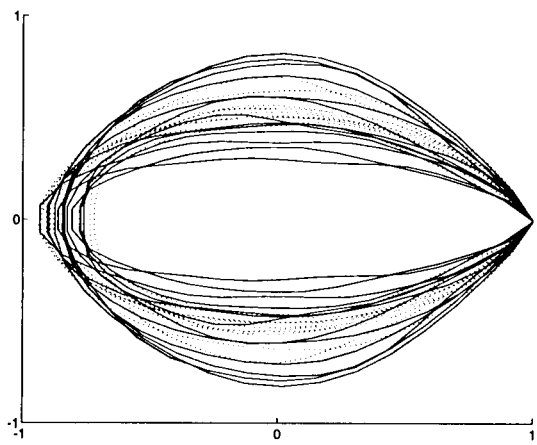


Figure 1: Reconstruction of cells from reduced sets of Fourier coefficient. (Solid: normal cell shapes; dotted: amorphous cell shapes)

activation functions of the hidden neurons are hyperbolic tangents, while the activation functions of the output neurons are linear. The training error is the usual sum of squared errors. The learning algorithm adjusts the weights of the network in order to minimize the error function. For the hidden layer, a second order modification of the back-propagation algorithm with batch learning is used, and, for the output neuron, the matrix inversion algorithm, described in [9] is used.

The algorithm used to adapt the neural network architecture is in brief:

1. Train the fully connected neural network using the learning algorithm.
2. Use the Optimal Brain Damage method to calculate the saliencies of all the weights in the network.
3. Find the weight in the network with the lowest saliency and remove this weight by setting it (permanently) to zero. If a hidden neuron is disconnected from the output neuron the hidden neuron is removed.
4. Retrain the network using the learning algorithm.

4 THE IMAGE DATABASE

The image database used in this work consisted of digitized images of fixated and Papanicolau stained semen specimens collected at the Department of Growth and Reproduction, Rigshospitalet Copenhagen. Each image obtained contains a variable number of manually classified cells. The classifications were far from certain and show that the problem includes significant

overlap between the categories. Six independent laboratory technicians classified the images and on the average the technicians differed from the consensus classification on more than 15% of the samples. To simplify matters, we have restricted our attention to binary discrimination using two categories: "normal" and "abnormal". The set consisted of 50 cells; 25 from each category. The set was divided into a training set of 13 cells from each category, and a test set of the remaining 24 cells.

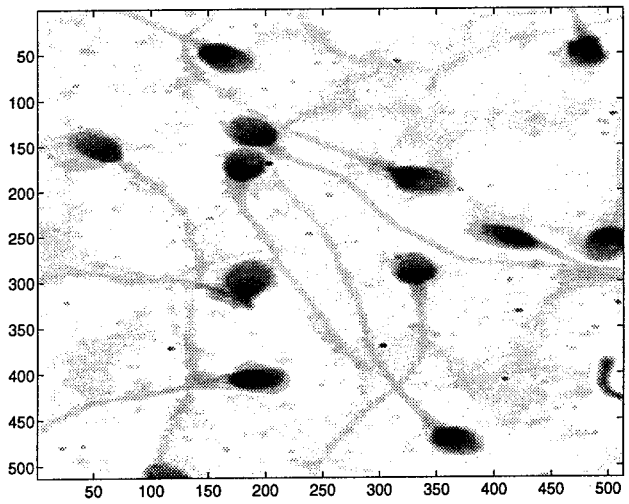


Figure 2: An original grey scale image used for manual labeling of the sperm cells

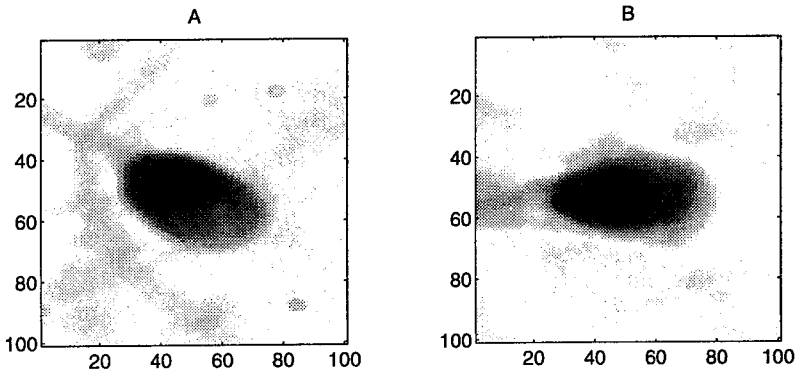


Figure 3: Examples of the two kinds of sperm cells. A) Normal cell shape B) Amorphous shape

5 EXPERIMENTAL RESULTS

The feed forward net was configured with 20 input units. The first 16 input units contained normalized Fourier amplitudes corresponding to 8 low-frequency pairs. The remaining four inputs represented the accumulated energies of 2 intermediate-frequency bands. The phases of all the Fourier components were discarded along with the amplitudes for the high-frequency components . The network has been initialized with random weights and a simple feed-forward architecture with 6 hidden neurons. It was found that the nets could be pruned significantly and that such pruning improved generalization. The training and test errors during a pruning run are depicted in figure 5. Contrary to previous experience using the pruning strategy [7, 6], the network with minimum test error was **not** the network with the smallest architecture which correctly implements the training set. Furthermore, we suspect that this is generally to be expected for problems with small training sets and significant overlap between categories. We are presently working on a crossvalidation methodology for choosing the best architecture along a pruning sequence.

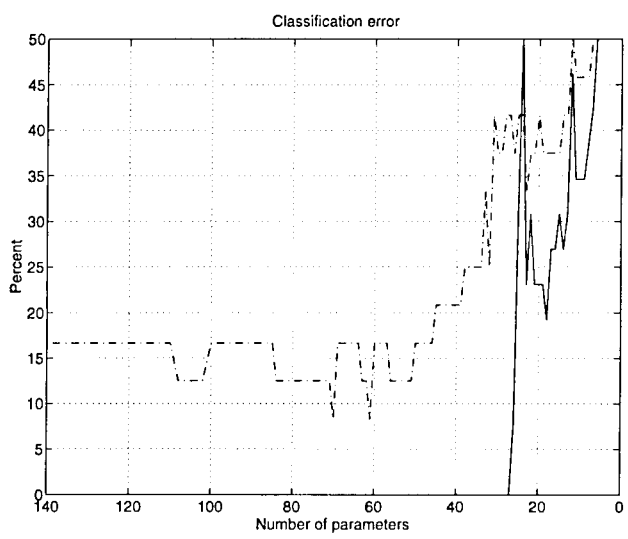


Figure 4: Training (solid line) and test errors (dot-dashed) as pruning progresses. For reference, the test error level of a technician is about 15%.

Inspecting the test errors it is found that pruning can decrease the error by about 25%.

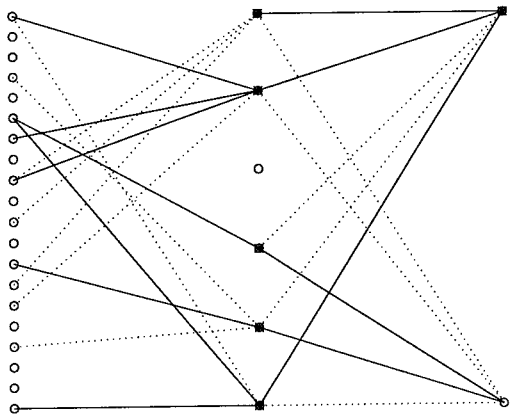


Figure 5: Typical pruned network that can learn the training set (Solid line - Positive weight, Dotted line - Negative weight).

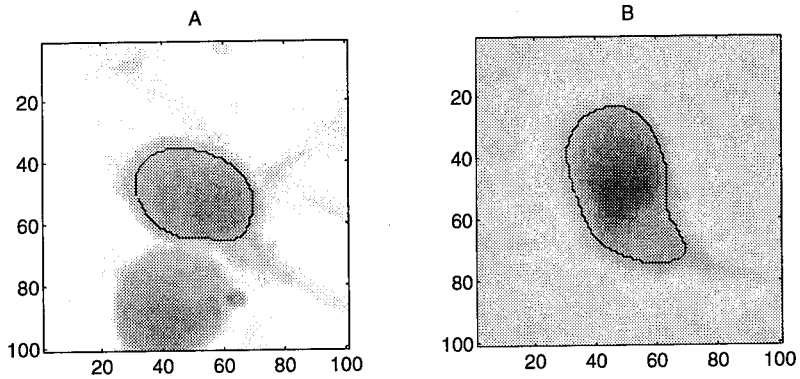


Figure 6: Two cell shapes from the test set that were not learned by the above network. A) normal cell B) amorphous cell.

6 CONCLUSION

It has been shown that neural networks are useful for morphological classification of cell shapes. Application specific architectures can be designed automatically which generalize well to an independent test set. The optimal architectures showed that only a few neurons and connections are necessary. Also it has been shown that preprocessing the cell shape coordinates using the complex Fourier transformation provides a well-suited and compact representation of the data. The optimal networks use less input information than the human eye apparently found necessary. Current work concerns larger databases, multi-class classification, and crossvalidation methodologies for choosing the best network among a pruning sequence.

ACKNOWLEDGMENTS

This research is supported by the Danish Research Councils for the Natural and Technical Sciences through the Danish Computational Neural Network Center. We thank A. Gewercman and N. Jørgensen (Dept. of Growth and Reproduction, Rigshospitalet, Copenhagen) and Erik Bostofte (Fertilitetsklinikken, Hvidovre Hospital and The Clinic of the medical doctors of Copenhagen) for useful discussions and for providing data for this project.

REFERENCES

[1] "WHO Laboratory Manual for the Examination of Human Semen and Semen-Cervical Mucus Interactions" Cambridge University Press, Cambridge 1992.

[2] M.M. Adelman and E.M. Cahill "Atlas of Sperm Morphology" ASCP Press, Chicago (1989).

[3] G.W. Strang "Introduction to Applied Mathematics" Cambridge University Press, Cambridge 1985.

[4] E. Carlsen, A. Gewercman, and N.E. Skakkebæk: "Evidence for decreasing quality of semen during past 50 years" British Medical Journal **305**, 609-613 (1992).

[5] R.M. Sharpe and N.E. Skakkebæk: "Are oestrogens involved in falling sperm counts and disorders of the male reproductive tract?" The Lancet **341**, 1392-1395 (1993).

[6] J. Gorodkin, L.K. Hansen, A. Krogh, C. Svarer, and O. Winter: "A Quantitative Study of Pruning by Optimal Brain Damage" Int. Journ. Neural Systems (1993).

[7] C. Svarer, L.K. Hansen, and J. Larsen: "On Design and Evaluation of Tapped-Delay Neural Network Architectures" The 1993 IEEE Int. Conference on Neural Networks San Francisco. Eds. H.R. Berenji et al., 45-51, (1993)

[8] C. Svarer, L.K. Hansen B. Wellhouse and P. Salamon: "Classification of Cell Shapes using Designer Networks". In preparation (1994).

[9] Simon A. Barton: "A Matrix Method for Optimizing a Neural Network", Neural Computation **3**, 450-459, (1991)

[10] Yann Le Cun, John S. Denker and Sara A. Solla: "Optimal Brain Damage", In Advances in Neural Information Processing Systems II (Denver 1989), ed. D.S. Touretzky, 396-404. San Mateo: Morgan Kaufmann, (1989)

[11] F.P. Kuhl and C.R. Giardina: "Elliptic Fourier Features of a Closed Contour" Computer graphics and image processing **18**, 226-258 (1982)

[12] Bill Wellhouse, Department of Mathematical Science, San Diego State University: "Description of cell shape data", Private communication, (1992)

[13] H-H. Wu and R.A. Schowengerdt: "Shape Discrimination Using inveriant Fourier Representation and a neural network classifier" SPIE **1569** Stochastic and Neural Methods in Signal Processing, Image Processing and, Computer Vision 147-154 (1991).

[14] L.M. Kerley and J.R. Knisley: "Complex Vectors and Image Identification" The College Mathematics Journal **24**, 166-174 (1993).

[15] J.C. Schon, J.T. Torre-Bueno, and G.B. Stefano: "Microscopic computer-assisted analysis of conformational state: reference to neuroimmunology" Advances in Neuroimmonology **1**, 252-259 (1991).

[16] G. Diaz, A. Zuccarelli, I. Pelligra, and A. Giani: "Elliptic Fourier Analysis of Cell and Nuclear Shapes" Computers and Biomedical Research **22**, 405-414 (1989).

[17] G. Diaz, D. Quacci, and C. Dell'Orbo: "Recognition of cell surface modulation by elliptic Fourier analysis" Computer Methods and Programs in Biomedicine **31**, 57-62 (1990).

[18] G.H. Granlund: "Fourier Preprocessing for Hand Print Character Recognition" IEEE Transactions on Computers **C-21**, 195-201 (1972).

[19] A.W. Partin, J.S. Schoeniger, J.L. Mohler, and D.S. Coffey: "Fourier analysis of cell motility: Correlation of motility with metastatic potential" Proc.Natl.Acad.Sci **86**, 1254-1258, (1989).

USE OF NEURAL NETWORKS IN DETECTION OF ISCHEMIC EPISODES FROM ECG LEADS.

Nicos Maglaveras[1], Member IEEE, Telemachos Stamkopoulos[1], Costas Pappas[1] and Michael Strintzis[2], Senior Member IEEE

Aristotelian University,
Lab of Medical Informatics[1] and Information Processing Chair[2],
54006 Thessaloniki, Macedonia, GREECE

Abstract. A supervised neural network (NN) algorithm was used for automated detection of ischemic episodes resulting from ST segment elevation or depression. The performance of the method was measured using the European ST-T database. In particular the performance was measured in terms of beat-by-beat ischemia detection and in terms of ischemic episodes detection. Aggregate statistics for the description of the detector performance were used due to the small number of events. The algorihtm used to train the NN was an adaptive backpropagation (BP) algorithm. This algorithm reduces dramatically training time (10-fold decrease in our case) when compared to the classical BP algorithm. The resulting NN is capable of detecting ischemia independently of the lead used. It was found that the average ischemia episode sensitivity is 88.62% while the average ischemia sensitivity is 72.22%. This drop in ischemia sensitivity could be attributed to the diverse statistical properties of the ECGs within the same patient. The results show that NN can be used in ECG processing in cases where fast and reliable detection of ischemic episodes is desired as in the case of critical care units (CCUs).

INTRODUCTION

Ischemia is considered to be a major complication of the cardiac function, and a prime cause for the occurence of cardiac infarction and dangerous cardiac arrhythmias. The main characteristic of ischemia in the cellular level is the depolarisation of the cellular resting membrane potential. This causes a potential difference between the normal and ischemic tissue, which in turn causes the flow of an "injury current" [1]. This "injury current" is manifested in the ECG by an ST depression or elevation depending on the anatomical position of the heart and the dipoles with respect to the recording electrodes. Thus there are cases in the 12-lead standard electrode system that the ST depression is not as evident, or where we have ST depression while no ischemia is present such as can happen with leads III and AVF due to patient position [2].

0-7803-2026-3/94 $4.00 © 1994 IEEE

Major problems with detecting the ST segment in the ECG can be identified as follows: 1. Slow baseline drift, 2. Noise, 3. Sloped ST changes and 4. Numerous ST-T patterns within the same patient. A number of methods have been proposed in the literature until today on this problem based on digital filtering, time analysis of the signal's first derivative, or syntactic methods. None of these methods though were able to be tested in an annotated database to obtain a good measure of their ability to detect ST depression. Recently a new annotated database was developed that contains recordings with annotated ischemic episodes [3]. The database contains two leads. A couple of new algorithms were developed to identify ischemia using information from both leads which improved sensitivity [2].

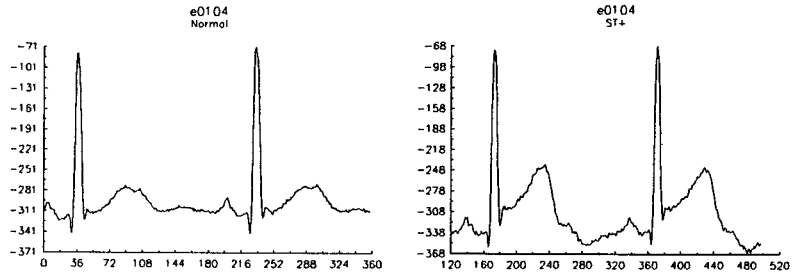
Neural networks have appeared over the past few years as pattern and statistical classifiers [4] and have been used in many areas of science. They have also been used in medicine and in ECG analysis in particular [5]-[7]. Since they can be trained to recognise patterns they have a good chance for recognising the complex patterns an ST segment can have.

In this paper we implement an adaptive backpropagation NN for ischemic episodes detection. The testing of the efficiency of the NN in detecting ischemic episodes is done using the European ST-T database. The training of the NN was found to be dramatically decreased by adapting the gain term in the delta rule, so that we can avoid local minima in the error phase plane. The test of the NN efficiency is made using aggregate statistics, and deriving specific indices for both ischemic episode and ischemia duration sensitivity and predictivity [8].

METHODS

The main stages we followed in our algorithm were:

1. Selection and preprocessing of the training set.: A training set is constructed using patterns from the ST-T database. Some patterns used in this set are shown in Figure 1. These patterns included normal, depressed, sloped and noisy ST segments. In particular 50% of the patterns used were normals, 25% had ST depression and 25% ST elevation. All patterns came from channel #1 of the European ST-T database (Table 1).



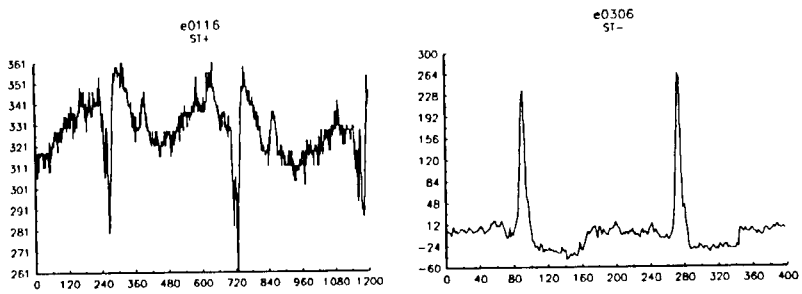


Figure 1:

A selection of representative patterns chosen to train the NN. We can observe normal patterns, ST depressed, and sloped ST changes.

Table 1: Training set statistical analysis.

Reference	V3	V4	V5	MLIII
Normal	10%	60%	10%	20%
ST+	0%	71.4%	14.3%	14.3%
ST-	28.6%	42.8%	0%	28.6%

2. Description and training of the NN: The NN used to identify the ST segment, consists of three layers. The first layer (input layer) has 20 neurons. This number is equal to the number of selected points from preprocessing stage. This number has been taken by experimental work and gives the best results. The second layer which is the hidden one, has 10 neurons. The number of 10 was chosen to avoid repetition problems and to minimize the training time. The number of hidden neurons represents the total memory of NN. For the third layer two neurons were used. The output of these neurons is a value between 0 and 1 and is considered to be 1 if it is greater than 0.5 and it is 0 otherwise. The patterns taken from output layer represent a type of ST segment. In a feedforward network (such as the BP NN), each unit has an activity level that is determined by the input received from units in the layer below. The total input to neuron j can be written as an inner product of input and weight vectors plus a constant T_j (bias). The output of neuron j is taken applying a sigmoid activation function f(.) to the total input. This adaptation algorithm changes backpropagation rates, depending on the derivative of energy function. When the energy reaches a local minimum which is detected observing the first derivative, the algorithm changes the training rates of NN in such a way that increasing the first derivative by a small amount, the energy decreases after a few iterations again.

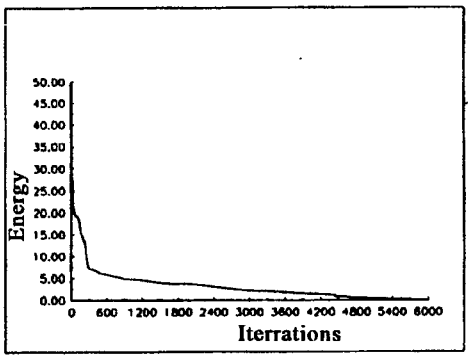


Figure 2:
The energy function as it changes during the NN training illustrating the monotonicity of the convergence of the error calculation

This results in a monotonicaly decreasing energy function (Figure 2). The most formal way to do this is to increase the delta rule rates for an amount equivalent to 10% of the former rate value and then to decrease it to a value smaller than the old one. This method cannot guarantee the minimization of energy function in all cases. But in our case it works well and could dramatically decrease the training time.

3. Preprocessing of the ECG signal - Recall phase of the NN: The main problem in detecting ischemia is to formalize the ST-segment in order to make an input suitable for NN without loosing any information. This is done here, finding the differences of ischemic ST segment template from this of normal one. At first, the detection of R-point is done using an LVQ neural network. The percentage of this NN in finding R-peaks is about 97% [7]. The ST segment begins at 60 msec after R-peak in normal case. In the case of tachycardia this value should decrease a little (here, 40 msec if RR-interval ‹ 600msec). The ST-segment also has predefined length which is equal to 160 msec. A baseline correction must be made in order to decrease false detections. Here a simple method is applied based upon the suggestion that the isoelectric level of the signal lies on the area approximately 80 msec left of the R-peak, where the first derivative becomes equal to zero for at least 10 msec or in the flattest 20 msec segment. Despite of its simplicity the algorithm has very good results in most cases. Because the interesting point here is the differences of ST-segment from normal ST, it is needed to substract the normal template from ST-segment. The normal template is constructed for each signal taking the average beat of the ten first beats of recording which are suggested to be normal. The average signal is taken to avoid noise problems due to the vibration of ECG signal. Thus, after this procedure the final part of signal consisting of N points (40 points in 250 Hz sample frequency) has only information that shows the differences between the

normal and testing waveform. This has been done also, in order to standarize the algorithm and to make it insensitive to differences from one lead to an other and from a patient to another. This number of N points which are taken from each beat, is reduced to 20, taking the mean value for every N/20 concecutive points. Finally a constant bias (different than the bias used in the BP algorithm of NN) is added to all the input vectors as polarization to avoid confusions in the classification and then all input vectors are normalized due to the BP algorithm. The BP method for training despite of his non-linearity, has some problems concerning the compatibility of input pattern with the NN-system. To avoid such problems, normalization of them should be applied. The pattern taken from the preprocessing stage, is divided by the euclidean distance of the vector from the zero point. One problem here is that using this method probably some useful information of ST-segment related with the length of input vector, may be lost. After the preprocessing stage, the signal is fed in the NN.

RESULTS - DISCUSSION

The algorithm was tested on the European ST-T Database [3]. This database contains 63 records with 160 annotated ischemic episodes for lead 1. For all the test, lead 1 was used because the signal quality was better compared to that of lead 0. For performance measure, four indices were calculated [8]. These indices refer to two distinct classes of detection. The two refer to the correct detection of the existence of an ischemic episode, while the other two to the correct detection of the duration of the episode. In particular these indices are:

1) Ischemic ST episode sensitivity (ST Se) defined as the ratio of the number of matching episodes and the number of annotated episodes.

2) Ischemic ST episode predictivity (ST+ P) defined as the ratio between the number of matching episodes and the number of detected episodes

3) Ischemia duration sensitivity (IS Se) defined as the ratio between the duration of true mathced ischemia and the total duration of annotated ischemia

4) Ischemia duration predictivity (IS + P) defined as the ratio between the duration of true matched ischemia and the total duration of detected ischemia

There are two types of statistical measures performed using the above mentioned indices. The first one is termed average statistics. This one gives equal weight to the ischemic episode at each file. The second one is termed Gross Statistics and gives equal weight to each ischemic episode. Thus we have calculated the four indices for each file, and for each lead separately. This show us the change of performance of algorithm from lead to lead. In table 2 the results are shown. To predict performance in clinical practice, it is important to model how well a detector behaves on a randomly chosen recording. For this reason, one might expect average statistics, in which each recording is equally weihted to be better predictor of ischemia episodes occurence than gross statistics.

Table 2 summarises the results for average and gross statistics. As can be seen, the NN performs equally well for leads not belonging to the training set (such as ML I), and we can observe that the average sensitivity and predictivity of the ischemia episodes is quite high (88.62% and 78.38%).

Table 2: Performance measures of the adaptive backpropagation NN in ischemia detection in the European ST-T database.

AVERAGE				
LEAD	**Episode Sensitivity (%)**	**Episode Predictivity (%)**	**Ischemia Sensitivity (%)**	**Ischemia Predictivity (%)**
MLI	97.22	90.28	93.55	77.88
MLIII	79.90	66.46	55.53	57.19
V1	87.50	80.00	71.05	83.73
V2	33.33	33.33	100.00	20.79
V3	100.00	100.00	58.87	64.86
V4	90.63	77.08	69.88	70.78
V5	94.44	86.51	80.09	62.19
TOTAL	**88.62**	**78.38**	**72.22**	**67.49**
GROSS				
MLI	95.24	76.92	94.19	78.69
MLIII	74.42	56.14	36.22	59.23
V1	85.71	78.26	79.98	93.09
V2	33.33	33.33	100.00	20.79
V3	100.00	100.00	50.18	66.81
V4	86.67	68.42	78.01	79.92
V5	93.75	71.42	89.73	59.93
TOTAL	85.00	68.69	73.00	69.45

The fact that in certain leads (such as the V2) the figures of merit are low, can be attributed to the fact that only one record of the database contains lead V2 as lead #1, and thus even aggregate statistics cannot give an accurate measure of the NN performance. Also, another point of interest is the use of an average template for each patient. It is well known that pathological levels of ST depression can vary in the same patient, and in different patients, Thus an initial estimate of the physiological ST depression (or elevation) can be taken by averaging the ST segments of the first ten beats. This on the other hand, may cause reduced noise levels due to averaging, and thus it may cause problems when trying to detect ischemia in areas of elevated noise, since only baseline correction filtering is performed. Baseline correction, is another possible source

of error, but the major problems seem to be the identification of the training set, the off-line training procedure, which should be done on-line, and the nonlinearities involved in the BP NN rendering theoretical analysis almost impossible. From a detailed study of each file in the database, it was concluded that another problem was the adjustment of the assumed ST segment starting point according to the heart rate. It is finally important to note here that ischemia cannot be conclusive only from the ST segment changes on the ECG, since there are no golden standards in identifying ischemia merely by looking the ECG, although the European ST-T database is a major step towards the solution of this problem. The positive points on the other hand, are the good general performance of the NN even though the training set is relatively small, the extremely fast recall phase, and the fact that in certain areas where the J point is impossible to be detected, the NN performs very well.

REFERENCES

[1]. LS Gettes, WE Cascio, "Effect of acute ischemia on cardiac electrophysiology" In *The Heart and Cardiovascular System*, HA Fozzard et al (Edts), Raven Press, vol.2, pp. 2021-2054, 1991.

[2]. F Jager, GB Moody, A Taddei, RG Mark, "Analysis of transient ST segment changes during ambulatory monitoring", *Computers in Cardiology*, IEEE Comp Soc Press, pp. 453-456, 1991.

[3]. A Taddei, G Distante, M Edmin, P Pisani, GB Moody, C Zeelenberg, C Marchesi, "The European ST-T Database: standard for evaluating systems for the analysis of ST-T changes in ambulatory electrocardiography", *Europ. Heart J*, vol. 13, pp. 1164-1172, 1992.

[4]. JA Freeman, DM Skapura, *Neural Networks: Algorithms, applications and programming techniques*, Addison Wesley, 1991.

[5]. Y Suzuki, K Ono, "Personal computer system for ECG ST-segment recognition bbased on neural networks", *Med & Biol Eng & Comput*, vol. 30, pp. 2-8, 1992

[6]. L Ebenbbrandt, B Devine, PW MacFarlane, "Neural networks for classification of ECG ST-T segments", *J of Electrocardiology*, vol. 25, pp. 167-173, 1992.

[7]. M Strintzis, X Magnisalis, G Stalidis, N Maglaveras, "Use of Neural Networks for Electrocardiogram (ECG) feature extraction recognition and classification", *Neural Network World Journal*, vol. 3-4, pp. 313-327, 1992.

[8]. F Jager, GB Moody, A Taddei, RG Mark, "Performance measures for algorithms to detect transient ischemic ST segment changes", *Computers in Cardiology*, IEEE Comp Soc Press, pp. 369-372, 1991.

Adaptive Processing And Communication

A Neural Network Trained with the Extended Kalman Algorithm Used for the Equalization of a Binary Communication Channel

Martin Birgmeier
Institut für Nachrichtentechnik und Hochfrequenztechnik
Technische Universität Wien
Gußhausstraße 25/E389, 1040 Vienna, Austria
Tel.: (+43 1) 58801 x 3661, Fax: (+43 1) 5870583
Email: Martin.Birgmeier@nt.tuwien.ac.at

Abstract — This paper describes a feedforward neural network architecture trained with the extended Kalman filter algorithm instead of the standard (LMS) method. It presents a simplified recursive procedure for calculating the necessary derivatives. The resulting algorithm is then used to train a network to adapt to the decision boundary of an optimal receiver for a binary communication channel, resulting in increased convergence speed and better approximation properties.

INTRODUCTION

This paper describes the application of a neural network to the task of equalizing a binary communication channel. This problem has been considered before in the literature, and various network architectures have been employed, showing that it is indeed possible to get close to the performance of an optimal receiver by using a neural network in its place (see the papers by Cowan, Mulgrew, and others [1], [2], [3], [4], Al-Mashouq and Reed [5], and Ramamurti, Rao, and Gandhi [6]). In these papers, either feedforward neural networks using the simple backpropagation (LMS) rule or radial basis function networks are described.

For complex decision boundaries, the standard backpropagation algorithm converges very slowly. In order to improve convergence speed, in this paper a feedforward neural network is trained using the extended Kalman filter algorithm. Kalman-trained neural networks have been described previously in the literature, having been applied mostly to the task of adapting to some predefined, artificial partitioning of the input space (see for example [7], [8], [9], [10], [11]). In the current paper, the Kalman-trained neural network is used to reduce the number of training steps required for the network to learn a partitioning of the input space implicitly given by the characteristics of the transmission channel. The results obtained show that this partitioning approximates the decision boundary of an optimal receiver, and that the average error produced by the neural network is close to that of the optimum receiver. Furthermore, the number of training steps are reduced significantly when compared to the standard LMS algorithm.

0-7803-2026-3/94 $4.00 © 1994 IEEE

THE EXTENDED KALMAN FILTER APPLIED TO A FEED-FORWARD NEURAL NETWORK

This section presents a derivation of the extended Kalman filter equations for the update of the link weights of a feedforward neural network. It is loosely based on the algorithm presented by Iiguni et al. [7], but includes a simplified and generalized version of the recursive equations for determining the partial derivatives of the link weights with respect to the output values.

The following definitions are used for the variables in the feedforward network:

x_i^l ... output of node i in layer l

s_i^l ... sum of inputs of node i in layer l

$w_{i,j}^l$ ... weight from node $(l-1, j)$ to node (l, i)

L ... number of layers, excluding layer 0

N_l ... number of nodes in layer l, excluding node 0

$f(\cdot)$ ... sigmoid function

The following conventions are used:

$x_0^l \equiv 1$ ... node zero of each layer provides the offset for the following layer.

$x_i^0 = r(n-i)$ ... nodes in layer zero correspond to input values.

$\mathbf{x}^l = \left[x_1^l \ \ldots \ x_{N_l}^l\right]^T$ ... outputs of layer l, excluding constant value.

$\overline{\mathbf{x}}^l = \left[x_0^l \ x_1^l \ \ldots \ x_{N_l}^l\right]^T$ ... outputs of layer l, including constant value.

$\mathbf{s}^l = \left[s_1^l \ \ldots \ s_{N_l}^l\right]^T$ ... sum of inputs to layer l.

$$\mathbf{w}^l = \begin{bmatrix} w_{1,1}^l & \cdots & w_{1,N_{l-1}}^l \\ \vdots & \ddots & \vdots \\ w_{N_l,1}^l & \cdots & w_{N_l,N_{l-1}}^l \end{bmatrix} \quad \ldots \text{ weights between layers } l-1 \text{ and } l, \text{ excluding offset values.}$$

$$\overline{\mathbf{w}}^l = \begin{bmatrix} w_{1,0}^l & w_{1,1}^l & \cdots & w_{1,N_{l-1}}^l \\ \vdots & \vdots & \ddots & \vdots \\ w_{N_l,0}^l & w_{N_l,1}^l & \cdots & w_{N_l,N_{l-1}}^l \end{bmatrix} \quad \ldots \text{ weights between layers } l-1 \text{ and } l, \text{ including offset values.}$$

$$\overline{\mathbf{w}}_i^l = \left[w_{i,0}^l \ w_{i,1}^l \ \cdots \ w_{i,N_{l-1}}^l\right]^T \quad \ldots \text{ weights leading to node } i \text{ in layer } l, \text{ including offset values.}$$

$$\mathbf{w} = \left[w_{1,0}^1 \ w_{1,1}^1 \ \ldots \ w_{N_1,N_0}^1 \ w_{1,0}^2 \ \ldots \ w_{N_L,N_{L-1}}^L\right] \quad \ldots \text{ all weights in the network.}$$

$\mathbf{d}(n) = d(n) = a \cdot s(n-k) + b$ ... (scaled version of) desired response.

Using this, the forward pass through the neural network is

$$\mathbf{x}^l = f(\overline{\mathbf{w}}^l \overline{\mathbf{x}}^{l-1}), \qquad 1 \le l \le L \tag{1}$$

In a stationary environment it is assumed that the optimum setting of the weights in the network is constant. Hence, the state transition matrix of the Kalman state model is the identity matrix, and the process noise vector is

zero (cf. Haykin [12]). For this case, the Kalman filter equations reduce to

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{G}(n)[\mathbf{d}(n) - \mathbf{x}^L(n)] \tag{2}$$

$$\mathbf{G}(n) = \mathbf{K}(n-1)\mathbf{C}(n)^H \left[\mathbf{C}(n)\mathbf{K}(n-1)\mathbf{C}(n)^H + \mathbf{R}_{min}\right]^{-1} \tag{3}$$

$$\mathbf{K}(n) = \mathbf{K}(n-1) - \mathbf{G}(n)\mathbf{C}(n)\mathbf{K}(n-1) \tag{4}$$

where $\mathbf{w}$ is the concatenation of all weights in the network and $\mathbf{C}(n)$ is the first term in the Taylor approximation of $\mathbf{x}^L(n) = h(\mathbf{w}(n))$, i.e. the partial derivative of $\mathbf{x}^L(n)$ with respect to $\mathbf{w}$, evaluated at $\mathbf{w} = \mathbf{w}(n-1)$. This latter approximation to $h(\cdot)$ constitutes the extended Kalman algorithm (cf. [13]).

For networks of even moderate sizes, the resulting gain and correlation matrices $\mathbf{G}(n)$ and $\mathbf{K}(n)$ would become unmanageable. Hence, the Kalman filter equations are applied independently to each node in the network for estimating the weights leading to that node only, as proposed by Iiguni [7] and others (see for example the NEKA algorithm in [8]). In this way, the Kalman filter equations for the weights leading to a single node are

$$\overline{\mathbf{w}}_i^l(n) = \overline{\mathbf{w}}_i^l(n-1) + \mathbf{G}_i^l(n)[\mathbf{d}(n) - \mathbf{x}^L(n)] \tag{5}$$

$$\mathbf{G}_i^l(n) = \mathbf{K}_i^l(n-1)\mathbf{C}_i^l(n)^H \left[\mathbf{C}_i^l(n)\mathbf{K}_i^l(n-1)\mathbf{C}_i^l(n)^H + \mathbf{R}_{i,min}^l\right]^{-1} \tag{6}$$

$$\mathbf{K}_i^l(n) = \mathbf{K}_i^l(n-1) - \mathbf{G}_i^l(n)\mathbf{C}_i^l(n)\mathbf{K}_i^l(n-1) \tag{7}$$

A simplified recursion formula for $\mathbf{C}_i^l(n)$ can be derived as follows. $\mathbf{C}_i^l(n)$ can be evaluated if equation 1 is alternatively written as

$$\mathbf{s}^{l+1} = \overline{\mathbf{w}}^{l+1} \begin{bmatrix} f(\mathbf{s}^l) \\ x_{N_l}^l \equiv 1 \end{bmatrix} \tag{8}$$

which when derivatives with respect to $\mathbf{s}^l$ are taken (the constant value is dropped) yields

$$\frac{\partial \mathbf{s}^{l+1}}{\partial \mathbf{s}^l} = \mathbf{w}^{l+1} \cdot \mathrm{diag}(f'(s_1^l), f'(s_2^l), \ldots, f'(s_{N_l}^l)) \tag{9}$$

which, using the chain rule, yields the recursion

$$\frac{\partial \mathbf{x}^L}{\partial \mathbf{s}^l} = \begin{bmatrix} \frac{\partial x_1^L}{\partial s_1^l} & \cdots & \frac{\partial x_1^L}{\partial s_{N_l}^l} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_{N_L}^L}{\partial s_1^l} & \cdots & \frac{\partial x_{N_L}^L}{\partial s_{N_l}^l} \end{bmatrix} = \frac{\partial \mathbf{x}^L}{\partial \mathbf{s}^{l+1}} \mathbf{w}^{l+1} \cdot$$
$$\cdot \mathrm{diag}(f'(s_1^l), f'(s_2^l), \ldots, f'(s_{N_l}^l)) \tag{10}$$

and initial condition

$$\frac{\partial \mathbf{x}^L}{\partial \mathbf{s}^L} = \mathrm{diag}(f'(s_1^L), f'(s_2^L), \ldots, f'(s_{N_L}^L)) \tag{11}$$

This then enables us to compute the partial derivatives needed in the evaluation of $\mathbf{C}_i^l(n)$

$$\mathbf{C}_{i,j}^l = \frac{\partial \mathbf{x}^L}{\partial w_{i,j}^l} = \left(\frac{\partial \mathbf{x}^L}{\partial \mathbf{s}_i^l}\right) x_j^{l-1} \stackrel{p.d.}{=} \mathbf{\Delta}_i^l x_j^{l-1} \tag{12}$$

or in vector notation for the combined weights leading to node i in layer l

$$\mathbf{C}_i^l = \frac{\partial \mathbf{x}^L}{\partial \overline{\mathbf{w}}_i^l} = \left(\frac{\partial \mathbf{x}^L}{\partial \mathbf{s}_i^l}\right) (\overline{\mathbf{x}}^{l-1})^H \stackrel{p.d.}{=} \mathbf{\Delta}_i^l (\overline{\mathbf{x}}^{l-1})^H \tag{13}$$

Note that $\mathbf{\Delta}_i^l = \frac{\partial \mathbf{x}^L}{\partial \mathbf{s}_i^l}$ actually is a scalar if we assume there to be only one node in the output layer, and $\mathbf{C}_i^l$ correspondingly is a row vector.

Using this recursion, equation 6 can be rewritten as

$$\begin{aligned}\mathbf{G}_i^l(n) = \mathbf{K}_i^l(n-1)\overline{\mathbf{x}}^{l-1}(n)\mathbf{\Delta}_i^l(n)^H \cdot \\ \left[\mathbf{\Delta}_i^l(n)\overline{\mathbf{x}}^{l-1}(n)^H\mathbf{K}_i^l(n-1)\overline{\mathbf{x}}^{l-1}(n)\mathbf{\Delta}_i^l(n)^H + \mathbf{R}_{i,min}^l\right]^{-1}\end{aligned} \tag{14}$$

Closer inspection of this formula reveals that $\overline{\mathbf{x}}^{l-1}(n)^H\mathbf{K}_i^l(n-1)\overline{\mathbf{x}}^{l-1}(n)$ is a scalar; therefore the matrix inversion lemma can be applied to arrive at an update equation for the Kalman gain which only requires a simple division. For the term in brackets this yields

$$\begin{aligned}&\left[\mathbf{\Delta}_i^l(n)\overline{\mathbf{x}}^{l-1}(n)^H\mathbf{K}_i^l(n-1)\overline{\mathbf{x}}^{l-1}(n)\mathbf{\Delta}_i^l(n)^H + \mathbf{R}_{i,min}^l\right]^{-1} = \\ &(\mathbf{R}_{i,min}^l)^{-1} - (\mathbf{R}_{i,min}^l)^{-1}\mathbf{\Delta}_i^l(n) \cdot \\ &\left[\left(\overline{\mathbf{x}}^{l-1}(n)^H\mathbf{K}_i^l(n-1)\overline{\mathbf{x}}^{l-1}(n)\right)^{-1} + \mathbf{\Delta}_i^l(n)^H(\mathbf{R}_{i,min}^l)^{-1}\mathbf{\Delta}_i^l(n)\right]^{-1} \cdot \\ &\mathbf{\Delta}_i^l(n)^H(\mathbf{R}_{i,min}^l)^{-1}\end{aligned} \tag{15}$$

Following [7] and setting $\mathbf{R}_{i,min}^l = \lambda\mathbf{I}(n)$, no matrix inversion is necessary, and with $\alpha_i^l(n) = \overline{\mathbf{x}}^{l-1}(n)^H\mathbf{K}_i^l(n-1)\overline{\mathbf{x}}^{l-1}(n)$ and $\beta_i^l(n) = \mathbf{\Delta}_i^l(n)^H\mathbf{\Delta}_i^l(n)$, equation 15 can finally be rewritten as

$$\begin{aligned}&\left[\mathbf{\Delta}_i^l(n)\overline{\mathbf{x}}^{l-1}(n)^H\mathbf{K}_i^l(n-1)\overline{\mathbf{x}}^{l-1}(n)\mathbf{\Delta}_i^l(n)^H + \mathbf{R}_{i,min}^l\right]^{-1} = \\ &= \frac{1}{\lambda}\left[\mathbf{I} - \mathbf{\Delta}_i^l(n)\frac{\alpha_i^l(n)}{\lambda + \alpha_i^l(n)\beta_i^l(n)}\mathbf{\Delta}_i^l(n)^H\right]\end{aligned} \tag{16}$$

The remaining derivation follows that presented by [7]; the reader is kindly asked to consult their paper.

Figure 1: Optimum decision boundary for a channel with $H(z) = 1 - 2z + 2z^2$, noise variance = 0.5. The small circles and crosses mark the positions which correspond to the set of possible values of $\mathbf{r}(n)$, for $s(n) = s_0$ and $s(n) = s_1$ respectively, when no noise is present.

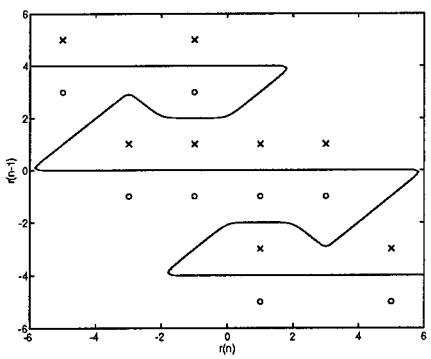
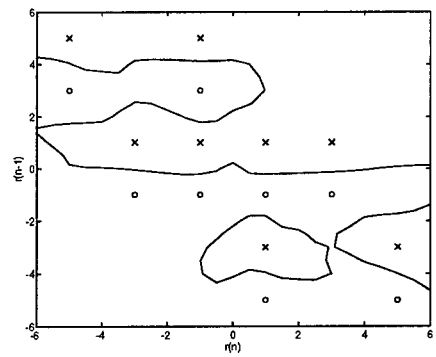


Figure 2: Final mapping for a 2-20-20-1 Kalman network, 50000 iterations, noise variance = 0.5.



COMMUNICATION CHANNEL MODEL

The communication channel is modeled in discrete time (with time index n) and consists of a transmitter producing the binary symbols $\{s_0 = -1, s_1 = 1\}$, a channel which distorts the transmitted signal either linearly or nonlinearly, a noise source which adds[1] statistically independent noise, and a receiver which computes estimates $\hat{s}(n)$ of the transmitted symbols $s(n)$ based on the received symbols $r(n)$.

Since it is assumed that the channel is not intersymbol-interference free, in the most general case the whole received sequence $\mathbf{r}$ must be used to feed an optimal receiver which in turn computes the maximum-a-posteriori (MAP) estimate $\hat{\mathbf{s}}$ of the sequence sent. Using the MAP criterion minimizes the probability of error (see Lee and Messerschmitt, [14]). For most practical purposes, however, the resulting delay between transmission and reception of a message is unacceptable. Thus normally only part (a window) of the received signal sequence is used to compute an estimate for part of the transmitted signal sequence. A common approach is then to use the window $\mathbf{r}(n) = [r(n)\ r(n-1)\ \ldots\ r(n-d+1)]^T$ to compute an estimate $\hat{s}(n-k)$ for $s(n-k)$, where d is the window length and k is the delay allowed between the transmission of the symbol $s(n)$ and the output of its estimate $\hat{s}(n)$.

[1] In fact, it would be possible to consider non-additive interaction of the noise with the signal as well, however in the simulations carried out this was not done.

Figure 3: Averaged error for a 2-20-20-1 Kalman network, 50000 iterations, noise variance = 0.5. Mean over 10 runs.

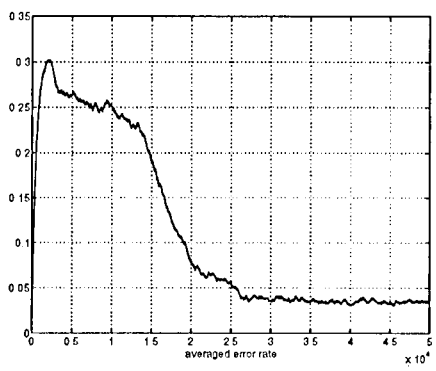
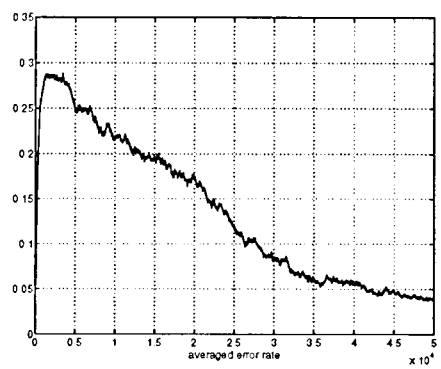


Figure 4: Averaged error for a 2-20-20-1 LMS network, 50000 iterations, noise variance = 0.5. Mean over 10 runs.



Using an input window of length d to produce estimates for the transmitted symbol $s(n-k)$ corresponds to a nonlinear mapping from a d-dimensional input space to a one-dimensional output space. Therefore a neural network can be employed to learn this mapping, which has already been demonstrated by several authors as noted in the introduction. However, using the standard LMS backpropagation training algorithm results in slow convergence. Therefore, Kalman training of the network is being introduced, as described in section . This yields a decrease in the number of training samples required by a factor of 4 to 10, plus an additional decrease in the residual error of the learned decision boundary, when compared to training with the standard LMS backpropagation algorithm.

SIMULATION RESULTS

In order to be able to show results graphically, the input vector dimension was fixed at $d = 2$. The simulated channel transfer function was $H(z) = 1 - 2z + 2z^2$, with the delay paramter k set to zero (i.e. no delay). Gaussian noise with adjustable variance was added at the input to the receiver. This choice of transfer function and delay parameter yields a highly nonlinear decision boundary, as shown in figure 1.

The resulting decision boundary after a total of 50000 training steps for a network trained with the extended Kalman algorithm is shown in figure 2.

Figure 3 shows the plot of the mean of ten runs of the averaged error[2] during training for the same network. It can be seen that after an initial

[2]The averaged error is computed by averaging the symbol error sequence using an exponentially decaying window.

Figure 5: Error sequence for a 2-20-20-1 Kalman network, 50000 iterations, noise variance = 0.2.

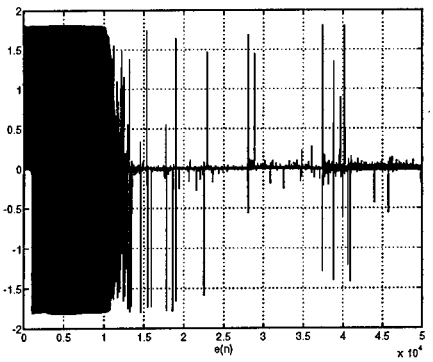
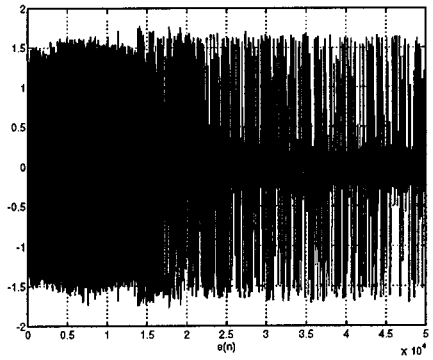


Figure 6: Error sequence for a 2-20-20-1 LMS network, 50000 iterations, noise variance = 0.2.



period of training rapid convergence to the final solution is obtained. Also, the averaged error after convergence is close to the theoretical bound of 0.03.

Figure 4 shows a plot of the averaged error during training of a standard LMS network given the same conditions, again averaged over ten runs. It is clearly visible that convergence is slower. The difference in convergence speed is still greater at lower noise variances or when adding non-Gaussian noise. When e.g. impulse noise of a fixed amplitude occuring with a predefined probability is added, the decision boundary is essentially quadrupled in the two-dimensional input space, thus becoming even more complex (see [15]). Using the Kalman-based training algorithm, a (larger) network converges to an acceptable solution in 50000 training steps, whereas the LMS-trained network needs an excessively high number of iterations.

Finally, figures 5 and 6 show typical plots of the error sequence (the desired output minus the actual output of the network) for Kalman- and LMS-trained networks, respectively. In this case the noise variance used is 0.2, so that the final error is close to zero. It can be seen that after an initial period the Kalman-trained network quite rapidly converges to the optimum decision boundary.

CONCLUSIONS

Based on existing implementations of the Kalman algorithm applied to a feedforward neural network, this paper has presented a simplified derivation of the recursion formulas needed in the operation of the algorithm. The resulting algorithm has then been applied to the task of implementing an optimum receiver for a binary communication channel. It has been shown that the convergence speed of the neural network has improved when

compared to the standard LMS algorithm. Also, after a comparable number of training steps, the Kalman-trained neural network provides a better approximation to the ideal decision boundary than the standard network.

REFERENCES

[1] Gibson, Siu, and Cowan, "The application of nonlinear structures to the reconstruction of binary signals", *IEEE Trans. Signal Proc.*, pp. 1877–1884, August 1991.

[2] Mulgrew and Cowan, "Equalisation techniques using non-linear adaptive filters", in *Adaptive Algorithms: Applications and Non Classical Schemes*, D. Docampo and A. R. Figueras, Eds. Universidad de Vigo, 1991, pp. 1–19.

[3] Chen, Mulgrew, and Grant, "A clustering technique for digital communications channel equalization using radial basis function networks", *IEEE Trans. Neural Networks*, pp. 570–579, July 1993.

[4] Chen, Mulgrew, and McLaughlin, "Adaptive bayesian equalizer with decision feedback", *IEEE Trans. Signal Proc.*, pp. 2918–2927, September 1993.

[5] Al-Mashouq and Reed, "The use of neural nets to combine equalization with decoding", in *Proc. ICASSP*, 1993, vol. I, pp. 469–472.

[6] Ramamurti, Rao, and Gandhi, "Neural detectors for signals in non-gaussian noise", in *Proc. ICASSP*, 1993, vol. I, pp. 481–484.

[7] Iiguni, Sakai, and Tokumaru, "A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter", *IEEE Trans. Signal Proc.*, pp. 959–966, April 1992.

[8] Shah, Palmieri, and Datum, "Optimal filtering algorithms for fast learning in feedforward neural networks", *Neural Networks*, vol. 5, pp. 779–787, 1992.

[9] Singhal and Wu, "Training feed-forward networks with the extended kalman algorithm", in *Proc. ICASSP*, 1989, vol. II, pp. 1187–1190.

[10] Puskorius and Feldkamp, "Decoupled extended kalman filter training of feedforward layered networks", in *Proc. IJCNN*, 1991, vol. I, pp. 771–777.

[11] Palmieri, Datum, and Shah, "Sound localization with a neural network trained with the multiple extended kalman algorithm", in *Proc. IJCNN*, 1991, vol. I, pp. 125–131.

[12] Haykin, *Adaptive Filter Theory*, Prentice-Hall, 1986.

[13] Anderson and Moore, *Optimal Filtering*, Prentice-Hall, 1979.

[14] Lee and Messerschmitt, *Digital Communication*, Kluwer Academic Publishers, 1988.

[15] Birgmeier, "A digital communication channel equalizer using a kalman-trained neural network", submitted to the 1994 IEEE International Conference on Neural Networks.

NEURAL-NET BASED RECEIVER STRUCTURES FOR SINGLE- AND MULTI-AMPLITUDE SIGNALS IN INTERFERENCE CHANNELS*

Dimitrios P. Bouras, *Student Member, IEEE*
P. Takis Mathiopoulos†, *Member, IEEE*
Dept. of Electrical Engineering
University of British Columbia
Vancouver, B.C., V6T 1Z4, Canada
Tel: (604) 822–6942, FAX: (604) 822–5949
E-mail: mathio@ee.ubc.ca

D. Makrakis *Member, IEEE*
Dept. of Electrical Engineering
University of Ottawa
161 Louis Pasteur
P.O. Box 450 STN A, Ottawa
Ontario, K1N 6N5, Canada

Abstract. This paper presents analysis and performance evaluation results for several neural-net based receiver structures which effectively combat additive channel interference, such as co-channel interference (CCI) and adjacent channel interference (ACI). Although the idea of employing neural net based receivers for interference channels is not new, the novel technical contributions of our paper can be summarized as follows. (i) Propose, analyze and evaluate a training algorithm for Nyquist filtered single- and multi-amplitude signals which is based upon a novel non-uniform signal sampling technique. (ii) Propose and evaluate neural net structures employing a novel non-linear activation function for the detection of multi-amplitude signals. (iii) Present novel bit error rate (BER) performance evaluation results for coherent and noncoherent single- and multi-amplitude signals, including binary phase shift keying (BPSK), quadrature phase shift keying (QPSK) and quadrature amplitude modulation (QAM), operated in generalized CCI and ACI channels. Our research has demonstrated that, as compared to more conventional detection techniques, the proposed neural net receivers provide significant performance improvements in CCI and/or ACI channels. Their tolerance for inaccuracies in symbol timing synchronization also makes them good candidates for practical modem implementation.

INTRODUCTION

In recent years, multilayer perceptron neural networks have been extensively applied to many fields in Electrical Engineering, including signal classification, pattern recognition, adaptive control, learning systems, very large scale integration (VLSI) and optimization methods (see for example [1–4] and the references therein). As compared to the aforementioned areas of research, the application of neural networks (or neural nets, as they are often referred to) in communication systems has received relatively little attention. Furthermore, as in this paper we are dealing with the physical layer of digital communication systems, there have been relatively few publications dealing with neural-net based receivers. For example, in [5] a decision feedback equalizer using the multilayer perceptron structure, for equalization in digital communication systems has been investigated. In [6], artificial neural network receivers have been employed for demodulation of spread-spectrum signals in a multiple-access environment. Neural-net based receiver structures for constant envelope continuous phase modulation (CPM) signals transmitted over an additive white Gaussian noise (AWGN) channel have been investigated in [7]. Related work for quadrature-quadrature phase shift keying (Q^2PSK) signals can be found in [8], whereas in [9] a programmable analog VLSI neural network processor designed for communication receivers has been proposed and implemented. There have been also some papers dealing with the application of neural networks for the decoding of error correcting codes (e.g., [10]).

In a recent conference publication [11], it was suggested that combating certain types of channel interference can be achieved by employing neural network techniques. In particular, the authors of [11] have presented some very limited performance evaluation results (see [11, Fig. 3]) of a neural-net based receiver for a Butterworth filtered binary digital communication

* This work has been supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants OGP-44312 and STR-0100720, the Centre for Integrated Computer Systems Research (CICSR), a University of British Columbia Graduate Fellowship and a B.C. Advanced Systems Institute (ASI) Fellowship.

† Please address any correspondence to this author.

system operating in the presence of one co-channel interferer. Their results have indicated that improvements in the bit error rate (BER) can be obtained by employing a multi layer perceptron (MLP) neural network, trained for this purpose under co-channel interference (CCI) channel conditions, with the aid of the backpropagation learning algorithm. There are several limitations on the work reported in [11]. It is well known that for bandwidth efficient digital communication systems, Nyquist type filters must be employed [12]. Furthermore, in order to increase system capacity, more bandwidth efficient modulation schemes (as compared to binary signalling), using multi-amplitude signal constellations, are required. It should also be pointed out that with the recent very rapid developments in the field of wireless personal communications, CCI [13], and to a lesser degree adjacent channel interference (ACI) [14], have become the main source of performance degradation. A digital communication system which can tolerate any amount of CCI and ACI, while providing reliable transmission, is of great interest since it automatically translates to higher capacity for the existing network.

Motivated by the above, in this paper we present analysis and performance evaluation results for neural-net based receiver structures, for single-amplitude modulation formats, such as binary-phase-shift-keying (BPSK) and quadrature-phase-shift-keying (QPSK), as well as multi-amplitude schemes, namely 4-signal pulse-amplitude-modulation (4-PAM) and 16-signal quadrature-amplitude-modulation (16-QAM), which effectively combat additive channel interference, such as CCI and ACI, as well as additive-white-Gaussian-noise (AWGN). More specifically we (i) Propose, analyze and evaluate a training algorithm for Nyquist filtered single- and multi-amplitude signals utilizing a novel non-uniform signal sampling technique. Problems related to training under certain channel conditions are also addressed, both for single- and multi-amplitude modulation schemes, and both types of signal filters, Nyquist and Butterworth, (ii) Propose and evaluate neural net structures employing a novel non-linear activation function for the detection of multi-amplitude signals. Although presented for the 4-PAM and 16-QAM schemes, the derivation of this function is readily extendable to modulation formats employing more than 4 signal levels, and (iii) Present novel bit error rate (BER) performance evaluation results for coherent and noncoherent single- and multi-amplitude signals operated in channels including CCI or ACI and AWGN. These include cases of single and multiple co-channel interferers, one or two adjacent channel interferers, differentially and non-differentially encoded signal constellations, and Nyquist or Butterworth signal filtering.

COMMUNICATION SYSTEM MODEL

The transmitter of the communication system under consideration consists of a signal mapper (SM), an optional differential encoder (DE), a pre-modulation pulse-shaping filter with transfer function $H_T(f)$, and a complex modulator. The input to the signal mapper consists of the N-bit information words $\overline{a_k^N} = \left[a_k^1, a_k^2, ..., a_k^N\right]$ of independent and equiprobable bits a_k^i, $1 \leq i \leq N$ from the alphabet $\{0, 1\}$. Each $\overline{a_k^N}$ is converted by the SM to a symbol $u_k = R_k \exp(j\Omega_k)$, where R_k represents the amplitude and Ω_k the phase of u_k, respectively. Possible differential encoding of the sequence of u_k's yields the sequence of differentially encoded symbols c_k. For example, QAM signals are encoded differentially as

$$c_k = u_k \frac{c_{k-1}}{R_{k-1}} = R_k \exp\left[j(\Phi_{k-1} \oplus \Omega_k)\right] \tag{1}$$

with Φ_k denoting the phase of c_k, R_k its amplitude and $\oplus$ modulo 2π addition. If differential encoding is not employed, we assume that $c_k = u_k$. After being passed through the pulse-shaping filter $H_T(f)$ and translated to the carrier frequency f_c by the modulator, the transmitted signal can be expressed as

$$x(t) = \mathrm{Re}\left\{ s(t) = \left[\sum_{k=-\infty}^{\infty} c_k h_T(t - kT) \right] exp(j2\pi f_c) \right\} \tag{2}$$

with $h_T(t)$ denoting the impulse response of $H_T(f)$ and T the symbol interval.

A block diagram for the channel and receiver front-end model assumed is illustrated in Fig. 1. Although the number of co-channel interferers in this model is not restricted to a particular number, in the bit-error-rate (BER) evaluation of the neural-net based receiver structures derived (see Section 4), one and three co-channel interferers are assumed. The ACI is assumed to be generated by users occupying frequencies on either side of the channel under consideration. For the computer simulation results presented in Section 4, either one interferer is assumed at frequency $f_c + (1/T)$, or two interferers at $f_c \pm (1/T)$.

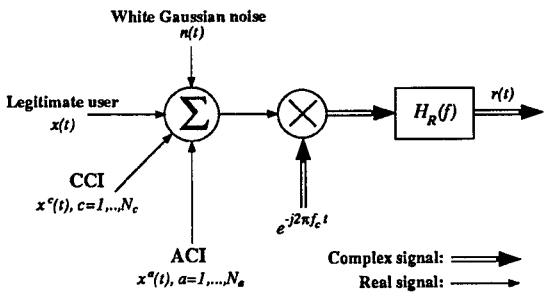


Fig. 1. Block diagram of the channel model and receiver front-end assumed; $H_R(f)$: receiver pre-detection filter.

The signal, after being distorted by CCI or ACI and additive-white-Gaussian noise (AWGN), is demodulated at the receiver. Assuming a receiver pre-detection filter transfer function $H_R(f)$ corresponding to an impulse response of $h_R(t)$, and using $h(t) = h_T(t) \otimes h_R(t)$ with $\otimes$ denoting convolution, the received demodulated signal for N_C independent co-channel interferers and N_A independent adjacent-channel interferers can be expressed as

$$r(t) = \sum_{k=-\infty}^{\infty} c_k h(t) + \sum_{l=1}^{N_C} \sum_{k=-\infty}^{\infty} e^{-j\theta_l^C} c_{l,k}^C h\left(t - lT - \tau_l^C\right) + \sum_{l=1}^{N_A} \sum_{k=-\infty}^{\infty} e^{-j\left(2\pi\left(f_c - f_l^A\right)t + \theta_l^A\right)} c_{l,k}^A h\left(t - lT - \tau_l^A\right) + n(t) \tag{3}$$

with the first term in the above equation expressing the legitimate user signal, the second term the CCI, the third the ACI and the fourth, the AWGN. In Eq. 3, c_l^C, θ_l^C and τ_l^C represent the co-channel interferer data stream, carrier phase shift and symbol timing delay respectively. The same symbols but with a superscript of A are used for the adjacent channel interferer. Note, however, that the phase shift for ACI is continuously increasing by $2\pi(f_c - f_l^A)t$ due to the frequency difference between the user carrier f_c and the l'th adjacent channel interferer carrier f_l^A. In general, we consider that θ_l^C and θ_l^A are independent random variables, uniformly distributed over $[0, 2\pi)$. Also, τ_l^C and τ_l^A are independent random variables uniformly distributed over $[0, T)$. As far as receiver performance is concerned, the parameters of interest are the signal-to-noise power ratio (SNR) and the signal-to-interference power ratio (SIR) at the output of the receiver filter. Before feeding the demodulated signal to the receiver, we sample it at a number of "appropriate" time instances. Depending on the sampling rate we can obtain any desired number of samples per symbol interval.

NEURAL-NET RECEIVER STRUCTURE

The neural-net receiver used for implementing a detector for a single- or multi-level scheme falls in the class of time delay neural networks (TDNN) [15]. According to this configuration, an otherwise static network is processing data arranged as a series of samples obtained from the incoming signal at specific time intervals. The total length of time corresponding to all samples appearing as inputs to the TDNN will henceforth referred to as a sample or data "window". In our case, the data window length will be an odd multiple of the symbol period, the symbol of interest (or current symbol) located at the middle of the window. The number of signal samples in this window is equal to the number of neural network inputs. The network is of the MLP type, having L layers and N_l number of neurons (or perceptrons) [16] in layer l, $1 \leq l \leq L$. The number of neurons on the input layer for all neural-net receivers presented here is determined by (as it is equal to) the desired number of samples in the data window. The MLP's employed are all fully interconnected[3] and possess three layers, the input, output and one hidden layer. Note however that every neuron on the input layer is connected only to its corresponding input. For notational purposes, such a MLP will be henceforth referred to by three numbers in parentheses, namely the number of

[3] A fully interconnected MLP is that having every input of any neuron in a given layer connected to all outputs in the previous layer, or to the inputs, if we are considering the input layer.

neurons in each layer, i.e., (N_1, N_2, N_3). For our study, 3-layer MLP's were chosen although a 2-layer MLP has been shown capable of forming an arbitrarily close approximation to any non-linear decision boundary [17]. The reason is that, for a given problem, 3 layers typically result in much smaller neural net size as compared to the equivalent 2-layer MLP's [18].

Neural-net receiver training

The training method employed is the backpropagation learning algorithm. A small number of known inputs and outputs consists the *training set*. Verification of the generalization capabilities of the network is performed using a new, randomly generated, *verification set*, after the initial training phase has reduced the RMS error to a value below an acceptable level. If the RMS error for this new set is sufficiently close to that obtained for the training set, network learning is done. Else, the training parameters and/or the network configuration is changed, and the training process is restarted. All weights are initialized to small random values before the net teaching process begins. This provides the algorithm with a relatively "safe" starting point [19].

The signal after the receiver filter, as given in Eq. 3, is sampled at a specified rate with respect to the symbol rate used. This yields a number of samples from which the decision device must recover the actual information transmitted. Two cases were considered for the transmit and receive filters: i) both $H_T(f)$ and $H_R(f)$ being 5-pole Butterworth having a 3 dB corner frequency equal to $1/T$, and ii) $H_T(f)$ being a $x/\sin(x)$ equalized $\sqrt{a}$ Nyquist filter and $H_R(f)$ a $\sqrt{a}$ Nyquist filter, both of excess bandwidth $a = 1.0$. The coherent receivers, against which the neural-net based ones are compared, decide upon the symbol transmitted by observing the value of a single sample, at the middle of the symbol interval; the noncoherent receivers do the same but operating on the output of a 1-symbol differential detector. The neural-net receiver, on the other hand, employs an observation window spanning over an odd number of symbol intervals, the middle one of which is assumed to be the "present" symbol for which the decision will be made. This essentially translates to a time-lag in decoding at the receiver equal to the number of "future" samples in the observation window. The observation window size employed was 7 symbol-periods long, 3 symbols on either side of the symbol detected. For BPSK and QPSK the received signal was sampled at twice the symbol rate yielding 14 signal samples for processing by the neural net. For 4-PAM and 16-QAM, frequencies of 2 and 4 times the symbol rate were used for sampling, yielding 14 and 28 signal samples respectively.

The neural network used for BPSK and QPSK has 14 neurons, equal to the number of samples available (7 symbols, 2 samples per symbol). The hidden layer has 5 and the output layer has 1 neuron. The same net is used to process the I- and Q-channel samples alternatively. This is possible since the inphase and quadrature channels carry independent BPSK signals [12]. It's training involves choosing at random a relatively small number of 7-symbol sets for the user and each interferer. For BPSK and QPSK, 128 such groups of 7 symbols were chosen, whereas for PAM and QAM, due to the much higher number of possible signal combinations, this number was increased to 1024. Each interferer group is scaled according to the given SIR, and then added to the corresponding user signal group in order to distort it. A very important issue while training the net is the minimum usable SIR value, which depends on the number of signal levels of the modulation format under consideration. At SIR slightly below this minimum value, the interfering signal will distort the user signal to such an extent as to have the combined signal level cross decision boundaries on the signal constellation. This, in turn, confuses the neural net by essentially presenting it with randomized "lessons" it is unable to "learn" from. As an example, for BPSK and QPSK, this minimum SIR value is 0 dB. Since, in this case, user and interferer have equal power, the interferer can cause the combined signal level to be around 0 (e.g., assuming signal power normalized to 1, user = 1, interferer = -1) which also happens to be the decision threshold. This confuses the neural net and prevents the search towards weight and offset values for minimum RMS error from converging. As a last note on the neural net receiver training, results were deemed acceptable if the RMS error calculated when estimating the network generalization performance was at most 10% over that obtained during training.

A new non-linear activation function for multi-level schemes

For the 4-level 4-PAM and 16-QAM (two independent 4-PAM channels) schemes, 2 net configurations were investigated. Both observe a 7-symbol signal window, one sampling at twice and the other at four times the symbol rate. This yields 14 signal samples for the first case and 28 signal samples for the second. The number of hidden layer nodes is equal to

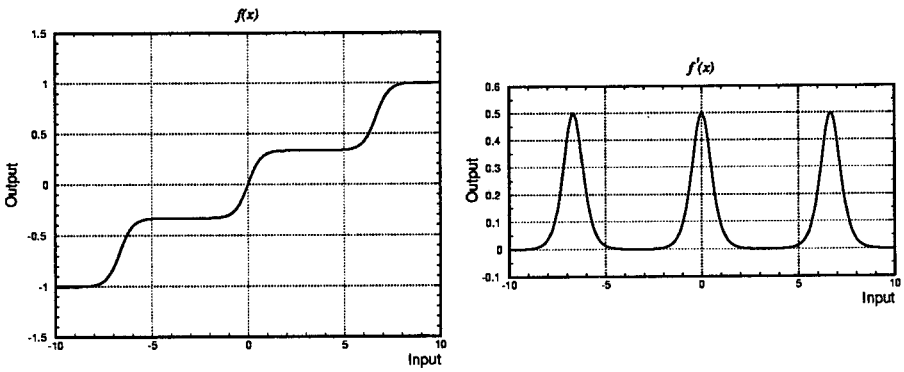


Fig. 2. The four-level sigmoid function used for 4-PAM and 16-QAM and its derivative.

7, as compared to 5 used for the single level schemes, in order to provide better network generalization[4] and shorter convergence time during training. In order to accommodate the four signal levels, the output non-linearity had to be modified. For this purpose a new function was constructed by using scaled and shifted versions of the well known sigmoid non-linearity. This new function $f_3(x)$ is given by Eq. 4

$$
\begin{aligned}
g(x) &= \frac{1}{1+e^{-\beta x}} \quad , \quad g'(x) = \beta g(x)[1-g(x)] \\
f_3(x) &= \left[-1+\frac{2}{3}g\left(x+\frac{20}{3}\right)\right] + \left[-\frac{1}{3}+\frac{2}{3}g(x)\right] + \left[\frac{1}{3}+\frac{2}{3}g\left(x-\frac{20}{3}\right)\right] \qquad (4) \\
f_3'(x) &= \frac{2}{3}g'\left(x+\frac{20}{3}\right) + \frac{2}{3}g'(x) + \frac{2}{3}g'\left(x-\frac{20}{3}\right).
\end{aligned}
$$

The β factor determining the steepness of each individual "step" in the function is set equal to 3. $f_3(x)$ and $f_3'(x)$, plotted for $-10 \leq x \leq 10$ are depicted in Fig. 2. Note that the same technique can be used for constructing non-linear activation functions for neural-net based receivers designed for other multi-level modulation formats.

For a 4-PAM signal, the minimum SIR at which we can train the neural network is 10 dB, for the same reason that we can't train the net for BPSK at a SIR of less than 0 dB. In order to understand the 10 dB limitation, assuming maximum signal power normalized to 1, consider a case where the user level is -1/3 while the interferer is +1. Assuming a SIR of 10 dB we imply that the interferer power is 1/10 with respect to the user. As far as signal amplitude levels are concerned this translates to $1/\sqrt{10} \simeq 0.3162$ which is very close to 1/3, effectively bringing the resulting signal level to approximately 0. It is important to note here that due to filtering, the signal levels are not constant; they fluctuate before and after each level transition. Hence the amplitude in the aforementioned case will fluctuate around 0. This creates the same effect as in the case of SIR = 0 dB for BPSK, confusing the neural net and preventing the training process from converging.

TRAINING AND BER PERFORMANCE RESULTS

This section presents results from training of specific neural net structures, and BER performance evaluation results for single- and multi-level modulation schemes employing the trained neural nets, obtained via computer simulation. Since Monte-Carlo error counting techniques were employed, the BER results presented cover error rates down to 10^{-4}, due to memory and time limitations in the simulation. The digital simulation also introduced a finite resolution on the signal representations, namely a number of signal samples per symbol (SPS). When employing Butterworth filtering, SPS was set to 8, while with Nyquist filters, 16 samples per symbol were employed. The discrete time simulation was carried out in baseband and perfect symbol timing synchronization was assumed for the coherent receivers against which the neural-net based structures were tested. Note that, although the results presented for the neural-net receivers also assume perfect symbol synchronization, simulations have

[4] By better network generalization we imply that smaller RMS error for a random new signal set can be obtained after training is completed.

shown that symbol timing errors of up to approximately 10-15% have negligible effects on their BER performance.

Single-amplitude schemes

As mentioned in the previous section, for BPSK and QPSK a (14,5,1) neural net was employed. Training in a single interferer CCI environment, at an SIR of 3 dB, with $\mu = 0.8$ and $\xi = 0.01$, resulted in an RMS error of approximately 0.005 in 11800 iterations. The interferer symbol timing delay τ_1^C was random (uniformly distributed over one symbol duration T), while its carrier phase offset θ_1^C was set to 0[5]. The BER performance of neural-net (NN) assisted BPSK (NN-BPSK) as a function of the SNR ratio, as compared to a coherent BPSK scheme, both operating in the aforementioned CCI environment, is illustrated in Fig. 3. Note that the performance shown also holds for NN-QPSK versus coherent QPSK. The filters employed in this case are of the Butterworth type. The gain in performance, for an operating SIR of 3 dB and a BER level of 10^{-2} is approximately 5 dB. For an SIR of 5 dB at BER of 2×10^{-4} it is approximately 4 dB and for SIR equal to 7 dB, at the same BER level it's around 4.3 dB.

Fig. 4 illustrates the performance of a (14,5,1) neural net used for a Nyquist filtered BPSK scheme. The excess bandwidth $\alpha = 1.0$, the learning $\mu = 0.8$, the momentum gain $\xi = 0.012$ and SIR during training equal to 3 dB. In this figure, the performance is plotted as a function of the E_s/N_0, E_s denoting the signal energy per transmitted symbol and N_0 the noise one-sided power spectral density of the AWGN $n(t)$. The gains in performance when the signals are Nyquist filtered, are somewhat smaller than the case where Butterworth filters are employed, but nevertheless still significant. At a BER level of 10^{-3}, for SIR = 3 dB the gain is approximately 3.9 dB, for SIR = 5 dB it's 3.6 dB, and for SIR = 7 dB and 9 dB, approximately 3.3 dB. As it is intuitively expected, the gain provided by the neural-net receiver will decrease as the SIR is increased. Note, however, that it remains roughly within 10% with an increase in SIR of approximately 50%. A (14,7,1) neural net was also trained and evaluated under the same conditions but there was no observable gain in performance with this increased number of hidden layer nodes. For the results of Fig. 4, the non-uniform signal sampling technique was employed. Using SPS = 16 for the digital simulation, only two center samples from each symbol period were used as input to the neural net, both in the training and evaluation phase.

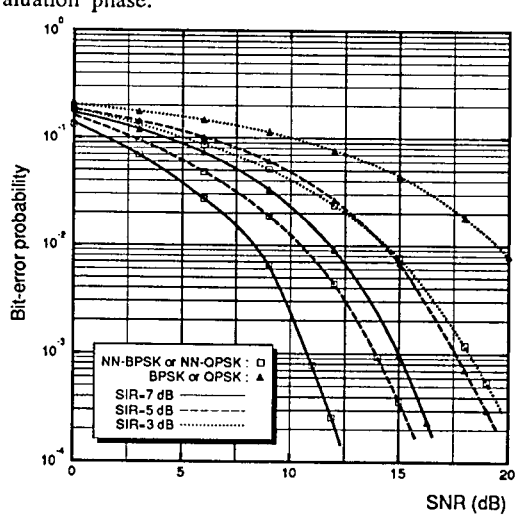


Fig. 3. Performance of Butterworth filtered NN-BPSK employing a (14,5,1) MLP, versus coherent BPSK in combined CCI and AWGN; 1 interferer with random symbol timing delay τ_1^C (uniform over $[0, T)$) and 0 carrier phase offset θ_1^C.

[5] Note that although this is not a realistic assumption, for one-dimensional schemes (e.g., BPSK), carrier offset equal to 0 for the interfering signals is indeed the worst case. Any other phase offset value will result in reduced interference amplitude since it will be scaled down by $\cos\left(\theta_1^C\right)$. However, this is not the case for two-dimensional schemes.

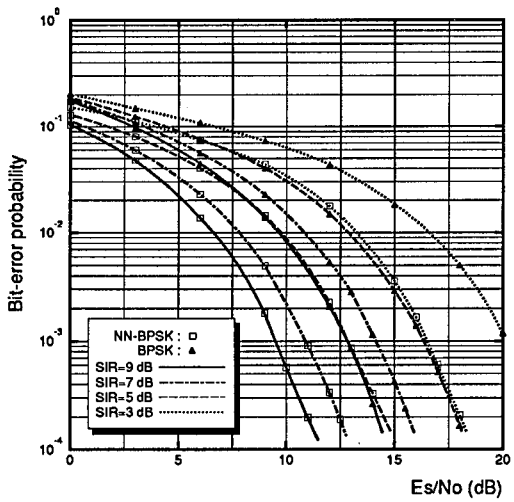


Fig. 4. Performance of Nyquist filtered (α = 1.0) NN-BPSK employing a (14,5,1) MLP, versus coherent BPSK in combined CCI and AWGN; 1 interferer with random symbol timing delay τ_1^C (uniform over $[0, T)$) and carrier phase offset $\theta_1^C = 0$.

The same neural-net structure was also trained and evaluated in a 3-interferer CCI environment (i.e., $N_C = 3$), using BPSK. Note that when having 3 interferers instead of 1, the minimum SIR level at which the network can be trained is less than the 3 dB used for the single interferer case and. As expected, this number depends on the number of interferers considered and can be found by taking the worst case of aggregate interference, i.e., all interferers having the same signal level, with sign opposite to that of the legitimate user. Assuming user power equal to 1, this case would correspond to a user level of +1 and all three interfering signal levels equal to -1/3. As we assumed statistically independent interferers[6], these numbers correspond to a SIR of 5 dB. For the aforementioned neural net structure, the SIR employed during the training phase was equal to 7.5 dB. This somewhat higher value accounts for the signal envelope fluctuations due to filtering, providing training patterns with no level crossings. The BER evaluation results for 3 co-channel interferers, training parameters of $\mu = 0.7$ and $\xi = 0.01$, are depicted in Fig. 5. It can be seen that the gain in performance, for SIR = 7 and 9 dB, is less than the single interferer case (at a BER level of 10^{-3}), but still a respectable 2.8 dB, as compared to the coherent BPSK case. The reason for this gain reduction is attributed to the fact that the co-channel interference appears more and more like noise, as the number of interferers N_C is increased. This, in turn, prevents the receiver from taking advantage of the neural-net pattern classification capabilities, with the limiting case being AWGN, where no additional gain is available.

The (14,5,1) neural-net receiver structure was also evaluated in an ACI environment consisting of a single interferer having $f_1^A = f_c + 1/T$. For Butterworth filtered signals, training was carried out with $\mu = 0.5$ and $\xi = 0.2$ at SIR = 1.5 dB, yielding an RMS error of about 0.01 after 6600 iterations. For Nyquist filters (α = 1.0), $\mu = 0.4$ and $\xi = 0.008$ at SIR = 6 dB, yielded an RMS error of 0.01 after 7100 iterations. The much higher value of SIR employed for the Nyquist case is due to the high signal fluctuation on the adjacent channel signal after the receiver filter. This is specific to the root-of-raised-cosine Nyquist filter; it's not the case with Butterworth filter employed in the previous case. Results for the Nyquist filtered case are illustrated in Fig. 6; at BER = 10^{-3} and SIR = 6 dB, the performance gain is approximately 3.2 dB. Note the error floor due to decision-level crossings caused by the aforementioned signal amplitude fluctuations after the receive filter, at SIR = 3 dB.

The same (14,5,1) neural net, trained with single interferer and Nyquist filtering, was also evaluated in an ACI environment with 2 adjacent channel interferers, one at $f_1^A = f_c + 1/T$ and the other at $f_2^A = f_c - 1/T$, with τ_1^A, τ_2^A uniformly distributed over one symbol duration

[6] When the interfering signals can be assumed statistically independent, the aggregate signal power is simply equal to the sum of individual powers.

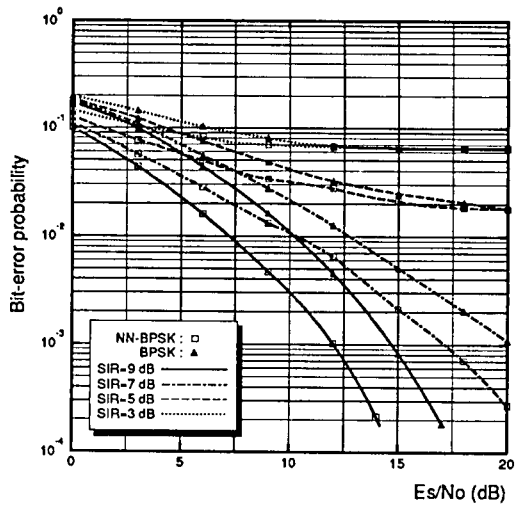


Fig. 5. Performance of Nyquist filtered (α = 1.0) NN-BPSK employing a (14,5,1) MLP, versus coherent BPSK in combined CCI and AWGN; 3 interferers with independent symbol timing delays $\tau_1^C, \tau_2^C, \tau_3^C$ (uniform over $[0, T)$) and carrier phase offsets $\theta_1^C = \theta_2^C = \theta_3^C = 0$. T and θ_1^A, θ_2^A uniformly distributed over $[0, 2\pi)$. The results presented in Fig. 7, show approximately 3.5 dB gain for both SIR = 10 and 12 dB. The error floors are more noticeable in this case than that of Fig. 6 since there are now two low-frequency, modulated sinusoids distorting the baseband legitimate user signal, each one belonging to one of two adjacent channel interferers.

Multi-amplitude schemes

Neural-net structures were employed for the 4-PAM and 16-QAM schemes, operated in a single interferer CCI environment, with τ_1^C uniformly distributed over one symbol duration T and θ_1^C equal to 0. The two net structures investigated were (14,7,1) and (28,7,1). The latter processes a signal window having double the number of samples as compared to the

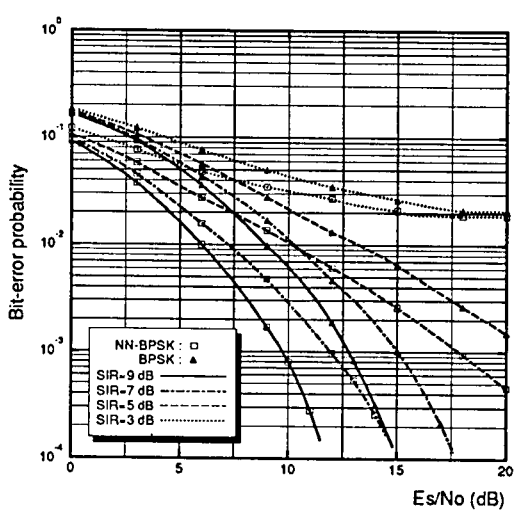


Fig. 6. Performance of Nyquist filtered (α = 1.0) NN-BPSK employing a (14,5,1) MLP, versus coherent BPSK in combined ACI and AWGN; 1 interferer with symbol timing delay τ_1^A (uniform over $[0, T)$) and carrier phase offset $\theta_1^A = 0$.

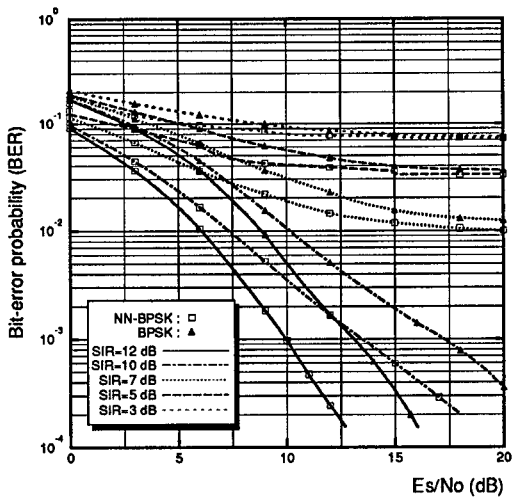


Fig. 7. Performance of Nyquist filtered (α = 1.0) NN-BPSK employing a (14,5,1) MLP, versus coherent BPSK in combined ACI and AWGN; 2 interferers with symbol timing delays τ_1^A, τ_2^A (uniform over $[0, T)$) and carrier phase offsets θ_1^A, θ_2^A (uniform over $[0, 2\pi)$).

former; namely 28 signal samples versus 14. The neural net receivers were trained at SIR = 13 dB and 15 dB, the minimum being equal to 10 dB, as explained in Section 2. The learning rate was set to $\mu = 0.7$ and the momentum gain to $\xi = 0.2$. These relatively large values help the algorithm advance quickly in the first tens of thousands of iterations but do not work well once the RMS error has dropped to a relatively low value. For this purpose, both parameters were halved when the error would drop below 0.1 and 0.05. Thus, most of the training was performed using $\mu = 0.175$, $\xi = 0.05$. The convergence time is quite longer than for the (14,5,1) net used with BPSK and QPSK, and the RMS error change is not as smooth. It falls below 0.1 at approximately 45150 iterations and bellow the target value of 0.005 at approximately 99000 iterations. For the 28-input net the number of iterations rises to approximately 568000.

The BER performance of a Butterworth filtered 16-QAM scheme is illustrated in Fig. 8.

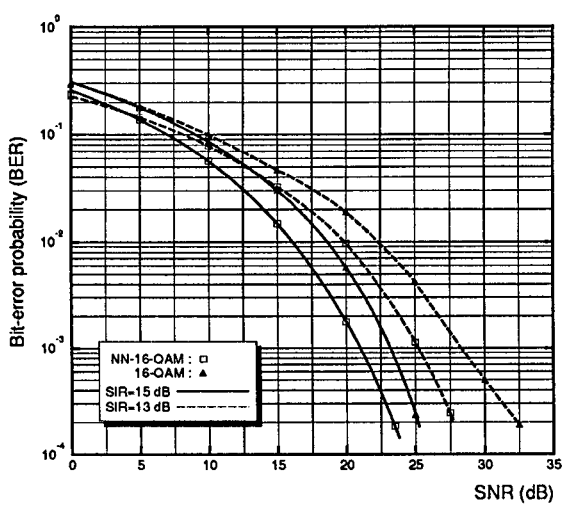


Fig. 8. Performance of Butterworth filtered neural-net assisted 16-QAM (NN-16-QAM) employing a (14,7,1) MLP, versus coherent 16-QAM, in combined CCI and AWGN; 1 interferer with symbol timing delay τ_1^C uniform over $[0, T)$

Results are presented for a neural-net based receiver employing a (14,7,1) net, the training for which was carried out at the SIR values indicated on the figure, namely 13 and 15 dB. The same values were used during evaluation. At a BER level of 10^{-3}, for SIR = 13 dB the gain with respect to coherently detected 16-QAM is approximately 3.3 dB, while for the SIR = 15 dB, it falls down to 2.4 dB. Increasing the number of input samples processed by the net does indeed have a positive effect on the gain. The (28,7,1) neural-net based receiver, operating at a SIR level of 13 dB, yields approximately an additional 1.9 dB of gain, at BER = 10^{-3}. This makes the overall gain with respect to the coherent receiver approximately equal to 5.3 dB.

REFERENCES

[1] R. P. Lippman, "An introduction to computing with neural nets," *IEEE Acoustics, Speech and Signal Processing Magazine*, pp. 4(2):4–22, April 1987.

[2] J. A. Freeman and D. M. Skapura, *Neural Networks: Algorithms, Applications and Programming Techniques*. Reading, MA: Addison-Wesley, 1991.

[3] D. B. Fogel, *System Identification through Simulated Evolution: A Machine Learning Approach to Modelling*. Needham Heights, MA: Ginn Press, 1991.

[4] N. Alon, A. K. Dewdney, and T. J. Ott, "Efficient simulation of finite automata by neural nets," *Journal of the Association of Computing Machinery*, pp. 38(2):495–514, 1991.

[5] S. Siu, G. J. Gibson, and C. F. N. Cowan, "Decision feedback equalization using neural network structures and performance comparisson with standard architecture," *IEEE Proceedings*, vol. 137, Part I, pp. 221–225, Aug. 1990.

[6] B. Aazhang, B.-P. Paris, and G. C. Orsak, "Neural networks for multiuser detection in code-division multiple-access communications," *IEEE Trans. Commun.*, vol. COM-40, pp. 1212–1222, July 1992.

[7] G. d. Veciana and A. Zakhor, "Neural net-based continuous phase modulation receivers," *IEEE Trans. Commun.*, vol. COM-40, pp. 1396–1408, Aug. 1992.

[8] S. Feiz and S. S. Soliman, "Adaptive ml neural network based receiver for Q^2PSK modulated data-transmission systems," in *Proc. of the 39th Vehicular Technology Conference*, pp. 263–269, May 1989.

[9] J. Choi, S. H. Bang, and B. J. Shev, "A programmable analog VLSI neural network processor for communication receivers," *IEEE Trans. Neural Networks*, vol. NN-4, pp. 484–495, May 1993.

[10] W. R. Caid and R. W. Means, "Neural network error correcting decoders for convolutional codes," in *Proc. of GLOBECOM'90*, pp. 1028–1031, Dec. 1990.

[11] H. M. Hafez and G. K. Chan, "Interference reduction using neural networks," in *Proc. of the 3rd Canadian Conf. on Electrical and Computer Engineering, Quebec, Canada*, pp. 33.1.1–33.1.4, Sept. 1991.

[12] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, 1983.

[13] T. S. Rappaport, "The wireless revolution," *IEEE Communications Magazine*, pp. 52–71, Nov. 1991.

[14] L. B. Milstein, R. L. Pickholtz, and D. L. Schilling, "Comparison of performance of digital modulation techniques in the presence of adjacent channel interference," *IEEE Trans. Commun.*, pp. 1984–1993, Aug. 1982.

[15] J. Hertz, A. Krogh, and R. G. Palmer, *Introductions to the Theory of Neural Computation*. Addison-Wesley, Redwood City, CA, 1987.

[16] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65:386–408, 1958.

[17] M. J, A. El-Jaroudi, and R. Schwartz, "Formation of disconnected decision regions with a single hidden layer," in *Proc. of the International Joint Conference on Neural Networks, volume 1*, pp. 455–460, 1989.

[18] D. L. Chester, "Why two hidden layers are better than one," in *Proc. of the International Joint Conference on Neural Networks, volume 1*, pp. 265–268, 1990.

[19] D. R. Hush, J. M. Salas, and B. G. Horne, "Error surfaces for multi-layer perceptrons," *IEEE Trans. on Systems, Man and Cybernetics*, 22(5) 1992.

A HYBRID DIGITAL COMPUTER - HOPFIELD NEURAL NETWORK CDMA DETECTOR FOR REAL-TIME MULTI-USER DEMODULATION

George I. Kechriotis and Elias S. Manolakos

COMMUNICATIONS AND DIGITAL SIGNAL PROCESSING (CDSP)
CENTER FOR RESEARCH AND GRADUATE STUDIES

Electrical and Computer Engineering Department
409 Dana Research Building
Northeastern University, Boston, MA 02115
e-mail: elias@athina.cdsp.neu.edu george@cdsp.neu.edu

Abstract - We propose a hybrid digital computer-neural network multi-user detector whose small computational complexity makes it attractive for real-time CDMA detection. Theoretical results on the nature of the local minima of the Optimal Multi-User Detector (OMD) objective function are summarized, and a method that leads to a significant reduction on the size of the optimization problem to be solved is outlined. The preprocessing problem size reduction stage is followed by a Hopfield Neural Network employed to solve the irreducible (residual) problem. The performance of the proposed detector is evaluated via simulations and it is shown to exceed that of other suboptimal schemes at a much lower computational cost.

INTRODUCTION

Code Division Multiple Access (CDMA) is rapidly emerging as a spectrum efficient method of choice for the simultaneous transmission of digital information sent by multiple users over a shared channel. The spectral efficiency as well as the anti-jamming and other attractive properties make CDMA Spread Spectrum techniques useful in a number of communication technologies such as cellular and mobile telephony and

0-7803-2026-3/94 $4.00 © 1994 IEEE

satellite communications. The major limitation of the CDMA techniques however, is the so called *near-far* problem: When the power of the signals transmitted by the users becomes very dissimilar the conventional matched-filter detector exhibits severe performance degradation, so that more complicated detectors have to be employed.

It has been shown by Verdu et al. [1] that Optimal CDMA Multiuser Detection (OMD) can be formulated as the solution to a quadratic integer programming problem that is NP-complete. Therefore research efforts have focused on deriving suboptimal schemes that are near-far resistant and achieve near-optimal Bit-Error-Rate (BER) performance. Among those reported in the literature we mention the *multistage detector* (MD) proposed by Aazhang et al. [2], the *decorrelating* detector by Verdu et al. [3], as well as the Viterbi based sequential decoding algorithms in [4].

Recently in [5], Aazhang et al. showed that a multi-layer perceptron can be trained to approximate the OMD discriminant function at a very small performance loss relatively to the OMD. In [6], the authors of this paper showed how Hopfield Neural Networks (HNNs) [7], can be employed to solve the same problem with considerable performance gains over the conventional detector. However both neural networks based receivers suffer from scalability problems. In the feedforward neural network case the number of neurons increases exponentially with the number of the users and so does the training time. The problem is not that severe in the case of the HNN receiver where the number of interconnections increases only with the square of the number of users.

Since with currently available technology only relatively small size neural networks can be manufactured, *hybrid* schemes that take advantage of both digital signal processing and neural network based approaches at a much smaller computational and hardware cost seem to be the most attractive alternative. In this paper we propose a novel detector that employs a digital computer (post-processing of the outputs of the conventional detector) stage reducing the size of the OMD optimization problem, with a small size HNN employed to solve a remaining (irreducible) problem of the same form as the OMD.

BACKGROUND

In Spread-Spectrum CDMA a number of users share a communication channel by transmitting information modulated by different *signature* waveforms (codes) $s_k(t)$. At the receiver's end, the signal is the superposition of all the individual transmitted signals and additive channel noise:

$$r(t) = \sum_{i=-P}^{P} \sum_{k=1}^{K} b_k^{(i)} s_k(t - iT - \tau_k) + n(t), \quad t \in \mathbf{R} \tag{1}$$

where $s_k(t)$ is the signature waveform of the kth user which is assumed to be time-limited to the interval $[0,\ T]$, $\tau_k \in [0,\ T)$ are the relative time delays between the users, $b_k^{(i)} \in [-1, +1]$ is the ith information bit transmitted by the kth user, and $2P+1$ is the packet length. Moreover, $n(t)$ is additive zero mean Gaussian channel noise.

The objective of multi-user CDMA detection is to recover the information bit streams of all the users from the received signal $r(t)$. The *conventional* CDMA receiver, consists of a bank of filters matched to the signature waveforms of the users, followed by a threshold decision logic. In the asynchronous CDMA case, the matched filter output corresponding to the ith bit of the kth user becomes:

$$y_k^{(i)} = \int_{iT-\tau_k}^{iT-\tau_k+T} r(t) s_k(t - iT - \tau_k) dt, \quad \text{for } k = 1, \ldots, K \tag{2}$$

The decisions on the ith information bit of the kth user are made according to the sign of $y_k^{(i)}$, i.e. $\hat{b}_k^{(i)} = sign(y_k^{(i)})$. In the presence of severe near-far problems (i.e. when the energies of the users are very dissimilar) the performance of the conventional detector degrades severely. The *optimal multiuser detector* (OMD) on the other hand is near-far resistant, and can be formulated for the most general asynchronous case as the solution of the following quadratic integer optimization problem [3]:

$$\mathbf{b}_{opt} = \arg \max_{\mathbf{b} \in \{+1,\ -1\}^{(2P+1)K}} \{2\mathbf{y}^T\mathbf{b} - \mathbf{b}^T\tilde{\mathbf{H}}\mathbf{b}\} \tag{3}$$

In (3), $\mathbf{y} = [y_1^{-P}\ y_2^{-P}\ \ldots\ y_K^{-P} \mid y_1^{-P+1}\ \ldots\ y_K^{-P+1} \mid \ldots \mid y_1^{P-1}\ \ldots\ y_{K-1}^{P-1} \mid y_1^{P}\ \ldots\ y_K^{P}]^T \in \mathbf{R}^{(2P+1)K\times 1}$ and $\tilde{\mathbf{H}} \in \mathbf{R}^{(2P+1)K\times(2P+1)K}$ is defined as the *symmetric* cross-correlation matrix of the appropriately time delayed signature waveforms.

In [6], [8] we have shown how an analog Hopfield neural network can be used to solve the OMD problem. An HNN is a collection of simple analog amplifiers that can be used to perform an almost instantaneous gradient descent algorithm in hardware. In particular if T_{ij} is the "synaptic weight" of the connection from the output of the ith amplifier to the input of the jth one, and I_i is the bias current running into the ith amplifier, then as Hopfield showed in his seminal paper [7], the output voltages of the OP-AMPs V_i will finally converge to a stable state, regardless of their initial values. If the weights matrix $\mathbf{T}$ is symmetric and no self- feedback in the OP-AMPs is present ($T_{ii} = 0$), the final stable state reached by a network of N neuron units will be a *local minimum* of the network *energy* function:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij} V_i V_j - \sum_{i=1}^{N} V_i I_i \tag{4}$$

By setting the weights T_{ij} and the biases I_i to reflect the objective function to be minimized, a fast gradient descent algorithm can be performed in hardware by such an OP-AMP network. As shown in [6], [8] such a suitable weights and biases assignment is: $\mathbf{T} = -2\widehat{\mathbf{H}}$ and $\mathbf{I} = \mathbf{y} + \widehat{\mathbf{H}}\bar{\mathbf{1}}$, where $\widehat{\mathbf{H}}$ is obtained by fixing all the diagonal elements of $\tilde{\mathbf{H}}$ to a value of zero, and $\bar{\mathbf{1}}$ is an $K(2P+1) \times 1$ vector of ones.

THE HYBRID DETECTOR

In this section we summarize our theoretical results on the nature of the local minima of the objective function (3) and we describe a very efficient preprocessing stage that leads to a significant reduction in the size of the optimization problem to be solved. The proofs of the propositions stated here can be found in [8].

PROPOSITION 1: If for some element i of the vector $\mathbf{y}$ as defined in (3) it holds that:

$$\sum_{j=1,\ j\neq i}^{(2P+1)K} |\tilde{H}_{ij}| < |y_i| \tag{5}$$

then the OMD's estimate for the corresponding transmitted information bit will be: $b_{opt,i} = \text{sign}(y_i)$ i.e. it will necessarily coincide with the conventional detector's estimate, $\hat{b}_i$.

In other words, inspection of conditions (5) can help us to derive information about the location of the solution of the OMD problem and therefore to restrict the search space over which the optimization has to be performed. Let us now denote by $S_r = \{i_1, i_2, \ldots i_{n_r}\}$ the set of index values for which the inequalities (5) are satisfied (are "right"), and $S_w = \{j_1, j_2, \ldots j_{n_w}\}$ the remaining set of ("wrong") indices. We can then partition the observation vector $\mathbf{y}$ according to the sets S_r and S_w as follows: $\mathbf{y}^T = [\mathbf{y}_r^T | \mathbf{y}_w^T] = [y_{i_1} y_{i_2} \ldots y_{i_{n_r}} | y_{j_1} y_{j_2} \ldots y_{j_{n_w}}]$. If the matrix $\mathbf{H}$ and the unknown vector $\mathbf{b}$ are partitioned accordingly as:

$$\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{H}_{rr} & \mathbf{H}_{rw} \\ \mathbf{H}_{wr} & \mathbf{H}_{ww} \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_r \\ \mathbf{b}_w \end{bmatrix}$$

then the following proposition becomes true.

PROPOSITION 2: The OMD problem (3) can be reduced to a smaller equivalent problem of the same form, in which $\mathbf{y}^T$ is replaced by $\mathbf{y}_{new}^T = \mathbf{y}_w^T - \mathbf{b}_r^T \mathbf{H}_{rw}$ and $\tilde{\mathbf{H}}$ is replaced by $\mathbf{H}_{ww}$.

Note that the same size-reduction procedure can be applied again to the reduced OMD problem as well, leading to the algorithm described in Table 1. The problem size reduction phase ends when either all conditions (5) are violated, in which case either a Hopfield Neural Network algorithm can be employed to solve the *residual* problem or an exhaustive search may be performed, or when all of the conditions (5) are satisfied, in which case the OMD solution is found.

While the computational cost of the multistage detector is equal to: (number of stages) x $(3(2P+1)K^2)$ additions, the computational cost of the *Reduced Detector* (RD) depends on how many of the conditions (5) are met during each iteration. If $n_r(m)$ ($n_w(m)$) is the number of symbols at the mth reduction step, for which conditions (5) are (are not) satisfied, then it can be shown [8] that an upper bound on the computational cost of the Reduced Detector per data packet is:

$$N_{rd}(add) = (2P+1)K + n_w(1) \cdot n_r(1) + \sum_{m=2}^{R} \left(n_w^2(m) + n_w(m) \cdot n_r(m)\right) \tag{6}$$

where R is the number of iterations required. During each step of the algorithm the size of the optimization problem decreases and so does the number of additions per step.

Initialization

$\mathbf{y}$ = output of conventional detector;

$\mathbf{b}_{rd} = sign(\mathbf{y})$;

$S_w = \{1, 2, \ldots, (2P+1)K\}$; $\quad n_w = (2P+1)K$;

repeat

  call reduce()

until ($n_w = 0$ or $n_r = 0$)

procedure reduce( )

$S_r = \{\ \}$; $\quad n_r = 0$;

for all i , $j \in S_w$

  if $\left(\sum_{i,j \in S_w,\, j \neq i} |\tilde{H}_{ij}| < |y_i|\right)$

    $S_r = S_r \cup \{i\}$; $\quad S_w = S_w - \{i\}$;

    $n_r = n_r + 1$; $\quad n_w = n_w - 1$;

end for

if ($n_w = 0$ or $n_r = 0$) break

else

  for all $i \in S_w$

    $y_i = y_i - \sum_{j \in S_r} \tilde{H}_{ij} \cdot b_{rd,j}$

    $b_{rd,i} = sign(y_i)$

  end for

Upon exit

$\mathbf{b}_{rd}$ holds the final estimate.

If $n_w \neq 0$ then S_w holds the indices of the irreducible problem.

Table 1: The Reduced Detector Algorithm.

In all cases that we simulated using MATLAB in a conventional workstation, the size of the irreducible OMD problem was smaller than 32, so that any one of the CLNN32 recently announced analog HNN chips [9] could be used to provide good suboptimal solutions to the residual problem, thus avoiding the need for computationally expensive objective function evaluations. The HNN was initialized to the estimate of the conventional detector for the irreducible problem, and then let to converge to its final state. Note that if more accurate results are desired, numerous tries of the HNN with different initial conditions can be performed in real time, and additional logic (at the expense of larger computational cost) can be used to decide on the best of the resulting local minima.

SIMULATION RESULTS

In all cases the Direct-Sequence Spread-Spectrum Binary PSK (DSSP-BPSK) [10] signaling system was used. The proposed hybrid detector (Reduced Detector followed by an HNN) was compared against the conventional matched filters and the 10-stage multistage detectors.

Example 1: In this case we simulated a set of $K = 3$ asynchronous users employed spreading codes of length $L = 4$ and transmitting packets of length $2P + 1 = 31$. The relative delays of the users were chosen such that the the conditions of worst case interference presented in [4] are satisfied. The energy of user 1 was 10 times larger than the energy of the other users. The RD detector was compared against the 10-stage MS and the conventional detectors. The size of the irreducible problem never exceeded 12 for the set of symbols that we simulated. It is clear from Table 2 that the RD outperforms the 10-stage MS detector, whereas the matched filters detector completely fails to demodulate the received information.

Example 2: A set of $K = 8$ asynchronous users is employing spreading sequences of length $L = 127$ (Gold sequences [10]). The energy of one of the users is 10 times larger than the energy of each of the other users as in the previous examples, and the length of the packet is again $2P + 1 = 31$. In Table 3 we compare the BER performance of the hybrid detector to that of the 10-stages multistage and the conventional detector. As the results suggest, the hybrid detector, at a much

SNR (dB)	BER multistage	BER hybrid (reduced)	BER conventional
5	-1.3348	-1.3529 (-1.342)	-0.9096
6	-1.5091	-1.5409 (-1.5233)	-0.9297
7	-1.6448	-1.7016 (-1.6552)	-0.9422
8	-1.8434	-1.9220 (-1.8982)	-0.9515
9	-2.134	-2.2354 (-2.202)	-0.9614
10	-2.4636	-2.5178 (-2.5747)	-0.9723

Table 2: $K = 3$ asynchronous users, $2P + 1 = 31$, $L = 4$ (conditions of worst case interference). BER performance comparison of the conventional, multistage (10-stages) and hybrid detectors. For the proposed detector, the numbers in parenthesis correspond to the BER value achieved without the HNN post-processing stage (RD stage only).

lower computational cost, outperforms slightly the multistage detector. When a HNN post-processing stage is added, the improvement over the multistage detector, becomes even larger. The maximum number of operations (additions) required for the demodulation of one 8 x 31-bit long data packet (evaluated over 10000 such packets) was 1204 and the average number of additions was 706, whereas the number of additions of the 10-stage multistage detector is about 84 times larger (5952 additions per packet per stage).

SNR (dB)	BER multistage	BER hybrid (reduced)	BER conventional
4	-1.3004	-1.3176 (-1.2989)	-1.2337
5	-1.4813	-1.5051 (-1.4797)	-1.3780
6	-1.6432	-1.6825 (-1.6455)	-1.5430
7	-1.8941	-1.9230 (-1.9008)	-1.7320
8	-2.2570	-2.3388 (-2.3010)	-1.9360
9	-2.6582	-2.7591 (-2.6702)	-2.1868

Table 3: BER performance comparison of the Multistage, Hybrid (Reduced) and Conventional CDMA multi-user detectors: $K = 8$ asynchronous users, $2P + 1 = 31$, $L = 127$ (Gold sequences), maximum near-far-ratio = 10

CONCLUSIONS - FURTHER RESEARCH DIRECTIONS

A novel hybrid digital computer - neural network CDMA multiuser detector has been introduced. For a similar level of BER performance, the computational complexity of the Reduced Detector is smaller than that of other proposed schemes by more than one order of magnitude in some cases. The Reduced Detector can be used in conjunction with any other suboptimal scheme since the irreducible problem has the exact same structure of the original OMD problem.

The small size of the irreducible problem allows for the use of off-the-shelf available HNN neural chips to further improve the performance. Issues that we are currently investigating include: The experimental evaluation of the proposed scheme on available HNN chips; the efficient implementation of the preprocessing stage on either standard DSP microprocessors or dedicated ASIC/systolic array VLSI architectures; the optimal initialization of the HNN as well as iterative schemes that increase the probability of converging to the global minimum; annealing and other proposed HNN like algorithms implementable in hardware to further improve the performance.

References

[1] S. Verdu. Computational Complexity of Optimum Multiuser Detection. *Algorithmica*, 4:303–312, 1989.

[2] M. K. Varanasi and B. Aazhang. Multistage Detection in Asynchronous Code-Division Multiple Access Communications. *IEEE Trans. on Comm.*, 38:509–519, April 1990.

[3] R. Lupas and S. Verdu. Near-Far Resistance of Multiuser Detectors in Asynchronous Channels. *IEEE Trans. on Comm.*, 38:496–508, April 1990.

[4] Z Xie, C. K. Rushforth, and R. T. Short. Multiuser Signal Detection Using Sequential Decoding. *IEEE Trans. on Comm.*, 38:578–582, May 1990.

[5] B.-P. Paris B. Aazhang and G. Orsak. Neural Networks for Multiuser Detection in CDMA Communication. *IEEE Trans. on Comm.*, 40:1212–1222, July 1992.

[6] G. Kechriotis and E. S. Manolakos. Implementing the Optimal CDMA Multiuser Detector with Hopfield Neural Networks. In *Proceedings of the Int'l Workshop on Applications of Neural Networks to Telecommunications*, pages 60–67, Princeton, New Jersey, October 1993.

[7] J. J. Hopfield. Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons. *Proc. Natl. Acad. Sci. USA*, 81:3088–3092, 1984.

[8] G. Kechriotis. *Feedback Neural Networks in Digital Communications: Algorithms, Architectures and Applications.* PhD thesis, Northeastern University, Boston, 1994.

[9] A. Jayakumar and J. Alspector. An Analog Neural-Network Coprocessor System for Rapid Prototyping of Telecommunications Applications. In *Proc. Int. Workshop on Appl. of Neural Networks to Telecommunications*, pages 13–19, October 1993.

[10] J. Proakis. *Digital Communications.* Prentice Hall Inc., N.J., 1988.

A HOPFIELD NETWORK BASED ADAPTATION ALGORITHM FOR PHASED ANTENNA ARRAYS

Mathäus Alberti
Dept. of Communications Engineering FB14/NT
University of Paderborn
33095 Paderborn
Germany

Abstract — One of the problems of adaptive antennas is to find the weight factors for an array pattern optimizing the signal to noise and interference ratio for the actual signal situation. A neural Hopfield network is able to find the optimal factors, if the direction to the desired transmitter and the interfering transmitters are known [1]. To actualize altering directions, the proposed random search algorithm analyses the signal power of the antenna output. In combination with the Hopfield network it can track the desired signal and suppress interfering sources. This is shown in simulations, which were carried out using a digital controller of an array antenna (algorithm and Hopfield network) and a host computer (signal situation, antenna pattern and output power).

INTRODUCTION

Compared to omnidirectional antennas, phased antenna arrays are more advantageous because of their ability to adapt the antenna group pattern to a certain signal situation. A major task is to find the weights w_i for the antenna element signals. Several algorithms are known [2] to perform this with more or less hardware. A method without need of element signals is to update the setting of a phase shifter according to a quality signal from the receiver [3, 4]. A random change in the setting of the phase shifters is maintained or abandoned depending on the change of this

0-7803-2026-3/94 $4.00 © 1994 IEEE

quality signal.

Figure 1 shows a block diagram of an adaptive array. The signal to noise and interference ratio of the receiver input signal is a function of the signal situation and the weight setting. Two possible quality signals (there are certainly more) are the array output power and the code error rate of a transmitted signal.

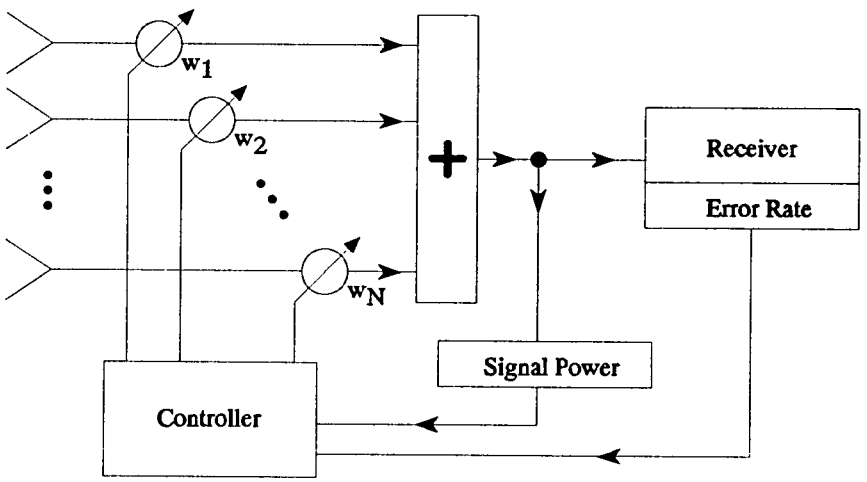


Fig. 1: Block diagram of the adaptive antenna system. w_i: Setting of the ith phase shifter; N: Number of antenna elements

Maximizing the output power of the array antenna alone is more than fast enough to adapt the group pattern for the demands of mobile applications [5]. The exclusive use of this quality signal is restricted to the case that the desired transmitter is the strongest or the only one in the interesting frequency band.

Suppressing an interfering transmitter needs a quality signal with more information. As an example, in [4] the error rate in the synchronisation frame of the digitally broadcasted audio signal of a DBS-Satellite[1] was evaluated. This enables the array antenna to

[1]Direct Broadcasting by Satellite, e.g. TV-Sat

suppress an interfering source, but slows down the angular speed for tracking from $100°/s$ to $0.5°/s$. This is due to the small dynamic of the quality signal used.

If both signals directly control the setting of a phase shifter, the signal with the larger dynamic and rate of repetition suppresses the influence of the other one.

The combination of both quality signals for a relatively fast adaptation with regard to interfering transmitters is possible, if each signal is used for a different task. Estimated directions of signal sources, e.g. determined by an initial sweep of an pencil beam, are classified by the low dynamic quality signal from the receiver as direction of a desired or interfering source. The high dynamic quality signal from the output power of the array is used to track the desired signal and the interferers. So the algorithm tries to optimize the position of a certain transmitter, not the setting of a phase shifter directly.

The last – but not least – step is to compute the weights $w_i(k)$ for the estimated positions of the transmitters. This task can be performed with a Hopfield network [1]. Due to its parallel computing, the delay for each iteration needs not too much time compared with the other parts of the algorithm. A drawback of this network is the necessity to update the complete set of network coefficients caused by the changing directions to the respective signal sources.

ALGORITHM

Estimation of signal direction

The algorithm based on the output power of the adaptive array after classification of possible signal sources is shown in Figure 2. It changes the estimated direction to a signal source in an arbitrary direction with a stepwidth $\Delta\theta$.

Two cases must be considered in the algorithm:
— Altering the estimated direction to the desired source, an increase of output power can be interpreted as an improvement of the estimation, a decrease as a step in the wrong direction.
— Adjusting the estimation of the position of an interfering source, an increase of output power can be interpreted as a step in the wrong direction, a decrease indicates a better suppression of the interfering transmitter.

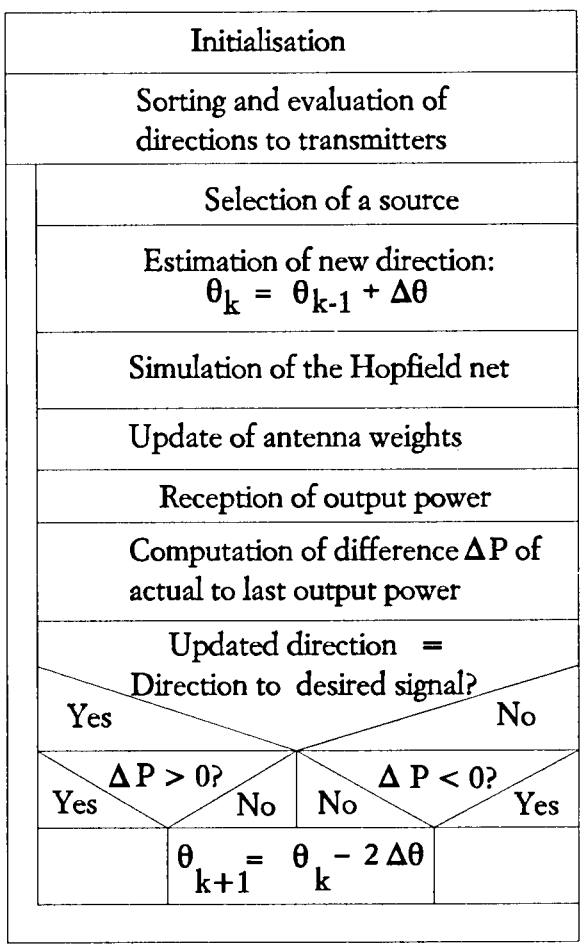


Fig. 2: Block diagram of the search algorithm.

Hopfield network

The structure of the network is shown in Figure 3. Because it is documented in [1], only a short rewiew for is given here.

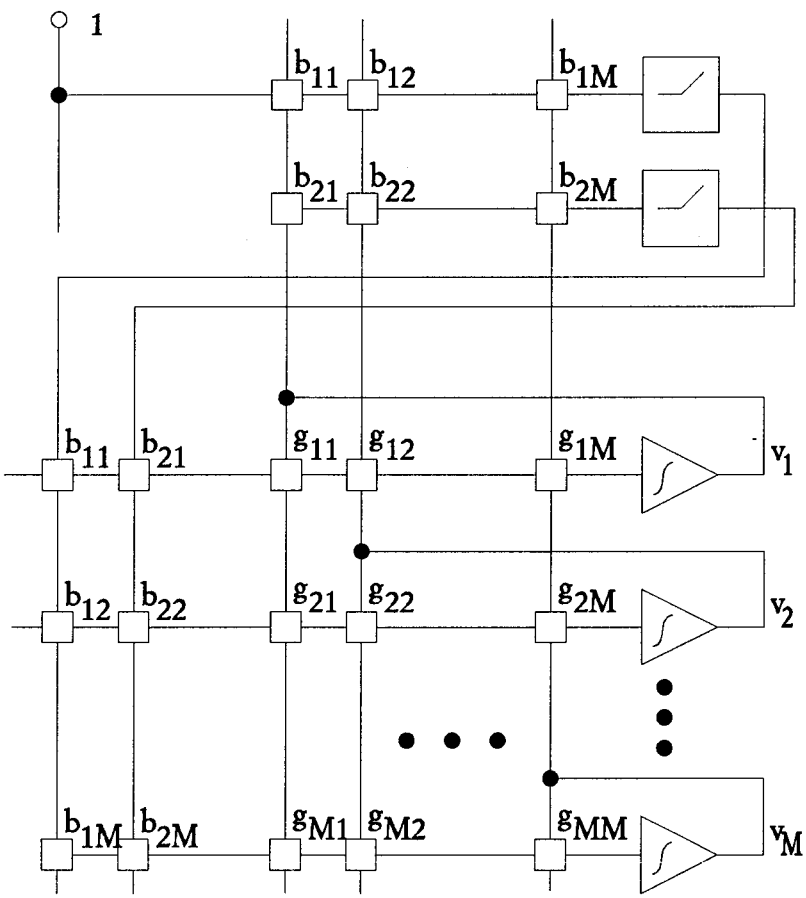


Fig. 3: Network to find array weights [1], $M = 2N$

The problem to find the weights w_i for the estimated signal situation can be written in the form [1, 6]:

$$\underset{v}{\text{Minimize}} P(v) = \frac{1}{2} v^T G v \tag{1}$$

with regard to

$$Bv - e = 0 \tag{2}$$

and

$$G = \begin{pmatrix} 2R_r & -2R_i \\ 2R_i & 2R_r \end{pmatrix}, \tag{3}$$

$$B = \begin{pmatrix} S_{0r}^T & S_{0i}^T \\ -S_{0i}^T & S_{0r}^T \end{pmatrix}, \tag{4}$$

$$e = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{5}$$

R_r and R_i are real and imaginary part of the covariance matrix $R = E(X(t)X^*(t))$, S_0 the steering vector of the desired signal and $v = (w_{1r}, w_{2r}, \cdots, w_{Nr}, w_{1i}, w_{2i}, \cdots, w_{Ni})^T$. G and B are the coefficients of the Hopfield network, w is the vector of array weights. X^* denotes the transpose conjugate of X.

The covariance matrix R is computed from the estimated signal situation to save the amount of hardware necessary to access the single antenna element signals.

SIMULATION

The random search algorithm together with the hopfield network is programmed on a TMS320E17 signal processor. Because this device has a single accumulator architecture, the parallel architecture of a hopfield network can only be simulated and its advantage in speed is lost. In this study the ability to change the algorithm and the network together with existing hardware was the reason for this choice. As the weights w_i are quantized with a resolution of $m = 4$ bit in real and imaginary part [7], the Hopfield network finds its stable solution within five iterations. Figure 4 shows the setup for the simulations.

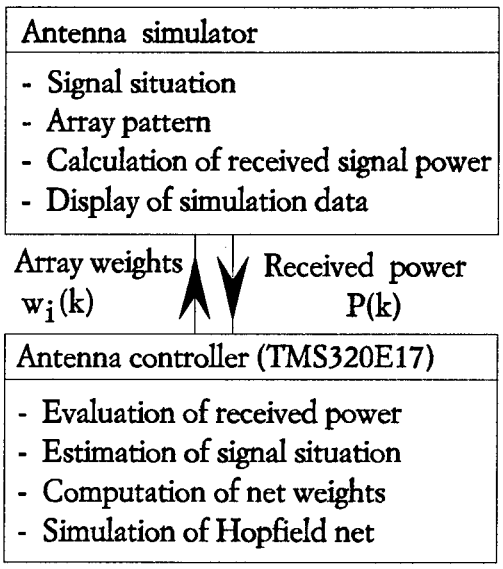


Fig. 4: Simulation configuration

Given the locations of desired signal and interfering transmitters, the Hopfield network is able to compute a set of weights w_i even for a more complicated signal situation, taken from [3]. The resulting array pattern is shown in Figure 5 after selecting the possible settings from a table [7] nearest to the output of the network.

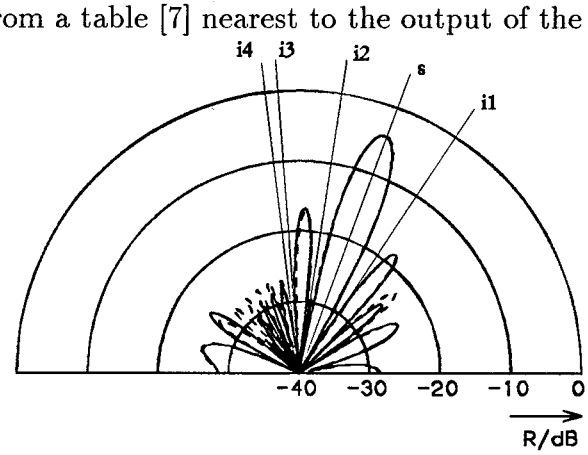


Fig. 5: Beam pattern after adapting to signal s and four interferences i, — discrete w_i, - - continuous w_i, $N = 16$, $\theta_s = -20°$, $\theta_{i1-4} = -33°,\ -7°,\ 5°,\ 7°$

Together with the random search algorithm based on the output power it is possible to track the desired transmitter. Figure 6 shows a signal situation with an interfering source at a position where otherways would be one of the first sidelobes of the simulated linear array with $N = 16$ elements spaced $\lambda/2$ apart (λ as average wavelength of the interesting frequency band). Figure 7 displays the signal to noise and interference ratio while tracking the situation given in Figure 6.

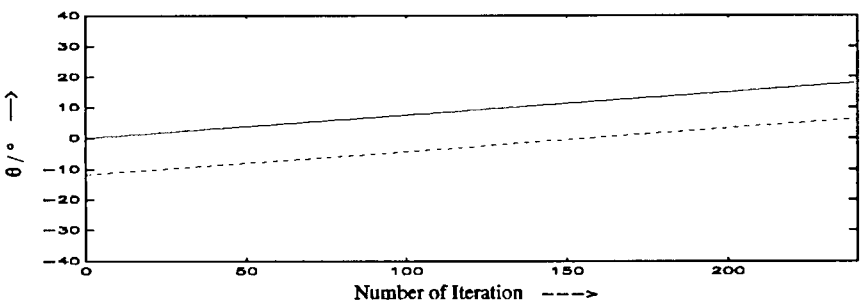


Fig. 6: Signal positions of — desired signal; - - interfering transmitter

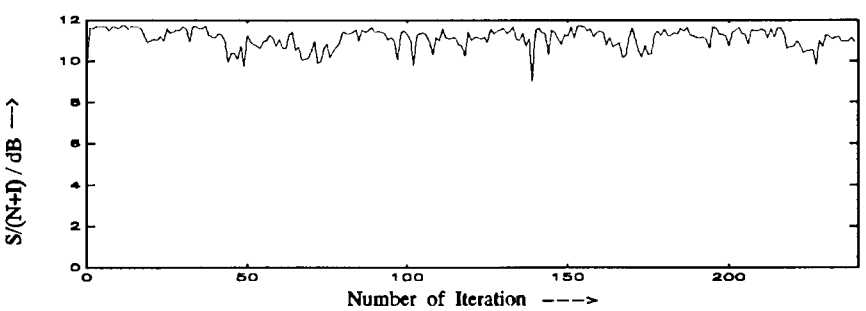


Fig. 7: Signal to noise plus interference ratio; $(S/N)_{max} = 12\text{dB}$

The time for the antenna simulator to compute the signal situation

and the communications between the simulator and the controller are not present in a complete antenna system. Subtracting them from the time of an iteration, the controller was able to track the signals with an angular speed of $\omega = (7°/s)(\pi/180°)$.
The relatively simple random search algorithm to provide the Hopfield network with the estimated positions of the transmitters is limited in the power quality signal. Shifting the estimated position of an interfering transmitter within a minimum of the antenna pattern influences the total received power as much as the slight change in the main beam due to this shift. The Algorithm is able to cope with the signal situation given in Figure 6 but fails for the situation given in Figure 5. Further work has to be done for improvements.

SUMMARY

In combination with a random search algorithm a hopfield network can be used to track a signal source for mobile communications and cancel out an interfering transmitter without need for the single antenna element signals. As quality signals for the random search only the output power of the array and a reference signal indicating the right transmitter is needed.

REFERENCES

[1] Chan, K., Chang, P., Yang, W.: A Neural Net Approach to Real-Time Adaptive Array, Conf. Proceedings of the 21. European Microwave Conference, 9.-12.9.1991, Vol.1, pp. 751-756

[2] Monzingo, R. A., Miller, T. W.: Introduction to Adaptive Arrays, Wiley & Sons, New York

[3] Grabow, W.: Numerische und experimentelle Untersuchungen zum mobilen Satellitenempfang zirkular polarisierter Wellen mit adaptiven Planarantennen, Dissertation D 14-34, Universität Paderborn, 1990

[4] Schrewe, H.-J.: Ein adaptives Antennensystem zum mobilen Empfang des digitalen Satellitenhörrundfunks, Dissertation D 14-65, Universität Paderborn, 1993

[5] Koschnick, P.: Eine adaptive Gruppenantenne für ein mobiles Verkehrsinformationssystem im Frequenzbereich um 12 Gigahertz Dissertation D 14-50, Universität Paderborn, 1991

[6] Kennedy, M. P., Chua, L. O.: Neural Networks for Nonlinear Programming, IEEE Trans. Circuits Syst., vol. CAS-35, pp. 554-562, May 1988

[7] Alberti, M.: Discrete weighting of digitized signals for adaptive phased arrays, Proceedings of the Third International Symposium on Antennas and EM Theory (ISAE'93), pp. 675-678, Sept. 1993, Nanjing, China

BLIND DECONVOLUTION OF SIGNALS USING A COMPLEX RECURRENT NETWORK

Andrew D. Back and Ah Chung Tsoi
Department of Electrical and Computer Engineering
University of Queensland, St. Lucia. 4072.
Australia.
back@sl.elec.uq.oz.au, act@sl.elec.uq.oz.au

Abstract– An algorithm for the separation of mixtures of signals was derived recently by Jutten and Herault under the assumption that the signals are independent. This algorithm is based on higher order moments and has also been applied to deconvolving signal mixtures. In practical problems where the order of the convolving filter may be high, frequency domain approaches are known to provide a more computationally efficient method of deconvolution. In this paper, we introduce a complex recurrent network structure for performing blind deconvolution. The aim is to investigate the performance of this approach for separating unknown, convolved signals which may occur in a situation such as the well-known 'cocktail-party problem'.

INTRODUCTION

Adaptive filtering techniques have been widely applied to noise cancellation and blind deconvolution problems, where the aim is to obtain an enhanced signal based on some knowledge of the signal or noise [20, 23, 35]. More recently, the problem of separating mixtures of signals has been considered. In this case, the only assumption made is that the signals are independent. In addition, there are multi-sensors which record the signal. This situation is commonly encountered in practice, e.g. in the "cocktail-party" problem, where the aim is to listen to a desired signal (speech) and other signals are considered as noise. In such cases, traditional methods for adaptive noise cancellation (e.g. [35]) are insufficient to solve the problem.

A novel method to overcome the problem of blind separation of sources has been proposed recently by Jutten and Herault [21]. They proposed an algorithm which only assumes that the sources are statistically independent. The blind separation of sources problem consists of a mixture model, with a number of unknown input signals, and a desired method of estimating those signals. The mixture model is described by

$$x_j(t) = \sum_{i=0}^{M} a_{ji} u_i(t) \tag{1}$$

where $u_i(t)$ is the ith signal source, a_{ji} is the (amplitude) weighting applied to the signal from the ith source and received by the jth sensor, and $x_j(t)$ is the sum of received signals at the jth sensor. Based on this model, Jutten and Herault proposed a method to estimate values for $u_i(t)$. The model developed by Jutten and Herault may be described as follows

$$Y(t) = X(t) - WY(t) \tag{2}$$

$$Y(t) = [y_0(t), y_1(t), ..., y_M(t)]^T \tag{3}$$

$$X(t) = [x_0(t), x_1(t), ..., x_M(t)]^T \tag{4}$$

$$W = \begin{bmatrix} 0 & w_{01} & \cdots & w_{0M} \\ w_{10} & 0 & \cdots & w_{1M} \\ \vdots & \vdots & \vdots & \vdots \\ w_{M0} & w_{M1} & \cdots & 0 \end{bmatrix}$$

0-7803-2026-3/94 $4.00 © 1994 IEEE

Where it is tacitly assumed that the number of sensors is equal to the number of sources, and the time constant of the network is small compared to that of the incoming signals, and that the network is stable. Thus, the network output can be described as

$$Y(t) = (I + W)^{-1} X(t) \tag{5}$$

where I is the identity matrix. We define the vector of zero mean output signals as

$$\underline{y}(t) = [\underline{y}_0(t), \underline{y}_1(t), \ldots, \underline{y}_M(t)]^T \tag{6}$$

The weights w_{ji} are updated by a stochastic gradient descent algorithm, a cost criterion (for each output) is defined as

$$J_i(t) = \underline{y}_i^2(t) \tag{7}$$

$$= y_i^2(t) - E[y_i^2(t)] \tag{8}$$

which is the power of the zero mean output signal. The algorithm obtained by this process is modified to update the weights using nonlinear functions which provide the test for higher order moments [21]. Thus, we have

$$\Delta w_{jk} = \eta f(\underline{y}_j(t)) g(\underline{y}_k(t)) \tag{9}$$

where η is a learning rate constant, and $f(x)$, $g(x)$ are odd nonlinear functions. For the simulations reported in Section 4 we used $f(x) = x^3$, and $g(x) = \arctan(x)$.

Note that in the Jutten and Herault formulation, the network architecture used is a recurrent network [21]. A related situation is that involving a convolutive model for dispersive media, this was considered by Nguyen, Jutten [26, 22], Platt and Faggin [16], Lacoume and Ruiz [24], Soon, Tong et. al. [30, 31] and Van Gerven and Van Compernolle [28] (Figure 1). In this case,

$$x_j(t) = \sum_{i=0}^{M} \sum_{k=0}^{P} a_{jik} u_i(t - d_k) \tag{10}$$

In a similar manner to that given by Jutten and Herault, Platt and Faggin gave a weight update equation for their model as

$$\Delta w_{ikj} = \eta f(\underline{y}_i(t)) g(\underline{y}_k(t - d_j)) \tag{11}$$

where the best reported results were obtained with $f(x) = |x|$, and $g(x) = x$. Previously, there has been significant work done on this topic, closed form solutions have been presented in [3, 6, 7, 33, 34], while other models have been considered in [30, 31].

The models developed by Jutten and Herault, Platt and others are based on real signals. In some situations however, it is desirable to be able to use complex signals. It is well known, for example, that frequency domain adaptive filters can have advantages over time-domain filters [17], (though for some adaptive noise-cancelling applications, these advantages are not apparent [9]). For conventional filter structures, frequency domain adaptive filters (using complex signals) have advantages in two main areas. The use of the Fast Fourier Transform (FFT) for complex weight update leads to a reduction[1] in computation with increased data [36]. Secondly, it is known that for

[1]The computational advantages are dependent on which scheme is used for processing, that is, whether the algorithm implements a circular convolution [11], or linear convolution [10], [12].

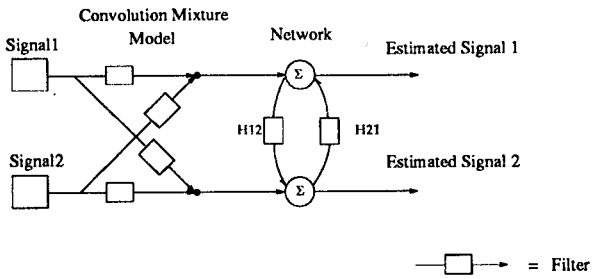


Figure 1: Convolutive mixture model and the network proposed by Platt and Faggin. In this case, each of the synapses is a filter. Note that we show only a two neuron network. In practice, the principles discussed here apply to any size network, (provided there are at least as many inputs as outputs).

the usual stochastic gradient algorithms, the convergence rate is proportional to the eigenvalue spread of the inputs [37]. This serves as a limiting factor in the time-domain filter. In a frequency-domain filter, the use of frequency bins provides an approximate orthogonalization of the data [17], and the learning rate in each frequency bin can be adjusted according to the power level present. This overcomes the problem of eigenvalue spread [19]. In effect, a normalized algorithm can be applied in each frequency bin [17], allowing the convergence rate to be improved for dependent inputs such as speech. For those applications where a frequency domain implementation does offer advantages, we present the algorithm contained herein.

For these reasons, we present a complex algorithm for blind source separation. The complex algorithm is derived in Section 2. Computational complexity details are given in Section 3, simulation results are given in Section 4 and conclusions are given in Section 5.

A COMPLEX ALGORITHM FOR BLIND SOURCE SEPARATION

The algorithm is developed by considering a complex model analogous to the real case. Figure 2 shows the complex model for the case of two mixture inputs and two outputs. The same techniques are applicable to larger network structures. From (3), we define

$$\bar{Y}(t) = [Y(t), Y(t-1), ..., Y(t-N+1)] \tag{12}$$

Taking the Fourier transform, and noting that the sampling rate in the frequency domain is reduced by a factor of N, where N is the number of points in the FFT (assuming no zero padding to overcome the problem of circular convolution), we have

$$\begin{aligned} \bar{\mathbf{Y}}(n) &= \text{FFT}\left[\bar{Y}(t)\right] \qquad t = nN \qquad (13) \\ &= \begin{bmatrix} \mathbf{y}_{00}(n) & \mathbf{y}_{01}(n) & \cdots & \mathbf{y}_{0N-1}(n) \\ \mathbf{y}_{10}(n) & \mathbf{y}_{11}(n) & \cdots & \mathbf{y}_{1N-1}(n) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{y}_{M0}(n) & \mathbf{y}_{M1}(n) & \cdots & \mathbf{y}_{MN-1}(n) \end{bmatrix} \end{aligned}$$

where n is the new sampling index in the frequency domain and the bold typeface represents variables in the frequency domain (following the convention used in [17]). Define the complex model as

$$\mathbf{Y}_j(n) = \left[\mathbf{y}_{j0}(n), \mathbf{y}_{j1}(n), \cdots, \mathbf{y}_{jM}(n)\right]^T \tag{14}$$

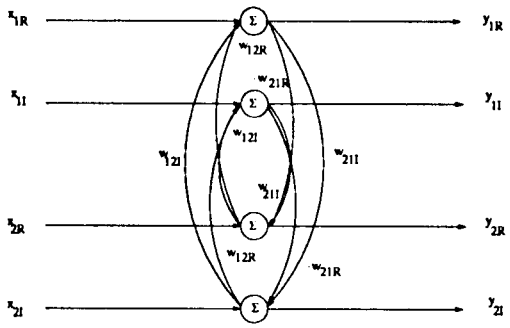


Figure 2: Complex blind source separation model, with two mixture inputs $\mathbf{x}_1$, $\mathbf{x}_2$, and two outputs $\mathbf{y}_1$, $\mathbf{y}_2$. The network is used in every frequency bin.

$$\mathbf{X}_j(n) = \left[\mathbf{x}_{j0}(n), \mathbf{x}_{j1}(n), \cdots, \mathbf{x}_{jM}(n)\right]^T \tag{15}$$

$$\mathbf{W}_j = \begin{bmatrix} 0 & \mathbf{w}_{j01} & \cdots & \mathbf{w}_{j0M} \\ \mathbf{w}_{j10} & 0 & \cdots & \mathbf{w}_{j1M} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{w}_{jM0} & \mathbf{w}_{jM1} & \cdots & 0 \end{bmatrix}$$

$$\mathbf{Y}_j(n) = (I + \mathbf{W})^{-1}\mathbf{X}_j(n) \tag{16}$$

We now derive the weight update equations. For clarity, the j subscript indicating the frequency component index is dropped from the remainder of the paper. Following [36], we minimize both the real and imaginary components of the weights simultaneously. Therefore, the total power $\underline{\mathbf{y}}_i(n)\underline{\mathbf{y}}_i^*(n)$ is minimized, ($*$ represents the complex conjugate), and

$$J_i = \underline{\mathbf{y}}_i(n)\underline{\mathbf{y}}_i^*(n) = \underline{y}_{iR}^2(n) + \underline{y}_{iI}^2(n)$$

The weights are adjusted by minimizing the instantaneous total power,

$$\mathbf{w}_{jk}(n+1) = \mathbf{w}_{jk}(n) - \eta\frac{\partial J_i}{\partial \mathbf{w}_{jk}} \tag{17}$$

where, $\mathbf{w}_{jk}(n) = w_{jkR}(n) + jw_{jkI}(n)$. Following the procedure used by Jutten and Herault, and updating the complex weights, we have

$$\Delta w_{jkR}(n) = -\eta\frac{\partial J_i}{\partial w_{jkR}}$$

$$\Delta w_{jkI}(n) = -\eta\frac{\partial J_i}{\partial w_{jkI}} \tag{18}$$

$$\frac{\partial \underline{\mathbf{Y}}(n)}{\partial w_{jkR}} = -(I + \mathbf{W})^{-1}\frac{\partial(I + \mathbf{W})}{\partial w_{jkR}}\underline{\mathbf{Y}}(n) \tag{19}$$

Omitting details, the weight changes (with $i = j$) are

$$\mathbf{w}_{jk}(n+1) = \mathbf{w}_{jk}(n) + \Delta\mathbf{w}_{jk}(n)$$

$$
\begin{aligned}
&= \mathbf{w}_{jk}(n) + -\eta\left(\frac{\partial J_j}{\partial w_{jkR}} + \jmath\frac{\partial J_j}{\partial w_{jkI}}\right) \\
&= \mathbf{w}_{jk}(n) + 2\eta\underline{\mathbf{y}}_k^*(n)\underline{\mathbf{y}}_j(n) \qquad (20)
\end{aligned}
$$

Previously, a higher order independence test was used in the weight update equation, which, if applied in the frequency domain case, would result in an equation of the form

$$
\mathbf{w}_{jk}(n+1) = \mathbf{w}_{jk}(n) + 2\eta\mathbf{f}(\underline{\mathbf{y}}_k^*(n))\mathbf{g}(\underline{\mathbf{y}}_j(n)) \qquad (21)
$$

where $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ are complex functions which operate on real and imaginary components independently, i.e.,

$$
\mathbf{f}(x+\jmath y) = f_R(x) + \jmath f_I(y) \qquad (22)
$$

A function of this type was proposed also by Benvenuto and Piazza [2]. Note that normally $f_R(\alpha) = f_I(\alpha)$. If the above method was used in this case however, the problem of obtaining the correct permutation matrix for the signals would be extremely difficult to solve. The permutations would occur in the frequency domain, and be complicated by an additional multiplication by a diagonal matrix of unit modulus complex values. The approach used in this paper, is to perform the test for independence in the time domain as before and transform the outputs to the frequency domain *before* multiplying them together. Since the signals are not multiplied in the time domain, but the multiplication of $f(y)^*g(y)$ is performed in each frequency domain bin, and the problem of permutations between different frequency bands is overcome.

In this way we select a criterion which allows the algorithm to perform a task of essentially the same difficulty as before in the time domain case (down to a permutation matrix in the real valued output signals). The final weight update equations are

$$
\mathbf{w}_{jk}(n+1) = \mathbf{w}_{jk}(n) + 2\eta\hat{\mathbf{F}}_k(n)\mathbf{G}_j(n) \qquad (23)
$$

where the Fourier transform of each block of output signals passed through the non-linear functions is computed, viz

$$
\begin{aligned}
\mathbf{F}_k(n) &= \mathrm{FFT}\left[f(\underline{y}_k(t))\right] & \mathbf{G}_j(n) &= \mathrm{FFT}\left[g(\underline{y}_j(t))\right] \\
\hat{\mathbf{F}}_k(n) &= \mathrm{FFT}\left[f(\underline{\bar{y}}_k(t))\right] & \hat{\mathbf{G}}_j(n) &= \mathrm{FFT}\left[g(\underline{\bar{y}}_j(t))\right] \qquad (24)
\end{aligned}
$$

where

$$
\underline{\bar{y}}_k(t) = \mathrm{FFT}^{-1}\left[\underline{\mathbf{y}}_k^*(n)\right] \qquad (25)
$$

Note that $\mathbf{F}_k$, $\mathbf{G}_j$, represent vector processes, with the length determined by the number of points in the FFT computation. A method of improving the performance of the standard algorithm, is to include power normalization in each frequency band [17].

It is possible that minimization of the above criteria may lead to problems of unstable convergence as in the Jutten and Herault algorithm [13]. It is expected that the algorithm presented here and the Jutten and Herault algorithm will have similar properties. For example, convergence is only expected if the probability density functions of the zero mean sources are even, while swapping the $f(\cdot)$ and $\mathbf{g}(\cdot)$ functions may cause a stable solution to become unstable [32]. Thus, the presentation of this algorithm does not address some of the more fundamental problems associated with the Jutten and Herault algorithm, but rather introduces a method for overcoming problems of convolved and correlated input signals.

Table 1: Computational Complexity Ratios for the Frequency Domain Blind Source Separation Model versus the Time Domain Model with M=2 and L=20.

N	8	16	32	64	128	256	512
Complexity Ratio	0.5	0.25	0.12	0.0614	0.0306	0.0153	0.0076

COMPUTATIONAL COMPLEXITY

Consider the convolutive mixture model, where there are N real weights in each filter, M sensors, M model outputs, and the model is applied to N data points.

The computational complexity is determined as the sum of multiplications for both the model outputs and the weight updates. The fast acting recurrent loop [21] is assumed to have L cycles per presentation of input data.

The complexity depends on whether the algorithm uses circular convolution or linear convolution [11], [12], [17], we will only consider the circular convolution here. For the circular convolution case, there are $2M$ FFTs required for the computation of outputs, and $3M$ FFTs required to compute the higher order independence test variables ($\mathbf{F}_k$, $\mathbf{G}_j$) (assuming $g(\alpha) \neq \alpha$), giving a total of $5M$ FFTs required for every N data points. To compute the outputs, there will be $4LN(M^2 - M)$ real multiplications required (over L cycles). For the weight updates with real inputs, there will be $6N(M^2 - M)$ real multiplications required, giving a total of $5MN\log_2(N) + (6 + 4L)N(M^2 - M)$ real multiplications for every N data points. The method of computing the linear convolution for either the overlap-add or overlap-save methods are described in [17], and are not considered further in this paper. Table 1 shows the comparison between the computational complexity of the time domain with the frequency domain methods. It is expressed as a complexity ratio of the computation required by the frequency domain method and that required by the time domain method.

SIMULATION RESULTS

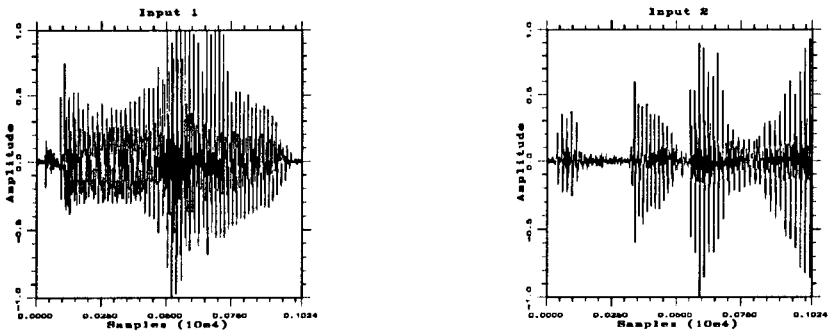
In this section, results are presented in using the above complex algorithm for simulations involving the separation of two speech signals. The voices were digitized at 8kHz and mixed according to the mixture model shown in Figure 1. The actual input voice signals are shown in Figure 3 and the mixture signals are shown in Figure 4

To maintain the symmetry of real outputs for real inputs, the imaginary weights were set to zero [18].

As a means of verifying performance of the complex model, various lengths of data were tried, with the model performing well in the ranges tested (16 - 128). In the results shown, $N = 64$ (Figure 5).

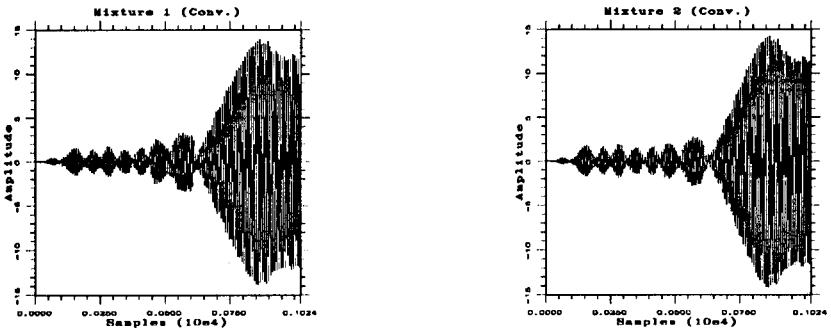
While it is possible to implement a linear convolution model, for the example given here, only the circular convolution model was used. The observed results show that the complex model is indeed capable of performing separation of signals mixed in a convolution model. For signal $y_1(t)$, the signal-to-noise ratio (SNR) is 8dB, and for $y_2(t)$, the SNR is 9.4dB[2]. It should be noted that, as is usual with gradient descent algorithms, better results would be possible by allowing the network to learn for a longer period of time. The results obtained here are less favourable than those reported by Platt and Faggin [16], who obtained around 20dB SNR for separating speech and music using the architecture shown in Fig. 2. In their case however, the

[2]These results were computed by normalizing the output signals $y_1(t)$, $y_2(t)$ to the same root-mean-square (RMS) value as the corresponding inputs. Thus, we use $y_1'(t) = y_1(t)\gamma_1$, and $y_2'(t) = y_2(t)\gamma_2$ where $\gamma_1 = \tilde{u}_1/\tilde{y}_2$, and $\gamma_2 = \tilde{u}_2/\tilde{y}_1$. The RMS value of x is represented by $\tilde{x}$. Note that in the case presented here, the outputs are reversed with respect to the inputs.



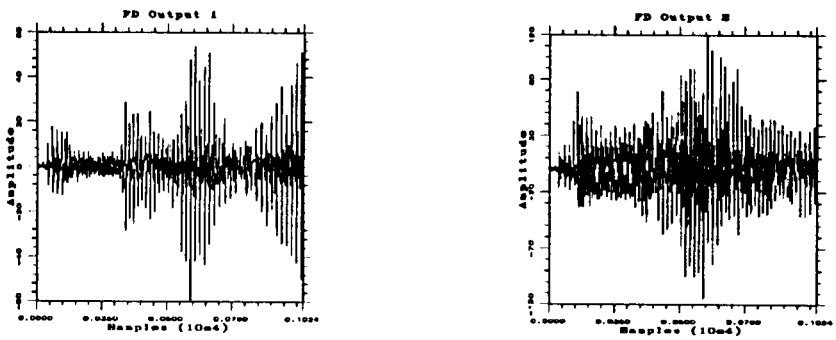
(a) (b)

Figure 3: Input sources of two independent speakers taken while each speaker counted "One .. two .. three ..", (a) u1(t), (b) u2(t). Sampling was done at 8Khz.



(a) (b)

Figure 4: Output from the mixture model, with each signal being a linear combination of both speech signals: (a) x1(t), (b) x2(t).



(a) (b)

Figure 5: Outputs from the frequency domain network which estimate the original input signals: (a) y1(t), (b) y2(t). This test was performed without power normalization and the performance is similar to the original time domain model. For the case shown here, the FFTs used $N = 64$. Learning commenced at the first sample, and occurred over the whole interval shown. Note that the order of the output signals is reversed from that of the input signals, due to the algorithm having no mechanism to sort the outputs in any particular way. Such an addition could be readily implemented.

mixtures were such that one signal was attenuated; specific details were not given. This contrasts with the results presented here, the estimated signals were of similar magnitudes to the original ones (observe Figures 4(a) and 4(b)). Thus, the simulations presented are not aimed at showing the best performance possible, but simply to verify the operation of the model. It would be interesting to compare the performance of this algorithm with one which uses only second order statistics, however this is not done here.

CONCLUSIONS

The algorithm developed by Jutten and Herault for blind source signal separation has shown promising results for real signals. In many practical situations however, it is desirable to apply an algorithm to complex data. In this paper, we have derived a complex algorithm for blind source signal separation which uses higher-order moments as in the original approach by Jutten and Herault [21], but allows the use of complex coefficients and data. The adoption of this method results in a more efficient implemention of a blind source separation model for convolutive mixtures. Simulations have verified the use of the algorithm for separating real speech signals.

Acknowledgements

The first author gratefully acknowledges support from the Electronics Research Laboratory, Defence Science and Technology Organisation, Australia, and the Australian Research Council. Appreciation is expressed to the Director, ERL, for permission to allow this paper to be published. Thanks are expressed also to John Shynk for helpful advice.

References

[1] S.T. Alexander, "A derivation of the complex fast Kalman algorithm", *IEEE Trans. Acoust. Speech, Signal Proc.*, vol. ASSP-32, no. 6, pp. 1230-1232, 1984.

[2] N. Benvenuto, and F. Piazza, "On the complex backpropagation algorithm", *IEEE Trans. Sig. Proc.*, vol SP-40, pp. 967-969, 1992.
[3] J.F. Cardoso, "Source separation using higher order moments", *Proc ICASSP'89*, vol. IV, pp. 2109-2112, Scotland, 1989.
[4] J.F. Cardoso, "Blind Beamforming for Non-Gaussian Signals", *IEE Proc.-F, Radar and Signal Proc.*, vol. 140, pp. 362-370, Dec, 1993.
[5] J.F. Cardoso, "Iterative Techniques for Blind Source-Separation Using Only Fourth-Order Cumulants", Signal Processing VI: Theories and Applications, J. Vandewalle, R. Boite, M. Moonen, A. Oosterlinck (Eds), Elsevier Science Publishers B.V., pp. 739-742, 1992.
[6] P. Comon, "Independent Component Analysis", *Proc. Workshop on Higher-Order Spectral Analysis*, pp. 174-179, France, 1989.
[7] P. Comon, "Separation of Stochastic Processes", *Proc. Workshop on Higher-Order Spectral Analysis*, pp. , France, 1989.
[8] P. Comon, C. Jutten, and J. Herault, "Blind separation of sources, Part II: Problems Statement", *Signal Processing*, vol. 24, pp. 11-20, 1991.
[9] J.M Connell and C.S. Xydeas, "A comparison of acoustic noise cancellation techniques for telephone speech", *Proc. 6th Int. Conf. on Digital Proc. of Signals in Communications*, IEE, pp. 320-325, 1991.
[10] G.A. Clark, S.R. Parker, and S.K. Mitra, "Efficient realization of adaptive digital filters in the time and frequency domains", *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, pp. 1345-1348, 1982.
[11] M. Dentino, J. McCool, and B. Widrow, "Adaptive filtering in the frequency domain", *Proc. IEEE* , vol 66, pp. 1658-1659, 1978.
[12] E.R. Ferrara, "Fast implementation of LMS adaptive filters", *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol ASSP-28, no. 4, pp. 474-475, 1980.
[13] J-C. Fort, "Stability of the Source Separation Algorithm of Jutten and Herault", Artificial Neural Networks, T. Kohonen, K. Makisara, O. Simula, and J. Kangas (Eds), Elsevier Science Publishers B.V. (North-Holland), pp. 937-941, 1991.
[14] J.C Lee, C.K. Un, and D.H. Cho, "A frequency-weighted block LMS algorithm and its application to speech processing", *Proc. IEEE*, vol. 73, no. 6, pp. 1137-1138, 1985.
[15] J.C Ogue, T. Saito, and Y. Hoshiko, "A fast convergence frequency domain adaptive filter", *IEEE Trans. Acoust. Speech, Signal Proc.*, vol. ASSP-31, no. 5, pp. 1312-1314, 1983.
[16] J.C. Platt and F. Faggin, "Networks for the separation of sources that are superimposed and delayed", *Advances in Neural Information Processing Systems 4*, J.E. Moody, S.J. Hanson, R.P. Lippman (Eds), Morgan-Kauffmann Publishers, San Mateo, Cal., pp. 730-737, 1992.
[17] J.J. Shynk, "Frequency-Domain and multirate adaptive filtering", *IEEE SP Mag.*, pp. 14-37, 1992.
[18] J.J. Shynk, Personal Communication.
[19] P.C.W. Sommen, P.J. Van Gerwin, H.J. Kotmans, and A.J.E.M. Janssen, "Convergence analysis of a frequency-domain adaptive filter with exponential power averaging and generalized window function", *IEEE Trans. Circuits Syst.*, vol. CAS-34, no. 7, pp. 788-798, 1987.
[20] T.G. Stockham, Jr, T.M. Cannon, and R.B. Ingebretsen, "Blind deconvolution through digital signal processing", *Proc. IEEE*, vol. 63, pp. 678-692, 1975.
[21] C. Jutten, and J. Herault, "Blind separation of sources, Part I: an adaptive algorithm based on neuromimetic architecure", *Signal Processing*, vol. 24, pp. 1-10, 1991.
[22] C. Jutten et. al., Proceedings of the Int. Signal Processing Workshop on Higher Order Statistics, Chamrousse, France, July 10-12, 1991.
[23] Haykin S., "Adaptive Filter Theory", 2nd Ed., Prentice-Hall, 1991.
[24] J.L. Lacoume and P. Ruiz, "Separation of Independent Sources from Correlated Inputs", *IEEE Trans. Signal Processing*, vol 40, no. 12, pp. 2074-3078, 1992.
[25] M.A. Lagunas, A. Pages-Zamora, and A. Perez-Neira, "High Order Learning in Temporal References Array Beamforming", Signal Processing VI: Theories and Applications, J. Vandewalle, R. Boite, M. Moonen, A. Oosterlinck (Eds), Elsevier Science Publishers B.V., pp. 1085-1088, 1992.
[26] H.L. Nguyen, C. Jutten, and J. Caelen, "Speech Enhancement: Analysis and Comparison of Methods on Various Real Situations", Signal Processing VI: Theories and Applications, J. Vandewalle, R. Boite, M. Moonen, A. Oosterlinck (Eds), Elsevier Science Publishers B.V., pp. 303-306, 1992.
[27] D.T. Pham, P. Garat, and C. Jutten, "Separation of a Mixture of Independent Sources Through A Maximum Likelihood Approach", Signal Processing VI: Theories and Applications, J. Vandewalle, R. Boite, M. Moonen, A. Oosterlinck (Eds), Elsevier Science

Publishers B.V., pp. 771-774, 1992.
[28] S. Van Gerven, and D. Van Compernolle, "Feedforward and Feedback in a Symmetric Adaptive Noise Canceller: Stability Analysis in a Simplified Case", Signal Processing VI: Theories and Applications, J. Vandewalle, R. Boite, M. Moonen, A. Oosterlinck (Eds), Elsevier Science Publishers B.V., pp. 1081-1084, 1992.
[29] P. Ruiz, et. al. Proceedings of the Int. Signal Processing Workshop on Higher Order Statistics, Chamrousse, France, July 10-12, 1991.
[30] V.C. Soon, L. Tong, Y.F. Huang, and R. Liu, "A Wideband Blind Identification Approach to Speech Acquisition Using a Microphone Array", Proc. ICASSP , vol I, pp. 293-296, 1992.
[31] V.C. Soon, L. Tong, Y.F. Huang, and R. Liu, "A Robust Method for Wideband Signal Separation", Proc. IEEE Int. Symp. Circuits and Systems , vol I, pp. 703-706, 1993.
[32] E. Sorouchyari, "Blind separation of sources, Part III: Stability Analysis", *Signal Processing*, vol. 24, pp. 21-29, 1991.
[33] L. Tong, V.C. Soon, Y.F. Huang, and R. Liu, "Indeterminacy and Identfiability of Blind Identification", *IEEE Trans. Circuits and Systems*, vol. 38, no. 5, pp. 499-509, May, 1991.
[34] L. Tong, Y. Inoue, and R. Liu, "Waveform Preserving Blind Estimation of Multiple Independent Sources", *IEEE Trans. Signal Processing*, vol 41, no. 7, July, 1993.
[35] B. Widrow, J. Glover, Jr., J.M. McCool, J. Kaunitz, C.S. Williams, R.H. Hearn, J.R. Zeidler, E.Dong, and R.C. Goodlin, "Adaptive noise cancelling: principles and applications", *Proc. IEEE*, vol. 63, pp. 1692-1716, 1975.
[36] B. Widrow, J.M. McCool, M. Ball, "The complex LMS algorithm", *Proc. IEEE*, pp. 719-720, 1975.
[37] B. Widrow, S. Stearns, "Adaptive Signal Processing", Prentice Hall, 1985.

IMPROVING THE RESOLUTION OF A SENSOR ARRAY PATTERN BY NEURAL NETWORKS

Christian Bracco, Sylvie Marcos and Messaoud Benidir
Laboratoire des Signaux et Systèmes, E.S.E.
Plateau de Moulon, 91192 Gif–sur–Yvette Cedex, France
Phone :(33) 1 69 41 80 40, Fax :(33) 1 69 41 30 60
E-mail: Marcos@lss.supelec.fr

Abstract- *The sensor array pattern is the spatial response of an array of sensors to an incident monochromatic plane wave. It is known to have a resolution which is a function of the number of sensors. In some applications the number of sensors may be small for technical or economical reasons. We here present and analyse a feedforward neural network structure which is able, by learning, to improve the resolution of the array pattern for a fixed and small number of sensors.*

1 INTRODUCTION

Sensor array signal processing deals with the problem of extracting information concerning radiating sources from signals which are simultaneously received on M spatially distributed sensors . The information of interest is the number of sources and the directions of arrival (DOA) of the transmitted waves with respect to the array. Beamforming is the most usual method for dealing with this problem. It consists in weighting the sensor outputs and in constructing the variance of the resulting output signal as an estimator of the sources DOAs. Beamforming therefore performs a kind of spatial filtering which produces beams in the direction of a number of possible DOAs among which the true DOAs can be found. The array pattern is the response of the beamformer to a monochromatic plane wave. The width of the main lobe in the array pattern which is an approximation of the resolution capability of the array, is an increasing function of the number of sensors. In some applications the number of sensors may be small for technical or economical reasons. However, there are other array processing methods with a higher resolution than the beamforming method [1]. Most of them come from a generalization of the spectral analysis methods. They all rely on the statistical properties of the received signals. Our aim is to design a feedforward neural network with sigmoïds able to produce beams with a higher resolution than the classical beamforming for a fixed and small number of sensors, and which is independent of the statistical characteristics of the received signals. The improvement of the array response to a monochromatic plane wave will only take into account the array manifold. This paper is also devoted to open new insights and viewpoints concerning the use of neural networks as an alternative to the classical DOA estimation methods.

0-7803-2026-3/94 $4.00 © 1994 IEEE

In the following section, we recall the beamforming technique and the classical array pattern. In Section 3, we describe the two proposed neural networks structures and the learning process. In Section 4, the performances concerning both learning and generalization are presented and analysed for each of the proposed two networks. In particular, the choice of the number of hidden neurons and the choice of the learning set are investigated. Section 5 is the conclusion.

2 THE CLASSICAL ARRAY PATTERN

Consider an array of M sensors on which N incident waves impinge ($M > N$). At a given frequency and for a particular snapshot, the M – dimensional vector of the received signals can be written as

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{n} \tag{1}$$

where $\mathbf{A} = [\mathbf{a}(\theta_1), \mathbf{a}(\theta_2), \ldots, \mathbf{a}(\theta_N)]$ is the $M \times N$–dimensional matrix of the steering vectors $\mathbf{a}(\theta_n)$, θ_n being the DOA of the source with respect to the normal of the array, $\mathbf{s}$ is the N–dimensional vector of the complex amplitudes of the sources and $\mathbf{n}$ is a noise vector. Each steering vector is some multidimensional transfer function between the source signal and the signal received on the sensors.
The steering vectors belong to the manifold of the possible wavefronts defined by $\mathcal{A} = \{\mathbf{a}(\theta), \theta \in \Theta\}$ where Θ is the set of the possible DOAs. The definition of this manifold requires that the model of the propagation of the waves and the reception of the signals should be known so that this manifold only depends on the parameter θ.
Beamforming consists in weighting the sensors outputs and in constructing

$$F_{\mathrm{FV}}(\theta) = \mathrm{E}[\mid \mathbf{a}\dagger(\theta)\mathbf{x} \mid^2] \tag{2}$$

where $\dagger$ denotes Hermitian transposition. The beamformer performs a spatial filtering in the direction θ.
Let us assume that the received signal $\mathbf{x}$ corresponds to a deterministic monochromatic plane wave such that $\mathbf{x} = \mathbf{a}(\theta_0)$ in the noise free case. The response of the beamformer (2) denoted by

$$F_{\theta_0}(\theta) = \mid \mathbf{a}\dagger(\theta)\mathbf{a}(\theta_0) \mid^2 \tag{3}$$

is called the array pattern. For example, in the case of a linear array of equispaced sensors for which the intersensor spacing is half the received signal wavelength, the vectors of the manifold $\mathcal{A}$ are of the form

$$\mathbf{a}(\theta) = [1 e^{j\pi \sin\theta} e^{j2\pi \sin\theta} e^{j(M-1)\pi \sin\theta}]^T \tag{4}$$

The array pattern thus becomes

$$F_{\theta_0}(\theta) = \frac{sin(M\pi(sin(\theta) - sin(\theta_0))/2)}{M sin(\pi(sin(\theta) - sin(\theta_0))/2)} \tag{5}$$

The width of the main lobe which is an approximation of the resolution capability of the array, is a function of the number of sensors M. Figure 1 exhibits the array pattern for a linear array with 4 sensors steered in the direction of $\theta_0 = 0$. We can see that the main lobe width is approximately 30 degrees. We can check that two sources spaced by less than 30 degrees cannot be resolved.

Let us depict in Figure 2 a neuronal system which consists of a single layer of quadratic units and which can construct the classical array pattern (2). The weights relating the input $\mathbf{a}(\theta)$ to the quadratic units are steering vectors $\mathbf{a}(\theta_i)$ chosen in the manifold $\mathcal{A}$. The response as a function of θ of each unit i is the array pattern (2) (Figure 1) steered in the direction of θ_i.
We propose to replace the structure of Figure 2 by a two–layer feedforward networks (three layers with the input layer) implementing sigmoïds.

3 THE PROPOSED NEURAL NETWORKS

Two different structures are proposed.
The first one is depicted in Figure 3. It consists of subnetworks, each of them having a single output constrained by learning to give a desired array pattern. Each subnetwork is assigned to one DOA and has its own hidden layer. As it is well–known, a two–layer network can approximate any regular function. We can then train the proposed structure to approximate a desired response for each output unit.
The second structure is depicted in Figure 4. It consists of a single network with multiple outputs and with a single hidden layer shared by all the outputs. Each output is assigned to one DOA. The number of outputs is equal to the number of DOAs to which an array pattern is steered.
In both structures, input i of the network receives the information delivered by sensor $i+1$ of the array. We do not take into account the output of the reference sensor, which is a constant. Actually, the network inputs consist of the real and imaginary parts of the sensors outputs. The network therefore contains $2(M-1)$ inputs.
In order that the network finds out the DOAs θ_i of the N underlaying sources impinging on the array, it is necessary to train it by supervised learning. The training set consists of vectors $\mathbf{a}(\theta)$ of the manifold $\mathcal{A}$, corresponding to different values of θ in the range of [0, 60] degrees.
The target vector $\mathbf{t}(\theta)$ stands for the desired output of the network.
When vector $\mathbf{a}(\theta)$ is presented at the input of the network, the target $t_i(\theta)$ of the output neuron i assigned to a DOA θ_i, is a value of a desired pattern $f_{\theta_i}(\theta)$. An example of a desired pattern corresponds to the combination of a Gaussian function with mean θ_i and standard deviation $\delta\theta$, with a hyperbolic tangent function. Function $f_{\theta_i}(\theta)$ is sampled according to the angles present in the training set. Note that learning a target equal to 1 as a desired response when $\mathbf{a}(\theta_i)$ is present at the input of the network and equal to 0, when it is not present, was found to introduce oscillations during the generalization phase.
Learning is based on the multidimensional minimization of the cost function

$$C(W,B) = \sum_{\text{samples of } \theta} ||\mathbf{s}_{W,B}(\theta) - \mathbf{t}(\theta)||^2$$

where $\mathbf{s}_{W,B}(\theta)$ is the response computed by the network when $\mathbf{a}(\theta)$ is at the input, and W, B stand for the weights and the biases of the network, respectively. Each neuron implements a sigmoid as an activation function. The minimization of the cost function and, consequently, the adaptation of the weights and the biases are performed by the Levenberg-Marquardt algorithm [2]. The weights and biases are randomly initialized within [-1, 1] with a uniform probability.
Let us make some comments concerning the range of the angles present in the training set. Indeed, according to (4), the inputs of the network are of the form

$$f_m(\theta) = \cos(\pi m \sin\theta) \quad g_m(\theta) = \sin(\pi m \sin\theta) \tag{6}$$

These functions oscillate for θ in the range of [0, 60], even for small values of m. Between two successive extrema of the oscillations, there is a linear part that we refer to as a "slope". We have observed that the network performs its response by linear combinations of these slopes. The more elementary slopes there are, the easier it is for the network to build intermediary responses in its hidden layer to fit the desired output. In the range of [60, 90], there are fewer oscillations. The networks under consideration were clearly found to converge more slowly than for the range [0, 60].
Another comment is that we need to use both real and imaginary parts of the input vector to avoid any ambiguous behaviour.
The next Section is devoted to the analysis of the performances of the proposed networks. This is done with the help of computer simulations.

4 SIMULATION RESULTS

4.1 First Network consisting of subnetworks

As the subnetworks are independent, we here investigate the performances of one of the subnetworks.
The network has 4 input neurons corresponding to a 3-sensor array. The subnetwork consists of 6 hidden neurons. It is constrained to have an array pattern whose main lobe is 8 degrees. This resolution is much thinner than the resolution of the classical array pattern.
First, the subnetwork is trained by presenting at the input a vector $\mathbf{a}(\theta)$ corresponding to a single source with DOA θ taking its value within the range of [0, 60]. The DOAs which are presented during the learning phase are represented by small circles in Figure 5a. The convergence of the network is obtained in 150 epochs. We stop the algorithm when the total error is 0.09. Note that the input vector $\mathbf{a}(\theta)$ is normalized before being presented to the network in order to have the same response of the network, whatever the amplitude of the source. Figure 5a exhibits the array pattern obtained after learning. Clearly the network performs satisfactorily when inputs, which were not in the training set, are presented.
Now let us present at the input of the trained subnetwork, the vectors $\mathbf{a}(\theta) + \mathbf{a}(\theta_{\text{int}})$ where θ_{int} is the DOA of an interference source and where θ scans the range of [0, 60]. Figures 5b exhibits the response of the network steered in the direction of 20 degrees, for $\theta_{\text{int}} = 0$. It appears that the pattern in Figure 5b is deteriorated compared to the pattern of Figure 5a.
In order to compensate for this deterioration, the learning of the subnetwork is performed by presenting two sources at the input. The training set now consists of vectors of the form $\mathbf{a}(\theta_i) + \mathbf{a}(\theta_j)$, θ_i and θ_j scanning [0, 60]. The input vector is still normalized before being presented to the network. As the subnetwork has a single output, we have to introduce a single target. The value of the target is computed as follows

$$t_2(\theta_i, \theta_j) = max(t(\theta_i), t(\theta_j))$$

where $t(\theta_i) = f_{\theta_0}(\theta_i)$ is the desired array pattern steered to the DOA θ_0. This criterion which may not be the best one, especially allows us to take a decision when both θ_i and θ_j are in the same lobe.
There is a compromise to do between a thin sampling in order to avoid oscillations of the response during generalization, and a larger sampling to decrease the convergence time. We here make the following choice. For angles in the

range of [16, 36], the sampling step is 2 degrees inside the lobe and 4 outside. The hidden layer of the subnetwork consists of 4 neurons. The training is achieved after 10 seconds CPU time on a HP 735 for a squared error of 0.1 . For angles close to 0 or 60 degrees, as the oscillations of $f_m(\theta)$ and $g_m(\theta)$ are scarce, we need more hidden neurons. In adding neurons, we observe secondary lobes due to an interference. This requires a thinner sampling. We choose 2 degrees in the lobe and 1 degree outside it. CPU time is then around 10 mn for the same error.

Figure 6 exhibits the response of the subnetwork steered to 20 degrees when an interference source is presented. By opposition to Figure 5, the array pattern is found correct for every interference source. If the interference source is outside the main lobe, the array pattern keeps its form when θ scan the range of [0, 60]. See, for example, the case $\theta_{\text{int}} = 0$. When the interference source is inside the lobe, the subnetwok computes a value as close as possible to $f_{\theta_0}(\theta_{\text{int}}$. For example, in the case of $\theta_{\text{int}} = 20$, the response of the subnetwork remains approximately 1 as θ scan the range [0, 60]. Note that the generalization to the case where the two sources presented at the input of the subnetwork have different amplitudes failed.

Figure 7a illustrates the response of the entire network containing all the subnetworks, when one source is presented at the input with DOA θ scanning the range of [0, 60]. The training set included two parts. The first one consisted of the presentation of two sources and the second part consisted of the presentation of only one source. The second part is redundant but seems to have stabilizing effects on the generalization. We can see the 16 output neurons responses plotted together on a same graph. The overlapping of the lobes is such that the output neuron recognizes the DOA to which it has been assigned when its response is larger than 0.7. Note that the learning might have been pursued at the cost of an increase in complexity and convergence time.

Figures 7b–c exhibit the network response when two sources are presented during the generalization phase. In the first case, the interference source belongs to the training set while, in the second case, it does not belong to it. In both cases, the network performs satisfactorily.

4.2 The second network consisting of a single network

This network has a number of outputs equal to the number of DOAs for which we want to design array patterns. During the training, output neuron i is set to one when a source with DOA θ_i is presented at the input of the network, and zero when the source has DOA θ_j with $j \neq i$.

As shown in Figure 8a, the network creates its own lobes. A bad choice of the number of hidden neurons may be balanced, to a certain extend, by constraining the response of the network to have a desired shape around θ_i. This constraint helps the network to converge when the number of hidden units is too small and it does not prevent the network to converge when the number of hidden units is sufficient. The desired shape of the response is a part of the hypertangent-Gaussian function as already used for the first network.

An experimental result is that the number of hidden neurons is approximately $N_{\text{h}} = \frac{1}{2}C^1_{N_{\text{o}}} = \frac{N_{\text{o}}}{2}$ where N_{o} is the number of outputs. The responses of all the outputs of the network to the presentation of one source the DOA of which scans [0, 60], are plotted together on the same graph in Figure 8a.

As for the first network, the generalization capacity of this network fails when two sources are presented at the input. It is therefore necessary to process the learning with two sources presented at the input. In this case, the number of neurons in the hidden Layer was found to be approximately $N_h = \frac{1}{3}C^2_{N_{\text{o}}}$. Note that, in both evaluations of N_{h}, C^p_N which is the number of different

combinations of p elements among N elements, is the length of the training set. The coefficient 1/2 or 1/3 characterizes symmetries that the network seems to create for the corresponding training set.
In order to avoid oscillations in the generalization response, it is necessary as in the previous case to constrain the responses of the outputs to a desired shape around the DOA to which they are steered. To facilitate the convergence we introduce an additive sensor, i.e. an additive network input.
Figure 8b exhibits the behaviour of the network output when a source with DOA ranging over [0, 60] and an interference source with fixed DOA θ_{int} are presented at the input. The network here has 6 outputs. For example, when $\theta_{\text{int}} = 5$, the response of the output neuron assigned to the DOA 5 degrees is almost 1 when the DOA of the other source is scanning the range of [0, 60], whereas the responses of the other outputs have the form of the desired pattern.

5 CONCLUSION

The present paper is a preliminary work of a more general study of the possible use of neural networks as an alternative to the classical DOA estimation methods. The first advantageous consequence of the present work is that the necessity in the classical approach of increasing the number of sensors to reach a desired resolution is replaced by the use of feedforward neural networks and by the learning of a desired pattern with a given resolution. This can be of great interest when the use of a great number of sensors is not possible for economical or technical reasons. Another advantage of our approach is that, once the neural network is matched to a given manifold of possible received signals, the detection of sources and the DOA estimation can be performed in real time which is actually not the case with the classical high resolution methods.
In this paper we have presented two structures of feedforward networks to localize sources. We have experimentally investigated the choice of the number of hidden units and the choice of the training set in order to not only improve the resolution of the array pattern, given the number of sensors, but also to make the network behave satisfactorily during generalization. The design and the convergence of the first network consisting of subnetworks were found much simpler than those of the second proposed network consisting of a single network. In the second case, the convergence quickly becomes very slow, and we need, for example, more than one hour of CPU time on a HP 735 to have access to networks with more than ten outputs.
In order to further improve the capacity in generalization of the proposed networks, it is required to include in the training set examples of combinations of an arbitrary number of sources with different amplitudes.

References

[1] Don H. Johnson, Dan E. Dudgeon, Array signal processing. Concepts and techniques *Prentice Hall Signal Processing Series, A.V.Oppenheim editor*, 1993.

[2] MATLAB, *Neural Networks Toolbox*, 1993.

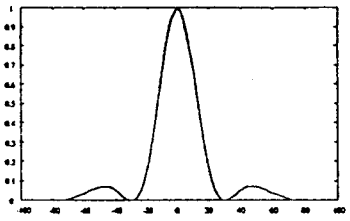


Figure 1 : The classical array pattern.

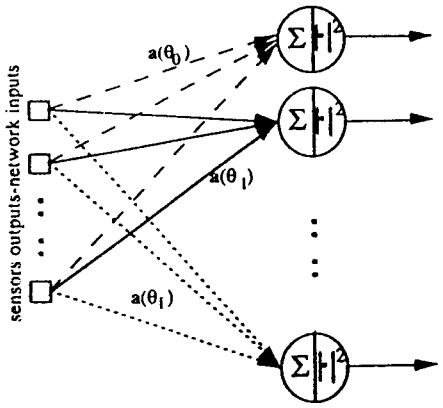


Figure 2 : The neuronal representation of the beamformer.

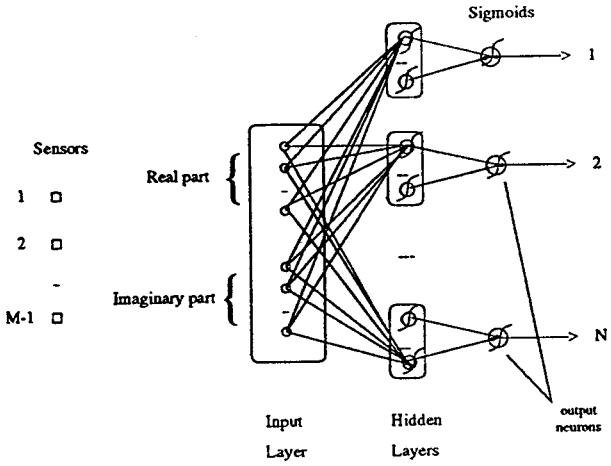


Figure 3 : Network consisting of sub-networks.

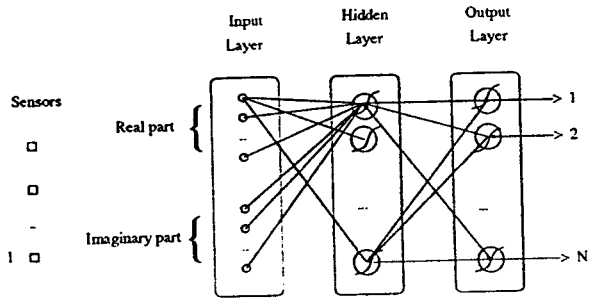


Figure 4 : Single network with multiple outputs.

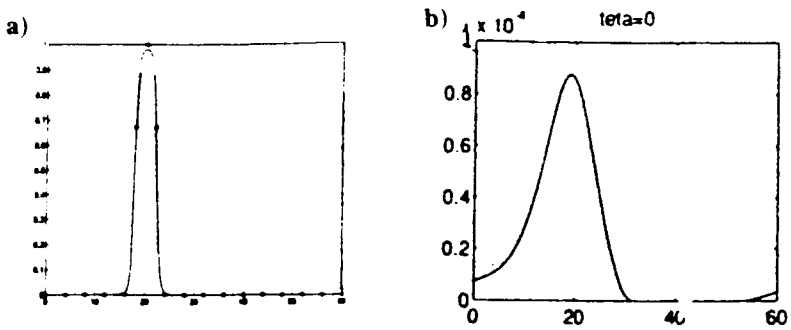


Figure 5 : Array pattern of a sub-network steered in the direction 20 deg. Learning is performed with a single source present at the input.
a) 'o' : angles in the training set, plain line : response of the network in the generalization phase with one source present,
b) 2 sources are presented with θ_{int}=0.

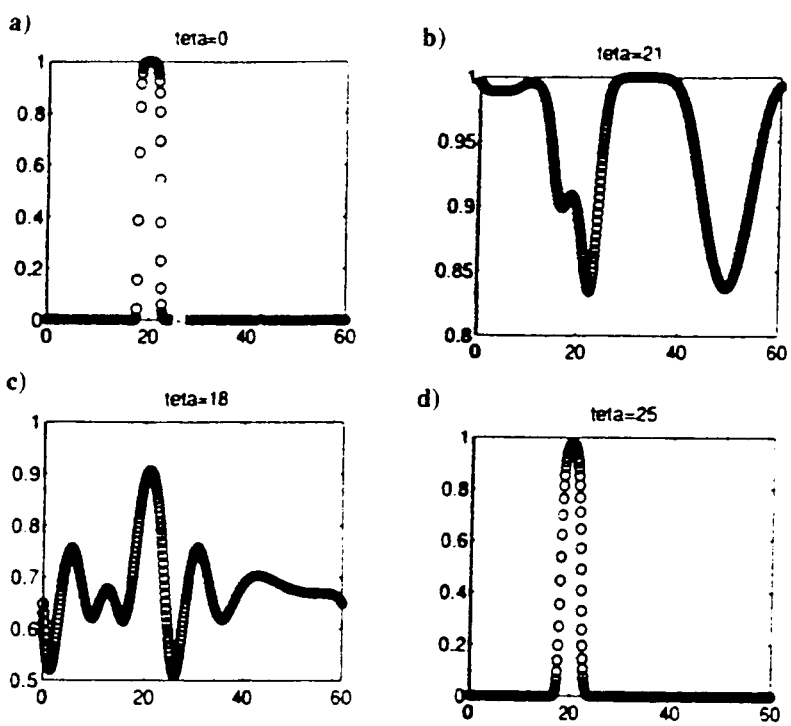
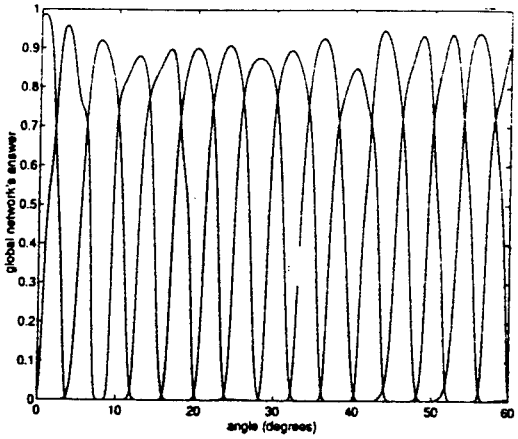
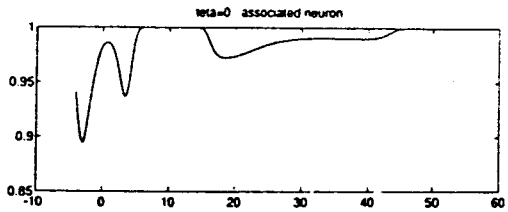


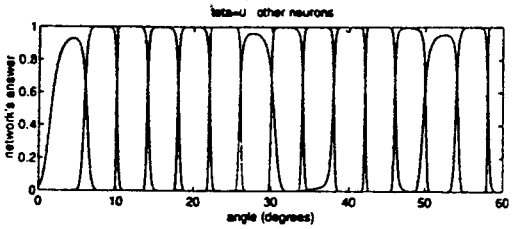
Figure 6 : Array pattern of a sub-network steered in the direction 20 deg Learning is performed with 2 sources at the input.
a) θ_{int}=0, **b)** θ_{int}=18, **c)** θ_{int}=21 , **d)** θ_{int}=25 .
θ=21 and θ=25 do not belong to the training set.

Figure 7 : Array patterns of all the sub-networks plotted together. Learning is performed with 2 sources.

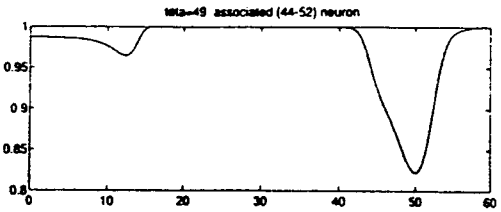


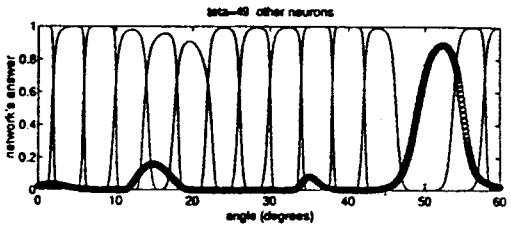
a) one source is presented to the network





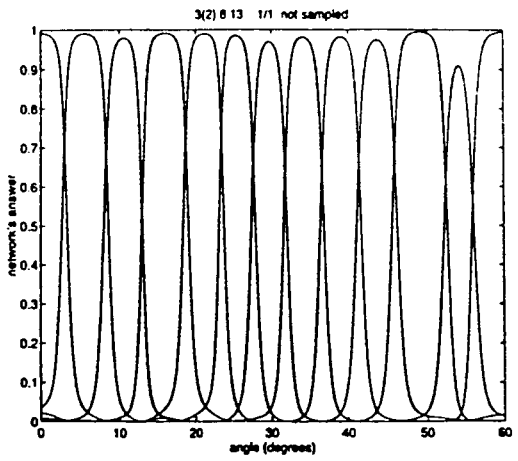
b) 2 sources are presented to the network and θ_{int}=0, which belongs to the training set.



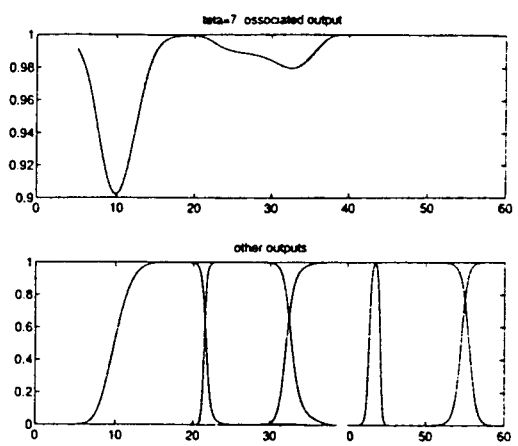


c) θ_{int}=49, which does not belong to the training set.

Figure 8 : Array patterns at the ouput of the network.



a) the learning is performed with one source and the target is 0 or 1, and one source is presented to the network



b) 2 sources are presented to the network and θ_{int}=7. learning is performed with 2 sources and θ_{int}=7 does not belong to the training set.

Other Applications

SENSITIVITY ANALYSIS ON NEURAL NETWORKS FOR METEOROLOGICAL VARIABLE FORECASTING

Castellanos, J**.; Pazos, A.*; Ríos, J.**; Zafra, J. L.**

**Facultad de Informática - Universidad Politécnica de Madrid
Campus de Montegancedo, s/n. Boadilla del Monte.
28660 Madrid (SPAIN)
E-MAIL: jcastellanos@fi.upm.es FAX: 34-1-3367412
*Universidade da Coruña. Castro de Elviña, 76. 15071 - A Coruña

Abstract - The problem that arises in a neural network with many inputs is being able to eliminate the irrelevant ones. In the particular case of short-term weather forecasting, there are variables that may have little or no impact on the forecasts. A technique of sensitivity analysis of outputs over inputs has been applied to the trained network. Thus the most relevant inputs have been determined, as have less important inputs that can be eliminated. By employing this technique, a smaller sized neural network is obtained which also has a greater capacity for generalization.

1. INTRODUCTION

Neural networks have proven their effectiveness in predicting the future behaviour of time series [5, 7]. In this paper, the time series traces the behaviour of a meteorological variable. Many studies have been conducted on weather forecasting in this century. Highly complex statistical models, such as those by Box and Jenkins [1], have come up with the best results in wide area weather prediction, employing a large amount of different data from satellites, ground stations, weather balloons, etc.

Our research has centred on forecasting one meteorological variable, using data collected at one ground weather station. Values of this variable and other complementary variables collected beforehand are used to this end. This complicates the already complex weather forecasting problem still further as local data are used. The meteorological variable selected for forecasting was temperature.

Several training sets were selected, where the number of days' recordings supplied to forecast the following day's observations varied. The

0-7803-2026-3/94 $4.00 © 1994 IEEE

sets also differed as to the number of meteorological variables they contained. Several network structures were tested as part of a multilayered perceptron. The number of hidden layers, the amount of neurons and the interconnection schema were varied. The learning process was carried out using the gradient backpropagation algorithm [2, 6].

Having observed that the various models receiving different information led to similar results, an empirical study was undertaken to determine the minimum number of necessary variables that had to be supplied to a neural network to get similar performance. The determination of the minimum set was to offer two major advantages. The first is the increase in efficiency and in the capacity for generalization produced by using a smaller sized network. The second benefit is the result of studying the relationship between the different variables, which may bring out knowledge that is possibly hidden by surplus information.

The technique employed to conduct the sensitivity analysis is based on the calculation of the partial derivative of the outputs over inputs to the network. This method of analysing the inputs to a neural network has been successfully applied to discriminate the relevant variables in modeling the behaviour of a nuclear power plant [4].

The derivative of the output of the network over the inputs is determined taking a description of the activation calculation of any neuron in the network:

$$o_j^n = f(net_j) = f(\sum_i w_{ji} o_i^{n-1}) \tag{1}$$

where o_j^n, o_j^{n-1} are outputs of the jth neuron of layer n and the ith neuron of layer n-1 of the network, respectively, f is the activation function and w_{ji} is the weight of the connection between the ith and the jth neurons.

To develop the expression of the partial derivative of the output of the network over the inputs:

$$\frac{\partial o_j^n}{\partial o_k^0} = \sum_i \frac{\partial o_j^n}{\partial o_i^{n-1}} \frac{\partial o_i^{n-1}}{\partial o_k^0} \tag{2}$$

The first term of the second member of the equation can be solved using equation (1):

$$\frac{\partial o_j^n}{\partial o_i^{n-1}} = w_{ji} f'(net_j) \tag{3}$$

Thus, the first term can be calculated from the known data at the level of neuron j of layer n. On the other hand, the second term can only be solved recursively by successively applying the formula:

$$\frac{\partial o_j^n}{\partial o_k^0} = f'(net_j) \sum_i w_{ji} \frac{\partial o_i^{m-1}}{\partial o_k^0} \tag{4}$$

The calculation process ends when the partial derivative of layer 1 over layer 0 (input) has been found. This is calculated using the expression:

$$\frac{\partial o_j^1}{\partial o_k^0} = f'(net_j) w_{ji} \tag{5}$$

Thus, the partial derivative of output over input would be calculated beginning with layer 1 and applying equation (5) and would be propagated to subsequent layers employing equation (3).

The process would work as follows. A pattern is presented to the network and is forward propagated, calculating the input derivatives for all the neurons. Then, the values of the partial derivatives are forward propagated for each neuron of the input layer until the desired output is obtained. This consists in the partial derivative of each unit of the output layer over that unit of the input layer. The values thus obtained are averaged out to an absolute value for all the outputs to obtain the importance of the input as compared to all the outputs.

2. DESCRIPTION OF THE DATA AND THE NEURAL NETWORK USED

The data employed in this study were collected at Barajas airport (Madrid). They were in METAR format, the code used to encode ordinary meteorological observations for aviation. The following five meteorological variables were chosen form the great abundance available: pressure, dry temperature, visibility and wind direction and speed. The observations were made every half an hour and data covering seven years were available, of which six have been employed in learning and the remaining year was reserved as a test set to check learning quality. The choice of this number of years, as well as the cadence of observations, amounting to a total of 48 observations a day per variable, give some idea of the huge size of the training sets.

The data were standardized taking the maximum and minimum values over the whole period and linearly mapped to the interval [-1, 1]. The

variable chosen for forecasting was temperature. Indeed, the aim was to forecast the 48 observations for one day, on the basis of the values of given meteorological variables, including temperature. The different observations matched a given number of days (1 to 3). Different combinations of the variables were tried, whereby at least three and at most the five of above-mentioned variables were always taken.

The architectures employed ranged from networks with no hidden layers to partially connected four-layer networks.

As regards the activation function used, different functions were assessed in this case too in order to compare their performance. These three functions were:

- **Systole 2**: This function discussed in [3] is defined as $f(x) = 2.5xe^{-x^2}$. It has the advantage of considerably reducing the number of iterations needed for the network to converge on a valid solution.
- **Sigmoidal function**, defined as:

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{6}$$

- **Wide sigmoid**, with a wider output range [-1.71, +1.71]. It is defined as:

$$f(x) = 1.71\frac{1 - e^{-1.33x}}{1 + e^{-1.33x}} \tag{7}$$

The model's error was measured in two ways. The first of these is the mean square error:

$$E_T = \frac{1}{P}\sum_{j}^{P} E_j = \frac{1}{P}\sum_{j}^{P}\sqrt{\frac{1}{N}\sum_{i}^{N}(d_i - o_i)^2} \tag{8}$$

where E_T is the total error for all of the patterns P, E_j is the error for the ith pattern, N is the length of the output layer, and d_i and o_i are the desired and real outputs, respectively. Standardized data were measured. The second of these measures is the percentage of the forecasts that are no further than 0.02 from the real value, also in the case of standardized data. This deviation, translated to the dynamic temperature range, represents a difference of at most half a degree centigrade.

The neural networks were trained using the gradient backpropagation algorithm. Learning was considered to be complete when test set performance was found to fall parallel to a fall in learning set error. The intention here was to maintain the network's generalization capacity.

A three-layer multilayered perceptron was selected from the networks providing higher performance, as it had a higher number of input variables and a more general structure. The sensitivity analysis was applied to this network in particular. The size of the input layer was 720 neurons (48*3*5), as it received three days' data on the five variables, and it was fully connected to the intermediate and the output layers. The intermediate layer had 20 neurons, fully connected to the output layer. The output layer was made up of 48 neurons, the same number as the temperature observations to be forecast per day.

The activation function that performed best was the above-mentioned Systole 2. Thanks to this function, fewer learning iterations were needed to obtain a mean square error level of 0.011671 in learning and 0.013014 in testing. These values are equivalent to a success rate of 94.57% and 90.08% for the learning and test sets, respectively. The above values were the best obtained under the stoppage criterion established to prevent the neural network's generalization performance from falling. The sensitivity analysis was applied to the network trained thus.

3. SENSITIVITY ANALYSIS RESULTS

The first study conducted on the basis of the sensitivity analysis was to determine the relative importance of each meteorological variable in forecasting temperature. The results are shown in Figures 1 to 5. These graphs plot the variable observations over the three days on record along the abscissa axis, the third day being the most recent. The variable's influence is represented along the ordinate axis. Note that all the graphs have been drawn to the same scale.

The results to be inferred from the analysis of the graphs would appear to be contrary to intuition, as the importance of atmospheric pressure (Figure 1) is negligible, for example, the visibility variable being more important (Figure 3).

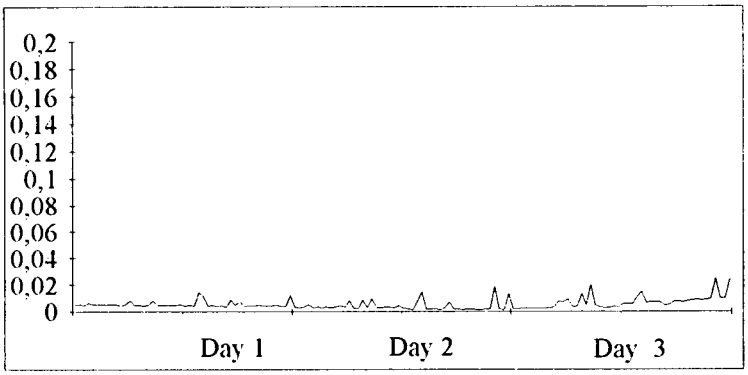


Figure 1. Importance of Pressure

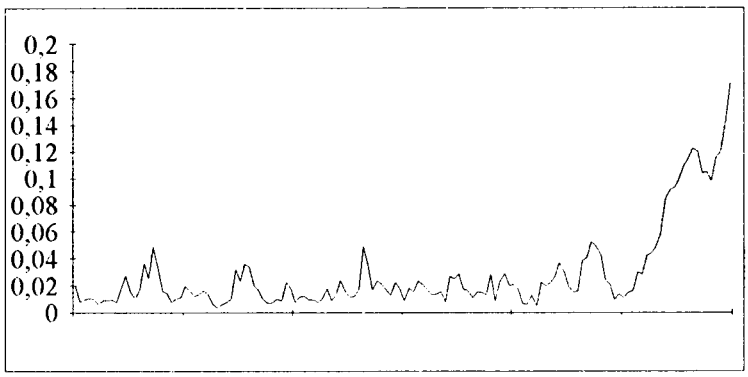


Figure 2. Importance of Temperature

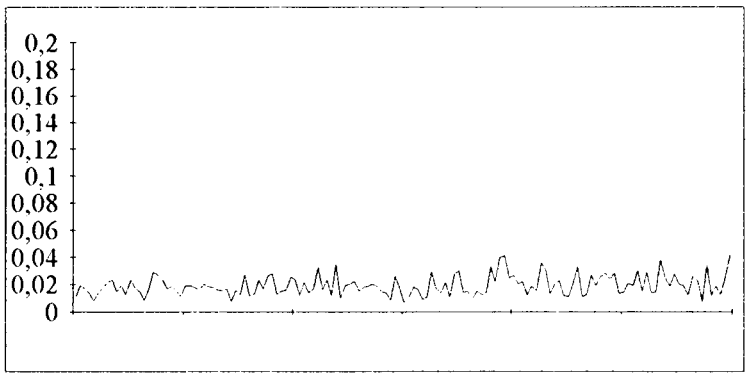


Figure 3. Importance of Visibility

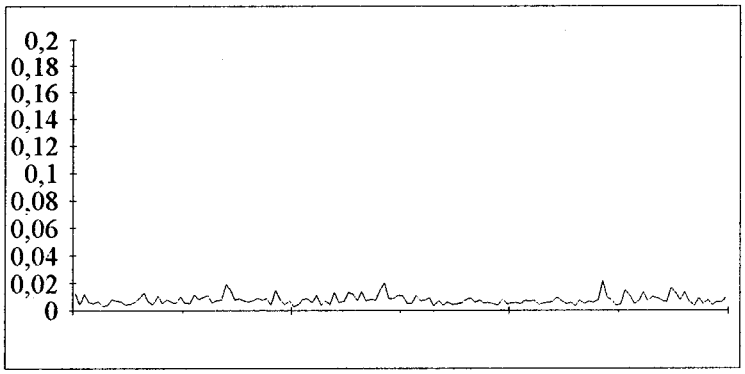


Figure 4. Importance of Wind Direction

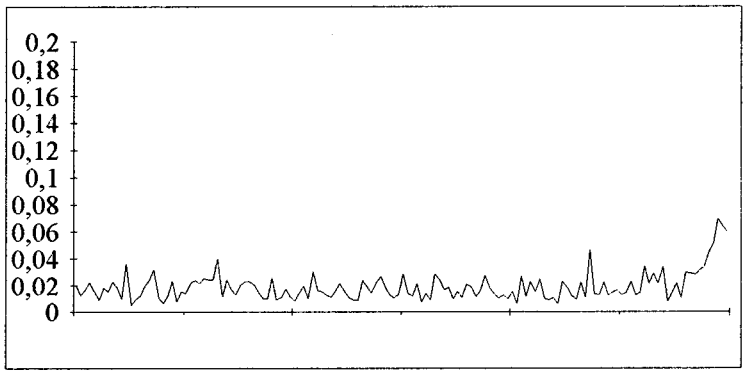


Figure 5. Importance of Wind Speed

The most important variable by far is temperature (Figure 2), though it is found to be of maximum importance in the later observations of the period.

As regards wind, it should be pointed out that wind direction (Figure 4) has little influence on forecasting and is of similar importance to atmospheric pressure. Wind speed (Figure 5) is considerably more influential, being of relatively significant importance in the later observations of the period.

The analysis of the graphs provides a guide for proceeding to eliminate given variables. They could have been eliminated manually, but the use of an automatic procedure to eliminate the less significant variables was preferred. The inputs were ordered on the basis of their importance, without taking into account the variable in question. Two networks have been generated on the basis of this classification, eliminating the less influential inputs and the connections between these and other neurons. Sixty per cent of the inputs were eliminated in the first and 85 % in the second. The

remaining network structure was maintained, as were the weights of the best performance obtained. The thresholds chosen led to the effective elimination of given variables, such as pressure.
The performance of the networks built thus was assessed in comparison to the original network, and it was found to fall. Bear in mind that the elimination of inputs and the resulting loss in connections produces an imbalance in the trained network. This phenomenon was solved after a short process of adjustment of the original weights (5 learning iterations using backpropagation) to obtain much improved performance. The comparison between the three neural networks is shown in Table 1. The performance of the learning set (Tr) and the test set (Tst) of the original network are shown, as is the performance of the nets after 60 % and 85 % of their inputs had been eliminated, both before and after the above-mentioned adjustment. It should be pointed out that network performance with 60 % elimination of inputs is fairly good compared with the original network. This gives some idea of just how unimportant the eliminated inputs are:

NETWORK	EAN SQUARE ERROR	FITNESS %
First Net (Tr.)	0.011671	94.57
First Net (Tst.)	0.013014	90.08
60% Net (Tr.)	0.015557	77.96
60% Net (Tst.)	0.015178	80.72
60% Net (5 It., Tr.)	0.011733	92.62
60% Net (5 It., Tst.)	0.011563	92.84
85% Net (Tr.)	0.025433	30.84
85% Net (Tst.)	0.024396	36.36
85% Net (5 It., Tr.)	0.012983	88.58
85% Net (5 It., Tst.)	0.012847	92.01

Table 1. Neural Network Comparison

In view of the performance offered by the less complex networks, it can be said that the process of eliminating inputs to forecast the temperature variable has been a success. The networks obtained using this method are an improvement on the original network in terms of generalization capacity, while there is a slight fall in learning set performance. This phenomenon can be explained by the smaller size of the network, which means that it does not concentrate on the peculiarities of the learning set but on the useful features that are repeated.

4. CONCLUSIONS

In this paper, a technique for forecasting an atmospheric variable is presented, which eliminates given input data and improves network generalization, leading to savings in computing time and the number of connections.

These results may also be applied to the other atmospheric variables, and data from other stations may be used to confirm the relative importance of given variables over others for one area or application field.

ACKNOWLEDGEMENT

This paper was prepared and written with the collaboration of CETTICO (Centre for Technology Transfer of Knowledge Engineering, Spain).

REFERENCES

[1] Ameen, J.R.M. "Discount Bayesian models and forecasting." PhD Dissertation. University of Warwick. 1984.

[2] Rumelhart, D., Hinton, G. and Williams, R. "Learning Internal representations by error propagation." *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, D. R. Rumelhart & J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986.

[3] Segovia, J. "Redes de Neuronas Recurrentes para el Reconocimiento de Patrones Temporales." PhD Dissertation. Madrid: Universidad Politécnica de Madrid. 1992.

[4] Uhrig, R. E. et al. "Enhancing Nuclear Power Plant Performance through the Use of Artificial Intelligence." Final Report. Department of Nuclear Engineering. University of Tennessee. 1992.

[5] Weigend, A., Huberman, B., Rumelhart, D. "Predicting Sunspots and Exchange Rates with Connectionist Networks." *Nonlinear Modeling and Forecasting, SFI Studies in the Science of Complexity*, Proc. Vol. XII, Eds. M. Casdagli & S. Eubank. 1992.

[6] Werbos, P. J. "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences." PhD Dissertation. University of Harvard. 1974.

[7] White H. "Economic prediction using networks the case of IBM daily stock returns," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, 1988, pp. II-451 - II-459.

Continuous-time nonlinear signal processing: A neural network based approach for gray box identification*

R. Rico-Martínez, J. S. Anderson and I. G. Kevrekidis
Department of Chemical Engineering, Princeton University,
Princeton NJ 08544

Abstract

Artificial neural networks (ANNs) are often used for short term discrete time series predictions. Continuous-time models are, however, required for qualitatively correct approximations to long-term dynamics (attractors) of nonlinear dynamical systems and their transitions (bifurcations) as system parameters are varied. In previous work we developed a black-box methodology for the characterization of experimental time series as continuous-time models (sets of ordinary differential equations) based on a neural network platform. This methodology naturally lends itself to the identification of partially known first principles dynamic models, and here we present its extension to "gray-box" identification.

1 Introduction

Artificial Neural Networks (ANNs) have proven to be a valuable tool in nonlinear signal processing applications. Exploiting ideas common to nonlinear dynamics (attractor reconstruction) and system identification (ARMA models), methodologies for the extraction of nonlinear models from experimental time series have been developed (e.g. [1, 2]) and applied to experimental data. In previous work, we have discussed some inherent limitations of these techniques (based on *discrete-time* schemes) in characterizing the instabilities and bifurcations of nonlinear systems depending on operating parameters.

An alternative approach, resulting in *continuous-time* models (sets of Ordinary Differential Equations (ODEs)), also based on a neural network platform, was devised and implemented [3, 4, 5]. The approximations constructed in that

*This work was partially supported by ARPA/ONR, the Exxon Education Foundation and an NSF PYI award. RRM acknowledges the support of CONACyT through a fellowship.

work can be described as black-box; no insight from first principles modeling of the system was incorporated in them.

In this work we extend the approach to cases where portions of the algebraic forms of the set of ODEs describing the dynamical evolution of the system are known. We attempt to capture the behavior of the overall system by "hard-wiring" the known parts and approximating the unknown parts using a neural network (gray-box identification).

In what follows we first briefly outline our black-box approach for the identification of continuous systems. This discussion naturally leads to the extension to gray-box identification. Finally, we illustrate its use through an application to the modeling of a reacting system with complicated nonlinear kinetics.

2 Black-box approach

Consider the autonomous ODE

$$\dot{\vec{X}} = \mathbf{F}(\vec{X};\vec{p}) \tag{1}$$

$$\vec{X} \in \mathcal{R}^n, \qquad \vec{p} \in \mathcal{R}^p, \qquad \mathbf{F} : \mathcal{R}^n \times \mathcal{R}^p \mapsto \mathcal{R}^n$$

where $\vec{X}$ is the vector of state variables, $\vec{p}$ is the vector of operating parameters and $\dot{\vec{X}}$ is the vector of derivatives of the state variables with respect to time. In previous work we showed a way of constructing such a set of ODEs from discrete-time experimental measurements of the state variables only ([3, 4, 5], see also [6]). We embedded the training of a neural network that approximates the function $\mathbf{F}(\vec{X};\vec{p})$ in a numerical integrator scheme. Both explicit and implicit integrators can be (and have been) used. In addition, we illustrated how the approach can be used when time series of only a single state variable are available.

Consider the simple implicit integrator (trapezoidal rule) formula for Eq. 1:

$$\vec{X}_{n+1} = \vec{X}_n + \frac{h}{2}[\mathbf{F}(\vec{X}_n;\vec{p}) + \mathbf{F}(\vec{X}_{n+1};\vec{p})] \tag{2}$$

where h is the time step of the integration, $\vec{X}_n$ is the value of the vector of states at time t and $\vec{X}_{n+1}$ is the (approximate) result of integrating the set of ODEs to time $(t+h)$. Figure 1(a) schematically depicts a neural network constructed using this numerical integrator as a template. The boxes labeled "neural network" represent the *same* neural network evaluated with two different sets of inputs for each training vector. Given the implicit nature of the integrator, the "prediction" of the integration depends on itself. Training was therefore done using standard recurrent network training ideas [7, 8]. Alternatively, a nonlinear algebraic equation solver can be used, coupled with the training, to solve exactly for the predicted value at every iteration and for every training

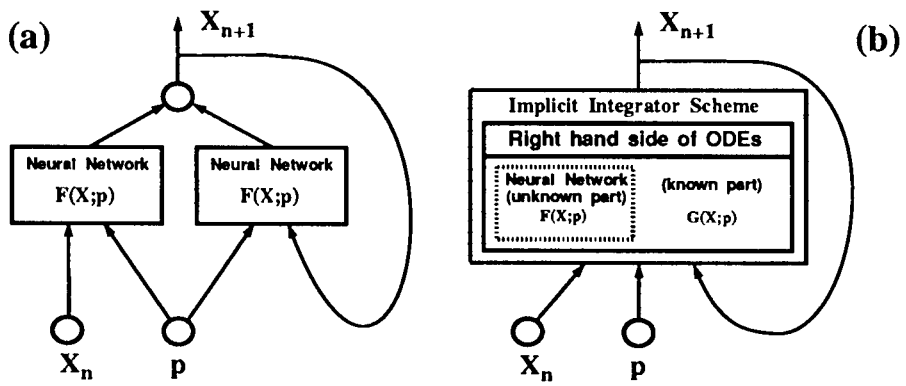


Figure 1: (a) Schematic of the evaluation of a neural network embedded in the implicit integrator (trapezoidal rule) of Eq. (2). The implicit dependence of the prediction of the state on itself results in a backward (recurrent) connection. (b) Schematic of the evaluation of a neural network for the gray-box approach. The known part of the model ($G(X;p)$) is evaluated along with the unknown part ($F(X;p)$), approximated by a neural network. In order to calculate errors (for training) the contribution of the known and unknown parts are combined using the integrator to give the state of the system at the next sampling time.

vector. Further details can be found in references [4, 5]. The use of explicit integrators is discussed in [3]. This identification procedure has been tested for experimental systems exhibiting complicated dynamics (see e.g. [3, 4]).

3 Gray-box approach

The approach discussed above can be combined with first principles modeling for cases where the full state vector is known while the understanding of the modeling of the system is only partial. Such an example is encountered in modeling reacting systems, when the kinetics of the reaction are not known *a priori* while inflow and outflow or heat transfer are well understood and easily modeled.

As in the case of black-box approximations, we embed the training of the neural network in a numerical integrator scheme. For gray-boxes, the known part of the right-hand-side of the ODEs is explicitly calculated ("hardwired") and the neural network is trained to approximate only the unknown parts.

Let us assume for the purposes of the illustration presented here that the first principles model of a given system takes the simple form:

$$\dot{\vec{X}} = \mathrm{G}(\vec{X};\vec{p}) + \mathrm{F}(\vec{X};\vec{p}) \tag{3}$$

where $\mathbf{G}(\vec{X};\vec{p})$ represents the known part of the model and $\mathbf{F}(\vec{X};\vec{p})$ is the unknown part. Note that the methodology is not restricted to models of the additive form of Eq. (3).

Figure 1(b) schematically depicts the training procedure for an implicit integrator. A "global" network is used to predict the state at the next time step. Some of the weights and nodes in this network are fixed because of the "known" part of the model; some are fixed because they pertain to the integration scheme and its constants. A neural "sub"-network is also contained in the scheme, which will upon training approximate the unknown parts of the right-hand-side of the system ODEs. We again use the implicit integrator of Eq. (2) as the basis for training this network, which – due to the implicit nature of the integrator – has recurrent connections and therefore requires multiple evaluations.

4 An illustrative example

In order to illustrate the capabilities of the gray-box approach we will make use of simulated data from a model reacting system [9]. It consists of a well-stirred reactor in which a single irreversible reaction $A \rightarrow B$ occurs on a catalytic surface. The mass balances for species A on the catalytic surface and the gas phase take the general (dimensionless) form:

$$\begin{aligned} \frac{d\theta}{d\tau} &= K_a\Pi(1-\theta) - K_d\theta e^{-\frac{(\alpha^*\theta+\beta)}{\gamma}} - K_R\theta e^{-\frac{1}{\gamma}} \\ \frac{d\Pi}{d\tau} &= 1 - \Pi + \Pi^*[K_d\theta e^{-\frac{(\alpha^*\theta+\beta)}{\gamma}} - K_a\Pi(1-\theta)] \end{aligned} \tag{4}$$

where θ is the fractional coverage of the catalytic surface, Π is the partial pressure of the reactant in the gas phase, γ is the dimensionless temperature, τ is the dimensionless time and K_a, K_R, K_d, α^*, β and Π^* are constants. This has been suggested as one of the simplest models that can give rise to oscillations in isothermal catalytic reactions; its main characteristic is the coverage-dependent desorption activation energy (the $e^{-\frac{\alpha^*\theta}{\gamma}}$ term in Eq. (4)) caused by adsorbate-adsorbate interactions.

To illustrate the dependence of the dynamics on an operating parameter, we obtained time series from this system for several values of the dimensionless temperature γ (keeping the remaining parameters $K_a = 35$, $\alpha^* = 30$, $K_d = 350$, $\Pi^* = 0.36$, $K_R = 8.5$ and $\beta = 0.2$ constant). Depending on the value of γ, the system may evolve towards a steady state, towards oscillatory behavior, or to either of the two depending on the initial conditions. The variegation in long-term dynamics makes this example a good test of the approximating capabilities of the neural network.

Figure 2 shows the bifurcation diagram for this system: a branch of steady states undergoes a *subcritical* Hopf bifurcation to oscillatory behavior for $\gamma \approx$

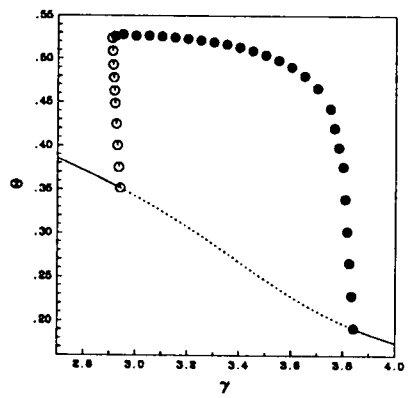


Figure 2: Bifurcation diagram for the single species surface reaction system with respect to the dimensionless temperature γ. Solid lines denote stable steady states, dashed lines unstable steady states, open circles unstable limit cycles and filled circles stable limit cycles. The maximum θ of the periodic trajectory at each value of γ is marked.

2.941. There is a small range of values of γ where a stable large amplitude oscillation coexists with a stable steady state (starting at about $\gamma \approx 2.9076$). As γ is increased the system exhibits, as its sole long-term attractor, a large amplitude limit cycle that disappears at $\gamma \approx 3.841$ via another (now *supercritical*) Hopf bifurcation.

Figure 3 shows phase portraits of the system for several values of γ in the range of the bifurcation diagram of Fig. 2.

5 Network construction and results

Using data representative of the periodic phenomena described above, we tested the neural network ODE-gray-box algorithm for identification. The training set included several time series (θ and Π vs τ) for values of γ before the subcritical Hopf (including the region of bistability), after the subcritical Hopf (limit cycle behavior), as well as after the supercritical Hopf at high values of γ.

For our illustration we assume that all terms in Eq. 4 are known except for the term representing the rate of desorption of the reactant from the catalytic surface. That is, we replace the term $K_d\theta e^{-\frac{(\alpha^*\theta+\beta)}{\gamma}}$, with an *unknown* function $f(\theta,\gamma)$ to be approximated through a neural network. The gray model we seek to construct is of the form:

$$\frac{d\theta}{d\tau} = K_a\Pi(1-\theta) - f(\theta,\gamma) - K_R\theta e^{-\frac{1}{\gamma}}$$

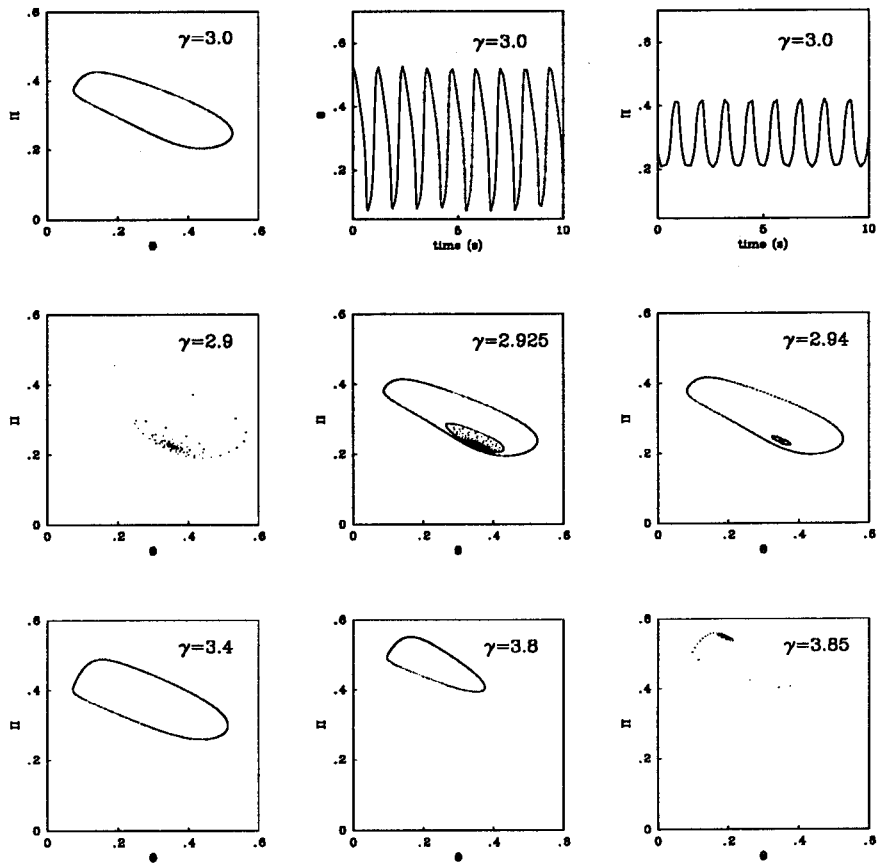


Figure 3: Long term attractors for γ values in the range of the bifurcation diagram of Fig. 2. Top row: phase portrait of the stable limit cycle at $\gamma = 3.0$ along with segments of the two corresponding time series. For the stable steady states ($\gamma = 2.9$ and $\gamma = 3.85$) phase portraits of transients approaching the steady state are shown. In the regions of bistability (γ in the range (2.925,2.94)) the unstable (and thus experimentally unobservable) limit cycle in the interior of the large amplitude stable limit cycle is also drawn.

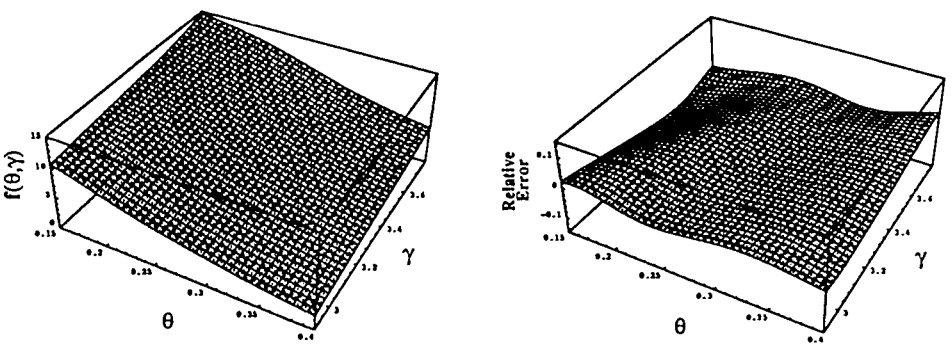


Figure 4: Predicted desorption rate as a function of surface coverage (θ) and dimensionless temperature (left) and relative prediction error (right). The plot on the right shows the difference of the predicted minus the actual desorption rate normalized by the actual rate.

$$\frac{d\Pi}{d\tau} = 1 - \Pi + \Pi^{*}[f(\theta,\gamma) - K_a \Pi(1-\theta)] \tag{5}$$

The (feedforward) neural sub-network, embedded in the numerical integrator of Fig. 1(b), involves two inputs (θ and γ), one output ($f(\theta,\gamma)$) and six neurons with sigmoidal (tanh-type) activation function in each of the two hidden layers. The derivatives of the error measure (energy function) with respect to network parameters needed for the training algorithm are obtained using the chain rule and (due to the recurrence) the implicit function theorem.

The training set consisted of a total of 2950 points allocated in the following manner: 250 points for $\gamma = 2.9$, 450 for $\gamma = 2.91$, 500 for $\gamma = 2.925$, 500 for $\gamma = 2.94$, 250 for $\gamma = 3.0$, 250 for $\gamma = 3.0$, 250 for $\gamma = 3.2$, 250 for $\gamma = 3.4$, 250 for $\gamma = 3.8$ and 250 for $\gamma = 3.85$. The time step of the integrator was 0.06 dimensionless units for all the time series used (roughly one twentieth of the period of the oscillation observed at $\gamma = 3.0$). More points are included in the region of multistability in an effort to capture accurately the hysteresis phenomena. Training was performed using a conjugate gradient algorithm with frequent restarts (see [4, 5] for a discussion). Convergence was achieved after approximately 300 network parameter updates.

The sub-network succeeds in capturing the basic form of the behavior of the

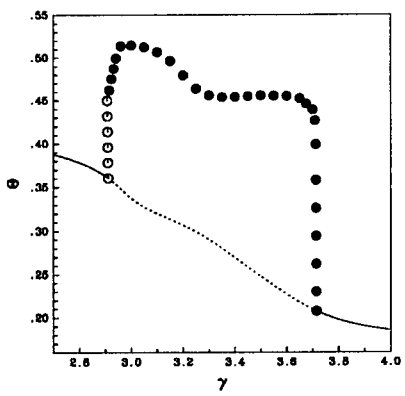


Figure 5: Predicted bifurcation diagram for the single species surface reaction system with the gray-box neural network approximation of the desorption rate.

rate of desorption with respect to θ and γ (surface coverage and temperature). Figure 4 compares the actual desorption rate (as a function of (θ, γ)) with the network predictions. More importantly, the dynamic behavior (including the *infinite-time* attractors) of the system (Eq. 5) also compares favorably with the original system (Eq. 4). Figure 5 shows the predicted bifurcation diagram using the form of the desorption rate given by the network. The network correctly predicts a subcritical Hopf bifurcation at low γ, as well as a supercritical Hopf bifurcation at higher values of γ (at a slightly lower value of γ than for the original system, Fig. 2).

The neural network gray-box approximation can be used to extract important mechanistic information pertaining to the fitted step – and thus possibly discriminate among rival candidate first principles models. For example, Fig. 6 shows that the network predicts a linear dependence of the logarithm of the desorption rate versus $\frac{1}{\gamma}$ at constant θ, in agreement with desorption being an activated process. Fig. 6 shows also that the predicted *slopes* of these plots (and thus, the activation energies) vary linearly with θ, consistent with an assumption of attractive adsorbate-adsorbate interactions (as was indeed the case).

6 Summary

We have extended a previously developed black-box neural network methodology for the characterization of experimental systems as continuous-time models, so as to allow the identification of unknown parts of first principles models. Such modeling efforts incorporate the insight obtained from the first principles modeling (algebraic forms of the ODEs describing the dynamical evolution of the system) in a neural network framework capable of approximating (after training)

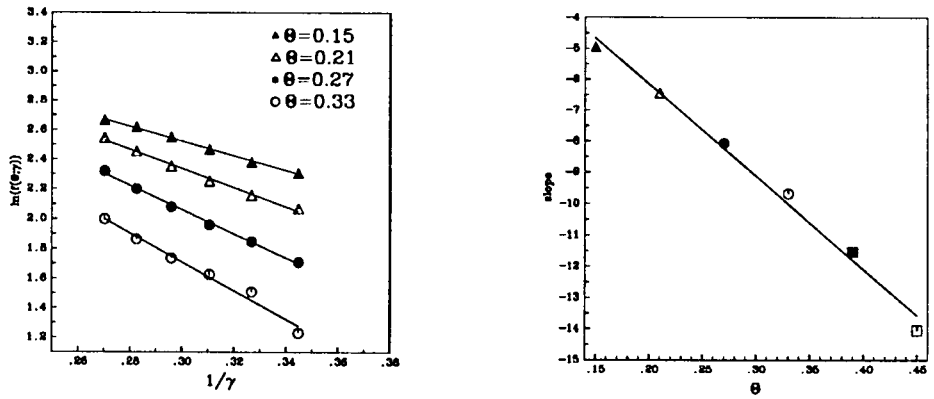


Figure 6: The linear dependence of the natural logarithm of the desorption rate with respect to $\frac{1}{\gamma}$ at constant θ is correctly captured by the neural network gray-box approximation (left); furthermore the predicted slope of the lines varies linearly with θ (right), in agreement with the assumption of adsorbate interactions used to generate the training data.

unknown parts of the model.

The capabilities of this gray-box approach were illustrated using a single species surface reaction system. In this illustration we assumed that the expression for the rate of desorption of the reactant is not known and approximated it through a neural network. Both the short- and long-term dynamic behavior of the system is well approximated by the hybrid model resulting from training. Furthermore, a study of the properties of the fitted desorption rate may yield insight in the physical mechanisms underlying it, and thus possibly assist in discriminating among rival first principles models.

Discrete-time models (based on neural networks) are trained to predict the *result* of integrating the model equations over some time period. It is difficult to "unravel" the contribution of known parts of the model to this result from the contribution of the unknown terms. When, on the other hand, the equations themselves are approximated (as opposed to the result of integrating them), the procedure naturally lends itself to incorporating processes whose modeling is established to the gray-box model.

The type of overall network presented here (with some parts of its architecture available for training, and some other parts fixed by either the known parts of the model or the integrator scheme) may prove to be a valuable tool towards understanding the dynamics of experimental systems. The particular choice of recurrent nets templated on implicit integrators presented here is motivated by the anticipated stiffness of chemical kinetic equations. Feedforward implementations based on explicit integrators are also possible. We are currently working on variants of training algorithms for recurrent nets and their implementation

on parallel computers.

References

[1] A. S. Lapedes and R. M. Farber. Nonlinear signal processing using neural networks: Prediction and system modeling. *Los Alamos Report LA-UR 87-2662* (1987).

[2] A. S. Weigend and N. A. Gershenfeld. Time series prediction: Forecasting the future and understanding the past. *Addison-Wesley* (1993).

[3] R. Rico-Martínez, K. Krischer, I. G. Kevrekidis, M. C. Kube and J. L. Hudson. Discrete- vs. continuous-time nonlinear signal processing of Cu electrodissolution data. *Chem. Eng. Comm.*, vol. 118, pp. 25–48 (1992).

[4] R. Rico-Martínez. Neural networks for the characterization of nonlinear deterministic systems. *Ph. D. Thesis*, Department of Chemical Engineering, Princeton University (1994).

[5] R. Rico-Martínez and I. G. Kevrekidis. Continuous-time modeling of nonlinear systems: A neural network approach. *Proc. 1993 IEEE Int. Conf. Neural Networks*, IEEE Publications, vol. III, pp. 1522–1525 (1993).

[6] S. R. Chu and R. Shoureshi. A neural network approach for identification of continuous-time nonlinear dynamic systems. *Proc. of the 1991 ACC*, vol. 1, pp. 1–5 (1991).

[7] F. J. Pineda. Generalization of back-propagation to recurrent neural networks. *Phys. Rev. Letters*, vol. 59, pp. 2229–2232 (1987).

[8] L. B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. *Proc. IEEE 1st Ann. Int. Conf. Neural Networks*, San Diego, CA., pp. 609–618 (1987).

[9] I. Kevrekidis, L. D. Schmidt and R. Aris. Rate multiplicity and oscillations in single species surface reactions. *Surf. Sci.*, vol. 137, pp. 151-166 (1984).

A QUANTITATIVE STUDY OF EVOKED POTENTIAL ESTIMATION USING A FEEDFORWARD NEURAL NETWORK

Adriana Dumitras* , Adrian T. Murgan*, Vasile Lazarescu*
Member, IEEE *Member, IEEE*
*** Electronics and Telecommunications Dept.**
Technical University of Bucharest, Romania
1-3 Armata Poporului Blv., Bucharest, Romania
phone: (401) 6 31 78 00 ext. 420, 322
email: adadum@vala.elia.pub.ro
atmurgan@vala.elia.pub.ro, vlazarescu@vala.elia.pub.ro

Abstract —We have used a multilayer perceptron to estimate the evoked potentials, masked by the EEG signal. The problem was studied on synthetic signals, generated as given in ([10]) and error criteria other than standard L2-norm were taken into account. We showed experimentally that, as suggested in ([2]), better results could be obtained this way, if the parameters were properly adjusted. An average performed on a few ensembles strongly improves the result and the number of ensembles is lower than quoted in other approaches. We have also studied the influence of the window length and of a different number of hidden units upon the convergence speed and test error. Though good results were obtained in this quantitative study, the trained network which resulted should be tested on real data, in order to get a complete outlook upon this problem.

INTRODUCTION

In the research of brain's electrical activity, the spontaneous and evoked activities were intensely studied, from both clinical and experimental points of view,. An important tool is the electroencephalogram (EEG), a recording of the brain's electrical activity. If there is no observable intervention, the EEG activity is called spontaneous ([11]). This background activity can be changed by central or peripheral stimuli, allowing the study of the evoked activity.

The evoked potential (EP) is an anwer to a stimulus and it can be noticed using microelectrodes, macroelectrods plased on the cortex or using electrodes placed on the skul. When using the third method, an extraction of the EP from the EEG signal, which masks it, it's necessary and one encounters a well-known case in biomedical signal processing, when a physiological signal has its components of interest obscured by much larger signals due to different processes ([11]).

A classical approach to enhance the EP's signal components, is based on filtering the data ([11], [1]), while another one takes into account artificial intelligence (AI) methods, including expert, neural and fuzzy systems ([1]). One should remark that: a) the EP is a very low amplitude signal, compared with the

0-7803-2026-3/94 $4.00 © 1994 IEEE

EEG; and b) the EEG signal is nonlinear. Neural network capabilities, as nonlinear models, ([7], [4], [10]), drove to good performances, when the signal characteristics were not known, though the results strongly depended on the neural architecture and the learning strategy ([5], [13]).

In this paper, we have considered the problem of EP estimation, using a feedforward network. A multilayer perceptron was chosen, trained with the standard backpropagation algorithm and some some of its extensions, concerning different error criteria. Changes in the structure of the network were also experimented , in order to improve the results and the convergence speed. The purpose was analysing from a quantitative point of view the performance of this structure in solving the EP problem. In the following, by network we are going to refer to a multilayer perceptron.

THEORETICAL APPROACH

The EP problem

The problem we approach is that of estimating the values of the evoked potential signal from the given data samples. The signal is present, but it is buried in the background EEG, which is obscuring it. This activity may be considered as noise, additive to the clean signal:

$$x_{noisy}(t) = x_{clean}(t) + n(t), \qquad (1)$$

where $x_{clean}(t)$ is the correct evoked potential, n(t) is the EEG activity and $x_{noisy}(t)$ is the measured signal. We assume that the EEG and EP signals are not correlated. The problem is finding an estimation $x^{\sim}(t)$ so that $\aleph(x)$ is minimized, where $\aleph$ is a general norm: $\|x_{clean}(t) - x^{\sim}(t)\|$ ([8], [12]). Many times, this is the L2-norm.

The Multilayer Perceptron

Among the neural networks, the multilayer perceptron has the most frequently encountered applications. It is widely used in biosignal processing, too. The results are promising, as long as the data were adequately preprocessed ([11]). In solving nonlinearly separable problems, at least two active layers are needed ([7], [4], [9]).

The architecture shown in Figure 1a consists of an input layer, which passes forward the input values, the hidden layer, which is responsible of an internal encoding during the learning process and the output layer, also active, like the hidden one. The layers are made up of nInput, nHidden and nOutput nodes, respectively.

The Learning Rule in its standard form uses a gradient search method, which updates the weights in the network for each input vector ([13]). The backpropagation algorithm (BKP) was built on the basis of the least-squares error (Euclidian or L2-norm) criterion ([2]), which is a first-order descent method.

Let the network be as depicted in Figure 1a, with $\underline{x}(t)$ the input vector, where $x_i = x(k-r)$, $r = \overline{0, nInput-1}$, are its components, given by a tapped delay line,

$\underline{y}^{hid}(t)$ and $\underline{y}^{out}(t)$ are output vectors from the hidden and the out layer, respectively. One should make an assumption regarding the relationship between the index "k" and time "t". The usual technique is k=t ([5]) and it was adopted here, too. We shall follow the notation: $\underline{w}^{in\text{-}hid}=[w_{ih}]$ and $\underline{w}^{hid\text{-}out}=[w_{hj}]$ for the weight vectors, where the neurons in the layers are:

$i = \overline{1,nInput}$, $h = \overline{1,nHidden}$, $j = \overline{1,nOutput}$.

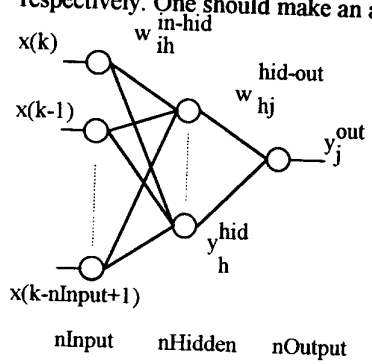


Figure1a:The network architecture

Each neuron in the active layers yields its output by forming the weighted sum of the outputs of the neurons in the immediately preceding layer, adding the bias and passing the result through its characteristic nonlinear activation function. For example, in the hidden layer, for the activation function chosen to be the sigmoid, one has:

$$net_h = \sum_{i=1}^{nInput} w_{ih} * x_i + \theta_i \quad ; \quad f(net) = a * \frac{1-\exp(-s*net)}{1+\exp(-s*net)} \tag{2}$$

where the parameter "s" specifies the steepness of the curve, the "a" value is a constant, x_i is the i-th component of the input vector and "θ" is the bias value. When passing from hidden to the output value, the values are calculated in the same way. Once the output values y_j^{out} are available, for each node j, the error can be evaluated, comparing the actual value with the desired value d_j. The error function ([2]) may be selected from Table 1, where $e_{jp} = d_{jp} - y_{jp}$. Usually, the L2-norm criterion is used, but as suggested in ([2]), when the signals are contaminated by non-Gaussian noise, more sophisticated error measures should be used, i.e. non-euclidian error signals, in order to achieve a better performance of the learning neural network.

Given the e_{jP}, the total output error for a pattern p, $p = \overline{1, nPat}$ can be written as (3) and for all the patterns, as (4) or (5).

$$E_p = \sum_{j=1}^{nOutput} \Im(e_{jp}) \tag{3}$$

$$E = \sum_{p=1}^{nPat} \sum_{j=1}^{nOutput} \Im(e_{jp}) \tag{4}$$

$$E = \frac{1}{nOutput * nPat} \sum_{p=1}^{nPat} \sum_{j=1}^{nOutput} \Im(e_{jp}) \tag{5}$$

The weights are updated backpropagating the error, in order to find the optimum weight vector that minimizes it. The gradient of E is calculated and the weight vector at iteration "t" is adjusted in the direction of the steepest descent ([7], [2], [9] et.al), with E_p given by (3) and η a learning constant:

$$w(t+1) = w(t) + \eta\,(-\nabla E_p) \tag{6}$$

The calculus of the gradient yields ([7], [2], [9]) eq. (7) for any two neurons "q" and "r", for a pattern "p", δ_r is node's "r" error and y_q is node's "q" output value, given by eq. (8) for the output nodes $j = \overline{1,nOutput}$, and by eq. (9) for the hidden nodes, with index "k" over all the nodes in the previous layer, the error is propagated from.

Table 1: ERROR FUNCTIONS

Error function	$\lvert e_{jp}\rvert \le \beta$	$\lvert e_{jp}\rvert > \beta$
L1 - norm	$\Im(e_{jp}) = \lvert e_{jp}\rvert$	$\Im(e_{jp}) = \lvert e_{jp}\rvert$
L2 - norm	$\Im(e_{jp}) = 0.5 * e_{jp}^2$	$\Im(e_{jp}) = 0.5 * e_{jp}^2$
Huber's error function	$\Im(e_{jp}) = 0.5 * e_{jp}^2$	$\Im(e_{jp}) = \beta \lvert e_{jp}\rvert - 0.5 * \beta^2$
Hampel's error function	$\Im(e_{jp}) = \frac{\beta^2}{\pi}(1 - \cos\frac{\pi e_{jp}}{\beta})$	$\Im(e_{jP}) = 2\frac{\beta^2}{\pi}$
Logistic function	$\Im(e_{jp}) = \beta^2 \ln[\cosh(\frac{e_{jp}}{\beta})]$	$\Im(e_{jp}) = \beta^2 \ln[\cosh(\frac{e_{jp}}{\beta})$

$$w_{qr}(t+1) - w_{qr}(t) = -\eta\frac{\partial E_p}{\partial w_{qr}} = -\eta\frac{\partial E_p}{\partial y_{pj}^{out}}\frac{\partial y_{pj}^{out}}{\partial net_{pj}}\frac{\partial net_{pj}}{\partial w_{qr}} = \eta\delta_r y_q \tag{7}$$

$$\delta_{pj}^{out} = -\frac{\partial E_p}{\partial y_{pj}^{out}}\frac{\partial y_{pj}^{out}}{\partial net_{pj}} = -\frac{\partial E_p}{\partial y_{pj}^{out}} f_j^{!}(net_{pj}) = f_j^{!}(net_{pj})\frac{\partial \Im(e_{jp})}{\partial y_{pj}^{out}} \tag{8}$$

$$\delta_{ph}^{hid} = f'_h(net_{ph}) \sum_k \delta_{pk} w_{kh} \tag{9}$$

There are several approaches to increase the convergence speed and to avoid the algorithm getting stuck in local minima. One could mention adaptive parameters (bias, momentum α, slope of the activation function) and other error criteria than L2-norm ([2], [9]).With a momentum term, $0 < \alpha < 1$, (7) becomes:

$$w_{qr}(t+1) - w_{qr}(t) = \eta \delta_r y_q + \alpha (w_{qr}(t) - w_{qr}(t-1)) \tag{10}$$

and α could be adaptively adjusted, too. This holds also for the bias, a relation similar to (10) can be written if one considers the bias as a weight value for a suplimentary node in the structure, having always as an input the "1.0" value.
Remark: in the equations (7), (8), (9), (10), which give the adaptation for both hidden - output and input - hidden weights, with $y_q = y_h^{hid}$, $h = \overline{1,nHidden}$ and $y_q = = x_i$, $i = \overline{1,nInput}$, when taking into account the error criteria in Table 1, only the expression for δ^{out} changes ([2]).

EXPERIMENTAL RESULTS

We used synthetic evoked potentials, generated as ([10]) quoted from literature: the "clean" signal consisted of one full sine wave, followed by a half attenuated sine wave. Noise resulted after filtering random numbers with uniform distribution, with the 11-point smoothing filter ([10]):

$$n_i = (-36 z_{i-5} + 9 z_{i-4} + 44 z_{i-3} + 69 z_{i-2} + 84 z_{i-1} + 89 z_i + 84 z_{i+1} + 69 z_{i+2} + 44 z_{i+3} + 9 z_{i+4} - 36 z_{i+5})/429 \tag{11}$$

There are plotted in Figures: 1b.) the unfiltered noise, 2.) the filtered noise (200 samples), 3) the clean and noisy EPs, 200 samples and 4) the clean and noisy EPs, 576 samples. We have considered N number of samples, and a window of length W (W odd). Each pattern consisted of W samples, read as the window was sliding one step (sample) to the right and a set of N input patterns resulted. The desired output value was always considered to be the correct value placed in the middle of the window. Experiments shown in Table 2 were carried out:

1. An 11-7-1 network was trained (see Table 2). The influence of N is given in Table 3, for N = 200 and N = 576. For a number of patterns N = 576, the training error decreases more rapidly to a lower value than for N = 200 (when error= 0.011 was the minimum we could achieve with this structure and N = 200).

2. The learning curves for the experiment 2 (Table 2) are shown in Figure 5. It was clear that adaptive biases and slopes speeded up the convergence.

3. We trained the network (see Table 2), requiring a maximum training error = 0.00066 for different nInput (odd) values. As nInput is the window length W, Table 4 shows its influence on the convergence speed and test error. If W is too small, the nonlinear relations between the current data sample and samples outside the window cannot be learned by the network ([13]). For nInput = 5 and 7, the test error is lower because the network is not making any encoding, as nInput $\leq$ nHidden, but still, the network performs a poor noise removal. If W is large, too

many correlations can be unexpectedly captured by the system ([13]). For nInput=W=13, the test error is higher as compared to other experim. cases.

Table 2: EXPERIMENTS

Constant parameters	Exp nr.	nInp. (=W)	nHid	nPat (N)	nEpochs	Bias	Err. crit.	Train EP
• $\alpha = 0.1$ • s = 0.5 • nOutp= 1 • a = 0.5 • f(net) as (1) • desired sign=clean EP • test sign =noisy EP	1	11	7	200	variable	ct.	L2	clean
		11	7	576	variable	ct.	L2	clean
	2	11	7	576	variable	ct.	L2	clean
		11	7	576	variable	adapt	L2	clean
	3	5,7,9, 11,13	7	576	variable	adapt	L2	clean
	4	11	6,7,8	576	variable	adapt	L2	clean
	5	11	7	576	1050	adapt	L2 Hub. Hamp Log.	noisy other than train.

Table 3: THE INFLUENCE OF N (NUMBER OF PATTERNS) UPON THE TRAINING ERROR

Nr. of patterns	Epochs	Total number of samples	Training error
200	512	102,400	0.0110
576	153	88,128	0.0011

4. The influence of the number of hidden units on the training error (experiment 4, Table 2) is shown in Figure 6. Adding a hidden unit did not improve the test error,

Table 4: THE INFLUENCE OF THE WINDOW LENGTH

nInput	nHidden	nOutput	Epochs	Total nr. of patt.	Test error
13	7	1	44	25,344	0.1199
11	7	1	69	39,744	0.1118
9	7	1	87	50,112	0.0802
7	7	1	10	5760	0.0597
5	7	1	10	5760	0.0378

on the contrary, though the convergence speed increased. There are methods given in literature, to determine, the number of hidden units, but our concern here was not finding an optimal structure. One could find out, though, following the suggestion of ([13]), whether the number of neurons in the hidden layer was too high, by checking the rank of the autocorrelation matrix of the output hidden units. This matrix would have had rank deficiency if at least one hidden unit's output would have been linearly dependent on the rest of the hidden units.

After these tests, we decided to use an 11-7-1 network, which proved to be a tradeoff between the convergence speed and performance in the test phase, in the

cases studied above. Training the network on a clean EP (input signal and desired output signal) and testing it on a noisy EP, as we did, showed what one could have told in advance: the network wasn't able to cope with the noisy input signal satisfactory. The output signal (test phase) vs. the desired one, is shown in Fig. 7.

5. This experiment (Table 2) took into account various error criteria. Training was performed for 1050 epochs (i.e. 604,800 input patterns), with input weights randomly chosen, but the same for all phases of the experiment. The output signals in the test phase, for the mentioned error functions are plotted in Figure 9,10,11,12. Better performance was obtained, for the same training conditions, if we considered other error criteria than the L2-norm. For the resulting networks, the mean square error when testing with noisy EP is given in Table 5.

Also, this experiment checked the assertion made in ([2]) regarding the strong dependance of the new error criteria on the value of the controlling parameter β. For a large value of β, the learning algorithm is practically equivalent to the standard BKP algorithm ([2]) and it may show a learning curve as given in Figure 8. On the other hand, if β is very small, then the first derivative of the error function closely approximates the signum (hard limiter) function ([2]). If β is too low, the network may not learn (for example, if $\beta \leq 0.1$, the network is not able to pass the first step, the output signal may be as given in Figure 12).

Table 5: THE MSE ERROR FOR THE RESULTING NETS

Training error function	MSE test error
L2-norm	0.09040
Huber, $\beta = 0.01$	0.08912
Logistic, $\beta = 0.001$	0.08705

It is also well-known, that average evoked potentials have a reduced variation in the framework of a specific stimulus ([11]). In ([10]), a strong improvement of the performance was shown, if average on a small set of ensembles was considered. We argue that, due to better results obtained when using other error criteria than standard L2-norm, performance is improved when averaging on a few ensembles (Figure 11, for the logistic function, 5 ensembles).

Experiment 5 (Table 2) was repeated, chosing the desired value at a specific moment of time, as the next value following the window, i.e. if the input values were noisy x_1,.....x_{11}, the desired value was the correct x_{12}, etc. There was no evident improvement due to this new context learning. The only apparent advantages were, the possibility of selecting the nInput value either odd or even, and the intuitive predictive quality of the network, as being able to give the following value, based on the past 11 input values.

CONCLUSION

Experimentally, we have chosen to use a multilayer perceptron with the structure 11-7-1. When trained on a clean EP signal and tested on noisy EP, it showed out poor performance, as the network did have in the training phase no information about the noise from the real EP so, we further trained it on nosiy EP and tested on other noisy EPs. Five experiments were carried out, which led us to the conclusion that for the same training parameters and number of patterns, better performance could be achieved when using error criteria other than the standard

L2-norm. We argue that, selecting such a new criteria and adjusting the parameters, an average of the results obtained in a few tests could be made, as usual in this particular problem, but the number of ensembles is lower than in other cases. Our results can be improved if further carefully chosing the β parameter. The influence of the window length and of a different number of hidden units upon the convergence speed and test error were also studied.

Our future work would have to take into account the idea of finding a minimum structure by one of the available methods. Also, we are aware of the fact that studying the performances on synthetic data, gives only a part of the overlook upon this matter, so it is necessary to test the obtained network on real data.

REFERENCES

[1] J Robert Boston, Rule Based System for Interpretation of Evoked Potential Waveforms; IEEE Engineering in Medicine & Biology Society 11th Annual Intl. Conference, IEEE, 1989;

[2] A. Cichoki, R. Unbehauen, Neural Networks for Optimization and Signal Processing; J.Wiley & Sons, UK, 1993;

[3] Hans Gaunholt, Adriana Dumitras, Digital Filtering with Neural Networks; Report ISSN 0105 - 8541, IT - 93 - 141; Institute of Circuit Theory and Telecom., Technical University of Denmark, Lyngby, Denmark, May, 1993;

[4] R. P. Lippmann, An Introduction to Computing with Neural Nets; IEEE ASSP Magazine, USA, 1987;

[5] O. Nerrand, P. RousseL, Ragot, L. Personnaz, G. Dreyfus et al., Neural Network Training Schemes for Non-Linear Adaptive Filtering and Modelling; IJCNN-91: Proceedings of the Intl. Joint Conference on Neural Networks, July 8-12, 1991, Seattle, WA, USA, IEEE Inc., Vol. 1;

[6] S.Roberts, L.Tarassenko, Analysis of the sleep EEG using a multilayer network with spatial organization; IEE Proceedings-*F*, Vol. 139, No. 6, December 1992;

[7] D.E.Rumelhart, J.L. McClelland and the PDP Research Group, Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol.1: Foundations; Cambridge, MA, MIT Press, USA, 1986;

[8] Mischa Schwartz, Leonard Shaw, Signal Processing: Discrete Spectral Analysis, Detection and Estimation; McGraw Hill Inc., USA, 1975;

[9] Nazif Tepedelenlioglu, Ali Rezgui, Robert Scalero, Ramona Rosario, Fast Algorithms for Training Multilayer Perceptrons; in Branko Soucek (ed.) and The IRIS Group: Neural and Intelligent Systems Integration; John Wiley & Sons, Inc., USA, 1991;

[10] A. Uncini, M. Marchesi, G. Orlandi, F. Piazza, Improved Evoked Potential Estimation Using Neural Network; *IJCNN - 90;* Proceedings of the Intl. Joint Conference on Neural Networks, San Diego, California, 1990; IEEE Inc., 1990, Vol. 2;

[11] R. Weitkunat (ed.), Digital Biosignal Processing; Elsevier Science Publishers, The Netherlands, 1991;

[12] B.Widrow, S.D. Stearns, Adaptive Signal Processing; Prentice-Hall, Engl. Cliffs, NJ, 1985;

[13] Quizhen Xue, Yu Hen Hu, Willis T.Tompkins, Neural-Network-Based Adaptive Matched Filtering for QRS Detection; IEEE Transactions on Biomedical Engineering, Vol. 39, No. 4, April 1992;

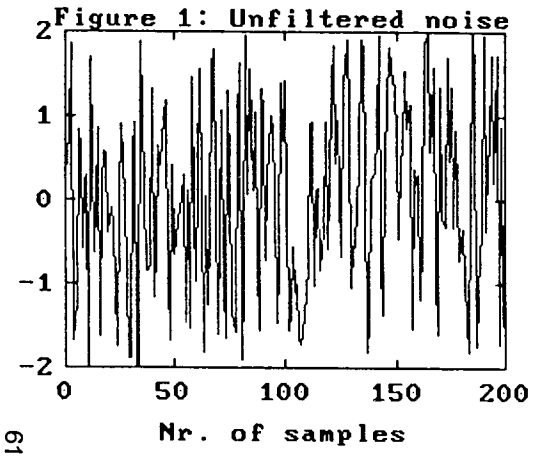


Figure 1: Unfiltered noise
2
1
0
-1
-2
0
50
100
150
200
Nr. of samples

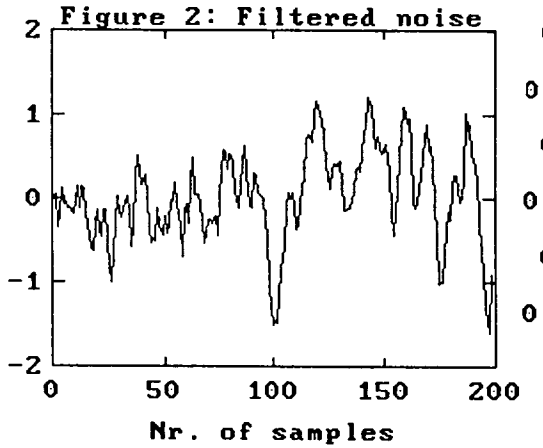


Figure 2: Filtered noise
2
1
0
-1
-2
0
50
100
150
200
Nr. of samples

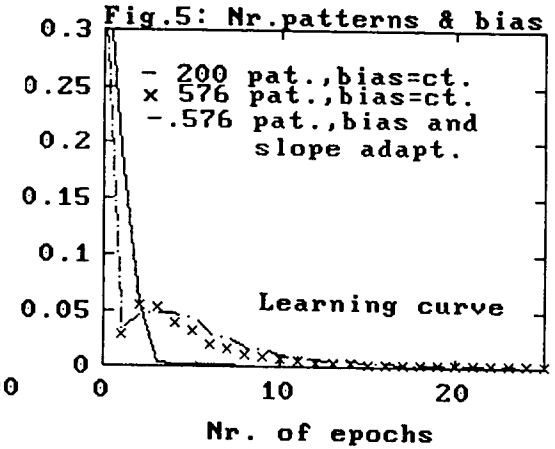


Fig.5: Nr.patterns & bias
- 200 pat.,bias=ct.
x 576 pat.,bias=ct.
-.576 pat.,bias and slope adapt.
Learning curve
0.3
0.25
0.2
0.15
0.1
0.05
0
0
10
20
Nr. of epochs

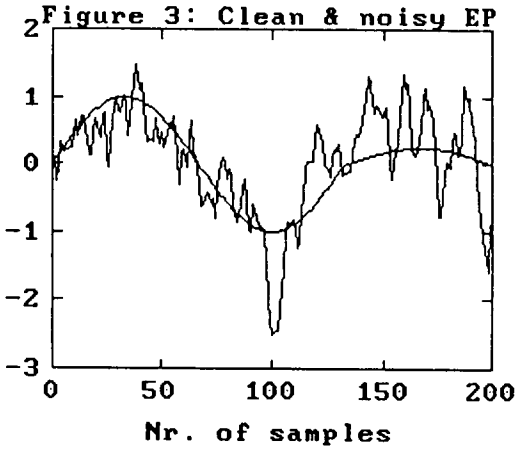


Figure 3: Clean & noisy EP
2
1
0
-1
-2
-3
0
50
100
150
200
Nr. of samples

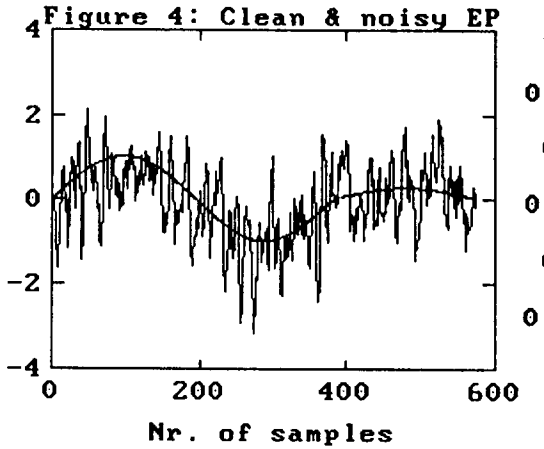


Figure 4: Clean & noisy EP
4
2
0
-2
-4
0
200
400
600
Nr. of samples

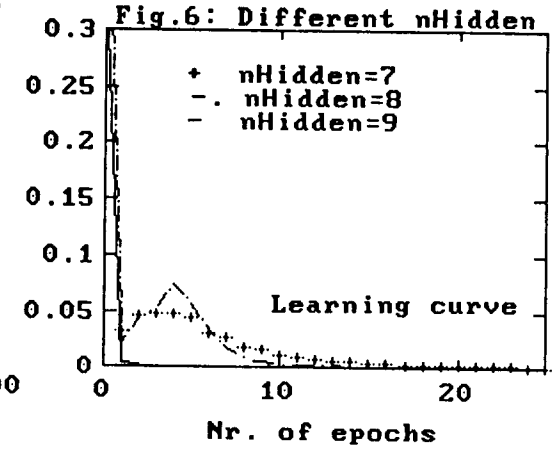


Fig.6: Different nHidden
+ nHidden=7
-. nHidden=8
- nHidden=9
Learning curve
0.3
0.25
0.2
0.15
0.1
0.05
0
0
10
20
Nr. of epochs

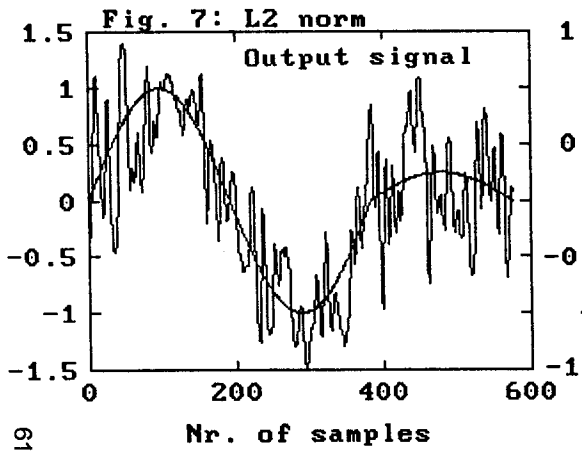


Fig. 7: L2 norm

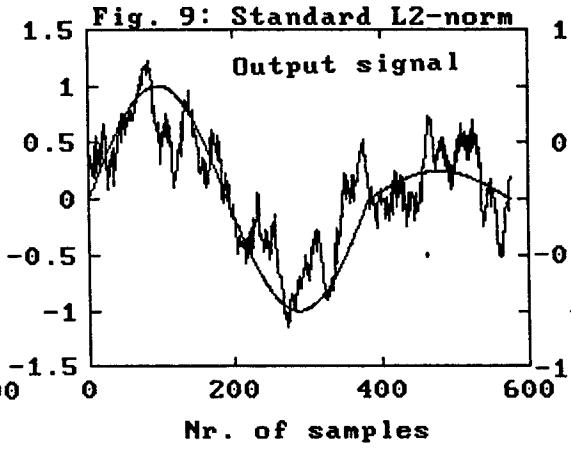


Fig. 9: Standard L2-norm

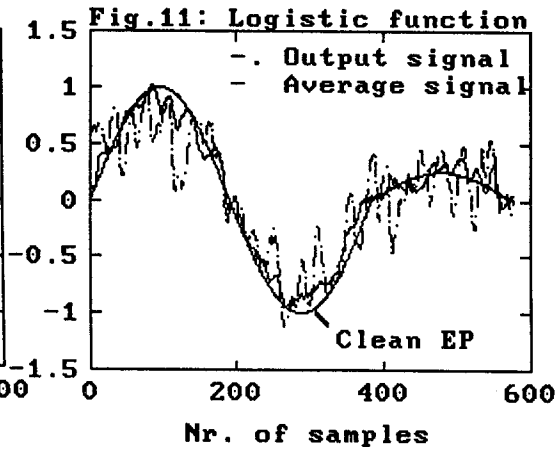


Fig.11: Logistic function

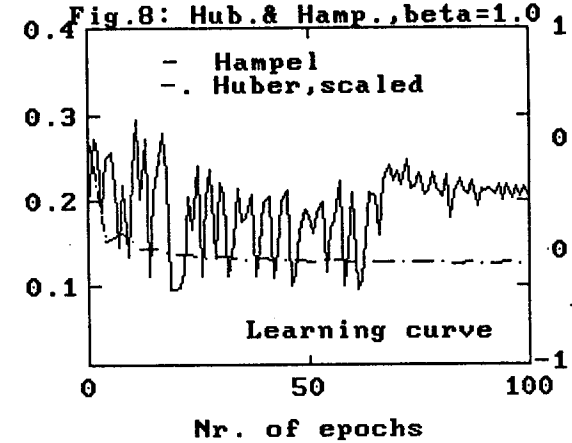


Fig.8: Hub.& Hamp.,beta=1.0

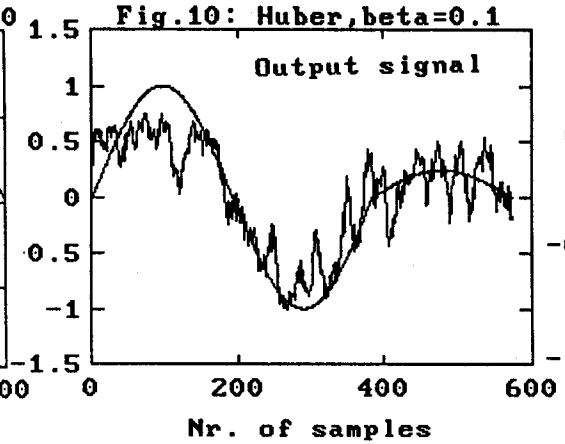


Fig.10: Huber,beta=0.1

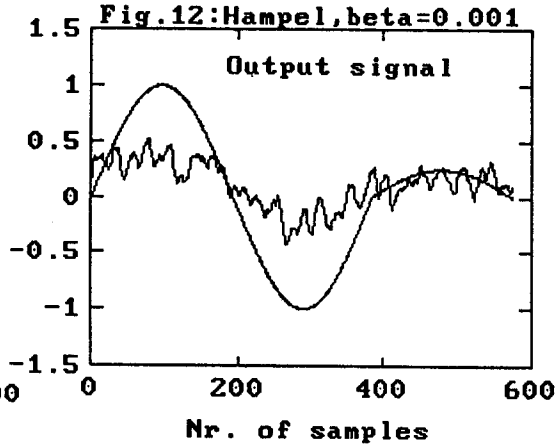


Fig.12:Hampel,beta=0.001

NEURAL ESTIMATION OF KINETIC RATE CONSTANTS FROM DYNAMIC PET-SCANS

Torben Fog, Lars Hupfeldt Nielsen and Lars Kai Hansen
CONNECT, Electronics Institute B349
Technical University of Denmark,
DK-2800 Lyngby, Denmark
email: lkhansen@eileen.ei.dth.dk

Søren Holm, Ian Law, Claus Svarer, and Olaf Paulson
Dept. of Neurology,
The University Hospital of Copenhagen
DK-2100 Copenhagen Ø, Denmark

Abstract. A feed forward neural net is trained to invert a simple three compartment model describing the tracer kinetics involved in the metabolism of [^{18}F]flourodeoxyglucose in the human brain. The network can estimate rate constants from Positron Emission Tomography sequences and is about 50 times faster than direct fitting of rate constants using the parametrized transients of the compartment model.

INTRODUCTION

Positron Emission Tomography (PET) is an important tool for mapping of brain metabolism and functionality [2]. The primary target of PET is reconstruction of concentrations of certain radioactive tracers. The usefull tracers emit positrons that are locally annihilated to produce two 511 keV gamma rays propagating in opposite directions. The 3D distribution of the tracer can be reconstructed from the geometric constraints of coincident counts, using standard techniques (filtered backprojection). An important class of tracers are chemically equivalent to compounds that enter the basic brain metabolism. By reconstructing such tracer distributions important aspects of brain metabolism have been revealed. Furthermore, by investigating the *transient response* to tracer injection, it is possible to identify fundamental kinetic rate constants. In this study we investigate the latter approach. The basic kinetic model was proposed by Sokoloff et al. [1]; in subsequent studies the model was used to estimate rate constant in lumped regions. In the

0-7803-2026-3/94 $4.00 © 1994 IEEE

work of Kanno et al. [3] pixel by pixel estimation of the rate-constants was introduced. This scheme has, however, not found wide spread use due to the complexity of the task of fitting the kinetic model transient to the large number of individual pixel transients. *In this work we show how a neural network may substitute for such tedious parameter fitting procedures. The neural network system is trained to produce a smooth map relating a given transient with its most likely rate constants. This will provide a much faster estimation time for the individual pixel rate constants*

SOKOLOFF's KINETIC MODEL

We consider here the kinetics of the compound [^{18}F]flourodeoxyglucose (FDG). The kinetics of this tracer is similar to glucose in the initial metabolic steps. It passes through the blood-brain barrier (BBB), and is phosphorylized in a process past the BBB analogous to glucose. Then it ceases to react further and is effectively trapped. The kinetics can be modelled by a compartemental model involving one compartment representing the tracer density in the arterial blood outside the BBB, C_P^*; one compartment representing the socalled *precursor pool*, C_E^*; and finally a compartment representing the phosphorylized fraction behind the BBB, C_M^*; see figure 1. In current experiments the arterial concentrations are measured continuously along with the scan, hence the concentration C_P^* can be considered a control parameter for the compartment model.

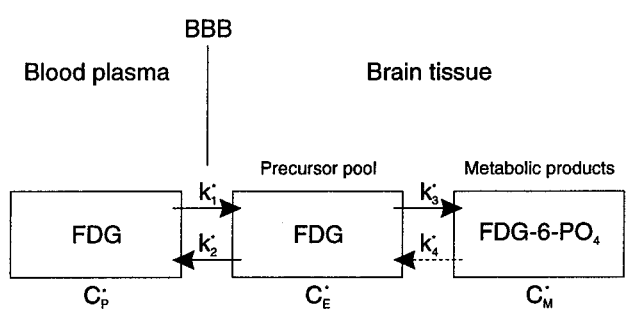


Figure 1: Sokoloff's three compartment model applied to phosphorylization of [^{18}F]flourodeoxyglucose (FDG). The star on the concentrations signifies that we consider tracer amounts and constants

Following the injection of the tracer, hence, the rise of the arterial blood concentration C_P^*, the flow through the BBB starts. The measured PET tracer activity is the sum of the activities of the two compartments to the right of the BBB c.f. figure 1,

$$C_i^* = C_E^* + C_M^* \tag{1}$$

The dynamics of the three compartment model is given by:

$$\frac{dC_i^*}{dt} = \frac{dC_E^*}{dt} + \frac{dC_M^*}{dt} \tag{2}$$

with

$$\frac{dC_M^*}{dt} = k_3^* C_E^*, \tag{3}$$

and

$$\frac{dC_E^*}{dt} = k_1^* C_P^* - k_2^* C_E^* - k_3^* C_E^*. \tag{4}$$

The reverse reaction rate constant (k_4^*) corresponding to k_3^* is neglected. These equations are straightforward to integrate providing the two time dependent concentrations,

$$C_E^*(t) = k_1^* e^{-(k_2^*+k_3^*)t} \int_0^t e^{(k_2^*+k_3^*)t'} C_P^*(t')\ dt' \tag{5}$$

$$C_M^*(t) = k_1^* k_3^* \int_0^t \left[e^{-(k_2^*+k_3^*)t'} \int_0^t e^{(k_2^*+k_3^*)t''} C_P^*(t'')\ dt'' \right] dt' \tag{6}$$

Following injection these solutions describe the transient activity in terms of the measured $C_P^*(t)$ and the three rate constants. Conversely, for a given transient $C_P^*(t)$ and for given measured sum of concentrations $C_i^*(t)$ we may fit the three rate constants. An example is shown in figure 2. We use a simple least squares cost function for the fit, hence implicitly assuming Gaussian residuals. Optimization over the three parameters was carried out using a second order Newton scheme[1]

There are two different approaches from here. Up til now most studies assume that the rate constant are homogeneous in regions, see e.g. [1, 2], and the rate constants are fitted from the regional average activity transient. Alternatively we can fit individual rate constants for each pixel in the reconstructed volume [3], and *analyse* for homogeneity. However, since it is quite tedious to fit the kinetic model, the latter approach has not found widespread use. In the upper panel of figure 3 we show the result of such a pixel by pixel fit. The artifacts outside the elliptic area of the brain are due to the reconstruction scheme used (Filtered backprojection).

The database used for these experiments are PET data collected at the PET center at Rigshospitalet, Copenhagen. The subject described in this paper is a 43 year old woman with multiple sclerosis. Data are aquired on a

[1]Based on the solution to the kinetic model it is straightforward to compute the second derivatives.

GE4096 plus (General Electric Medical Systems), sampling 15 slices simultaneously. The dynamic scans after injection of 200 MBq F-18 labelled FDG are performed over 60 minutes, yelding 34 contiguous time frames of increasing duration in order to provide a reasonable sampling of the C_i^* curve: (10@6 sec; 3@20 sec; 8@60 sec; 5@120 sec; 8@300 sec). A single such curve is shown in figure 2. Images were reconstructed in $128x128$ matrices ($2mm^2$ pixels) by standard Filtered backprojection (Ramp filter with Hann window). Correction for attenuation is based on a separate transmission scan with a rotating Germanium pin source. For further introduction to PET scan techniques see e.g., [2]

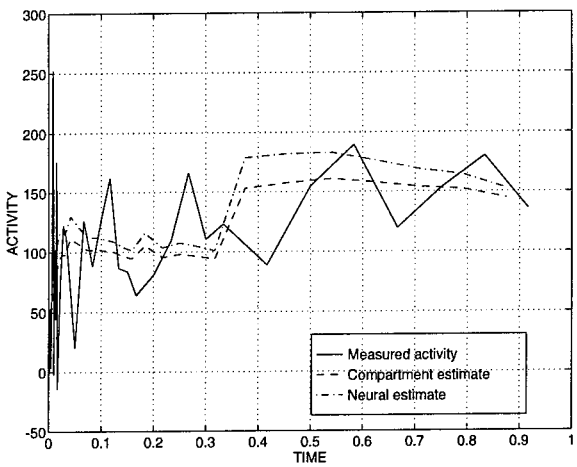


Figure 2: C_i^*(t) as measured by PET for a single pixel, as produced by the kinetic compartment model with the fitted rate constants, and as estimated by the neural network.

To avoid the tedious fitting procedure we here investigate the possibility of identifying the *inverse model* of the kinetics: we search for a *map* that provides an estimate of the three rate constants for a given observed transient. Our basic vehicle will be a simple feed-forward network.

What should be used for inputs?. The PET transients depend, c.f. equations (5-6), on the rate constants and on the time dependent arterial concentration (C_P^*). If we want to generalize from one set of pictures to another set (possibly another subject) we would need to provide both the observed PET transient and $C_P^*(t)$. This will be pursued in future studies. In this work we tentatively train the network to predict the rate constants for pictures of a *single sequence* of PET images, hence, C_P^* is the same for all pixels and we need not provide it as input. In particular we train the network on transients from a small subsample of one slice of the PET volume scan Subsequently we apply the trained network to get the rate constants for the (large) remaining set of pixels.

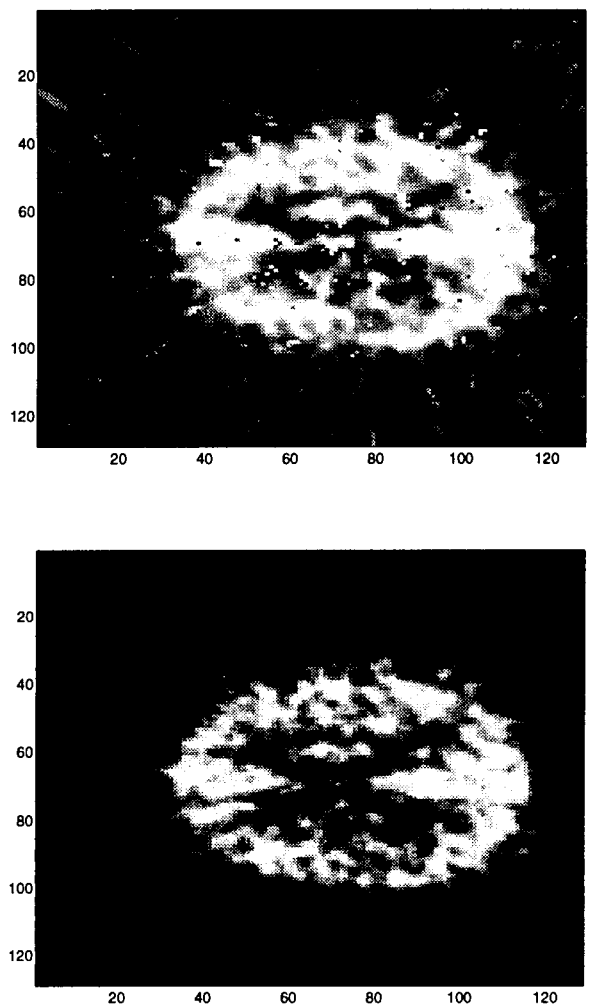


Figure 3: Image (slices) showing the k_1^* rate constant as determined by fitting the kinetic model (pixel by pixel) using a second Newton scheme (upper panel) and as determined by the neural net operating as inverse model for the kinetics (lower panel). The artifacts outside the elliptic area of the brain are due to the "Filtered backprojection" algorithm used for reconstruction.

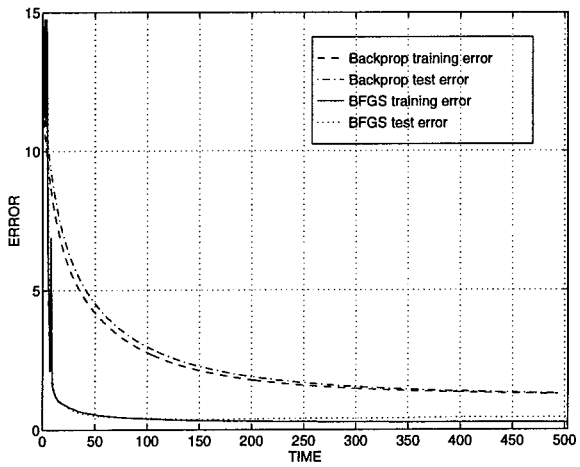


Figure 4: Comparison of time developments for the training process of standard Backpropagation and the BFGS scheme.

NETWORK DESIGN

The network considered is a standard feed forward net with a single layer of hidden units. The activation function of the hidden units are hyperbolic tangents, while the output units are linearly activated. The particular network for this study had 34 inputs corresponding to the activity transient of a given pixel. The net had ten hidden units and three output units. We trained the network by a pseudo second order scheme, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm see e.g. [4]. This algorithm is a variable metric method that constructs a sequence of matrices approximating the inverse Hessian. Using BFGS instead of e.g. standard Backpropagation [5] provides a significant speed-up. This is quantified in figure 4, showing the time development of the training process with backprop and with the BFGS method respectively. Note the scale is in arbitrary *CPU time units* not iterations, since a backprop iteration is faster than a BFGS iteration.

For most adaptive systems the objective is not to learn the training set, but rather to perform well on a much larger set of conceivable inputs, i.e., generalization. The generalization ability depends on architecture and on size of the training set. Hence, for a given architecture, it is important to estimate the necessary number of training cases (pixels) to obtain good generalization. This relation is quantified by the so-called *learning curve* of the architecture as shown in figure 5. We note that a mere 4000 pixels are needed to obtain the asymptotic level of the test error.

Finally we apply the trained network to produce a full estimate of a "slice" of the PET scan as presented in the lower panel of figure 3. It is quite remarkable that the image quality of the rate constants reconstructed by the

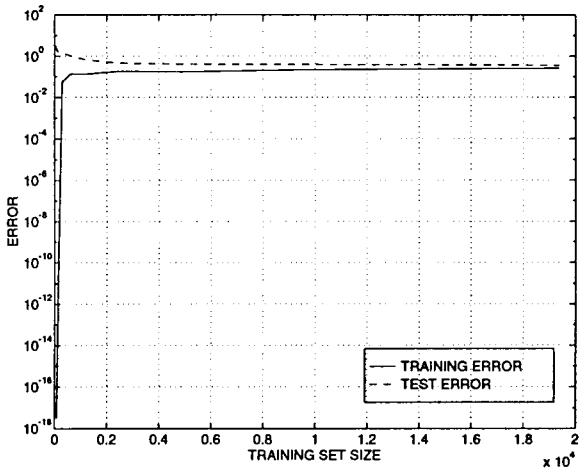


Figure 5: Learning curve for the feed forward net trained by BFGS, approximately 5000 pixels are needed to reach the asymptotic generalization level (about a quarter of the pixels in an image (slice)).

networks inverse model is less noisy than quality obtained from the fitting procedure. The reason is that the network capacity is limited by the inherent constraints of the network architecture, while the Newton fit can lead to arbitrarily (wrong) rate constants for a given pixel if the transient is very noisy. We also note that the execution time for the feed forward network is about two percent of the average time needed to obtain the kinetic constants by fitting the transients with the Newton method.

CONCLUSION

We have shown how a feed forward net may be used for identification of the inverse model for three compartment PET tracer kinetics. Not only is the use of the feed forward net significantly faster than fitting the kinetic model, but our tentative results seem to indicate that the rate constant distribution is regularized by the constraints imposed by the network architecture.

ACKNOWLEDGMENTS

This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT).

REFERENCES

[1] L. Sokoloff, M. Reivich, C. Kennedy, M.H. Des Rosiers, C.S. Patlak, K.D. Pettigrew, O. Sakurada, and M. Shinohara: "The C-14-Deoxyglucose Method for the Measurement of local Cerebral Glucose Utilization: Theory, Procedures and Normal Values in the Conscious and Anesthetized Albino Rat" Journal of Neurochemistry **28** 897-916, (1977).

[2] M.E. Phelps: "Positron Emission Tomography (PET)". In: J.C. Maziotta and S. Gilman (Eds.): *Clinical Brain Imaging, Principles and Applications*, F.A. Davis Company, Philadelphia (1992).

[3] H. Sasaki, I. Kanno, M. Murakami, F. Shishido, and K. Uemuda: "Tomographic Mapping of KInetic Rate Constants in the FDG model Using Dynamic PET". Journal of Cerebral Blood Flow **6** 447-454 (1986).

[4] W. Press et al.: *Numerical Recipes in 'C'*. Cambridge University Press.

[5] D. E. Rumelhart and J. L. McClelland: "Back-propagation of errors" in *Parallel Distributed Processing. Explorations in the Microstructure of Cognition*, Vols. 1–2, MIT Press, Cambridge, Massachusetts (1986).

AUDITORY STREAM SEGREGATION BASED ON OSCILLATORY CORRELATION

DeLiang Wang
Laboratory for AI Research, Department of Computer and Information Science
and Center for Cognitive Science, The Ohio State University
Columbus, OH 43210-1277, USA
Telephone: 614-292-6827; Fax: 614-292-2911
Email: dwang@cis.ohio-state.edu

Abstract - Auditory segmentation is critical for complex auditory pattern processing. We present a generic neural network framework for auditory pattern segmentation. The network is a laterally coupled two-dimensional neural oscillators with a global inhibitor. One dimension represents time and another one represents frequency. We show that this architecture can in real time group auditory features into a segment by phase synchrony and segregate different segments by desynchronization. The network demonstrates the phenomenon that auditory stream segregation critically depends on the rate of presentation. The neuroplausibility and possible extensions of the model are discussed.

1. INTRODUCTION

At any time a listener is being exposed to acoustic energy from many simultaneous auditory events. In order to recognize and understand such dynamic environment, the listener must first disentangle the acoustic wave and capture each event. This process of auditory segmentation is referred to as *auditory stream segregation* or *auditory scene analysis* [1]. It is a critical part of auditory perception.

Auditory segmentation was first reported by Miller and Heise [6] who noted that the listeners split the signal with two sine wave tones into two segments. Segmentation could be obtained with as little as a 15% difference in frequency and throughout the entire frequency range from about 150 Hz to 7000 Hz. Bregman and his collaborators have carried out a series of studies on this subject. In one of the early studies [1], subjects were asked to report the temporal order of the six tones in the sequence. Three of them were in a high frequency range, and the other three in a low frequency range. This situation is simplified into Figure 1. The results showed that at high rates of presentation, subjects perceived two separate sequences corresponding to the high and low frequency tones respectively, and they were able to report only the temporal order of the tones within each sequence, but not across the two sequences. This basic phenomenon of stream segregation was repeatedly verified in different contexts [4] [1]. In general, if auditory patterns are

0-7803-2026-3/94 $4.00 © 1994 IEEE

displayed on a spectrogram, the results are consistent with Gestalt laws of grouping that have been expressed in the visual domain.

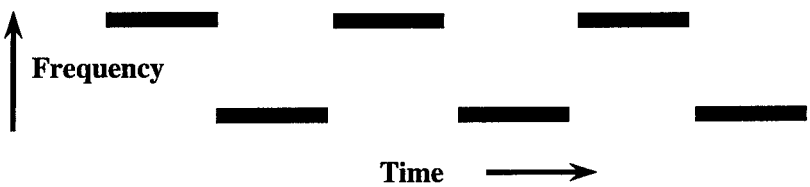


Figure 1. Six alternating high and low frequency tones as displayed in a spectrogram.

von der Malsburg and Schneider [9] proposed perhaps the only neural network model that addressed the problem of auditory segmentation. They described the idea of using neural oscillations for expressing segmentation, whereby a set of features forms the same segment if their corresponding oscillators oscillate in synchrony and oscillator groups representing different segments desynchronize from each other. Using a fully connected oscillator network, they demonstrated segmentation based on onset synchrony, i.e., oscillators simultaneously triggered (a segment) synchronize with each other. Generally, a fully connected network indiscriminately connects all the oscillators which are activated simultaneously by different objects, because the network is dimensionless and loses critical geometrical information. Because of this, their model could not extend very far. For example, the model cannot demonstrate stream segregation which requires an account of frequency proximity. Computer algorithms have been developed to separate two speakers on the basis of different fundamental frequencies [7] [12]. The success of these models is quite limited, and it is not clear how the models could be extended to handle sound separation beyond two talkers speaking voiced sounds.

In the following, we will present a model for auditory stream segregation. Similar to the model of von der Malsburg and Schneider [9], our model is based on the idea of *oscillatory correlation*, whereby phases of neural oscillators encode the binding of sensory components. However, both the single oscillator model and the neural architecture are fundamentally different from those used by von der Malsburg and Schneider. Simulations show that the model demonstrates the basic phenomenon of stream segregation. The framework proposed here promises to explain a variety of experimental observations and to provide an effective computational approach to auditory segmentation (see also [11]).

2. NEURAL ARCHITECTURE

The building block of an oscillator network model is a single oscillator, which in this model is defined as a feedback loop between an excitatory unit and an inhibitory unit

$$\frac{dx_i}{dt} = -x_i + g_x(x_i - \beta y_i + S_i + I_i + \rho) \tag{1a}$$

$$\frac{dy_i}{dt} = -\lambda y_i + g_y(\alpha x_i) \tag{1b}$$

$$g_r(v) = \frac{1}{1+\exp[-(v-\theta_r)/T]}, \qquad r \in \{x, y\} \tag{1c}$$

where α and β are the coupling parameters between the two units. S_i represents input from the other oscillators and I_i represents external stimulation. λ is the decay parameter, and ρ denotes the amplitude of a Gaussian noise term. $g_r(v)$ is a sigmoid gain function with threshold θ_r and parameter T. Eq. 1 is essentially a simplification of the Wilson and Cowan model [13], and the oscillations are driven (induced) by external stimulus.

Inspired by the idea of dynamic links [9], we recently introduced a mechanism called *dynamic normalization* [10]. In this scheme, there is a pair of connection weights from oscillator j to i, one permanent T_{ij}, and another dynamic J_{ij}. Permanent links reflect the hardwired structure of a network, while dynamic links quickly change their strengths from time to time, depending on the current state of the network. More specifically, dynamic normalization ensures that each oscillator has equal overall dynamic connection weights (J_{ij}) from all its input oscillators. With this mechanism, we showed that when triggered by a stimulus, a network of oscillators with just local connections exhibits emergent synchrony across the stimulated regions in the network [10]. Contrary to fully connected ones, locally coupled oscillator networks preserve geometry of input patterns.

The fact that auditory segmentation depends on the rate of presentation calls for a representation of time. In this model, time is treated as a separate dimension. In the simplest form, the segmentation network is a two-dimensional matrix: rows represent frequency (pitch), and columns represent time. This architecture is shown in Fig. 2. The input entering the network always stimulates the left end and the activity of the entire plane shifts towards the right end after a certain time delay, D. In other words, the value of the excitatory unit (x_i) of an oscillator, except the leftmost ones, is set equal to that of its left neighbor every shift step D. Thus, the duration of a tone is represented by the number of consecutive oscillators excited by the tone along its corresponding frequency channel.

There is a common inhibitor y which receives excitation from every oscillator of the network, and sends inhibition back to it:

$$\frac{dy}{dt} = -\lambda y + g_y\left(\frac{1}{N_x} \sum_i x_i\right) \tag{2}$$

Where λ is the decay parameter of the inhibitor as defined in Eq. 1b. $N_x = \sum_i h(x_i)$, a normalizing parameter equal to the number of active oscillators. The

global inhibitor prevents different segments from accidentally grouping together. Intuitively, when the oscillators of one segment reach the peak of their oscillations, they cause the inhibitor to fire strongly. This in turn exerts strong inhibition on other oscillators, and thus prevents them from reaching their peaks of activities.

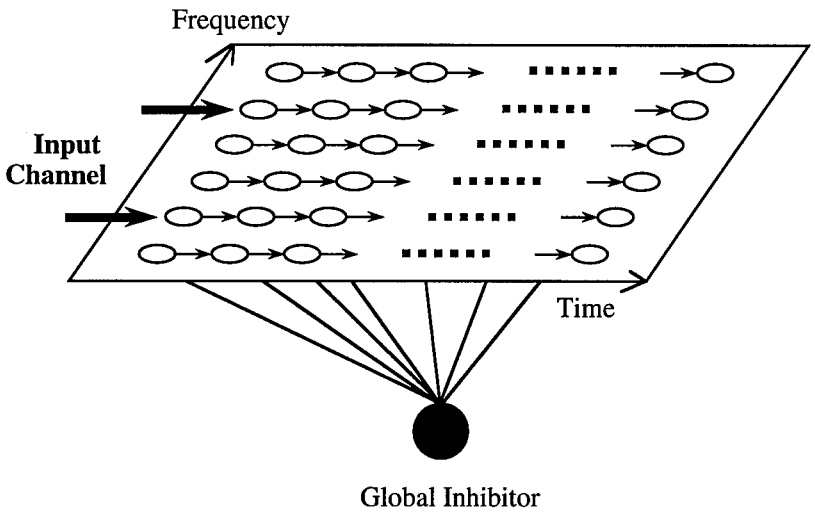


Figure 2. Two dimensional time-frequency matrix for auditory segmentation. The thin arrowheads indicate the direction of activity shift.

In addition to the shift connections, we assume a priori (genetically determined) permanent connectivity between oscillators in the segmentation network (Fig. 2), which, except for the self connection, takes on a two dimensional Gaussian kernel. Assume that the two dimensional indices of oscillator i are (t_i, f_i), representing the time and frequency coordinates respectively. Oscillator i connects to oscillator j with strength

$$T_{ij} = Exp[\frac{-(t_j-t_i)^2}{\sigma_t^2} - \frac{-(f_j-f_i)^2}{\sigma_f^2}]. \tag{3}$$

where the parameters σ_t and σ_f determine the widths along the time axis and the frequency axis of the Gaussian kernel, respectively. The self connectivity $T_{ii} = 0$. In sum, internal input S_i to oscillator i is (cf. Eq. 1a):

$$S_i = \sum_j J_{ij}x_i - \mu y \tag{4}$$

where μ is a parameter.

3. SIMULATION RESULTS

The above architecture for auditory stream segregation has been simulated using a matrix of oscillators with six rows representing distinct frequency channels, as shown in Fig. 2. A sequence of alternating tones *HLHLHL* is used as input, and it is presented to the network in real time. All *H* tones are assumed to trigger the same frequency channel, and so are all *L* tones. The two triggered channels are three rows apart (see Fig. 2). When an oscillator in the left end is triggered, a random phase is generated for it. After a fixed time interval, the activities of the network shift one column to the right. The sequence is repeatedly presented to the network, as in the psychological experiments.

In order to relate to real time, it is assumed that a basic step in simulation corresponds 0.05 ms. The shift time interval is 80 ms. In the time-frequency domain, the presentation rate of a tone corresponds to the number of oscillators occupied by the tone along the time axis. We conducted three groups of simulations with presentation rates 160 ms, 240 ms, and 320 ms per tone, corresponding to fast, medium, and slow presentations, respectively. Thus, for fast presentation each tone occupied two oscillators, for medium presentation three oscillators, and for slow presentation four oscillators.

When the presentation rate was fast (2 oscillators per tone), a network of 6x12 oscillators was simulated. Except a brief beginning period of each column (shift interval), all active oscillators of the *H* channel quickly reached synchronization, and so did the oscillators of the *L* channel. The oscillators of one channel desynchronize with those of the other channel. This phenomenon was particularly stable after the first cycle of sequence presentation was finished. Figure 3 shows the combined activity of the frequency channels for a typical interval after the entire sequence was presented. The top panel of the figure depicts the stimulation pattern, showing that one tone occupies two oscillators, while the middle and the bottom panels show the combined activities of the *H* and *L* frequency channels respectively. Synchronization within the same frequency channels and desynchronization across the two channels are clearly illustrated in this form of display. Relating to the experiments, stream segregation occurred for this rate of presentation, and two streams were segmented apart in real time.

When the presentation rate was medium (3 oscillators per tone), a network of 6x18 oscillators was simulated. Similar to Fig. 3, the top panel of Fig. 4 illustrates the stimulation condition. The lower two panels show the combined activities for the *H* and *L* channels, respectively. As shown in the figure, the oscillators within each channel did not synchronize, but instead exhibited two distinct phases after an initial transient. Actually, two neighboring *H* tones formed one segment, two neighboring *L* tones formed another segment, and one *H* tone and its neighboring *L* tone formed yet another segment. As can be seen from the display, there were three distinct phases in total. In sum, for the medium presentation rate, phase synchrony was not reached across the entire same frequency channel. Rather, partial stream segregation, e.g. among two consecutive *H* or *L* tones, was exhibited in the simulation.

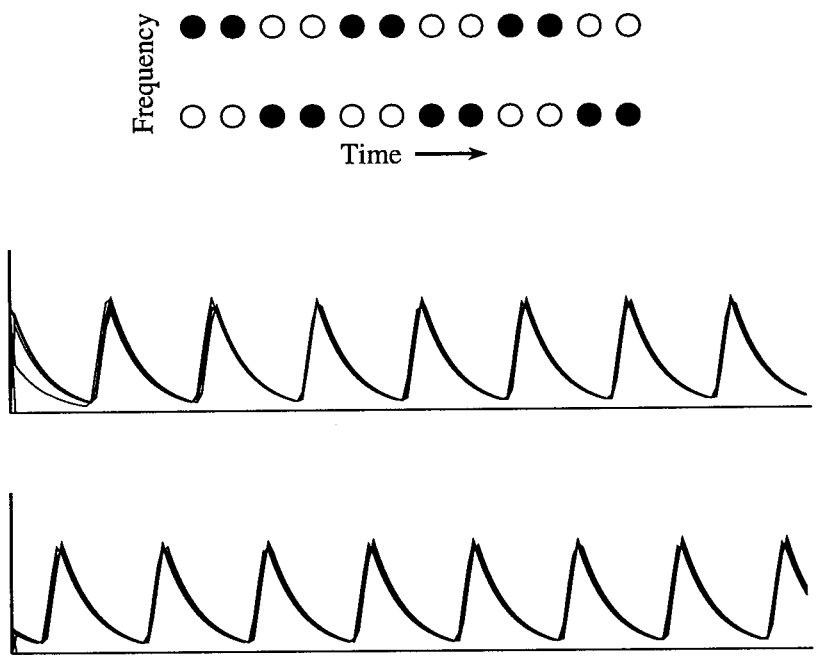


Figure 3. Top panel: a snapshot of the stimulus pattern in the H and L channels. **Middle and bottom panels**: The combined activities of all the oscillators in the H and L channels, respectively. The parameters $\alpha = 0.6$, $\beta = 2.5$, $\rho = 0.01$, $\lambda = 1.0$, $\theta_x = 0.6$, $\theta_y = 0.15$, $T = 0.025$, $\sigma_t = 2.75$, $\sigma_f = 1.8$, $\mu = 0.5$, $I_i = 0.7$ if oscillator i is externally stimulated, and $I_i = 0.0$ otherwise.

Finally, when the presentation rate was further slowed down to 4 oscillators per tone, a network of 6x24 oscillators was simulated. Again, the top panel of Fig. 5 shows the stimulation pattern. From the combined oscillations in the lower two panels, one can easily observe that there are three distinct phases within each frequency channel. Each phase corresponds to one tone. In other words, there was no phase synchronization at all between different tones of the same frequency. Rather, one tone was grouped into one segment with a neighboring tone of the other frequency. This can be seen by comparing the lower two panels with respect to time. In sum, with this rate of presentation, one tone formed a segment with one neighboring tone of another channel, and the six tones were segregated into three segments.

From these simulations, we can conclude that tones can be grouped together based on their similarities in frequency, and segmentation critically depends on the rate of presentation. Stream segregation is best for high rates of presentation, absent for low rates, and is intermediate for medium rates of tone presentation. Why does such behavior occur? The behavior can be explained by competition and cooperation in the network. The neighboring oscillators representing the same

tone always synchronize because they are strongly coupled with each other [see (3)]. For the fast presentation, neighboring tones of the same frequency are separated by only two oscillators (the top panel of Fig. 3). Thus, strong coupling among them group them into the same segment. With every thing else the same, slowing down the rate of presentation increases the distance between the neighboring tones of the same channel, and thus reduces their coupling. Recall that each oscillator has the same overall amount of incoming dynamic links. So the reduced coupling will increase the relative coupling across the different channels. That explains why with slower presentation, neighboring tones *in time* are more likely to be grouped together. In the case of Fig. 5, only neighboring tones in time are grouped into the same segments. The global inhibitor serves to segment the stimuli on the entire network into different segments.

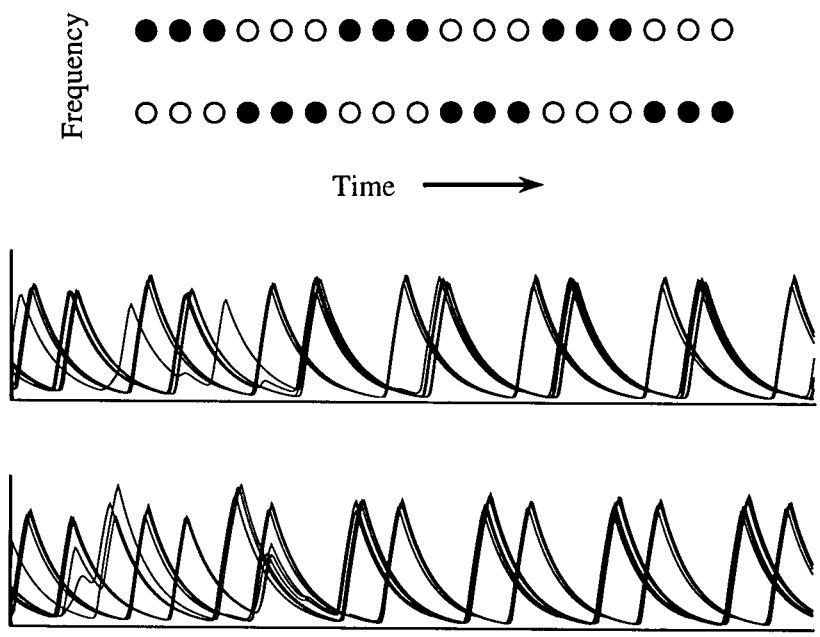


Figure 4. Top panel: a snapshot of the stimulus pattern in the *H* and *L* channels. **Middle and bottom panels**: The combined activities of all the oscillators in the *H* and *L* channels respectively. The parameter values are the same as in Fig. 3.

4. DISCUSSION

There is ample evidence suggesting the existence of neural oscillations in the brain. It has been observed that local field potentials in the visual cortex and the sensorimotor cortex show stimulus-driven synchronous oscillations [3] [8]. The range of the frequencies of these oscillations is between 20 and 80 Hz, often referred to as 40 Hz oscillations. 40 Hz oscillations of auditory evoked potentials

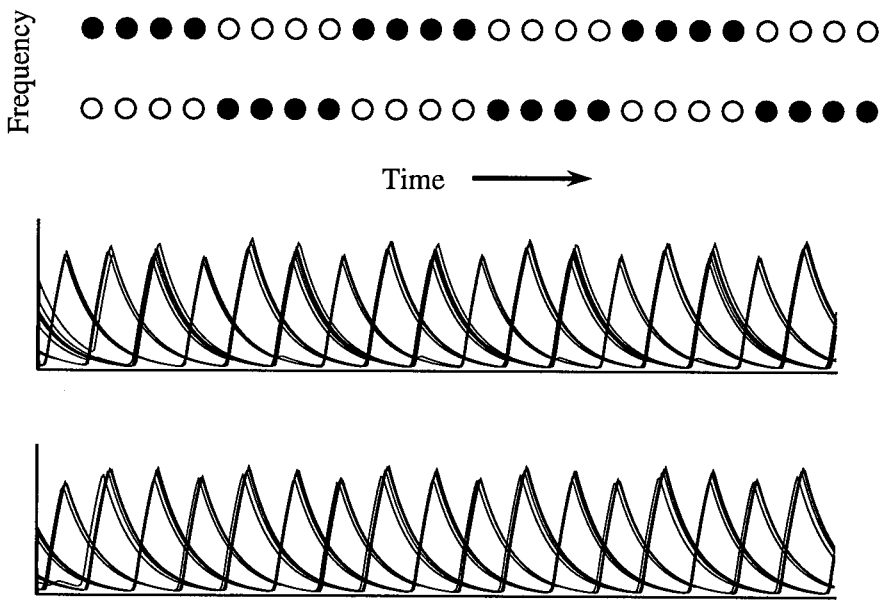


Figure 5. Top panel: a snapshot of the stimulus pattern in the *H* and *L* channels. **Middle and bottom panels**: The combined activities of all the oscillators in the *H* and *L* channels respectively. The parameter values are the same as in Fig. 3.

were also observed by Galambos et al. [2]. These oscillations can last for several cycles after the stimulus presentation is over. The observation was later confirmed by Madler and Pöppel [5], who further found that these characteristic oscillations were absent from the patients with deep anesthesia.

An important element of the architecture of the model is the use of shift circuits with delay lines. Time delays of neuronal responses have been found at various levels of the visual pathway, and it appears that delays become longer in higher auditory structures. In the cat auditory cortex, electrophysiological recordings identify up to 1.6 second delays in response to identical tones separated by certain periods or a sequence of different tones.

Although Sect. 3 shows only the preliminary simulation results of auditory stream segregation, the model is not limited by the stimuli used. For example, the three high (low) tones do not have to trigger the same frequency channel. As long as they trigger nearby frequency channels, auditory segmentation based on frequency similarity will occur. This is because synchronization of oscillators depends on the connection strengths between them, and oscillators with similar frequency coordinates have relatively large mutual connection strengths according to the Gaussian kernel (3). By the same token, each tone need not be a pure tone. A tone with frequency modulation will work similarly. In essence, the priori connectivity pattern of a Gaussian kernel strongly biases towards the grouping of sounds that have continuous frequency transitions, which is consistent with the analysis of speech perception [1]. Of course, the basic architecture of Fig. 2 must

be extended in order to incorporate other qualities of auditory stimuli, such as amplitude, rhythm, harmonics, timbre, etc. However, the basic principles of this model should remain the same.

ACKNOWLEDGMENTS The preparation of this paper was supported in part by ONR grant N00014-93-1-0335 and NSF grant IRI-9211419.

REFERENCES

[1] A.S. Bregman, Auditory Scene Analysis, Cambridge MA: MIT Press, 1990.

[2] R. Galambos, S. Makeig, and P.J. Talmachoff, "A 40-Hz auditory potential recorded from the human scalp," Proc. Natl. Acad. Sci. USA, vol. 78, pp. 2643-2647, 1981.

[3] C.M. Gray, P. König, A.K. Engel, and W. Singer, "Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties," Nature, vol. 338, pp. 334-337, 1989.

[4] M.R. Jones, "Time, our lost dimension: toward a new theory of perception, attention, and memory," Psychol. Rev., vol. 83, pp. 323-355, 1976.

[5] C. Madler and E. Pöppel, "Auditory evoked potentials indicate the loss of neuronal oscillations during general anesthesia," Naturwiss., vol. 74, pp. 42-43, 1987.

[6] G.A. Miller and G.A. Heise, "The trill threshold," J. Acoust. Soc. Am., vol. 22, pp. 637-638, 1950.

[7] T.W. Parsons, "Separation of speech from interfering speech by means of harmonic selection," J. Acoust. Soc. Am., vol. 60(4), pp. 911-918, 1976.

[8] J.N. Sanes and J.P. Donoghue, "Oscillations in local field potentials of the primate motor cortex during voluntary movement," Proc. Natl. Acad. Sci. USA, vol. 90, pp. 4470-4474, 1993.

[9] C. von der Malsburg and W. Schneider, "A neural cocktail-party processor," Biol. Cybern., vol. 54, pp. 29-40, 1986.

[10] D.L. Wang, "Modeling global synchrony in the visual cortex by locally coupled neural oscillators," in Proc. of the 15th Ann Conf. Cognit. Sci. Soc., Boulder CO, 1993, pp. 1058-1063. For a more extended version, see D.L. Wang, "Emergent synchrony in locally coupled neural oscillators," IEEE Trans. on Neural Networks, in press.

[11] D.L. Wang, "A computational theory of temporal pattern segmentation," in Neural representation of temporal patterns, H. Hawkins, T. McMullen, and R. Port, Ed. New York: Plenum, to appear, 1995.

[12] M. Weintraub, "A computational model for separating two simultaneous talkers," in IEEE ICASSP, Tokyo, 1986, pp. 81-84.

[13] H.R. Wilson and J.D. Cowan, "Excitatory and inhibitory interactions in localized populations of model neurons," Biophys. J., vol. 12, pp. 1-24, 1972.

APPLICATION OF NEURAL NETWORKS FOR SENSOR PERFORMANCE IMPROVEMENT

S. Poopalasingam, C.R. Reeves and N.C. Steele
Control Theory and Applications Centre, Coventry University,
Priory Street, Coventry CV1 5FB, U.Kingdom.
Tel: +44 203 838972, Fax: +44 203 838585.
email : NSTEELE@uk.cov.ac.

Abstract. Sensor technology has developed in parallel with advances in the fields of electronics and computing. Beyond obtaining a suitable sensing element, stringent demands on accuracy has led to continued developments in the improvement of compensation and calibration techniques. Typically, signal conditioning would attempt to minimise the effects of zero offsets and nonlinear temperature and pressure effects. Conventional analogue compensation methods have been phased out in favour of digital methods which provide a lower cost solution due to the reduction in test and calibration time. However, digital methods currently employed have been deemed to be insufficiently accurate or highly memory intensive, thus there is a need for an alternative approach that provides a compromise between the above. The use of neural networks may offer this compromise, with the added advantage of possessing certain characteristics that could contribute to the development of a smart transducer

Introduction

Classical transducers relied on analogue circuitry to store compensation and calibration data. Typically, analogue 'storage' components such as laser-trimmed resistors and potentiometers were used. With the advent of digital technology, non-volatile memory components such as EPROMs and RAMs were phased in to take the place of analogue components. The practice adopted was then to obtain sensor calibration data during the manufacturing process. Based on data provided, a table of correction coefficients is derived and stored in memory, thus making memory requirements an issue to be considered. In an attempt to reduce memory requirements, reduced density look-up tables with linear interpolation have been implemented.

Another widely used approach has been to use compensating algorithms. In this approach, the sensor output surface is modelled using polynomial fitting techniques. The memory requirements are dramatically reduced because only

0-7803-2026-3/94 $4.00 © 1994 IEEE

the polynomial coefficients need to be stored. It has been found, however, that such an approach is incapable of producing the required accuracy even when using high order polynomials.

In an attempt to find a compromise between the memory requirements of the look-up table approach and the poor accuracy for the polynomial fitting method, the possiblity of using neural networks in this application has been investigated. The work reported in this paper has been based on pressure transducers that have wide-ranging applications from consumer applications to the aerospace industry. The accuracy required is application dependent and ranges from 0.01% to 2.0% of the full scale output (FSO). The added advantage of using neural networks is the ability to realise a generic mapping for a specific type of sensor, thus eliminating the need to obtain individualized sensor data. It is also likely that the neural network based calibration module can be implemented within the sensor housing, eliminating the need for external hardware support and in line with the aims of developing a smart transducer. Initial work concentrated on compensating for temperature and pressure nonlinearities. The accuracy level after linearizing the temperature and pressure effects will be limited by the hysteresis effects and time-related stability.

A multilayer perceptron (MLP) based compensation module

Data from two types of devices were used. The first device was used in a feasibility study and was selected due to its relatively good temperature stability, however no validation data was available for this device. The second device, also a silicon based transducer, was then used to obtain sufficient data for validation and to realise a generic mapping for that particular type of sensor. This paper concentrates to a larger extent on the feasibility study carried out, although the findings from this study have been applied to the second device.

The ability of a neural network to realise an arbitrary nonlinear mapping is well established [1], [2]. The mapping to be realised was the inverse sensor characteristics. The training data consisted of calibration data obtained under controlled laboratory conditions. The response of the first device within a specified pressure range for different temperatures is shown in Figure 1. The sensor response readings were the output of an analogue to digital converter.

Training was carried out using the backpropagation learning algorithm. As a first approach, optimal network parameters were heuristically found. The required error specification for this transducer is an error range of within 0.1% of the FSO. The error based on the FSO is referred as the full scale error (FSE). As a network approximation ability measure, the average of the integral of the

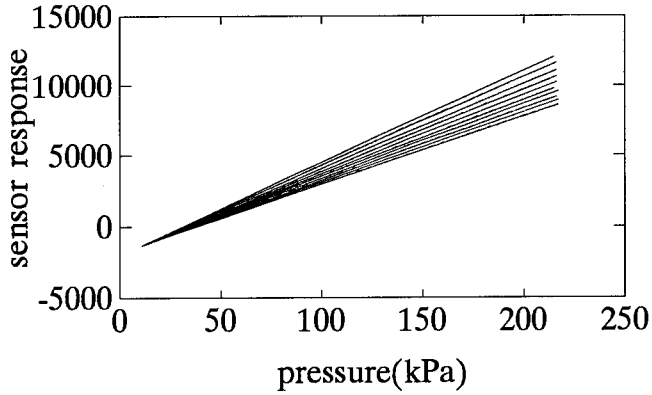


Figure 1: Characteristics of first device for different temperatures

squared full scale error measure (AIFSE) was used. Using single and two layer networks, very similar results were obtained. Analysis of the performance of the single layer network was carried out and the error distribution within the input space was plotted. From the results, it was apparent that the worst cases occur at the edges and in high temperature regions of the input space, this is shown in Figure 2. The error criterion for this plot has been increased to 0.3% to clearly illustrate where the points of poor approximation lie. The symbol '*' indicates cases classified as 'good' and 'o' the poor.

In an attempt to improve performance at the edges of the input space, several approaches were investigated. The first method was to use nonlinear scaling of inputs. It was hoped that this method would improve the edge effects as well as reduce the amount of nonlinearity the network was required to learn. However, this method fared very poorly. As a next step, redundant information was introduced. Although by no means conclusive, it was discovered that introducing fewer cases a larger number of times tended to produce better results. The introduction of redundant information gave an improved performance of approximately 10% but resulted in longer training times due to the larger training set. Due to the fact that both the above approaches did not give much improved results, it was conjectured that if the training set contained data over a wider region than the operating region of the sensor, the poor approximation at the edges of the input space could be improved. To test this, as data over a wider operating region were not available, the existing data set was reduced by removing cases lying on the edges of the input space. The network trained with the reduced data set produced cases exhibiting large errors at the edges of the reduced input space. The cases at the edges of the reduced data space previously lay on the centre of the original input space and exhibited low er-

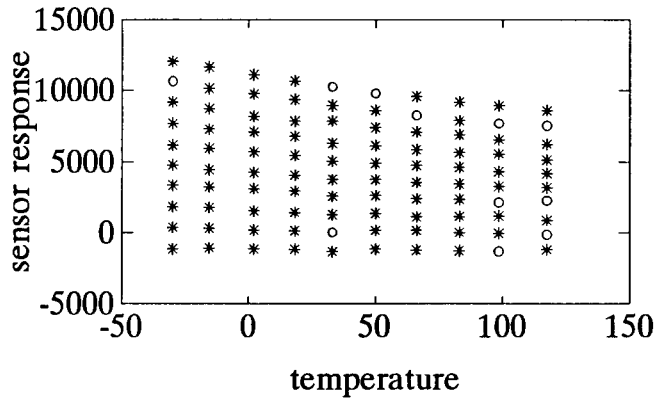


Figure 2: Distribution of cases within the input space

ror values. Thus, the above conjecture would appear to be supported by these findings.

Optimization of network structure using Genetic Algorithms(GA)

To confirm that optimal network structures were being utilised in the above, a GA was used for the full training set. Using an appropriate performance measure, found by the GA was a single layer network which resulted in an improvement of 10% over the previous findings obtained using the single layer network. Although this finding is comparable to that of the best result found by introducing redundant information, it is more efficient in terms of training set size, and thus training times. The results obtained using different MLP architectures are summarised in Table 2. With only 41% of the test cases lying within the error criterion at best, attention was focussed on an alternative learning paradigm.

The Radial Basis Function (RBF) approach

Due to the long training times required by the backpropagation and the poor match to the error criterion, the RBF network was implemented. As discussed by Poggio and Girosi in [3],[4], RBF networks have the best approximation properties. Moreover, the higher degree of local computation results in faster training

Table 1: SUMMARY OF MLP RESULTS

NETWORK	% OF CASES < 0.1% FSE
2-7-1 100 training cases	31
2-9-1 100 training cases	37
2-15-1 100 training cases	32
2-7-1 154 training cases(18*3)	41
2-8-1 100 training cases (selected by GA)	40
2-5-3-1 100 training cases	38

times. The kernel function used in the RBF was a Gaussian nonlinearity. Selection of the Gaussian centres was carried out using the k-means algorithm in the initial phase of implementation [5]. The scaling parameters in the Gaussian were set to unity. The number of centres were then chosen to obtain the best fit. The networks with centres found using the clustering algorithm fared poorly compared with the those with regularly spaced centres. This can be explained in terms of the distribution of the data within the input space. Using the RBF there was an improvement of 39% over the results obtained using the best MLP network, i.e. that found by the GA.

The choice of scaling parameters for a given application can be carried out in numerous ways. In [6], some possible approaches are discussed. For the problem of sensor calibration, the optimal scaling parameters values were selected using a GA [7]. Compared to the most favourable result obtained using the RBF networks with unity parameter values, similar results were obtained but with a reduction of hidden unit numbers by approximately 80%. Thus, with optimal kernel function parameters the RBF network outperformed the MLP network of comparable size in terms of both mapping accuracy and efficiency. A summary of the results obtained using the RBF network is given in Table 2. The three figures appearing in each block in the second and third columns of the table correspond to a measure of the AIFSE, the percentage of test cases lying within 0.1% of the desired response and those lying within 0.4% respectively.

Table 2: SUMMARY OF RBF RESULTS

No. of centres	k-means	regular spacing
13	6.356	6.282
($\sigma = 1$)	4	4
	14	15
30	0.176	0.021
($\sigma = 1$)	18	52
	69	99
50	0.008	0.011
($\sigma = 1$)	79	69
	100	100
10	0.008	-
($\sigma = 4.32$)	79	-
	100	-

Validation of inverse model and obtaining a generic model

Due to the lack of data, severe limitations were placed on the ability to validate the model obtained for the first device. Data from the second device was used to model the inverse behaviour, and as was the case with the first device, the network approximation was good over the entire operating temperature range. This is shown in Figure 3. The dotted line in the figure represents the mapping, the solid line represents the network approximation.

Validation of this model is currently being carried out. The next step would be to attempt to realise a generic inverse model for this particular sensor type as opposed to individual sensor model. Data from three sensors of this particular type is available to realise this general model. Analysis of the data seems to indicate that the bridge resistance readings may be critical as an input in training the network.

Hysteresis Effects

Having considered issues pertaining to temperature and pressure nonlinearities and because hysteresis imposes a limit on the attainable sensor accuracy, some consideration needs to be given to the elimination of hysteresis. Hysteresis effects exist for both pressure and temperature variations. The data was obtained under laboratory conditions for pressure and temperature ramped across

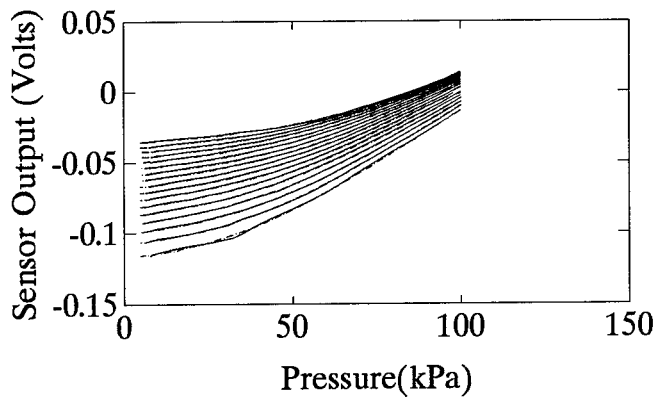


Figure 3: Characteristics of second device for different temperatures

the operating range of the sensor. Only pressure hysteresis is being considered currently due to constraints in time available for testing. The data was treated as a time series and a recurrent network with a Widrow delay network at the output was used. Using a method proposed in [8], the δ-test, the minimum embedding dimension of this series was found to be two. Using this information to design the network configuration, the hysteresis time-series was successfully reproduced by the neural network. The next stage of this work will involve obtaining data from the transducer based on the outcome of an experimental design as it is a realistic assumption that the pressure excursions that a sensor is subjected to in a practical application will not range from minimum to maximum in a regular fashion. As such, these issues need to be carefully considered in order to obtain representative data for purposes of training the compensation module to minimise the hysteresis effects.

References

[1] V.Y. Kreinovich. Arbitrary nonlinearity is sufficient to represent all functions by neural networks : A theorem. *Neural Networks*, Vol. **4**:pp 381–383, 1991.

[2] K. Funanashi. On the approximate realisation of continuous mappings by neural networks. *Neural Networks*, Vol. **2**:pp 183–192, 1989.

[3] T. Poggio and F. Girosi. Networks for Approximation and Learning. *Proceedings of the IEEE*, Vol. **78**:pp 1481–1497, September 1990.

[4] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, Vol. **63**:pp 169–176, 1990.

[5] M.T. Musavi et al. On the training of radial basis function classifiers. *Neural Networks*, Vol. **5**:pp 595–603, 1992.

[6] J. Moody and C.J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, Vol. **1** , 1990.

[7] A. Whitwood. Neural Networks and Genetic Algorithm Coursework. *MSc. Mathematical Modelling and Computer Simulation, Coventry University*, 1993.

[8] P. Hong and C. Peterson. Finding the Embedding Dimension and Variable Dependences in Time Series. *Neural Computation (submitted)*, March 1993.

NEURODEVICE - NEURAL NETWORK DEVICE MODELLING INTERFACE FOR VLSI DESIGN

Pekka Ojala, Jukka Saarinen and Kimmo Kaski
Tampere University of Technology, Electronics Laboratory,
P.O. Box 692, FIN-33101 Tampere, Finland
e-mail: ojala@ee.tut.fi, jukkas@ee.tut.fi, kaski@ee.tut.fi

Abstract. A novel, fast and accurate neural network tool is proposed for efficient technology independent implementation of the interface between device modelling and circuit simulation. Modified backpropagation, conjugate gradient and Levenberg-Marquardt optimization algorithms are applied in network teaching. Simulations show fast convergence and an excellent fit of recalled characteristics to the measured device data. The utilized algorithms are robust and capable of presenting the entire device characteristics unaltered even with largely reduced amount of the teaching material. The good monotonicity of the neural network generated device data facilitates the usage of the method in circuit simulation purposes. Possible further applications of implementing circuit level macromodels with this technique are discussed.

INTRODUCTION

The integration of measured semiconductor device behaviour or data from numerical device simulators into a circuit simulator has been a long standing problem for the integrated circuit design. For digital circuits the requirement of accurate functional modelling includes knowledge of device currents together with the internal and external RC-products to facilitate proper timing simulations. For analog circuit designs the simulation has proven more difficult. The linear circuit gain is governed by the small-signal parameters, *transconductance* and *output conductance*, and their frequency behaviour. Therefore, accurate and continuous models for small-signal parameters over the complete operation region are required. Precision modelling of analog circuits requires accurate presentation of the substrate effects in dc- and ac-operation, as well. Other inaccuracies have been attributed to poor or non-existent modelling of the subthreshold region, non-linearities in device operation and voltage dependent capacitances.

Typical approaches for the device modelling interface to circuit simulation have included analytical, parameterized semiconductor device models [1,2], table look-up models with various interpolation techniques [3,4], and a more atypical method of tensor product splines [5].

The most widely applied approach of using *parameterized analytical models* for presenting electrical characteristics of a device has been troubled with several difficulties. The parameter extraction presents a difficult problem even if the models are

0-7803-2026-3/94 $4.00 © 1994 IEEE

physically sound and include every possible physical phenomena that completely describes the device. Two approaches can be utilized in the device parameter extraction. First, the parameters for physically based models can be defined from the time consuming *extracting measurements* for the device. Second, the extracted inaccurate set of parameters can be *fine-tuned in a general non-linear optimizer* to fit the model to the measured or numerically simulated device data [6,7,8]. This approach is, however, capable of producing highly unphysical interpretations of the device parameters with unexpected and detrimental effects during circuit simulation.

To overcome the difficulties in parameterized models, a variety of table look-up methods with different interpolation techniques have been used [3]. These models typically store the device current in a table. For a four terminal device such as MOSFET or MESFET, a 3D table is formed as a function of gate, drain and substrate potential. The table size, therefore, grows as the third power of the number of input vectors, and becomes the limiting factor in the modelling accuracy.

NETWORK STRUCTURE AND METHODOLOGY FOR MODELLING

The neural network that is implemented in this study is a three-layer feedforward perceptron network that consists of input, hidden, and output layers. Each layer contains several processing elements with sigmoidal nonlinearities. Cybenko [9] has shown that this network can be used to approximate arbitrary functions, *i.e.* it can model any continuous nonlinear transformation. The network is feedforward in the sense that each unit receives inputs only from the units in the preceding layer. The network converts input signals according to connection weights. During learning, connection weights are adjusted in a direction to minimize the sum of squared errors between the desired outputs and the network outputs.

In the following, the subscripts k, j and i refer to any unit in the output, hidden, and input layers, respectively. The total inputs to unit j in the hidden layer or unit k in the output layer is

$$net_r = \sum_S \left(\sum_{i,j,k} w_{rs} O_S \right), \qquad r = k, j; \quad s = j, i; \tag{1}$$

where w_{rs} is the weight from the sth unit to the rth unit and O_S represents the output of unit s in the hidden and input layers. A sigmoidal nonlinearity is then applied to each unit r to obtain the output as

$$O_r = \frac{1}{\exp\{-(net_r - \theta_r)\}} \qquad or \qquad O_r = \tanh(net_r - \theta_r), \tag{2}$$

where θ_r serves as a threshold of unit r. Hence, each layer communicates with all successive layers. There is no feedback within the network between layers of individual units and no communication with other units in the same layer.

In the learning process, the network is presented with a pair of patterns, an input pattern, and a corresponding desired output pattern. Learning comprises of changing the connection weights and unit thresholds to minimize the mean squared error between the actual outputs and the desired output patterns with the gradient descent

method. The activity of each unit is propagated forward through each layer of the network by using (1) and (2). The resulting output pattern is then compared with the desired output pattern, and an error for each output is calculated.

We have minimized the error in the network using three different gradient based optimization algorithms. These include the modified back-propagation algorithm [10], the conjugate gradient [11], and Levenberg-Marquardt algorithms [11].

To accelerate the convergence of the standard back-propagation algorithm, we have used the modified back-propagation method presented by Vogl *et.al.* [10]. This algorithm includes three main modifications. The first modification is that the network weights are not updated after each learning pattern. Instead, the weights are modified only after all input patterns have been presented. The changes for each weight are summed over all of the input patterns and the sum is applied to modify the weight after each iteration over all the patterns. Other modifications include the altered learning rate η which controls the step size and the momentum factor α. The learning rate η is varied according to whether or not an iteration decreases the total error for all patterns. If an update results in reduced total error, η is multiplied by a factor $\phi > 1$ for the next iteration. If a step produces a network with a total error more than a few percent above the previous value, all changes to the weights are rejected, η is multiplied by a factor $\beta < 1$, α is set to zero, and the step is repeated. When a successful step is taken, α is reset to its original value. For a successful step α resembles momentum as it tends to favour the change of weights to the earlier successful direction.

The conjugate gradient algorithms are standard optimization algorithms [11] which apply information from second order Taylor-series approximation to choose the search direction more carefully than the steepest descent algorithm. The search is chosen in a conjugate direction which means that the new search direction is orthogonal to all the previous directions. This makes the method faster than the steepest descent method with a line search optimized step size to provide a direction that is orthogonal only to most recent iteration. In our implementation no line search was performed to optimize the step size but a constant step in the new conjugate gradient direction was chosen. The information from the numerical estimate of the second derivative was used to modify the size of the step to be taken to the previous conjugate direction and this together with the present gradient direction provided the new conjugate direction.

The Levenberg-Marquardt optimization algorithm [11] applies, similarly as the conjugate gradient method, numerically estimated information from the second derivative of the cost function. In addition, the diagonal elements of the second derivative matrix are assumed to contain information on the scale of the problem. This information is used to optimize the step size. The Marquardt parameter, which is altered according to the success in optimization, is summed to diagonal elements of the Hessian matrix. The Levenberg-Marquardt method changes, therefore, from the inverse Hessian type method far from minimum to the steepest descent method close to the minimum. Both the conjugate gradient and the Levenberg-Marquardt methods were used in a batch mode to update the network weigths. Similarly, as for the modified back-propagation algorithm, the weigths were modified only after all input patterns were presented to the network.

RESULTS, COMPARISONS AND DISCUSSIONS

For the circuit simulation several characteristic data from the devices are required to facilitate the variety of analyses of interest, namely the operation point, transient, dc- and ac-analysis. The required data, for example for FET-devices includes the terminal currents versus bias voltages and terminal non-reciprocal non-linear relationships of capacitances versus bias voltages. Typically, these models also present small-signal parameters analytically for the FET. In Fig. 1 we present the methodology for providing these data to the circuit simulation environment by using a perceptron network that was described above.

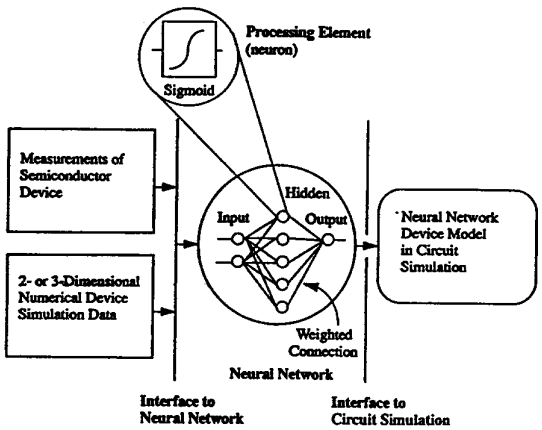
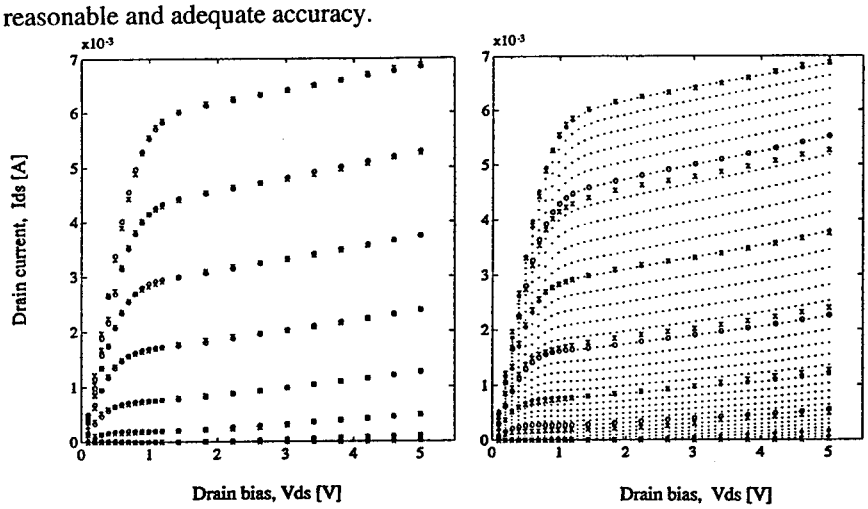


Figure 1:
Neural network device modelling methodology.

As the targets of our neural network device characteristic modeller, we have chosen a GaAs MESFET data and short channel device modelling data from BSIM MOSFET model. The GaAs MESFET data serves as a fairly difficult task because the simulation accuracy of analog GaAs circuits has consistently been worse than for comparable Si MOSFET technology owing to less evolved device models. The BSIM model data was chosen to study the effects of device geometry on the modelling accuracy. In Fig. 2 a) we present the modelling results for the drain current curve family of GaAs MESFET with a geometrical form factor $W/L = 50\mu m/2\mu m$ (data from [12]). The modelling output from the network is marked with 'o' and it is superimposed on the training data that is marked with 'x'. A very good fit with an average relative error of 1.41 per cent was reached. The precision modelling of the dc-curve family for GaAs MESFET has been difficult with the analytical models because of the complicated second order phenomena in device physics. These device anomalies include backgating or sidegating effects from adjacent devices, impact ionization triggered leakage current to substrate and gate terminal, drain potential induced barrier lowering for short devices, and deep level trapping dependent leakage current and subthreshold characteristics [12]. Therefore, even with a set of optimized model parameters the above scale of modelling error has been unavailable.

A more demanding test for the network is performed by reducing the amount of

training material to one half of the original data by removing every other gate bias value from the data set and calculating the network weights again for the remaining data. These weights are further used to recall the original data set to interpolate the curves for every other value of gate bias. Fig. 2 b) presents the result of modelling the interpolation of data. The measured data in this particular test is marked with 'x' and the learnt data for the trained points of data set with '*'. The interpolated, recalled data corresponding measured points is marked with 'o' and the recalled mesh data with '.'. The interpolated data points show 6.25 per cent residual error in modelling, compared with 1.81 per cent for the data that was used in teaching. The substantial reduction of the teaching material has not lead into a catastrophic modelling failure. Instead, the network is still able to present the intermediate data points with reasonable and adequate accuracy.



a) b)

Figure 2:
a) Measured and neural network generated characteristics for GaAs MESFET drain current versus drain and gate voltages, 'x': measured data, 'o': simulated data. b) Interpolation capability of the neural network, 'x': measured material, '*': learnt data corresponding teaching points, 'o': interpolated data, '.': mesh data.

The evolution of the modelling error for the above data is presented in Fig. 3. Here we have shown the modelling error from 0 to 500 epochs, where each epoch represents one update for the weigth vector. The conjugate gradient method requires larger number of epochs than the Levenberg-Marquardt method. On the other hand, the Levenberg-Marquardt algorithm uses an order of magnitude more CPU-time for one epoch updating than the conjugate gradient method. Both algorithms are capable of finding a better minimum with considerably less epochs than the standard and modified backpropagation method.

Next, we present an example of small-signal parameter modelling, namely the GaAs MESFET output conductance and transconductance. The output conductance modelling with analytical models has suffered from discontinuous models between regions of device operation. In comparison, the neural model provides fully contin-

uous and monotonous modelling (Fig. 4). The network learnt the teaching material with the final relative error of 2.26 per cent. In comparison the typical accuracy for analytic output conductance modelling with optimized model parameters has remained at 5-10 per cent for analog CMOS technologies [7]. The result of the transconductance modelling is presented in Fig. 5 with the temperature and gate potential as input variables. The residual relative error in Fig. 5 is 4.6 per cent.

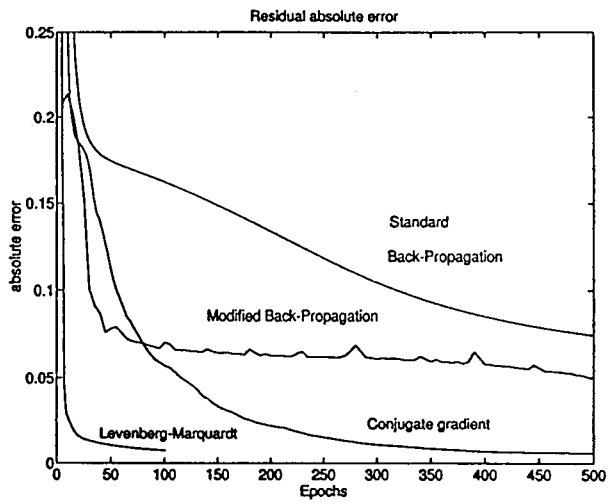


Figure 3:
Evolution of the sum of squares error with the used algorithms.

For transient and ac-analysis the device capacitances will have to be accurately modelled. In Fig. 6 we present the measured [13] and recalled gate to source capacitance modelling with the network. The nonlinear relationship of capacitance wrt gate bias is accurately reproduced.

In order to provide a useful and fast device modelling interface neural network will have to be capable of modelling device data with more than two input dimensions. The 3D table look-up models allow presentation of the FET device data with respect to all three terminal potentials differences (Vgs, Vds, Vsb). We demonstrate the similar modelling by adding a third processing element for the 3D input data-vector in our network. In Fig. 7, recalled GaAs MESFET drain current characteristic is presented for the subset of learnt gate bias regime, from 0.0 to 0.3 V. The average relative error for the network was 2.28 per cent. Previously, a 3-4 per cent relative error for short-channel Si MOSFET had been demonstrated with 3D table look-up technique [3].

To estimate the capability of the network to present variable device geometries, we have generated device length and width dependent small geometry MOSFET data using BSIM-model with realistic model parameters [1]. The device channel length and width were varied from 2μm to 8μm. Nine different device geometries were used in teaching the network. We used a two-stage network where the first stage implements the electrical dependences and the second stage the geometrical factor of the device. The representative modelling results are given in Fig. 8 for de-

vice size of W/L=4μm/4μm with V_{bs}=0. The overall relative sum of squares error after the second stage for all data was 7 per cent with the average error of 0.8 per cent. The teaching was performed using the Levenberg-Marquardt method with 1000 epochs.

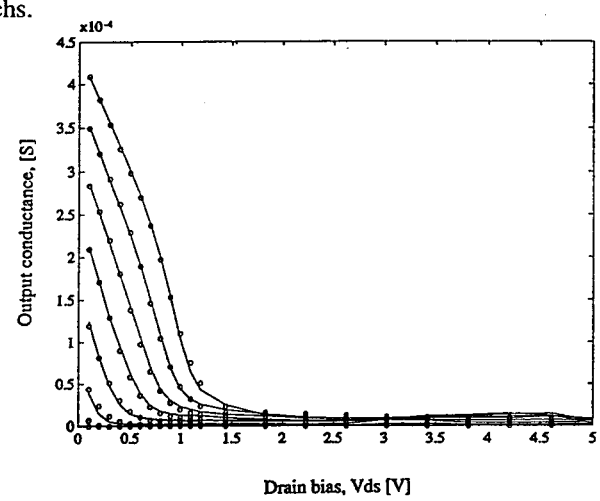


Figure 4:
GaAs MESFET output conductance modelling with the neural network. Solid lines as measured data, 'o': data recalled by network.

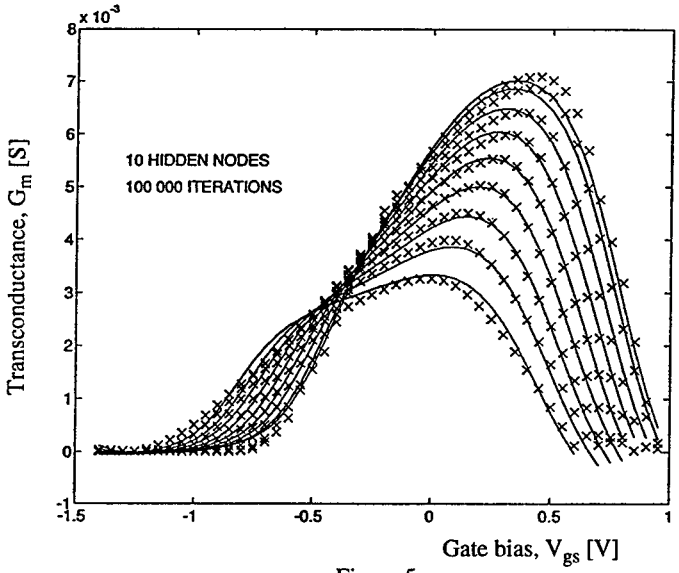


Figure 5:
GaAs MESFET transconductance modelling wrt gate bias and temperature. Solid lines as measured data, 'x': data recalled by network.

We have also implemented the perceptron network in SPICE circuit simulator, and tested the operation in dc- and transient analysis. There is no convergence problems in simulating the *NeuroDevice* since the modelling functions are smooth and continuous. In our example, the perceptron network device modelling interface for

one device size of Fig. 2 requires total of 21 network weights for two dimensional data and two bias units. If the third input dimension is included in network with five hidden layer elements, total of 26 weights are required. For more than five processing elements in the hidden layer, the number of weights will be higher. These weights will have to be stored in memory for each presentation of data. The amount of stored weights presents a remarkable saving of memory when it is compared to the table look-up method. About 500-1000 datapoints will have to be saved for each channel length for the table to attain the same accuracy. Therefore, we use only 3-5 per cent of the memory that is used in a typical implementation of the table look-up method.

In terms of the CPU-time the implemented neural network interface can be more efficient for circuit simulation than analytical models if only one device size and one stage of network is used. The device modelling task is required to be performed only once and circuit simulation is performed sequentially with fast recollection of the model from the stored weight-vector. Therefore, the implemented neural network device modelling interface is also capable of reducing the time-consumption of circuit simulations.

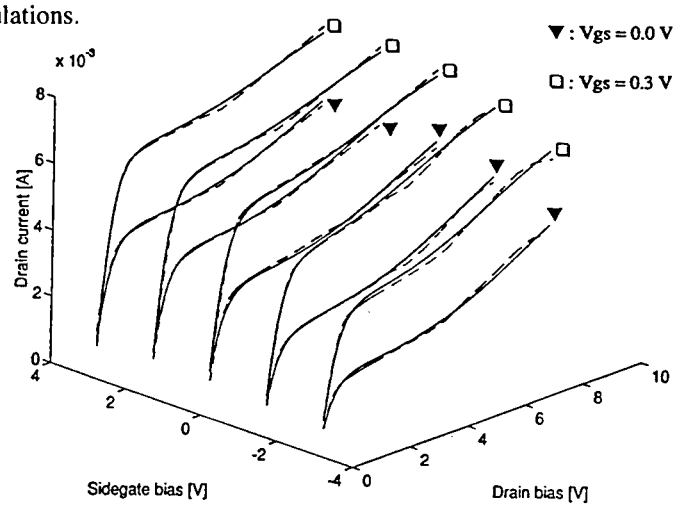


Figure 6:
3D drain current modelling as a function of drain, gate and substrate (sidegate) bias for GaAs MESFET with a feature size of W/L = 50μm/ 2μm. Solid lines as measured datapoints, dashed lines as data recalled by network for gate bias values 0.0 and 0.3 V.

CONCLUSION

The method of realizing the interface between device modelling and circuit simulation using neural network algorithms has been shown to produce excellent fit to the measured data. The objective of presenting a general device characteristics in circuit simulator environment can be reached. Any kind of circuit element can be ac-

curately modelled and represented, and a standard automatable neural network can be set up for the construction of these representations.

The implemented algorithms combine a fast learning rate with efficient and accurate recall of the learnt material. The method is especially suitable for applications where physically justified analytic device models lack the required accuracy. These include deep submicrometer devices and novel device structures with as of yet unclear physical phenomena. Also, the technology independent approach for the modelling facilitates quick adjustment to the new device structures, materials and technologies.

The macromodelling of complex circuit structures with easy neural network parameter presentation vastly simplifies the required simulations for large systems. It reduces efficiently the required memory for the circuit presentation and simulation. The proposed *NeuroDevice* facilitates and encourages the user to model complex topologies. In simulation it promotes the inclusion of desired behavioural information of general phenomena with easy to extract neural network weights, which are a fully compact and ideal form to present, save and transfer knowledge.

REFERENCES

[1] B. J. Sheu, D. L. Scharfetter, P.-K. Ko, and M.-C. Jeng, "BSIM: Berkeley Short Channel IGFET Model for MOS-Transistors," *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 4, Aug. 1983, pp. 558-566.

[2] Y. P. Tsividis and G. Masetti, "Problems in Precision Modeling of the MOS Transistor for Analog Applications," *IEEE Transactions on Computer-Aided Design*, vol. CAD-3, no. 1, Jan. 1984, pp. 72-79.

[3] A. Rofougaran and A. A. Abidi, "A Table Lookup FET Model for Accurate Analog Circuit Simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-12, no. 2, Feb. 1993, pp. 324-335.

[4] T. Shima, H. Yamada, and R. L. M. Dang, "Table Look-up MOSFET Modeling System Using a 2-D Device Simulator and Monotonic Piecewise Cubic Interpolation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-2, no. 2, Apr. 1983, pp. 121-126.

[5] G. Bischoff and J. P. Krusius, "Technology Independent Device Modeling for Simulation of Integrated Circuits for FET Technologies," *IEEE Transactions on Computer-Aided Design*, vol. CAD-4, no. 1, Jan. 1985, pp. 99-110.

[6] K. Doganis and D. E. Scharfetter, "General Optimization and Extraction of IC Device Model Parameters," *IEEE Transactions on Electron Devices*, vol. ED-30, no. 9, Sep. 1983, pp. 1219-1229.

[7] P. Ojala, K. Kankaala, H. Tenhunen, and K. Kaski, "Advanced Techniques for Circuit Parameter Extraction", Report 5-88, Tampere University of Technology, Department of Electrical Engineering, Laboratory of Microelectronics, July 1988.

[8] P. O. A. Yang and P. Chatterjee, "An Optimal Parameter Extraction Program for MOSFET Models," *IEEE Transactions on Electron Devices*, vol. ED-30, no. 9, Sep. 1983, pp. 1214-1219.

[9] C. Cybenko, "Approximations by Superpositions of a Sigmoidal Function," *Math. Contr., Signal, Syst.*, vol. 2, 1989, pp. 303-314.

[10] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the Convergence of the Back-propagation Method," *Biological Cybernetics*, 59, 1988, pp. 257-263.

[11] W. H. Press, B. P. Flannering, S. A. Teukolsky and W. T. Wetterling, Numerical Recipes in C, New York: Cambridge University Press, 1988.

[12] F. S. Shoucair and P. K. Ojala, "High-Temperature Electrical Characteristics of GaAs MESFET's (25-400°C)," *IEEE Transactions on Electron Devices*, vol. 39, no. 7, Jul. 1992, pp. 1551-1557.

[13] N. Scheinberg and E. Chisholm, "A Capacitance model for GaAs MESFET," *IEEE Journal of Solid-State Circuits*, vol. SC-26, no. 10, Oct. 1991, pp. 1467-1470.

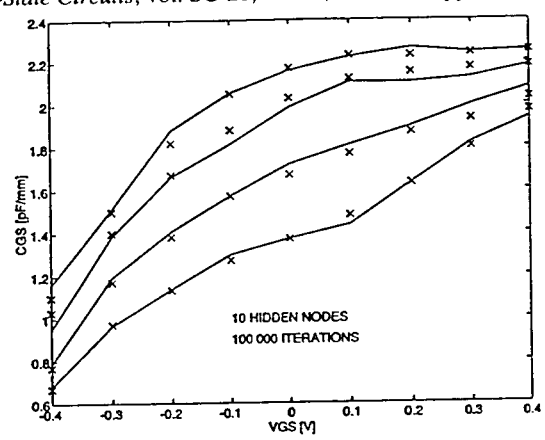


Figure 7:
GaAs MESFET gate to source capacitance wrt gate to source bias with neural network. Solid lines as measured data [13], 'x': data recalled by network.

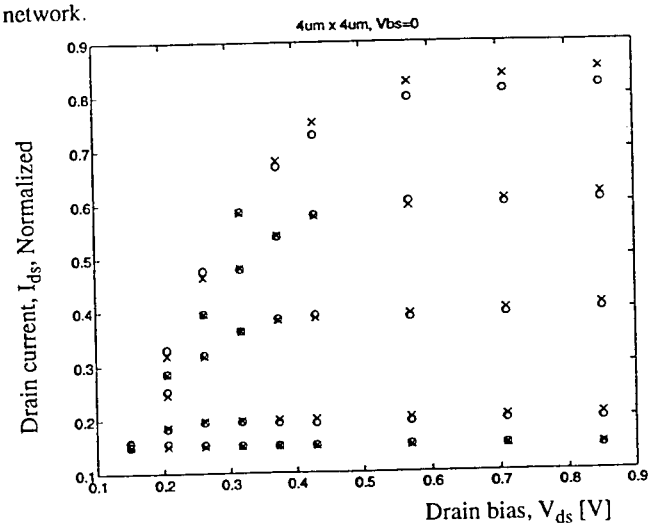


Figure 8:
Neural network device modelling with device geometry dependent input vector. Recalling of device data W/L=4μm/4μm, with no body bias.

ENCODING PYRAMIDS BY LABELING RAAM

Stefano Lonardi Alessandro Sperduti Antonina Starita
Dipartimento di Informatica
Università di Pisa
Corso Italia 40, 56125 Pisa, Italy
e-mail: perso@di.unipi.it

Abstract

In this paper we present preliminary results on the application of Labeling RAAM for encoding pyramids. The LRAAM is a neural network able to develop reduced representations of labeled directed graphs. The capability of such representations to preserve structural similarities is demonstrated on a pyramid. We suggest to exploit this skill in data compression and/or to discover scale affine autosimilarities.

INTRODUCTION

A model frequently used in image analysis is the *quadtree*, a hierarchical data structure based on the principle of recursive decomposition of space [10]. Different instances of quadtree can be obtained depending on the type of data represented, the decomposition principle, and the resolution (variable or not). In this paper, we discuss how a new type of neural network, the Labeling RAAM, seems specially suited to code it. Specifically, we consider a complete quadtree implementing a *pyramid*, a data structure used to represent a multiresolution version of an image using nonoverlapping 2 by 2 blocks of pixels (see Fig. 1). The aim is twofold: to get a compressed representation and/or to discover scale affine redundancy in the image represented by the pyramid. Firstly, we introduce the LRAAM model. Then, preliminary results obtained on pyramids are presented and discussed.

LABELING RAAM

The *Labeling RAAM (LRAAM)* [12, 15, 16, 17] is an extension of the RAAM model [9] which allows one to encode labeled structures. The general structure of the network for an LRAAM is shown in Figure 2.

0-7803-2026-3/94 $4.00 © 1994 IEEE

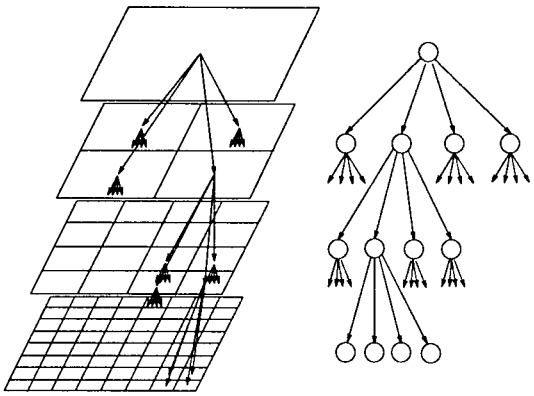


Figure 1: A pyramid implemented by a quadtree.

The network is trained by backpropagation to learn the identity function. The idea is to obtain a compressed representation (hidden layer activation) of a node of a labeled directed graph by allocating a part of the input (output) of the network to represent the label (N_L units) and the rest to represent one or more pointers. This representation is then used as pointer to the node. To allow the recursive use of these compressed representations, the part of the input (output) layer which represents a pointer must be of the same dimension as the hidden layer (N_H units). Thus, a general LRAAM is implemented by a $N_I - N_H - N_I$ feed-forward network, where $N_I = N_L + nN_H$, and n is the number of pointer fields.

Labeled directed graphs can be easily encoded using an LRAAM. Each node of the graph only needs to be represented as a record, with one field for the label and one field for each pointer to a connected node. The pointers only need to be logical pointers, since their actual values will be the patterns of hidden activation of the network. At the beginning of learning, their values are set at random. A graph is represented by a list of these records, and

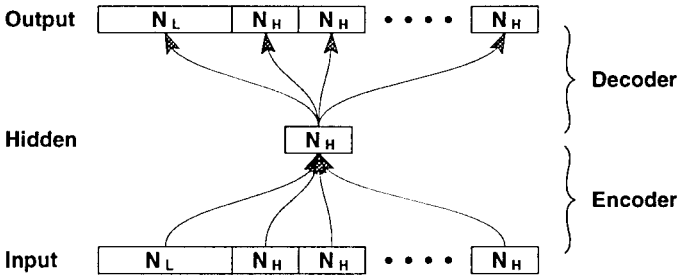


Figure 2: The network for a general Labeling RAAM.

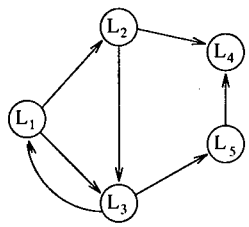


Figure 3: An example of a labeled directed graph.

this list constitutes the initial training set for the LRAAM. During training the representations of the pointers are consistently updated according to the hidden activations. Consequently, the training set is dynamic. For example, the network for the graph shown in figure 3 can be trained as follows:

input		hidden		output
$(L_1\ P_{n_2}(t)\ P_{n_3}(t))$	$\rightarrow$	$P'_{n_1}(t)$	$\rightarrow$	$(L''_1(t)\ P''_{n_2}(t)\ P''_{n_3}(t))$
$(L_2\ P_{n_3}(t)\ P_{n_4}(t))$	$\rightarrow$	$P'_{n_2}(t)$	$\rightarrow$	$(L''_2(t)\ P''_{n_3}(t)\ P''_{n_4}(t))$
$(L_3\ P_{n_1}(t)\ P_{n_5}(t))$	$\rightarrow$	$P'_{n_3}(t)$	$\rightarrow$	$(L''_3(t)\ P''_{n_1}(t)\ P''_{n_5}(t))$
$(L_4\ nil_1(t)\ nil_2(t))$	$\rightarrow$	$P'_{n_4}(t)$	$\rightarrow$	$(L''_4(t)\ nil''_1(t)\ nil''_2(t))$
$(L_5\ P_{n_4}(t)\ nil_3(t))$	$\rightarrow$	$P'_{n_5}(t)$	$\rightarrow$	$(L''_5(t)\ P''_{n_4}(t)\ nil''_3(t))$

where L_i and P_{ni} are the label of and the pointer to the ith node, respectively, and t represents the time, or epoch, of training. At the beginning of training ($t = 1$) the representations for the non-void pointers ($P_{n_i}(1)$) and void pointers ($nil_i(1)$) in the training set are set at random. After each epoch, the representations for the non-void pointers in the training set are updated depending on the hidden activation obtained in the previous epoch for each pattern: $\forall i\ P_{n_i}(t+1) = P'_{n_i}(t)$. The void representations are, on the other hand, copied from the output: $nil_i(t+1) = nil'_i(t)$.

If the backpropagation algorithm converges to zero error, it can be stated that:

$$\begin{array}{cccc} L_1 = L''_1 & L_2 = L''_2 & L_3 = L''_3 & L_4 = L''_4 \\ L_5 = L''_5 & P_{n_2} = P''_{n_2} & P_{n_3} = P''_{n_3} & P_{n_4} = P''_{n_4} \\ P_{n_5} = P''_{n_5} & nil_1 = nil''_1 & nil_2 = nil''_2 & nil_3 = nil''_3 \end{array}$$

Once the training is complete, the patterns of activation representing pointers can be used to retrieve information. Thus, for example, if the activity of the hidden units of the network is clamped to P_{n1}, the output of the network becomes (L_1,P_{n2},P_{n3}), enabling further retrieval of information by decoding P_{n2} or P_{n3}, and so on. In order to decide whether a pointer is void or not, one bit of the label is allocated for each pointer field to represent the void condition. This convention allows us to avoid a commitment to any predefined representation for the void pointer. Consequently, copying

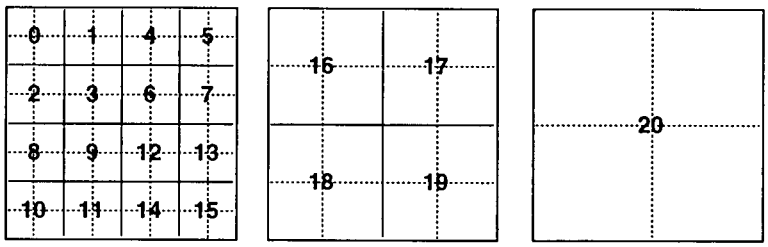


Figure 4: An example of label encoding of a pyramid for an 8×8 image.

the representations for the void pointers from the output of the network to the training set results in a faster training, where multiple representations for the void pointer are developed by the network itself. For more details on this issue the reader is reffered to [15]. Note that multiple labeled directed graphs can be encoded in the same LRAAM.

ENCODING PYRAMIDS

Since a pyramid can be represented as a labeled tree, it can be easily encoded by an LRAAM. The aim in using an LRAAM is to obtain a compact representation of the pyramid where the same pattern at different scales is uniquely represented, i.e., without affine redundancy [1]. The definition of the label for a pattern in the training set of the LRAAM can proceed from the leaf level to the root of the pyramid (see Fig. 4). One drawback of this approach is that the number of patterns in the training set grows exponentially in the dimension of the image. Given a $2^n \times 2^n$ image, if the scale factor is $1/2$, the total number of patterns in the training set is $\sum_{i=0}^{n-1} 4^i$. Thus, it is clear that images of large dimensions are difficult to handle with a single LRAAM. For example, the training set for a 256×256 image would be composed by slightly less than 22,000 patterns. One possible solution to this problem, that we have to verify, is the use of a modular LRAAM, where every module is responsible for the encoding of a given subtree. Another solution is to start learning with a training set representing the lower resolution levels of the pyramid and, after convergence, augmenting the training set to the complete one.

Once the training set has been generated, the LRAAM is trained until it can decode successfully the pyramid. We have observed that, with this stopping criterion, and under the condition of affine redundancy in the image, learning converges rather early. For example, using the encoding scheme shown in figure 5 on an 8×8 version of the Sierpinski triangle, the mean number of epochs employed by the corresponding $12 - 2 - 12$ LRAAM was slightly more than 100.

We verified faster learning using the *descending-epsilon* heuristic technique [18]: during the learning phase we maintain a list of the patterns hav-

ing a decoding error higher than a specified value. The backpropagation is performed only on the patterns of the list: when all patterns are below the threshold, we lower the threshold and resume the backpropagation. The procedure stops when we obtain the perfect decoding.

Analysis

An important property of the LRAAM model is *the capability to develop similar hidden representations for pointers to similar labeled trees.* An example of this capability is given by the hidden activation deviced by the $12-2-12$ LRAAM encoding the Sierpinski triangle. In figure 6 we show, on the left side, the label map obtained by decoding a set of points sampled from the pointer space, and on the right side, the vector fields obtained by transforming the same set of points through the children transformations. A vector field is represented by plotting the sampled points (domain points) and their transformed results (image points) as vectors starting from the domain (dots) and arriving to the image points. Note how the network exploits the same pointer transformation for pointer fields encoding the same set of subtrees. This allows the LRAAM *to decode correctly this image at a resolution higher than the one used in the training set.* This property, however, must be verified on images which are not so regular.

The pointers' dynamics is the subject of figure 7: in this representation the application of the same pointer transformation is repeated until convergence to a fixed point. The gray scale denotes the number of steps to reach the fixed point where darker areas mean a higher number of iterations. Note that the network places the fixed points in the vertices of the $[-1,1]^{N_H}$ space. This property can be explained by probabilistic arguments concerning the stability of the decoding (for a discussion on stability properties of the LRAAM see [16]).

Another example of pyramid (Fig. 8) with the corresponding label map, children vector fields and pointers' dynamics maps are shown in figures 9 and 10.

From our analysis of the decoding of the label and the pointers' dynamics, the fractal approximation developed by the LRAAM seems to have a close relationship with the *hierarchical iterated function system* model [3, 4], where a graph of hierarchical IFS generates the image. In our case, nodes of the graph represent fixed points of the pointers' dynamics. Each node is labeled by the label obtained by decoding the corresponding fixed point. Arcs of the graph represent pointer transformations of the fixed points (see Figure 11).

It must be stressed that the construction of these graphs can be done, in this case, only because any fixed point is actually transformed in another one (or in itself) in just one transformational step. In general, however, this may not be the case. We are currently investigating under which conditions the graph construction scheme holds.

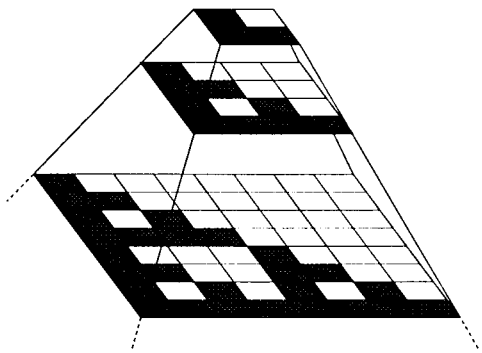


Figure 5: The pyramid coding the Sierpinski triangle.

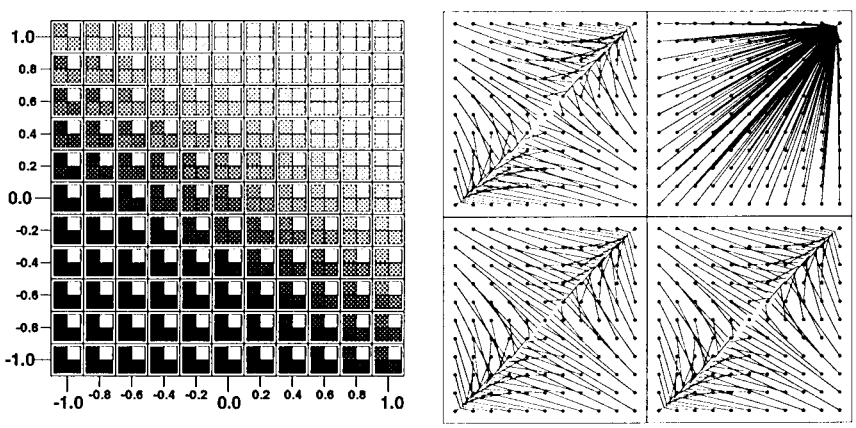


Figure 6: Representations devised by an LRAAM encoding an 8×8 Sierpinski triangle; **left:** label map on the pointer space; **right:** vector fields for the children transformations.

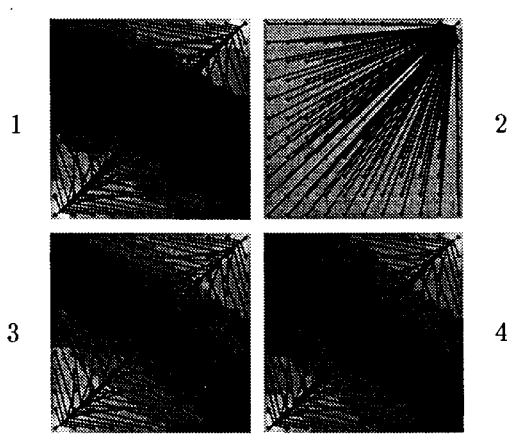


Figure 7: The pointers' dynamics maps for the Sierpinski triangle.

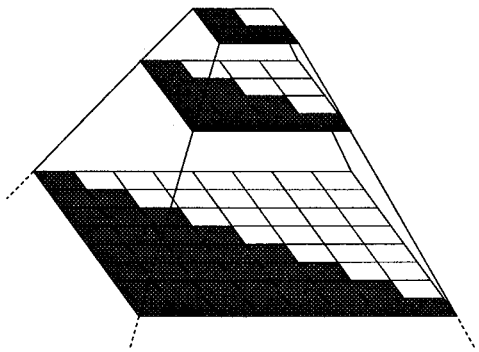


Figure 8: The pyramid coding the simple triangle.

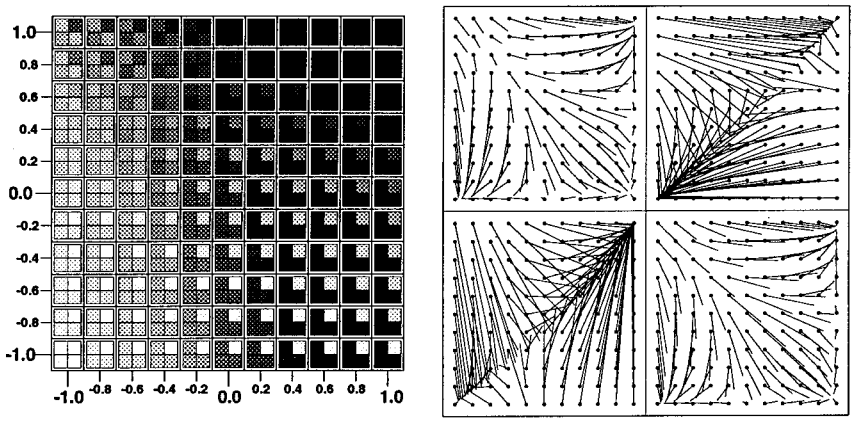


Figure 9: Representations devised by an LRAAM encoding an 8×8 triangle. **left:** label map on the pointer space; **right:** vector fields for the children transformations.

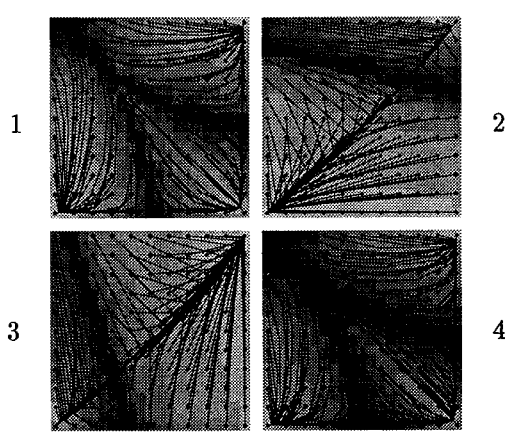


Figure 10: The pointers' dynamics maps for the simple triangle.

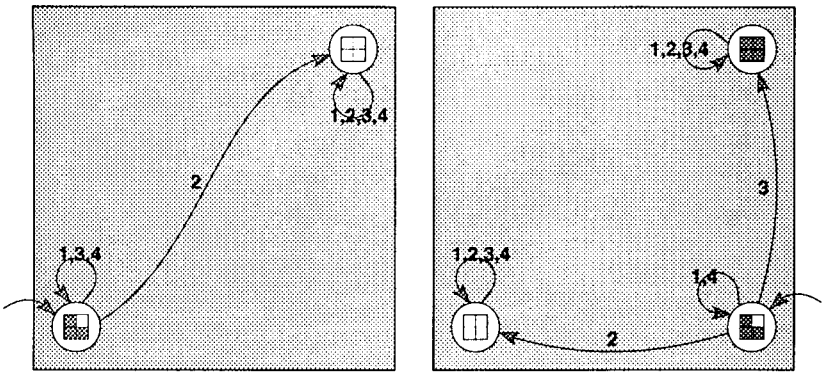


Figure 11: The graphs, derived by the LRAAM, generating the Sierpinski triangle pyramid (left) and the triangle pyramid (right). Each node represents a fixed point in the pointers' dynamics and its label is the label obtained by decoding the fixed point. Arcs represent pointer transformations. Node x is connected to node y by an arc labeled i if $T_i(x) = y$, i.e., if the fixed point associated to x is transformed to y by the pointer transformation $T_i()$.

Applications

There are two possible applications of encoding pyramids by Labeling RAAM:

- *data compression*; a set of pyramids can be described by the set of pointers to the roots plus the decoder part of the LRAAM;
- *affine redundancy discovery*; the likeness among pointers can be used to establish similarities among patterns at different scales, so to device an efficient fractal compression.

Using an LRAAM for data compression requires a careful choice of the number of units in the hidden layer. More units in the hidden layer guarantee a better reconstruction of the image represented by the pyramid, but the compression factor decrease because the number of parameters (weights) grows as a quadratic function. Specifically, given pointers of dimension N_H and label of dimension N_L, we have

$$4N_H^2 + (N_L + 5)N_H + N_L$$

parameters. In fact, the dimension of the hidden layer affects the number of different labels that the LRAAM can encode. Having an insufficient number of hidden units constrains the network to find the minimal approximation with the most used label present in the image at every scale, which is the simplest fractal approximation allowed by the pyramidal representation. We are still working on the evaluation of the best trade-off between quality of

the image and data compression. Note that *pruning* techniques like OBD [7], and OBS [6] can be used to reduce the number of weights in the network.

Classification of fractal images by a neural network was explored in the paper [5]. The feed-forward network used in that work, where the pyramid is explicitly represented into the topology of the network, can be considered as a special case of our model. In our case, however, we are not interested in classification, but in discovering scale affine autosimilarity. The ability of the LRAAM to represent similar patterns by similar pointers can be exploited to identify a fractal approximation of the image [1, 2, 11, 8].

CONCLUSIONS

The LRAAM model seems ideal to code a pyramid representing an image with scale affine redundancy. The hidden representations developed by the LRAAM seem to capture redundancies present in the image at different scales. The LRAAM can be used either to compress the pyramid or to discover autosimilarities which can be exploited by a fractal compression method. The compression rate can also be improved by coding the image into an incomplete pyramid using a decomposition principle based on statistical regularities. Incomplete pyramids can be encoded without problems by an LRAAM which can represent every kind of tree.

Besides the present line of research, it is likely that all areas utilizing quadtrees for different applications could use the interesting characteristics of the subsymbolic coding developed by the LRAAM.

Acknowledgments

The authors wish to thank Ahmad Zandi for comments and suggestions.

References

[1] M. F. Barnsley, Fractals Everywhere, Academic Press, 1988.

[2] M. F. Barnsley, L. P. Hurd, Fractal Image Compression, AK Peters Ltd, 1993.

[3] K. Culik, S. Dube, "Rational and Affine Expressions for Image Description", Discrete Applied Mathematics, 41, pp. 85–120, 1993.

[4] K. Culik, S. Dube, "Balancing Order and Chaos in Image Generation", Computer & Graphics, 17(4), pp.465–486, 1993.

[5] B. Freisleben, J. H. Greve, J. Lober, "Recognition of Fractals Images Using a Neural Network", Proceedings of Int. Workshop on Artificial Neural Networks, Lecture notes in Computer Science 686, pp.632–637, Springer Verlag, 1993.

[6] B. Hassibi, D.G. Stork, "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon", Advances in Neural Information Processing Systems, San Mateo: Morgan Kaufmann, Vol. 5, pp.164–171, 1993.

[7] Y. Le Cun, J.S. Denker, S.A. Solla, "Optimal Brain Surgeon", Advances in Neural Information Processing Systems, San Mateo: Morgan Kaufmann, Vol. 2, pp.598–605, 1990.

[8] S. Lonardi, Analisi e sintesi frattale di immagini, Thesis, Computer Science Department, University of Pisa, Italy, 1994.

[9] J. B. Pollack, "Recursive distributed representations", Artificial Intelligence, 46(1-2), pp.77–106, 1990.

[10] H. Samet, "The Quad-Tree and Related Hierarchical Data Structures", ACM Computing Surveys, 16(2), pp.188–260, 1984.

[11] P. Sommaruga, S. Lonardi, "Best Fractal Orthogonal Approximation and Block Coding of Images", submitted to Signal Processing.

[12] A. Sperduti, A. Starita, "An Example of Neural Code: Neural Trees Implemented by LRAAMs", Intl. Conf. on Neural Networks and Genetic Algorithms, Innsbruck 1993, pp.33–39.

[13] A. Sperduti, A. Starita, "Modular Neural Codes Implementing Neural Trees", to appear in 6th Italian Workshop on Parallel Architectures and Neural Networks, 1993.

[14] A. Sperduti, Optimization and Functional Reduced Descriptors in Neural Networks, PhD Thesis, Computer Science Department, University of Pisa, Italy, TD-22/93, 1993.

[15] A. Sperduti, "Labeling RAAM", Technical Report 93-029, International Computer Science Institute, 1993. Accepted for pubblication on *Connection Science.*

[16] A. Sperduti, "On Some Stability Properties of the LRAAM Model", Technical Report 93-031, International Computer Science Institute, 1993.

[17] A. Sperduti, "Encoding of Labeled Graphs by Labeling RAAM", to appear in Advances in Neural Information Processing Systems, San Mateo: Morgan Kaufmann,Vol. 6, 1994.

[18] Y. Yu, R. Simmons, "Descending Epsilon in Backpropagation: a Technique for Better Generalization", Proceedings of IJCNN , San Diego 1990, pp.167–172.

Reconstructed Dynamics and Chaotic Signal Modeling

Jyh-Ming Kuo and Jose C. Principe

Computational NeuoEngineering Lab.
University of Florida, Gainesville, FL 32611
kuo@synapse.ee.ufl.edu

ABSTRACT: A nonlinear AR model is derived from the reconstructed dynamics of a signal. The underlying system is assumed to be nonlinear, autonomous, and deterministic. In this formulation, the output error scheme is shown to be more suitable than the equation error scheme in training a network as a NAR model of the signal. A method to incorporate the information of the dynamical invariants in signal modeling is proposed.

1. INTRODUCTION

Signal modeling in either time domain or frequency domain has a long history in the research of science and engineering. But, not until recently has this problem been studied based upon the underlying dynamics of signals [Crutchfield][Casdagli]. Under the assumption that the original system is nonlinear, deterministic, and autonomous, a state-space system trajectory can be reconstructed from its output signal [Packard][Takens]. Thanks to this reconstruction, the measurement of two important dynamical properties, namely dimensionality and Lyapunov exponents of a system attractor, becomes possible. The "uncertainties" observed in the signals of this class are originated by the system dynamics rather than some external random sources. A dynamical system with this unpredictable characteristics has been referred to as a chaotic system [Eckmann].

According to the assumption of the underlying system, the reconstruction suggests a nonlinear autoregressive (NAR) model of a signal [Casdagli][Kuo]. This NAR model can also be used to describe the underlying dynamic of the signal. In this approach, a nonlinear modeling tool is required. Through training, multilayer perceptrons (MLPs) can become an accurate nonlinear mapper [Hornik][Hecht-Nielsen]. In this work we train MLPs as NAR models of some test signals, which were produced by nonlinear, deterministic systems. We compare the results of two adaptation schemes in AR modeling, namely equation error and output error schemes [Shynk]. If the length of training sequences can be properly chosen, the output error scheme is shown to be more suitable in modeling signal dynamics than equation error scheme which has been adopted by most research groups [Lapedes][Casdagli][Mead]. In this paper, we propose a method to determine the length of training sequences based upon the dynamical measures of the reconstruction.

In model validation, we show that using the mean squared one-step-

0-7803-2026-3/94 $4.00 © 1994 IEEE

prediction error as the criterion, although of practical importance, sometimes is very misleading about how well the model approximates the underlying dynamics of a signal. We propose that the validation should include the comparison of the dynamical measures of the model output and the original signal [Principe][Kuo][Crutchfield].

2. RECONSTRUCTED DYNAMICS

In the study of experimental dynamics, the delay coordinate method was proposed to reconstruct a state-space trajectory from a signal [Packard][Takens]. Under certain conditions, the reconstruction is topologically equivalent to the state-space trajectory of the system that produced the signal. According to this method, the elements of a reconstructed state vector, $\vec{y}(t)$, are signal samples and delayed versions, or

$$\vec{y}(t) = \left[x(t) \;\; x(t+\tau) \;\; \ldots \;\; x(t+2m\tau)\right]^T \qquad \text{eq. 1}$$

where x(t) is the data sample at time t, τ is a constant delay, and 2m+1 is the dimension of the construction space. Takens showed that the sufficient condition for the equivalence of the reconstructed and the original system trajectories is $2m+1 \geq 2d+1$, where d is the dimension of the original system attractor [Takens]. For most real-life problems, the dimensions of the underlying dynamics are usually unknown. Therefore, we need a method to estimate the minimal reconstruction dimension. Later, we will show that the estimation of dimensionality can be related to the order estimation of the signal model.

By topological equivalence, we mean that two state-space trajectories have the same measures of dimension and Lyapunov exponents. The dimension of an attractor indicates the degrees of freedom of the system, while the Lyapunov exponents measure the growing or destroying rate of information in system dynamics [Eckmann][Parker]. To include the fractal case for chaotic dynamics, a generalized definition of dimensionality is required. In the paper, we adopt the correlation dimension because of its mediate requirement of computation time and memory size [Grassberger]. To estimate the correlation dimension of a state-space trajectory, we first compute the correlation integral, C(r),

$$C(\varepsilon) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} H(\varepsilon - \|\vec{y}(i) - \vec{y}(j)\|) \qquad \text{eq. 2}$$

where N is the total number of the reconstruction state vectors, H(r) is the Heaviside function (=1 for $r \geq 0$, and = 0 for $r < 0$), and ε is the radius. For small ε's, $C(\varepsilon)$ is proportional to ε^D. Since the dimension of the underlying attractor is unknown, a common approach is to reconstruct state-space trajectories from a

signal in spaces of different dimensions and plot the curves of log C(ε) v.s. log ε, which is referred to as correlation integral map or CIM for short [Principe]. When we increase the dimension of the reconstruction space to a certain value we may observe the saturation in the slopes of the CIM curves. The saturation value is the estimate of the correlation dimension.

A positive (negative) Lyapunov exponent is the measurement of exponentially divergent (convergent) rate between two nearby trajectories along a specific direction. For instance, we start trajectories from an initial state and points on a sphere surrounding this state. The sphere may be deformed into an ellipsoid along the flow of a dynamical system. We can compute the ith Lyapunov exponent, λ_i, as

$$\lambda_i = \lim_{t \to \infty} \frac{1}{t - t_0} \log_2 \frac{d_i(t)}{d(t_0)} \qquad \text{eq. 3}$$

where $d(t_0)$ is the radius of the sphere, and $d_i(t)$ is the ith principal axis of the ellipsoid as time evolves from t_0 to t. In signal modeling, we are more concerned about how fast the model output will diverge from the original signal. In other words, positive Lyapunov exponents are of more importance. Because the largest Lyapunov exponent will eventually dominate the divergence [Sano], we propose to consider its measurement only. An algorithm proposed by Wolf will be used to estimate the largest positive Lyapunov exponent [Wolf]. According to this algorithm, two spatially neighboring segments of a reconstructed trajectory are treated as two nearby trajectories to compute the exponent. And, we compute the average of the right-hand-side expression in eq. 3 instead of taking the limit.

3. SIGNAL MODELING

We assume the underlying dynamics of a signal are smooth. Consequently, there exists a function $F: R^{2m+1} \to R^{2m+1}$, which maps the current reconstructed vector to the next one, i.e.

$$\vec{y}(t+1) = F(\vec{y}(t)) \qquad \text{eq. 4}$$

$$\begin{bmatrix} x(t+1) \\ x(t+2) \\ \cdots \\ x(t+1+2m) \end{bmatrix} = F\left(\begin{bmatrix} x(t) \\ x(t+1) \\ \cdots \\ x(t+2m) \end{bmatrix} \right)$$

Here, we use delay coordinate reconstruction method with $\tau = 1$. We note that the mapping actually includes several trivial filters and a predictor. Let us denote the predictive mapping as $F^{\perp}: \mathrm{R}^{2m+1} \to R$, which can be expressed by

[Kuo][Casdagli]

$$x(t+1+2m) = F^{\perp}(x(t), x(t+1), \ldots, x(t+2m)) \quad \text{eq. 5}$$

$$\text{or} \qquad x(t+1) = F^{\perp}(\vec{x}(t))$$

where $\vec{x}(t) = \left[x(t-2m) \ \ldots \ x(t-1) \ x(t)\right]^T$. Since $F^{\perp}$ is assumed to be a nonlinear function, eq. 5 represents a NAR model of the signal x(t). *From this point of view, signal modeling is equivalent to the modeling of the underlying dynamics.* In conventional AR modeling, we construct a model that, in the average, fits only the local behaviors best. On the other hand, the dynamical measures provide global information of the signal dynamics. If we are able to incorporate these measurements into the model design, a better performance can be expected.

The order of an AR model is defined as the number of the past data samples we use to predict the current data sample. In eq. 5, we note that the model order is equal to the dimension of the reconstruction space. Once we compute the correlation dimension of the underlying dynamics, Takens theorem already gives us a lower-bound estimate of the model order. In other words, the reconstruction dimension or the order of the AR model, 2m+1, must be equal to or greater than 2D + 1. However, for experimental data we usually find two problems with this estimate. First, Takens theorem usually overestimates the minimal order of the model. Second, this estimate is sensitive to the selection of the sampling rate [Mead]. We have proposed to use CIM curves to estimate the lower bound of the model order [Kuo]. In the computation of the correlation dimension, we plot the CIM curves in reconstruction spaces of different dimensions. When we increase the dimension of the reconstruction space to a certain value, we can observe the saturation of the slopes of the CIM curves. This gives us a reconstruction space of the minimal dimension in which a topological equivalence of the original system dynamics can be established. This estimate of the minimal reconstruction dimension is also the estimate of the minimal model order. Our experimental results show that the minimal order estimate derived from this method is consistent with the minimal input temporal window (= sampling rate x model order) concluded in the work of Mead et al.[Mead].

In this paper, we would like to concentrate on how to apply the information of another dynamical measurement, the largest positive Lyapunov exponent, to signal modeling. Before we do that, we will discuss two possible schemes in training an MLP as a NAR model.

4. NETWORK TRAINING SCHEMES

For chaotic signal modeling, a common approach is as follows: a memory device, e.g. a delay line, is used to retain the data samples in the past. The parameters of the model are determined either analytically or adaptively to minimize the error between the model output and the current data sample. This approach is referred to as the equation error scheme in adaptive IIR filtering

[Shynk]. It is well known that this scheme may introduce bias to the solution. This is because after adaptation we will connect the output back to the input of the memory device instead of clocking in the data samples of the original signal to make the next prediction. In this work, we choose the other adaptation scheme --- output error scheme. A schematic diagram of this training is shown in Figure 1. Here, the input layer of the MLP is replaced with a delay line. This structure is referred to as Time-Delay Neural Network (TDNN) [Mozer]. We note that the network is trained as well as operated in the same manner. However, this training scheme requires a trajectory learning algorithm.

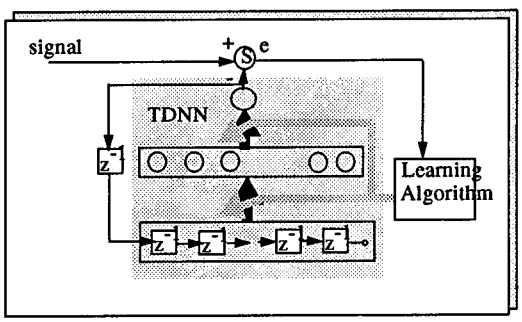


Figure 1. Output error training scheme

In a trajectory learning, we need to decide the length of training sequences. On one hand, training sequences should contain enough information about the global picture of an attractor to be modeled. On the other hand, the computation complexity of a trajectory learning algorithm usually increases dramatically with the length of training sequences. If the signal is periodic, this decision is quite straightforward. The number of data samples that covers one cycle is usually the best choice. However, if the signal is chaotic, this decision becomes very involved.

5. LENGTH OF TRAINING SEQUENCES

In modeling chaotic signals, a long training sequence not only increases computation time but also induces stability problem. Since the underling system of a chaotic signal possesses positive Lyapunov exponents, two nearby trajectories of the system will diverge eventually. In the time domain, this implies that system output signals with small deviation initially will diverge eventually. Assume the model already captures the underlying dynamics. A tiny modeling error will be magnified by the system dynamics such that the output of the model will diverge from the desired sequence in a long run. If we try to reduce this error by changing the network parameters, we will not improve modeling accuracy. Instead, we may deteriorate the modeling result and create oscillations in the adaptation. To present enough global information of an attractor to the network during training and avoid using a long training sequence, we propose to train the network with different segments of the signal. The training procedure is as follows: clock an initial condition into the network input layer, iterate the network to produce a data segment, and then adjust the network parameters to minimize the distance between

the iterates and the corresponding target sequence. We apply the same procedure to all of training sequences once in each training epoch.

The problem becomes how to choose the length of training sequences such that we can avoid the oscillation problem during training. To analyze this problem, let us assume we train a network with two data segments whose initial conditions are very close. They can be considered as two different segments of a reconstructed trajectory, say segment A and B, starting from the same neighborhood. When the network starts capturing the dynamics through training, the iterates of the model corresponding to these two training sequences can be used to reconstruct two trajectories around segment A and segment B respectively. This is illustrated in Figure 2. Instead of specifying the trajectories reconstructed from the outputs of the model, two uncertainty regions are delimited. They represent the possible divergence range between the model output and the signal due to the existence of the positive Lyapunov exponents. As time evolves, both uncertainty regions will grow and eventually overlap. When this overlap occurs, the training may become unstable. This is due to the fact that if the output of the model falls into the overlap region the model may be required to develop a map to follow the evolution of both segment A and segment B during training. Since segment A and segment B will diverge from each other eventually, it is not possible for the model to meet this conflicting requirement.

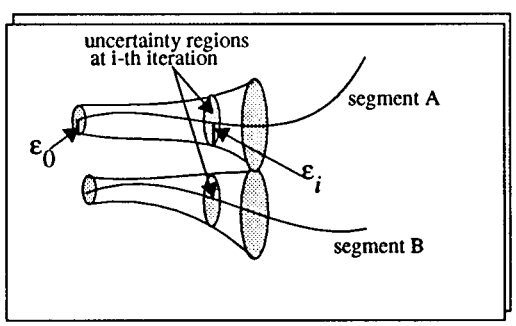


Figure 2. State space representation in training a model with two sequences whose initial conditions are close.

We assume the divergence occurs mainly along the direction corresponding to the largest Lyapunov exponent. The smallest number of iterations, i_s, before the overlap happens can be calculated according to the following inequality

$$\varsigma_{i_s} \leq 2\varepsilon_{i_s} = 2\varepsilon_0 e^{\lambda_{max} i_s \Delta t} \quad \text{eq. 6}$$

where ς_i is the distance between the i_sth points on both training segments, ε_i is the estimate of the largest principal axis after i_s iterations, ε_0 is the mean square root of one-step prediction errors, and Δt is the sampling period. This estimation is based on the assumption that the largest principal axes of both uncertainty regions are in the same line but in opposite directions. To avoid training the model with a conflicting

requirement, both training sequences should be smaller than i_s. Therefore, we propose to use the average of i_s's computed from all pairs of neighboring training sequences as the length of the training sequences.

6. EXPERIMENTAL RESULTS

The test signal was obtained by integrating the Mackey-Glass equation [Mackey](with a = 0.2, b = 0.1, c = 10, and D = 30) with 4-th order Runge-Kutta method at a step size of 1. Then, the signal was downsampled by 6 and normalized to the range of [-1,1] for training. The resulting signal is given in Figure 3. This signal will be referred to as mg30.

First, we train a TDNN with 8-14-1 (8 input units, 14 nonlinear hidden units, and 1 linear output unit) architecture using equation error scheme. In other words, the network is trained to be a one-step predictor of the mg30 signal. The training signal has 500 data samples. The learning rate is set to 0.001. The training was stopped after 500 epochs. We will refer to the network with the resulting weights as TDNN#1. The learning curve is given in Figure 4. We note that the change of the mean square error can almost be ignored after first 200 epochs. The final mean square error is 2.88×10^{-4}. After training, we clock in the first 8 data samples as the initial condition and iterate the network to produce 3000 data samples. The waveform of the first 500 data samples is shown in Figure 3(b). Compared with th original signal, these iterates seem much more regular.

Next, we train another TDNN of the same size using output error scheme. We compute the average of the i_s's for all pairs of nearby training sequences. The result is given in Figure 5. The average of these i_s's is 14, and this average is chosen as the length of the training sequences. Once we determine the length of the training sequences, each training pair, including an initial condition and a desired output sequence, is prepared from the signal every 3 points. In other words, two consecutive training sequences overlap 9 data samples. We use the Backpropagation Through Time algorithm to train the network [Werbos]. The learning curve is also given in Figure 3. We note that the final mean square error (m.s.e.) of the equation error training is about half of the m.s.e. of the output error training. After training, the network, referred to as TDNN#2, is also used to reproduce a sequence of 3000 data samples. The waveform of the first 500 points is shown in Figure 3(c). We note that although the final mean squared one-step-prediction error of TDNN#1 is much smaller than that of TDNN#2, the output of TDNN#2 for this given initial condition looks much closer to the mg30 signal. To illustrate the oscillation problem, we increase the length of training sequences to 20 samples. The learning curve is shown in Figure 6. We note that after the m.s.e. drops to 0.06 the performance can not be improved any longer, and oscillations occur.

To further compare the performances of TDNN#1 and TDNN#2, we compute the mean squared multi-step-prediction errors. The error curves are shown in Figure 7. We also plot Casdagli's conjecture curve, which indicates the divergence between the model output and the original signal due to the system dynamics [Casda-

gli]. We note that the error curve of TDNN#2 is very close to Casdagli's conjecture curve. This corroborates the fact that the TDNN trained by using output error scheme is a better dynamic model for the mg30 signal.

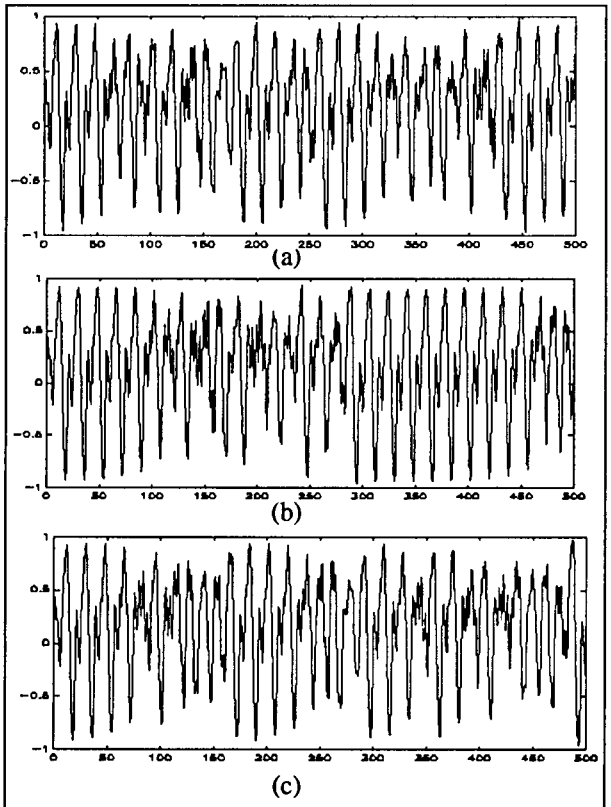


Figure 3. Waveforms of (a) mg30, (b) resulting signal with equation error training, and (c) resulting signal with output error training.

We also compute the dynamical invariants for the original signal and the outputs of both networks, the results are listed in Table 1. Based on these results we conclude that the output error training method can yield a better model if we are able to select the training length properly. And, the mean squared one-step prediction error is not a reliable indicator about how well a model can approximate the underlying dynamics.

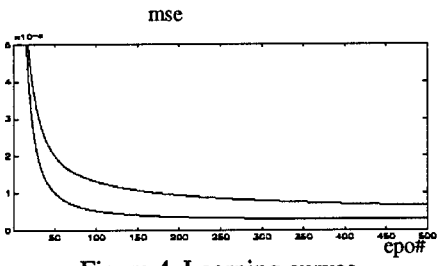


Figure 4. Learning curves

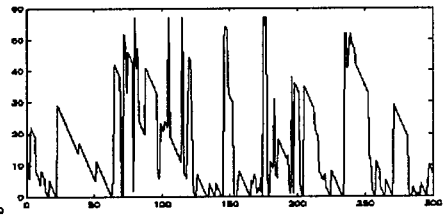


Figure 5. Estimates of i_s's

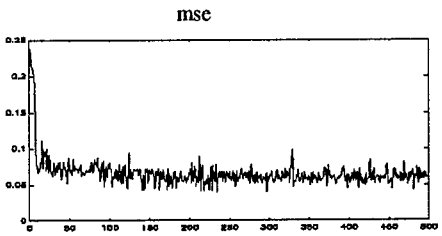
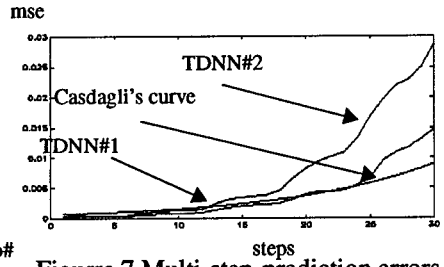


Figure 6 Learning curve for long training sequences



Figurre 7 Multi-step prediction errors

TABLE 1. Measurement of Dynamical Invariants

	dimension	**λ_{max}(nats/sec)**
mg30	2.70±0.05	0.0073 ± 0.0005
iterates of TDNN#1	1.60±0.10	0.0062 ± 0.0005
iterates of TDNN#2	2.65 ± 0.03	0.0074 ± 0.0004

7. CONCLUSIONS

In this work, we assume the underlying system of a signal is autonomous, nonlinear, and deterministic. The uncertainty in the signal waveforms is created by system dynamics. In this framework, we propose to use the measures of dynamical invariants in signal modeling. The preliminary results show that this approach can really improve the model accuracy. And, the oscillation problem in modeling chaotic signals can be avoided. The output error scheme is also shown to be more suitable than the equation error in this framework.

For some chaotic signals, the positive Lyapunov exponents may be very large such that the length estimate of training sequences may become very short (≈ 1). In this case, the training sequences can barely provide the global information about

the signal dynamics. We have proposed another method to prepare the training sequences [Kuo]. The method yields some promising results in modeling Lorenz attractor.

ACKNOWLEDGEMENTS: This work has been partially supported by NSF grant ECS #920878.

REFERENCES

[Cas89] Casdagli, M., "Nonlinear prediction of chaotic time series," Physica D 35, pp.335-356, 1989.

[Cru87] Crutchfield, J. P., and B. S. McNamara, "Equations of motion from a data series," Complex Systems 1, pp. 417-452, 1987.

[Eck85] Eckmann, J. P., and D. Ruelle, "Ergodic theory of chaos and strange attractors," Reviews of Modern Physics, vol. 57, no. 3, part 1, pp.617-656, 1985.

[Eis91] Eisenhammer, T., A. Hubler, N. Packard, and J. A. S. Kelso, "Modeling experimental time series with ordinary differential equations," Biological Cybernetics, vol. 65, pp. 107-112, 1991.

[Gra83b] Grassberger, P. and I. Procaccia, "Characterization of strange attractors," Physical Review Letters, vol. 50, no. 5, pp.346-349, 1983.

[Kuo93] Kuo, J.M., Nonlinear dynamic modeling with artificial neural networks, Ph.D. dissertation, University of Florida, 1993.

[Lap87] Lapedes, R., and R. Farber, "Nonlinear signal processing using neural networks: prediction and system modelling," Technical Report LA-UR87-2662, Los Alamos National Laboratory, Los Alamos, New Mexico, 1987.

[Mac77] Mackey, M. C., and L. Glass, "Oscillation and chaos in physiological control systems," Science, vol. 197, pp. 287-289, 1977.

[Mea91] Mead, W. C., R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, L. A. Lee, and M. K. O'rourke, "Prediction of chaotic time series using CNLS-NET -- example: the Mackey-Glass equation," Technical Report: LA-UR-91-720, Los Alamos National Laboratory, Los Alamos, New Mexico, 1991.

[Mor89] Mozer, M.C., "A focused back-propagation algorithm for temporal pattern recognition," Complex Systems 3, pp. 349-381, 1989.

[Pac80] Packard, N. H., J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series," Physical Review Letters, vol. 45, no. 9, pp. 712-716, 1980.

[Par87] Parker, T. S. and L. O. Chua, "Chaos: a tutorial for engineers," Proceedings of the IEEE, vol. 75, no. 8, pp. 982-1008, 1987.

[Pri92b] Principe, J. C., A. Rathie, and J. M. Kuo, "Prediction of chaotic time series with neural networks and the issue of dynamic modeling," International Journal of Biburcation and Chaos, vol. 2, no. 4, pp. 989-996, 1992.

[San85] Sano, M., and Y. Sawada, "Measurement of the Lyapunov spectrum from a chaotic time series," Physical Review Letters, vol. 55, no. 10, pp. 1082-1085, 1985.

[Shy89] Shynk, J. J., "Adaptive IIR filtering," IEEE ASSP Magazine, pp. 4-21, April, 1989.

[Tak81] Takens, F., "Detecting strange attractors in turbulence," In: Dynamical systems and turbulence, ed. D. A. Rand and L. S. Yang, Lecture Notes in Mathematics, vol. 898, pp. 365-381, Springer, Berlin, 1981.

[Wei90] Weigend, A. S., B. A. Huberman, and D. E. Rumelhart, "Predicting the future: a connectionist approach," International Journal of Neural Systems, vol. 1, pp. 193-209, 1990.

[Wol85] Wolf, A., J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining lyapunov exponents from a time series," Physica 4D, pp. 285-317, 1985.

EEG Signal Analysis Using a Multi-Layer Perceptron with Linear Preprocessing

S. A. Mylonas R.A. Comley
City University, Centre for Information Engineering
Northampton Square, London, EC1V 0HB, UK
e-mail za300@uk.ac.city

Abstract

A system for the detection of epileptic spikes and other transients in the EEG will be described. It consists of a number of adaptive linear filters combined with a non-linear detection unit to control their operation. This has been implemented as a multi-layer perceptron. Configurations using different input preprocessing, initialization and network sizes will be presented along with a discussion on their corresponding results.

1 Introduction

The generation of feeble electrical signals by the brain was known since the end of last century, although their study with the primitive equipment of that time was not easy and the restricted understanding of their origins made interpretation difficult. With the establishment of neurophysiology on a scientific basis and the advances in electronics in the 1950's it became possible to record these time-varying signals on paper by attaching electrodes on the surface of the scalp. Nowadays, such a recording, called the *electroencephalogram* (EEG) can be taken in many hospitals following a harmless as well as inexpensive procedure. Features that relate to the age and level of consciousness and general indications of the mental activity are registered in the EEG, giving a profile of mental health.

Despite recent advances in medical imaging, the EEG is still of great value in monitoring and screening patients suffering from neurological conditions or *idiopathic epilepsy*, where the abnormality is only functional and transient[17].

In epilepsy the abnormal EEG patterns that characterize seizures often occur isolated in "larval" form and are registered in the interictal clinical EEG. *Spikes*, so called because of their shape on a conventional recording are among the commonest patterns. Their presence, absense and frequency of occurrence are valuable clues in the diagnosis and the treatment of this condition[2]. Conventional EEG recordings are sometimes too brief to be trusted for the assessment of patients in order to prescribe some effective therapy. Hence prolonged recording has been suggested as a more reliable alternative[6, 16]. Human interpretation, however, is hindered by the enormous volume of data and the difficulty in their collection from several ambulatory patients. On-line analysis by

0-7803-2026-3/94 $4.00 © 1994 IEEE

a portable microcomputer-based unit was considered as a feasible solution. Early methods were simple because of the limited capabilities of the computers of the time[1]. Despite the sophistication and speed of modern processors no method has been developed as yet to analyze the EEG with total success. This is mainly because the only reference for comparison are human experts, who learn by experience how to perform the highly qualitative and often subjective task of EEG analysis. No formal criteria seem to be followed and even the definitions of the various patterns[5] act as mere guidelines. Intra and inter-reader variability is not uncommon.

The method for automatic EEG analysis described here attempts to combine some attributes of human decision making with the formalism of conventional signal processing tools to form a flexible, but consistent automatic system.

2 Signal modelling and analysis

2.1 A simplified model for the signal

The behaviour of the EEG signal, varies with the level of consciousness, eye opening and closure, mental activity etc. Experts usually separate the EEG into *background activity*, which is the signal present at all times and on which *transients* are superimposed. These have been discriminated into epileptic spikes, which are of medical interest, and other transients. Noise is often present also. Therefore the recorded EEG may be represented as a composite signal:

$$e(n) = b(n) + s(n) + t(n) + v(n) \tag{1}$$

where $b(n)$ is the background activity, $s(n)$ and $t(n)$ are spikes and other transients, respectively, and $v(n)$ is the noise component. Each one of these components was modelled as the output of an an all-pole system excited with either a sequence of impulses or a white uncorrelated sequence[12, 13, 14]. The components of the generating model of the EEG signal are shown in Figure 1.

2.2 EEG signal analysis

The analysis of the EEG signal was based on the inverse of the model. The inverses of $H(z)$, related to the background activity and $G(z)$, related to spike generation, are both transversal "linear predictor" filters[12, 13] with transfer functions of the form $P(z) = 1 + \sum_{i=1}^{L} h_i z^{-i}$. $H^{-1}(z)$ was estimated using on-line linear optimization (adaptive filtering)[19]. It can also track slow changes in the signal behaviour[12, 13], as explained in [18]. $G^{-1}(z)$ was estimated off-line from available spikes, but on-line adaptation of $G(z)$, in the neighbourhood of the optimal, was carried out for better modelling of individual spikes[13].

Other transients may be treated in a similar way. Only one transient was considered this, having the form of an exponential decay with arbitrary polarity, imitating interference potentials from movements of the recording electrodes on the scalp or of ocular origin.

When $H^{-1}(z)$ is applied to the (recorded) EEG signal $e(n)$ the output, $r(n)$, contains the white sequence, $u(n)$, transients due to $s(n)$ and $t(n)$ and additive distorted noise. Similarly, the application of $G^{-1}(z)$ on $e(n)$ would reproduce the sequence of impulses $d(n)$, and other components in the output $y(n)$. Likewise, application of the inverse transfer function of any other transient would produce its generating sequence in $y_i(n)$, among other components.

Spikes are registered in $r(n)$ a fact used by earlier systems for their detection[3, 4, 9, 11]. When other transients are present, they are also registered and the results are inconclusive. Impulses present in $y(n)$ and in any of the $y_i(n)$ are not very reliable indications, because they are burried in non-random signals, caused by $b(n)$.

The proposed method[12, 13], depicted in Figure 2, detected transients in $r(n)$. Making an initial assumption about the origin of the transient, by inspecting $y(n)$ and $y_i(n)$ for all modelled transients and deciding which one is the most likely to have occurred it was possible to generate an excitation impulse to the appropriate generating transfer function ($G(z)$ for spikes) whose output was subtracted from the recorded EEG signal, $e(n)$ to produce a supposedly transient-free signal, $e'(n)$. Processing this through $H^{-1}(z)$ and observing no disturbance in its output $r'(n)$ confirmed the presence of the suspected transient.

3 The need for decisions in signal classification

The reliability of the above scheme, is linked to the accuracy of the detection of transients in $r(n)$ and $r'(n)$ and the discrimination of their origin in either $y(n)$ or one of the $y_i(n)$.

Initially, simple statistical significance testing was employed for the detection of transients. Both $r(n)$ and $r'(n)$ are essentially random sequences, consisting mainly of $u(n)$ if the noise level is low. This follows a normal distribution with zero mean. Transients have amplitudes that are atypically large and may be detected with a certain degree of certainty, $p\%$ using the assumed probability distribution to derive a corresponding level of significance ν. Usually the standard normal distribution is employed, and hence the sample is normalized by the standard deviation of the signal.

This method, however, did not discriminate between *isolated* atypical samples, which are genuine transients and longer *bursts* which are related to certain extracerebral phenomena, like muscle artifacts. To overcome this problem, every time a new sample became available, a linear combination of the N most recent sample values was formed.

$$f(n) = \sum_{i=-N/2}^{N/2} w_i \frac{x(i)^2}{\sigma_x^2(n)} \tag{2}$$

where $x(n)$ is either $r(n)$ or $r'(n)$ and $\sigma_x^2(n)$ is an on-line estimate the power (variance) of $x(n)$, making $f(n)$ independent of the signal level. The weights, $\{w_i\}$ were positive for $i \in [-M/2, M/2]$ where M is a small

number of samples (1,3 or 5) and negative otherwise. The statistical significance level, $\frac{1}{\nu}$ is implicitly included in the weights. By applying a threshold θ on $f(n)$, a binary detection, $z(n)$ is formed.

$$z(n) = \begin{cases} 1 & f(n) > \theta \\ 0 & f(n) \leq \theta \end{cases} \tag{3}$$

Two such elements applied on $r(n)$ and $r'(n)$ verified whether transients were present ($z_1(n)$ and $z_2(n)$). Similar units were introduced to detect transients in $y(n)$ and $y_1(n)$ ($z_3(n)$ and $z_4(n)$). The outputs of these units were combined to produce activation signals for the spike ($z_s(n) = z_1(n)\text{AND}z_3(n)$) and for the other transient ($z_t(n) = z_1(n)\text{AND}z_4(n)$) as well as the "spike detected" output, $z_o(n) = z_1(n)\text{AND NOT}z_2(n)$. $z_s(n)$ and $z_t(n)$ were used as windows on $y(n)$ and $y_1(n)$ to generate the (presumed) impulse activation function of the spike or transient generating filters, for example $\hat{d}(n) = z_s(n)y(n)$. The logical operations required may also be implemented as weighted sums (e.g. $f_s(n) = z_1(n) + z_2(n)$ and $f_o(n) = z_1(n) - z_2(n)$) followed by thresholding ($\theta_s = 1.5$ and $\theta_o = -0.5$). This scheme is an extension of an earlier method which considered only spikes[12]. Unlike its predecessor, which was quite successful, the performance of the extended system was only moderate. The problems associated with the generalized structure are believed to be associated with the selection of an appropriate set of weights for the elements that act on the inputs, rather than with the structure itself.

4 A Multi-Layer Perceptron structure as a decision unit

The layered fixed-weight structure of the decision unit had a logical interpretation. The function of any of its elements may be described in terms of a linear combination of its inputs (equation 2) followed by thresholding (equation 3).

This bears a strong resemblance to the Multi-Layer Perceptron (MLP) neural network, originally described by Rumelhart and McClelland[15], as every layer receives inputs from the previous layer only. The only difference is that the intuitive system does not have all outputs of one layer connected to the next, but with the introduction of these with zero weights the two became equivalent.

Because all the inputs to the MLP are all treated in the same manner, they loose their individual significance and they may be grouped to form a single input vector: $\mathbf{x}(n) = [1r(n+N/2)\ldots r(n-N/2)r'(n+N/2)\ldots r'(n-N/2)y(n+N/2)\ldots y(n-N/2)y_1(n+N/2)\ldots y_1(n-N/2)\ldots]^T$. Describing the weights of the kth element of the lth layer in a similar way $\mathbf{w}_{lk} = [w_{lk0}w_{lk1}\ldots]^T$ the weighting operation may be defined as an inner product, $f_{1k}(n) = \mathbf{w}_{1k}^T\mathbf{x}(n)$. The bias weight, w_{1k0} is multiplied by unity and plays the role of the threshold in the earlier system. The output of the element, $y_{1k}(n) = \sigma[f_{1k}(n)]$ is produced by applying a limiting function (non-linearity), $\sigma[\cdot]$ on the linear output $f_{lk}(n)$. The operation of the element of any layer may be described in terms of the

inner product $f_{lk}(n) = \mathbf{w}_{lk}^T \mathbf{y}_{l-1}(n)$ as described above, using the vector of the outputs of the previous layer, $\mathbf{y}_{l-1}(n) = [1 y_{l-1,1}(n) y_{l-1,2}(n) \ldots]^T$.

The function of the MLP description of the system is defined as a relation between its inputs and its desired output rather than by the behaviour of its individual elements, which was the basis of the earlier system. This is specified as a set of examples, consisting of pairs $[\mathbf{x}(n), \mathbf{d}(n)]$, the training set, where $\mathbf{d}(n)$ is the desired output vector. An optimal weight vector may be found by minimizing the mean-squared error between the desired and the actual output vectors of the output layer $\xi = E\{[\mathbf{d}(n) - \mathbf{y}_L(n)]^2\}$ according to the well-known generalized delta rule (backpropagation algorithm)[15]. Many variants of the basic algorithm exist[10, 7]. The one used in this application has a momentum term and updates the weight vector on every sample, according to the following recursive relation, where for notational convenience, $\mathbf{y}_0(n)$ has been used to denote the input vector:

$$\mathbf{w}_{lk}(n+1) = \alpha \mathbf{w}_{lk}(n) + (1-\alpha)\mathbf{w}_{lk}(n-1) + 2\mu\delta_{lk}(n)\mathbf{y}_{l-1}(n) \qquad (4)$$

where

$$\delta_{lk} = \begin{cases} d_k(n) - y_{lk}(n) & \text{if } l = L \\ \sigma'[f_{lk}(n)] \sum_j \delta l + 1, j(n) w_{l+1,j,k}(n) & \text{otherwise} \end{cases} \qquad (5)$$

where $\sigma'(f) = y(1-y)$ is the derivative of the non-linear logistic function $y = sigma(f) = \frac{1}{1+e^{-f}}$, α is a filtering factor defining the "momentum" and μ is the learning rate of the algorithm.

5 Implementation and other issues

Although the structure and the learning algorithm for the MLP are well-defined several details and problems in the implementation needed to be resolved. The main ones are discussed here.

5.1 Forming a training set

A training set consisting of input-output examples needed to be defined. This is not directly possible from the real EEG records available, because spikes and other transients considered are fairly rare and the exact location of their occurrence unknown. Some spikes that were identified by an earlier system were used but the training set consisted mainly of artificial data, generated according to the described model (section 2.1). For these the location of the excitation functions for the various transients is known exactly. These were used to derive the desired output signals indicating the points of application of the excitation impulses for spikes, $z_s(n)$ and the other type of transients, $z_t(n)$, as well as a separate indication for the occurrence of spikes $z_o(n)$ (Figure 3). The real EEG training patterns were used as well, but their effect on the final weights was not visible.

5.2 Implementation of the MLP

One of the problems encountered during the implementation was the fact that the MLP is embedded in the rest of the system and that its spike detection output depends indirectly on the spike excitation signal, which is also an output. In other words, there is feedback. Hence training the network cannot be done independently and as a result it was not possible to use many of the available tools for developing and training the network.

It was therefore necessary to develop a library of functions that deal with the construction of networks of different configurations as well as the implementation of the backpropagation algorithm. The library, which was realized in the C programming language, like the rest of the system, also provides the means to read a network configuration and learning parameters from a file and save them in a format that is easily read by computers and humans alike.

5.3 Preprocessing of the inputs to the MLP

An important issue when using neural networks with natural signals like the EEG is the format of the inputs to the network. Although it is sometimes claimed that there is no need for preprocessing, it is obvious that performance may be affected by changes in the dynamic range of the inputs. To ensure that the input levels to the network are not affected much by such fluctuations, they were normalized in a way similar to the one described for the earlier system (equation 2). All inputs to the network were divided by the RMS value of $r(n)$. This is a convenient measure, because it is also representative of $r'(n)$ and is not affected by modifications to the system by the extension or restriction of the number of transients considered.

Preprocessing of the inputs using other operations to assist the network to learn or to simplify its structure were also considered. The latter is important if the system is to be implemented on a small portable microcomputer with limited processing capabilities and resources.

Two types of preprocessing were considered. The first consists of simply scaling all the elements in the input vector by σ_r, as already explained in the previous paragraph. The inputs to the network, $\hat{x}_i(n)$ are related to the corresponding "raw" inputs, $x_i(n)$ by the simple relation $\hat{x}_i(n) = \frac{x_i(n)}{\sigma_r}$. No information is lost during this operation, but the size of the network may be larger than other alternatives.

Squaring the elements of the input vector prior to their application to the MLP, so that $\hat{x}_i(n) = \frac{x_i^2(n)}{\sigma_r^2}$ was another simple form of processing. This resembles the operation performed by equation 2, which has an intuitive interpretation (section 3).

5.4 Initial conditions of the learning algorithm

When training the MLP using the backpropagation algorithm the weights of the processing elements are usually initialized to small random values. In this application the weights of the intuitive system of section 3

for which the performance was not perfect, but not unreasonable, were considered as a possible alternative, as they may be closer to an optimal solution than a random weight vector. Strictly speaking, these are only valid if the inputs are squared and normalized.

The elements on the first (hidden) layer were made to detect whether a sample at a specific position in the sequences of the input signals, $\{r(n)\}$, $\{r'(n)\}$, $\{y(n)\}$ and $\{y_1(n)\}$ has an atypically large amplitude compared to the ones in its neighbourhood. The result was then fed into one more (hidden) layer, implementing logical AND operations between the outputs of the first layer, before they are combined by the logical OR elements of the output layer for the producton of $z_s(n)$, $z_t(n)$ and $z_o(n)$.

6 Results

The system was tested for a number of combinations of input preprocessing and weight initialization methods. Training was primarily carried out using synthetic data, because the available EEG signals did not contain substantial numbers of other transients. Tests, however, were carried out on real EEG records as well.

Different MLP configurations were tested, either with normalized or with normalized-squared inputs, as explained in section 5.3. The convergence of the backpropagation algorithm for random and preset initial weight vectors (section 5.4) was also investigated.

Tests were carried out on a simulated EEG record containing 25 spikes and 20 other transients and a real EEG record containing 52 spikes. The number of successful spike detections was noted for every network configuration, as well as the number of iterations required for the network performance to stop improving. Some networks were also tested with a different number of elements in the first hidden layer. The results are shown in Table 1.

Net Config.	Iterations to Converge	Detections		
		Simulated EEG		Real EEG
		Spikes (25)	Transients (20)	Spikes (72)
1	60000	25 (0)	20 (0)	71 (5)
2	80000	25 (0)	20 (0)	71 (5)
3	60000	25 (2)	20 (0)	70 (5)
4	10000	25 (2)	20 (0)	70 (5)
5	*	*	*	*
6	30000	25 (2)	20 (0)	70 (12)

Note: * indicates complete failure of the system

Table 1: Results

All tested configurations had 44 inputs (11 samples from each input signal) and 3 outputs. The first four had two hidden layers with 8 and 5 elements respectively, and the other two had two hidden layers with 5 elements each. Configurations 1, 2 and 5 were for normalized inputs and

3, 4 and 6 for squared and normalized inputs. 1, 3, 5 and 6 had random initial weights, whereas 2 and 4 had preset weights.

Figure 4 shows a typical set of waveforms from the simulated EEG data records. The indications of spike detections under the input signal show the proper operation of the system, whereas the estimated spikes and transients in the third set demonstrate that $z_s(n)$ and $z_t(n)$ are generated correctly as well.

7 Comments, observations and conclusions

All but one of the configurations presented in Table 1 performed reasonably well with both simulated and real EEG data. In spite of training the network mainly with synthetic data, its behaviour with real EEG signals was still quite good. Although the number of data records considered in this study was limited, results show that the MLP-based system is capable of performing well even when its inputs deviate from those used for its training.

The number of iterations to obtain a satisfactory performance was considerably smaller for squared-normalized inputs, when the weights were initialized to those of the earlier system than when random initial values were used. This indicated that the former were closer to an optimal, which was also apparent from the small distance between the initial and the final weight vectors. The final weight sets for the two different initialization procedures were different, but this is not surprising, as there are many combinations of weights that give an optimal performance for a given network architecture[8]. For the normalized only input vector, this was not the case. It appears that starting from random weights produces faster convergence than when the weights of the earlier system were used. This is not surprising, because the latter corresponds to the weights for squared inputs, which is a completely different case with the preset initialization being perhaps far from the optimal.

The loss of the polarity of the input signal with squaring as preprocessing was evidenced in the form of the two false spike detections, indicated in brackets for the simulated EEG record. These corresponded to spikes with negative polarity which were intentionally introduced. Real epileptic spikes always have positive polarity. These have not been detected by the systems 1 and 2, which maintain the polarity of their inputs.

Finally, for configuration 6 with a reduced number of elements and the inputs squared, the system was still quite successful. This structure required a simpler network to produce a satisfactory output, because some of the burden of preprocessing was shifted to the the input. This was not the case in configuration 5, where the lack of an adequate number of elements led to a poor performance of the MLP leading to the degradation of the performance of the system.

The main advantage in using the MLP in this application lies in its ability to learn by example. This permits the inclusion of medical expertise which cannot be expressed in a set of rules. Hence the EEG analysis system presented may be trained by an individual expert to reflect his/her experience or by a group of analysts to act like a less

subjective analysis tool combining both medical experience and formal engineering methodology.

References

[1] J. H. Barlow: *Computerized Clinical Electroencephalography in Perspective,* IEEE Trans. Biomed. Eng., vol. BME-26, No. 7, pp. 377–391 (Jul. 1979).

[2] C. D. Binnie: *What's the use of EEG in epilepsy?* British Journal of Hospital Medicine, February 1988, p. 98, 1978, pp. 575–585.

[3] W. P. Birkemeier, et al.: *Pattern Recognition Techniques for the Detection of Epileptic Transients in the EEG,* IEEE Trans. Biomed. Eng., vol. BME-25, No. 3, pp. 213–216 (Jul. 1978).

[4] G. Bodenstein and M. Praetorius: *Feature Extraction from the EEG by Adaptive Segmentation,* Proc. IEEE, vol. 65, No. 5, pp. 642–657 (May. 1977).

[5] G. E. Chatrian et al.: (IFSECN International Assembly) *A Glossary of Terms Most Commonly Used By Clinical Electroencephalographers* Electroenceph. Clin. Neurophysiol., 37, (1974), pp. 538–548.

[6] R. A. Comley and J. E. Brignell: *Real-Time Detection of the Epileptic Precursor,* J. Phys. E: Sci. Instrum., vol. 14, pp. 963–967 (1981).

[7] R. Hecht-Nielsen: *Neurocomputing,* Addison–Wesley (1990).

[8] A. M. Chen and R. Hecht-Nielsen: *On the Geometry of Feedforward Neural Network Weight Spaces,* Proceedings, Second International Conference on Artificial Neural Networks, Bournemouth, (Nov. 1992), pp. 1–4.

[9] A. Isaksson, et al.: *Computer Analysis of EEG Signals with Parametric Models,* Proc. IEEE, vol. 69, No. 4, pp. 4151–4161 (Apr. 1981).

[10] R. P. Lippmann: *An Introduction to Computing with Neural Nets,* IEEE ASSP magazine, Apr. 1987, pp. 4–22.

[11] F. H. Lopes Da Silva, et al.: *Automatic Detection and Localization of Epileptic Foci,* Electroenceph. Clin. Neurophys., No. 43, pp. 1–13 (1977).

[12] S. A. Mylonas and R. A. Comley: *Detection of Epileptic Spikes in the EEG Using Adaptive Filters,* I Fórum Nacional de Ciência e Tecnologia em Saúde – XIII Congresso Brasileiro de Engenharia Biomedica, Caxambu (MG), Brazil (Nov. 1992)

[13] S. A. Mylonas and R. A. Comley: *Adaptive Predictive Modelling for the Analysis of the Epileptic EEG,* Singapore ICCS/ISITA'92, vol. 3, pp. 1214-1218 (Nov. 1992)

[14] S. A. Mylonas and R. A. Comley: *Linear Prediction, Neural Networks and the Analysis of EEG Signals,* Cyprus, Int. Conf. on DSP/II Int. Conf. on Comput. Appl. to Eng. Sys. (Jul. 1993)

[15] D. E. Rumelhart et al.: *Learning internal representations by error propagation,* in D. E. Rumelhart and J. L. McCLelland (Eds.) *Parallel Distributed Processing: Explorations in the microstructure of Cognition,* 1, pp. 318–362, (1986), MIT Press.

[16] A. L. Stelle and R. A. Comley: *Portable Analyser for Real-Time Detection of the Epileptic Precursor,* Proc., XI Brazilian Conf. in Biomed. Eng., pp. 101–107 (Sep. 1989).

[17] J. N. Walton: *Brain Diseases of the Nervous System (8th Ed.): Chapter 22, pp. 1093–1132,* Oxford Medical Publications.

[18] B. Widrow, et al.: *Stationary and Nonstationary Learning Characteristics of the LMS Adaptive filter,* Proc. IEEE, vol. 64, No. 8, pp. 1151–1162 (Aug. 1976).

[19] B. Widrow and S. D. Stearns: *Adaptive Signal Processing,* Prentice-Hall Inc., NJ (1985).

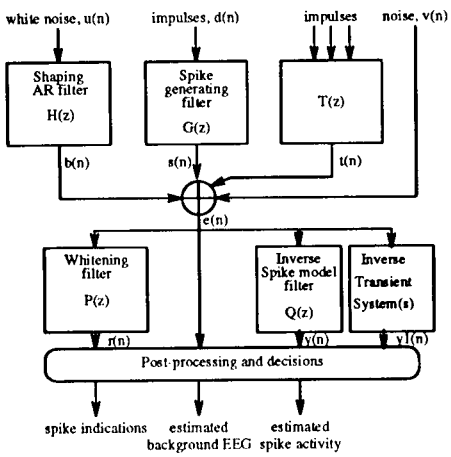


Figure 1: EEG modelling processes for synthesis and analysis

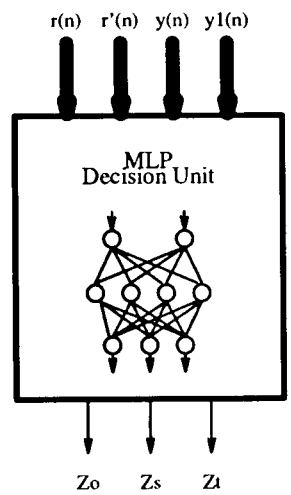


Figure 3: The MLP decision unit

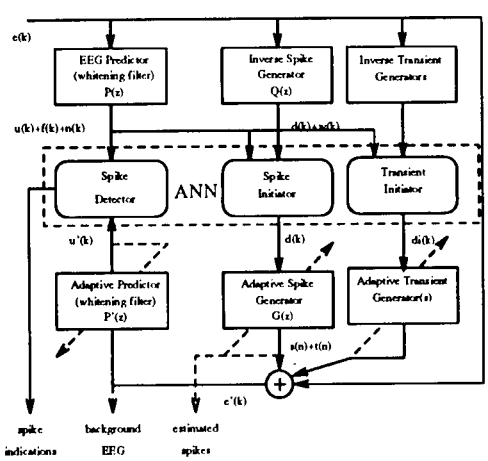


Figure 2: Block diagram of the proposed system

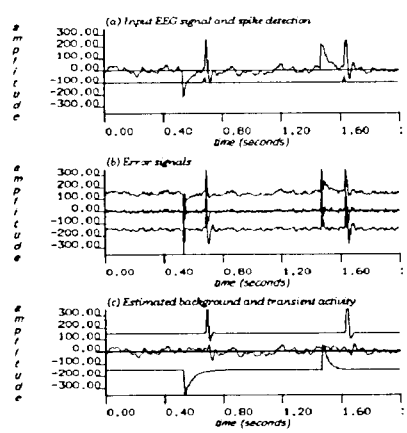


Figure 4: Waveforms at the various stages of the system

A Neural Network Scheme For Earthquake Prediction Based On The Seismic Electric Signals

Spiros Lakkos[1], Andreas Hadjiprocopis[2],
Richard Comley[1], Peter Smith[2]
City University,
Centre for Information Engineering*(1)*,
Department of Computer Science*(2)*,
Northampton Square, London, EC1V 0HB, UK

Abstract

Earthquake prediction based on the *Seismic Electric Signals* usually employs statistical linear models. In this paper, an alternative scheme for earthquake prediction and modelling of the geophysical characteristics based on Artificial Neural Networks, is presented. Several network configurations are investigated and the results are discussed and interpreted in various ways.

Introduction

It has been reported that transient variations of the *electrotelluric field* - called *Seismic Electric Signals*(SES) - are observed before an earthquake. The study of the physical properties of these signals is used for the determination of the parameters (epicentre and magnitude) of an impending event[1].

The occurrence of these precursors varies from a few hours to a few days before the earthquake and have a duration of one minute to a few hours. These signals appear as a transient change of the potential difference measured between two electrodes (up to a few tens of μV/m) (Figure 1) depending on the earthquake magnitude, Ms, the epicentre, local geophysical inhomogeneities, source characteristics and travel path. The

0-7803-2026-3/94 $4.00 © 1994 IEEE

components of the electric field are measured in two perpendicular directions (East-West and North-South) using dipoles with lengths varying from a few tenths of meters to a couple of kilometers.

Very often, noise obstructs the clarity of SES. It can be classified into three categories depending on the nature of the cause: *electrochemical, magnetotelluric and cultural.*

By using various techniques[9], [10] to eliminate the noise from the electrotelluric field measurements and by applying certain, well defined, criteria[2], the detection of SES is achieved.

The study of the physical properties of SES and their correlation to the impending seismic activity leads to the construction of an *empirical selectivity map* for a monitoring station. *Selectivity* is defined as the sensitivity of a station to signals from a restricted number of seismic areas while remaining insensitive to SES from other areas[3].

In this paper an alternative approach is suggested for the construction of the selectivity maps based on the use of *Artificial Neural Networks.*

Artificial Neural Networks

The most basic function of Artificial Neural Networks (ANN) is the mapping of an N-Dimensional space to M Dimensions. By adjusting the weights of the internal connections of the network, through training, a transformation function is approximated.

The accuracy of the resultant mapping depends on the amount of output error at the end of the training process, as well as, whether the training set is a representative sample of the domain of the application.

The problem was to find a suitable transformation which would map the two dimensional input data (the relative SES components [mV/m] in the directions East-West and North-South) collected by the monitoring station, into a three dimensional representation (the geographical location - longitude and lattitude - and the magnitude of the impending event), (Figure 2).

The XERION software package[16] was used to simulate a

feed-forward Network. Several combinations of network architectures and training algorithms were tested. The configuration that gave the most satisfactory results comprised of:

- Two input nodes corresponding to the two components of the SES,
- Fourty five hidden layer nodes,
- Three output nodes corresponding to longitude, lattitude and magnitude information.

The Delta-Bar-Delta[16] training algorithm was employed.

The training data was collected by a monitoring station based at Ioannina (western Greece) and presented in [1 to 8]. Due to its small size, expansion of the original set was necessary by addition of a small amount of Gaussian noise to each of the training vectors. The size of the data set has been increased by a factor of five, (Figure 3).

The mapping produced with the expanded data set works well since the network now has a better idea of what the input surface looks like and any misinterpretations due to restricted input data are avoided. Although the overall output error in this case increases, a continuous and smooth output is obtained, moreover, meaningless output values are avoided.

After convergence, the network can be used to predict impending earthquakes and construct the selectivity map for a monitoring station.

Interpretation of the Results

The network has been tested using a small subset of the available data which has not been presented to the network during the training process. The majority of the training vectors were associated with earthquakes from the geographical area $20.0^oE - 21.5^oE, 37.5^oN - 40.0^oN$. As a result the network prediction accuracy was higher in that area. The epicentre location can be predicted with an error of less than 0.3^o, and the magnitude with an error of less than 0.5 Ms. The most successful of the methods used so far for the same purpose, based on

traditional statistical linear models have approximately twice as much error.

Furthermore, by feeding the network with a data set occupying the whole input space, a surface related directly to the sensitivity properties of the station is obtained, thus, approximating its selectivity map, (Figure 4).

A possible way of investigating the travel paths of the SES and certain geophysical characteristics of the monitoring area is to present the network with a set of data as above and plot only the epicentre information, discarding the magnitude. The obtained curves or family of curves indicate paths where SES, sensitive to that station, possibly propagate, (Figure 5). Similarly, by discarding the epicentre information and plotting only the earthquake magnitude versus one component of the input vectors, while the other is kept constant, the correlation between the magnitude of the earthquake and the SES may be obtained, (Figure 6).

Comments and Conclusions

The method presented here is superior to the classical statistical method for the prediction of earthquakes based on the SES. The construction of the selectivity map has become a relatively easy and accurate task. Once the network is trained with a large data set, the results can be considered sufficiently accurate for practical purposes. Furthermore the trained network can be used as a model for geophysical research.

More work can be done to further investigate the behaviour of the Neural Net under unconventional conditions. Other network topologies, such as Self Organised Maps, could be employed. It is also worth considering the idea of a network with inputs from more than one monitoring station.

Despite the fact that so far a single network was employed assuming a strong correlation between the magnitude and epicentre of the earthquake, it could be possible to use two separate networks.

We would like to acknowlegde the help and advice of Mr. J.Makris, University of Athens.

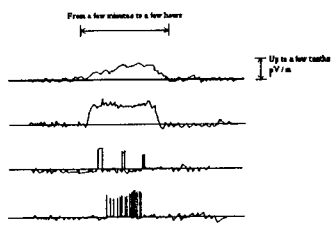


Figure 1: **Typical forms of Seismic Electric Signals.**

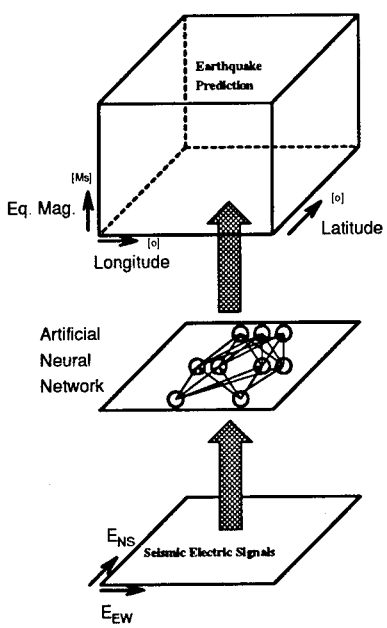


Figure 2: **Input to Output Space transformation by means of ANN.**

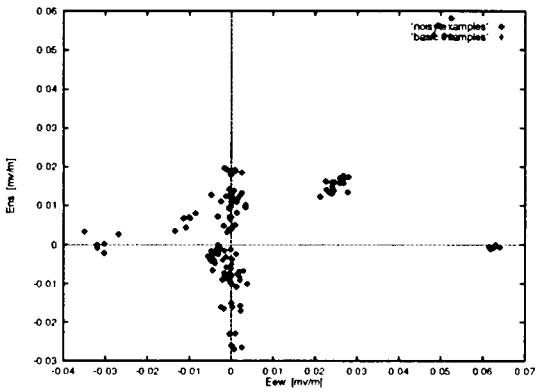


Figure 3: **The input training vectors.**

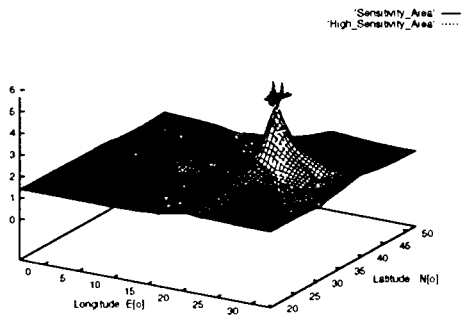


Figure 4: **Sensitivity map of a monitoring station.**

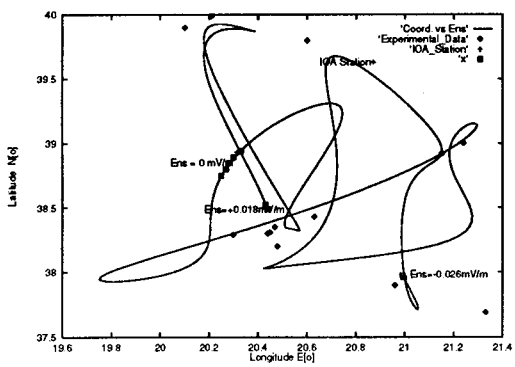


Figure 5: **Epicentral location as a function of E_{NS} with E_{EW} constant.**

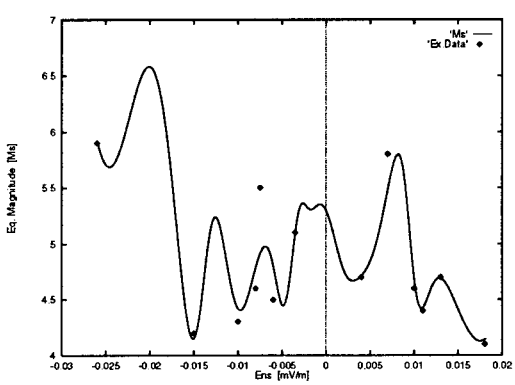


Figure 6: **Earthquake magnitude as a function of E_{NS} with E_{EW} constant.**

References

[1] Varotsos P. and Alexopoulos K.: *Physical Properties of the Variation of the Electric Field of the Earth Preceding Earthquakes I and II,* Techtonophysics, vol 110, (1984).

[2] Varotsos P. and Lazaridou M.: *Latest Aspects of Earthquake Prediction in Greece based on Seismic Electric Signals,* Techtonophysics, vol.188 (1991).

[3] Varotsos P., Alexopoulos K. and Lazaridou M.: *Latest Aspects of Earthquake Prediction in Greece based on Seismic Electric Signals II,* Techtonophysics (1993).

[4] Varotsos P., Alexopoulos K., Lazaridou M., Nagao T.: *Earthquake predictions issued in Greece by seismic electric signals since February 6, 1990,* Techtonophysics (1993).

[5] Varotsos P., Alexopoulos K., Lazaridou M.: *Recent earthquake predictions issued by the VAN-network in Greece. Period: Feb. 6, 1990 - May 31, 1991,* submitted in Techtonophysics.

[6] Shnirman M., Schreider S., Dmitrieva O.: *Statistical evaluation of the VAN predictions issued during the period 1987-1989,* Techtonophysics (1993).

[7] Dologlou E.: *A three year continuous sample of officially documented predictions issued in Greece using the VAN method: 1987-1989* Techtonophysics (1993).

[8] Eftaxias K., Hadjicontis: *Information material on earthquake prediction in Greece by means of seismic electric signals,* Int. Conf. Measurement and Theoretical Models of the earth's electric field variations related to earthquakes, Feb. 6-8 Athens).

[9] Chouliaras G. and Rasmussen T.M.: *The Application of the Magnetotelluric Impedance Tensors to Earthquake Prediction Research in Greece,* Techtonophysics, vol. 152 (1988).

[10] Lakkos S. and Comley R.A.: *An Adaptive System for the Estimation of the Magnetotelluric Impedance Tensor and its Application in Earthquake Prediction,* International Proceedings of the International Symposium of Information Theory and Applications, ISITA '92, Singapore (1992).

[11] McCullough W.S. and Pitts W.: *A Logical Calculus of the Ideas Immanent in Nervous Activity,* Bull. Math. Biophys. 5, 115-133 (1943).

[12] Rosenblatt F.: *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms,* Spartan Books, New York (1962).

[13] Widrow B. and Hoff M.E.: *Adaptive Switching Networks,* IRE Wescon Convention Record (1961).

[14] Freeman J.: *Neural Networks: Theory and Practice,* Addison-Wesley (1991).

[15] Hecht-Nielsen R.: *Neurocomputing,* Addison–Wesley (1990).

[16] Drew Van Camp: *The XERION Neural Network Simulator Users Guide,* University of Toronto (1993).

Pruning Recurrent Neural Networks for Improved Generalization Performance

Christian W. Omlin [a,b], C. Lee Giles [a,c]
[a] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540
[b] Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180
[c] UMIACS, U. of Maryland, College Park, MD 20742
Phone: (609) 951-2678/2642 FAX: (609) 951-2482
E-mail: {omlinc,giles}@research.nj.nec.com

Abstract

The experimental results in this paper demonstrate that a simple pruning/retraining method effectively improves the generalization performance of recurrent neural networks trained to recognize regular languages. The technique also permits the extraction of symbolic knowledge in the form of deterministic finite-state automata (DFA's) which are more consistent with the rules to be learned. Weight decay has also been shown to improve a network's generalization performance. Simulations with two small DFA's ($\leq$ 10 states) and a large finite-memory machine (64 states) demonstrate that the performance improvement due to pruning/retraining is generally superior to the improvement due to training with weight decay. In addition, there is no need to guess a 'good' decay rate.

1 MOTIVATION

We propose a simple, destructive training method for improving the generalization performance of recurrent neural networks trained to recognize regular languages. To our knowledge, no such techniques for recurrent networks has been previously published. In addition to improved generalization performance, we also demonstrate that the rules extracted in the form of deterministic finite-state automata is superior to those extracted from larger networks. Good generalization results have also be reported using weight decay ([9, 11]). We will compare our pruning method with weight decay for different decay rates.

2 PRUNING A RECURRENT NETWORK

We incrementally trained discrete-time, recurrent networks with second-order weights W_{ijk} to learn regular languages [2, 5, 12, 14]. The weights W_{ijk} were updated according to a second-order form of the RTRL learning algorithm for recurrent neural networks ([15]). For more details see [5]. The heuristic we use for extracting rules from recurrent networks in the form of deterministic finite-state automata (DFA's) is described in detail in [5]. Different approaches are discussed in [2, 14]. The quality of the extracted rules has been discussed in [6].

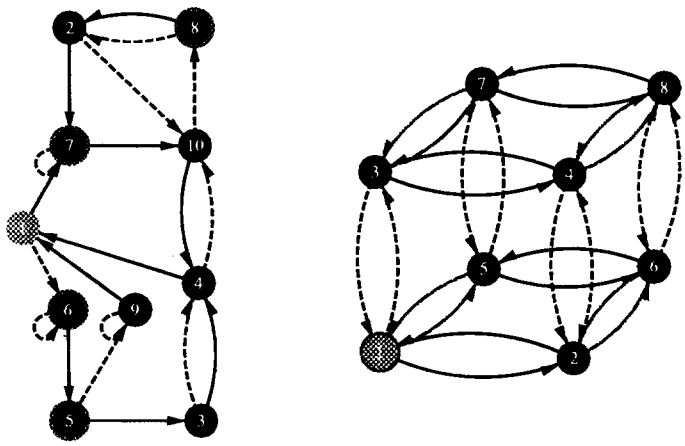
Our goal is to train networks of small size with improved generalization performance and also to improve the quality of the extracted rules. We start by training a large network for a known regular grammar and apply our network pruning and retraining strategy to the trained network. Whenever the training is successful, the state neuron with the smallest weight vector is removed and the network is retrained using the same training set. This process is repeated until either a network with satisfactory generalization performance is obtained or until the retraining fails to converge within a certain number of epochs. When the current network fails to converge, we choose the network trained in the previous prune/retrain cycle as our solution network.

3 SIMULATION RESULTS

3.1 Experiments

We trained recurrent networks on two different training sets: The first set was obtained from the randomly generated 10-state DFA shown in figure 1a. It consists of the first 500 positive and 500 negative example strings in alphabetical order with alternating positive and negative strings. Since this a second-order modification of RTRL, training can occur at the end of each presented string. The second training set was generated by the DFA shown in figure 1b. It accepts only strings which have an even number of 0's, 1's and 2's in it (triple parity). The initial training set consisted of 30 strings; the learning rate and the momentum were set to 0.5. We started training with a network with 15 state neurons and the weights were initialized to random values in the interval [-1.0, 1.0].

All networks were trained on the same training set. However, using an incremental training heuristic, none of the networks needed to be trained on all strings. The learning of the DFA states from short strings allowed the network to correctly classify longer strings without explicit training on these strings.



(a) (b)

Figure 1: **Inferred DFA's.** (a) randomly generated DFA with 10 states and two input symbols. (b) the DFA for triple parity accepts only strings with an even number of 0's, 1's and 2's.

3.2 Generalization Performance

For each (re)training/pruning cycle, we show in table 1 the number of state neurons, the (re)training time, the size of the training set necessary for successful training, the generalization performance of the trained network, the quantization level q used for DFA extraction, the size of a good minimized DFA extracted from the network and its generalization performance.

The results for the randomly generated 10-state DFA are shown in table 1. A network with 15 state neurons learned the training set relatively easily (197 epochs) and only a fraction of the entire training set was necessary (142 strings). The generalization performance of the trained network on all strings of length up to 20 (2,097,150 strings) is fairly good with only 6.75% of all strings misclassified. For DFA extraction, we only considered DFA's that were consistent with the training set, i.e. the DFA's correctly classified all strings of the training set. As a good model of the regular language, we chose the consistent DFA extracted with the smallest quantization level q ([6]). For the trained 15-neuron network, we were able to extract a consistent DFA for q=2; however, the minimized DFA had 382 states as compared to 10 states for the DFA that generated the training set. The extracted DFA's generalization performance is impressive with only 0.41% of all test strings misclassified, thus outperforming the trained network (this is often the case, see for example [6]).

After pruning the first state neuron of the network, the retraining time was negligible (7 epochs), indicating that the pruned state neuron did not contribute significantly to the internal representation of the learned DFA. The generalization performance of the pruned and retrained network and the extracted DFA were comparable to the performance of the larger network.

The retraining got harder with fewer state neurons while the network gen-

eralization performance improved by an order of magnitude (0.14% for the 7-neuron network). This improvement would come as no surprise if the maximum size of the training set were also increasing; with more training strings used, one would certainly expect the network to perform better. However, the generalization improvement was achieved in most cases *without* additional new training strings. At each stage of the pruning/retraining process, the size of the training set was smaller than the size of the training set used to train the 15-neuron network. This clearly shows that the performance improvement is due to the reduced size of the network.

Neurons	Time	Size	NN Performance	q-level	DFA States	DFA Performance
15	197	142	6.75%	2	382	0.41%
14	7	46	6.89%	2	484	1.57%
13	98	99	2.61%	2	314	0.35%
12	11	62	1.51%	2	10	0.00%
11	14	67	0.97%	2	10	0.00%
10	22	83	1.26%	2	135	0.05%
9	111	157	2.95%	2	151	0.62%
8	102	140	2.44%	4	505	1.21%
7	104	118	0.14%	2	10	0.00%

Table 1: **Random DFA:** Network performance after each pruning cycle; epochs; maximal size of the maximal training set; NN classification errors on test set; quantization level; size of extracted DFA; DFA classification errors.

After retraining the 7-neuron network, we attempted to further reduce the size of the network. However, the 6-neuron network failed to converge within 50,000 epochs. Thus, the 7-neuron network was our final network. Trained recurrent networks make generalization errors because the internal representation of DFA states is unstable, i.e. with increasing string length, well-separated neuron activation clusters formed during training begin to merge together ([16]). The extracted DFA's do not share this problem and thus show consistently better generalization performance. The DFA extracted from the smallest network (7 neurons) was identical with the original DFA. The quality of the extracted rules also tends to improve with decreasing network size.

The results shown for triple parity (table 2) confirm our findings that our pruning/retraining algorithm is an effective tool for improving the generalization performance of both the trained network as well as the extracted DFA. Note that in this case the resultant 3-state neural network is the least size neural network for "representing" the 8-state DFA *if* the internal state representations of the network are confined to the rails of the sigmoid ([16]).

Neurons	Time	Size	NN Performance	q-level	DFA States	DFA Performance
15	183	209	13.64%	3	3105	8.42%
14	3	42	13.85%	3	2560	4.17%
13	17	84	10.08%	2	128	0.00%
12	22	99	9.62%	2	81	0.00%
11	12	65	5.45%	2	8	0.00%
10	23	92	3.87%	2	46	0.00%
9	20	83	3.27%	3	124	0.00%
8	23	91	4.07%	2	8	0.00%
7	36	93	3.16%	2	8	0.00%
6	29	96	3.98%	3	8	0.00%
5	39	87	0.58%	2	8	0.00%
4	29	85	2.08%	2	8	0.00%
3	179	92	0.75%	3	8	0.00%

Table 2: **DFA for Triple-Parity:** Network performance after each pruning cycle; epochs; maximal size of the maximal training set; NN classification errors on test set; quantization level; size of extracted DFA; DFA classification errors.

3.3 Comparison with Weight Decay

It has been observed in simulations that weight decay can improve the generalization performance of feed-forward networks ([9, 11]). Weight decay suppresses irrelevant components of weight vectors by choosing a small vector that solves the learning problem.

For networks trained using weight decay, the error function is expanded to include an error term which penalizes large weights: The weight update then becomes

$$\Delta w_{ijk} = -\alpha \frac{\partial E_0}{\partial w_{ijk}} - \lambda w_{ijk}$$

The results in table 3 show a comparison of the performances of pruned networks with the generalization of networks trained with weight decay for varying decay rates λ. The training set was the same as the one used above to learn the random 10-state machine. In all but one case, the pruned networks outperformed the networks with weight decay. The training time for pruned networks includes the initial time necessary to train a 15-neuron network and the retraining time for each pruning step. The pruning always resulted in networks with 7 state neurons. The training times for pruned networks and networks with weight decay were comparable, although pruning causes fewer weight updates after each pruning/retraining cycle. The methods refer to plain training (none), training with pruning (pruning), and training with weight decay ($\lambda = 0.0001$, $\lambda = 0.0005$, $\lambda = 0.001$). The pruning heuristic always improved both the network generalization performance and the extracted DFA, especially when the ideal DFA with 10 states was not already extracted in the original 15-neuron network. The convergence time for training with weight decay increases with increasing decay rate. None of the runs converged for values of λ larger than the ones shown. In cases where the original network was not well trained (table 3a), weight decay improved network generalization and the extracted rules. However, in all other cases (tables 3b-d), networks trained with weight decay can show worse generalization performance and DFA's were extracted that were consistent with the training data, but not identical with the ideal 10-state DFA. We can conclude that our pruning heuristic generally results in better trained networks and smaller DFA's that explain the training data than weight decay methods. In addition we did not have the weight decay disadvantage of possible failure to converge to a good solution or the need to set the decay rate λ prior to training.

Method	Time	NN Performance	DFA States
none	197	6.75%	382
pruning	666	0.14%	10
$\lambda = 0.0001$	199	4.18%	10
$\lambda = 0.0005$	257	3.18%	30
$\lambda = 0.001$	401	2.20%	10

(a)

Method	Time	NN Performance	DFA States
none	175	2.76%	81
pruning	437	0.21%	10
$\lambda = 0.0001$	141	1.32%	13
$\lambda = 0.0005$	186	1.03%	10
$\lambda = 0.001$	362	2.92%	10

(b)

Method	Time	NN Performance	DFA States
none	151	0.90%	10
pruning	375	0.00%	10
$\lambda = 0.0001$	154	1.93%	87
$\lambda = 0.0005$	169	0.97%	10
$\lambda = 0.001$	305	1.74%	10

(c)

Method	Time	NN Performance	DFA States
none	161	2.14%	10
pruning	282	1.12%	10
$\lambda = 0.0001$	172	0.61%	10
$\lambda = 0.0005$	200	3.28%	72
$\lambda = 0.001$	351	2.49%	13

(d)

Table 3: **Comparison Pruning vs. Weight Decay for DFA:** The methods refer to plain training (none), training with pruning (pruning), and training with weight decay rates ($\lambda = 0.0001$, $\lambda = 0.0005$, $\lambda = 0.001$).

4 FINITE-MEMORY MACHINES

The example DFA's in the previous sections were small (≤ 10 states). Current learning algorithms based on gradient descent searches are useful tools for learning these small DFA's because they converge fast; however, they are currently inappropriate for learning larger DFA's because of problems with the propagation of error information - and thus state information - over long strings [1].

Method	Time	NN Performance	DFA States
none	189	1.10%	334
pruning	641	1.45%	73
$\lambda = 0.0001$	200	0.29%	230
$\lambda = 0.0005$	309	0.76%	376
$\lambda = 0.001$	769	3.16%	65

(a)

Method	Time	NN Performance	DFA States
none	281	1.59 %	296
pruning	956	1.03 %	67
$\lambda = 0.0001$	199	3.78 %	745
$\lambda = 0.0005$	274	1.98 %	316
$\lambda = 0.001$	818	1.30 %	65

(b)

Method	Time	NN Performance	DFA States
none	257	1.69%	110
pruning	512	0.60%	71
$\lambda = 0.0001$	234	1.97%	316
$\lambda = 0.0005$	356	2.60%	78
$\lambda = 0.001$	-	-	-

(c)

Method	Time	NN Performance	DFA States
none	205	1.71%	409
pruning	536	0.33%	190
$\lambda = 0.0001$	226	1.96%	197
$\lambda = 0.0005$	387	1.07%	173
$\lambda = 0.001$	-	-	-

(d)

Table 4: **Comparison Pruning vs. Weight Decay for FMM:** The methods refer to plain training (none), training with pruning (pruning), and training with weight decay rates ($\lambda = 0.0001$, $\lambda = 0.0005$, $\lambda = 0.001$). Training with weight decay factor $\lambda = 0.001$ did not converge within 5000 epochs in all cases.

There exists a subclass of DFA's called *finite-memory machines* [10]. Formally, a finite-memory machine (FMM) is a DFA with finite memory input-

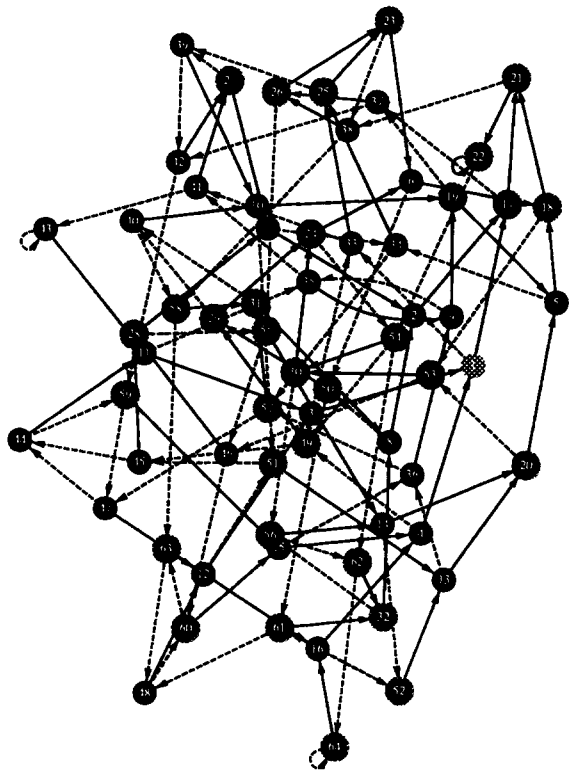


Figure 2: **Inferred Finite-Memory Machine.** This FMM with 64 states was generated with parameters $m = n = 3$.

order m and output-order l,n i.e. m and n are the least integers such that the present state of a DFA can always be uniquely determined from the knowledge of the last m inputs and the last n outputs. We suspect that large FMM's can be learned more easily than large DFA's because the interval over which state information has to be propagated is fixed and small compared to the length of the training strings. A FMM with 64 states (m=n=3) is shown in figure 2. We trained networks with 15 state neurons on the first 1000 strings (positive and negative strings) whose labels were determined by the FMM of figure 2. We compared the training time, the network generalization performance and the quality of the rules extracted from trained networks for plain training, training with pruning and training with weight decay for 4 networks with different initial conditions. The results are shown in table 4. We observed that smallest network which for which the pruning/retraining method converged always had 7 state neurons. The results confirm the observation we made earlier for the random DFA and the triple-parity DFA.

The results in the above table show that network pruning can be an effective tool for improving the performance of the network generalization and for reducing the size of the extracted FMM. We observe that the perfect FMM with 64 states was not extracted for any training method in any of the runs shown and one might thus conclude that the 64 state machine cannot be

successfully learned; however, this is not the case. When we chose all strings of length up to 11 (4096 strings) as the training set, we were able to extract the perfect DFA in all cases, regardless of the training method; since no significant differences in network generalization performance was found for the different methods when the larger training set was used, we chose to train with a smaller training set.

5 CONCLUSIONS

We have presented a destructive method for improving the generalization performance of recurrent neural networks, trained to recognize strings of regular languages. Our simulation results demonstrate that pruning combined with retraining can significantly improve the performance of the networks themselves and also of the extracted symbolic rules. The pruning procedure is a repetitive cycle of reducing the size of the architecture and retraining the network. Our method is based on a simple heuristic which assesses the relevance of recurrent state neurons according to the magnitude of the incoming weights. State neurons with small weights tend to contribute less to the overall computation and thus are promising pruning candidates. The pruned network needs to be retrained to achieve its performance prior to the pruning step. As to be expected, the retraining becomes harder as the size of the network decreases; the performance improves while generally using fewer strings than were necessary to train the original network. Our preliminary results where the generalization performance improves by an order of magnitude using a simple pruning heuristic are encouraging. The performance improvements of pruned networks are generally superior to networks trained with weight decay; training is faster due to the shrinking network size and there is no need for determining a decay rate prior to training. We found that these improvements also hold true for large finite-memory machines (64 states) [10]. It would be interesting to compare our method with other weight pruning methods ([3, 8]).

An open question is whether this pruning method produces the smallest network necessary to learn (or represent) the deterministic finite automata (DFA) to be learned. Certainly for the triple parity DFA, a 3-state neural network is the smallest possible if the neuron state activations are confined to the rails of the sigmoid. But the 10-state random DFA should have had a 4 state recurrent network. However, that did not occur; training failed to converge. It would be interesting to see if knowledge inserted into the network before or during training [4, 7, 13] aids or impedes the pruning process.

References

[1] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient-descent is difficult," *IEEE Transactions on Neural*

Networks, vol. 5, no. 3, pp. 157–166, 1994.

[2] A. Cleeremans, D. Servan-Schreiber, and J. McClelland, "Finite state automata and simple recurrent recurrent networks," *Neural Computation*, vol. 1, no. 3, pp. 372–381, 1989.

[3] Y. L. Cun, J. Denker, and S. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems 2* (D. Touretzky, ed.), (San Mateo, CA), pp. 598–605, Morgan Kaufmann Publishers, 1990.

[4] P. Frasconi, M. Gori, M. Maggini, and G. Soda, "A unified approach for integrating explicit knowledge and learning by example in recurrent networks," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, p. 811, IEEE 91CH3049-4, 1991.

[5] C. Giles, C. Miller, D. Chen, H. Chen, G. Sun, and Y. Lee, "Learning and extracting finite state automata with second-order recurrent neural networks," *Neural Computation*, vol. 4, no. 3, p. 380, 1992.

[6] C. Giles and C. Omlin, "Extraction, insertion and refinement of symbolic rules in dynamically-driven recurrent neural networks," *Connection Science*, vol. 5, no. 3-4, pp. 307–337, 1993.

[7] C. Giles and C. Omlin, "Inserting rules into recurrent neural networks," in *Neural Networks for Signal Processing II, Proceedings of The 1992 IEEE Workshop* (S. Kung, F. Fallside, J. A. Sorenson, and C. Kamm, eds.), pp. 13–22, IEEE Press, 1992.

[8] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems 5* (S. Hanson, J. Cowan, and C. Giles, eds.), (San Mateo, CA), pp. 164–171, Morgan Kaufmann Publishers, 1993.

[9] G. Hinton, "Learning translation invariant recognition in a massively parallel network," in *PARLE: Parallel Architectures and Languages Europe*, pp. 1–13, Berlin: Springer Verlag, 1987. Lecture Notes in Computer Science.

[10] Z. Kohavi, *Switching and Finite Automata Theory*. New York, NY: McGraw-Hill, Inc., second ed., 1978.

[11] A. Krogh and J. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems 4* (J. Moody, S. Hanson, and R. Lippmann, eds.), (San Mateo, CA), pp. 950–957, Morgan Kaufmann Publishers, 1992.

[12] J. Pollack, "The induction of dynamical recognizers," *Machine Learning*, vol. 7, pp. 227–252, 1991.

[13] J. W. Shavlik, "A framework of combining symbolic and neural learning," Tech. Rep. TR 1123, Computer Sciences Dept., Computer Sciences Dept, U of Wisconson - Madison, 1992.

[14] R. Watrous and G. Kuhn, "Induction of finite-state languages using second-order recurrent networks," *Neural Computation*, vol. 4, no. 3, p. 406, 1992.

[15] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.

[16] Z. Zeng, R. Goodman, and P. Smyth, "Learning finite state machines with self-clustering recurrent networks," *Neural Computation*, vol. 5, no. 6, pp. 976–990, 1993.

Analysis of Satellite Imagery Using a Neural Network Based Terrain Classifier*

Michael P. Perrone and **Michael J. Larkin**
Institute for Brain and Neural Systems
Campus Box 1843
Brown University
Providence, RI 02912
(401)863-3920
Fax:(401)863-3494
mpp@cns.brown.edu
and
Prometheus Inc.
21 Arnold Ave.
Newport, RI 02840
(401)849-5389
Fax:(401)848-7293
larkin@dam.brown.edu

Abstract

We present a novel method of detecting changes, such as erosion or deforestation, from time sequential pairs of remote images. After preprocessing the images and obtaining a difference image, we use a neural network-based system to adaptively threshold the difference image and resolve areas of pixel intensity with a terrain classifier which combines information in the original images. The result is that we detect precisely the types of changes in which we are interested, without being "distracted" by changes due to noise or natural within-terrain variability of pixel intensity.

1 Introduction

The objective of our research has been to design an automated system for detecting changes in the environment, based upon time sequential remote

*This work was supported in part by Army Contract No. DACA76-93-C-0005, under subcontract to SEA CORP and by the Office of Naval Research, the Army Research Office, and the National Science Foundation.

sensor images of the same area. Our approach was to apply image processing techniques to the original digital images in order to compensate as much as possible for errors due to registration (i.e., a given pixel in the second image does not necessarily correspond to the pixel in the identical position in the first image), as well as variations in pixel intensity due to illumination changes, clouds, and certain natural variabilities inherent in certain types of terrain that are not of importance for analysis purposes. At the same time, it is recognized that preprocessing will not necessarily correct all of these errors, so the system was designed to be robust to errors due to registration or pixel intensity variability, as well as other types of noise.

The basic premise is to take the two images and subtract one from the other, creating a difference image. Ideally, any non-zero pixel intensities in the difference image would indicate that a change in the environment had occurred. Of course, the problems of registration and other types of noise will also result in contributions to the difference image. Also, there will be certain types of changes in the image that are characteristic of certain types of textural terrain (trees or grasses, for instance) that are not of much interest. Thus, the problem is to determine what features in the difference image are representative of meaningful changes in the environment, such as deforestation, erosion or pollution; and which features are due to noise or various pixel intensity variabilities.

Our system runs a window over the difference image and computes the average pixel intensity within the window. If the pixel intensity exceeds a given threshold, the corresponding windows in the two preprocessed original images are compared, through the use of a neural network based terrain classifier. As described in the following sections, this system determines if any change has occurred in the window based on the results of the terrain classifier.

2 Overview of Algorithm

This section outlines the basic steps of our algorithm. These steps are detailed in subsequent sections.

- Preprocessing - image registration and normalization
- Generate smoothed difference image
- For each pixel above a fixed threshold, classify the texture in the corresponding regions from both preprocessed images.
- A pixel is interesting if the texture classifications differ.
- If the ratio of interesting to uninteresting pixels in a given region is greater than some threshold, then the region is interesting.

3 Image Preprocessing

This section describes the image preprocessing required to prepare the images for input into our classification algorithm. The goal of preprocessing is to bring the images into registration and to match local pixel intensities. We achieve this with the methods outlined below.

3.1 Registration Algorithm

Without registration, there may be little of no relation between the pixels of a one image and the corresponding pixels of another image. In particular, this will be the case when the camera is not in exactly the same location and orientation when each image is taken. One possible registration algorithm is outlined below.

- Select several regions from one image to be used as fiducial marks. Ideally, 3 fiducial marks are sufficient for to adjust for any translation, rotation and linear scaling but more may be used to increase accuracy in noise environments.
- Find the best match in the second image for each fiducial region. It may be necessary to locally normalize the intensity of the image regions for an appropriate match to be found.
- Use the fiducial mark matches to determine the appropriate scale, rotation and offsets between the two images such that the following linear transformation holds between the pixel locations in each image.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

3.2 Normalization Algorithm

In general, images will have varying degrees of illumination. If one image is taken at noon and another is taken at dusk, image subregions which are identical may have widely varying pixel intensities. Similar problems may arise from snow, ice, leaves, etc. To avoid these problems, we normalize each image such that they both have pixel intensities with zero mean and unit variance. In certain terrains where average illumination may vary over the image (e.g. shadows produced by mountains or tall buidings) it will be necessary to perform local normalization.

4 Difference Image

One fundamental aspect of our algorithm is the difference image which, in its simplest form, is the difference between pixel intensities in the overlapping

regions of the two images. If the images are identical, the difference image should be all zeroes. We impose the constraint that the algorithm should be insensitive to the order in which the two images are presented; therefore we define the difference image as

$$D_{ij} = |A_{ij} - B_{ij}| \tag{1}$$

where A_{ij} and B_{ij} are the pixel intensities in the ith row and jth column of images A and B respectively. Note that it is necessary to map the images onto the same grid if any rotation or scale transform is used for registration..

4.1 Image Smoothing

Difference images tend to be very noisy due to natural variations from image to image and "ghosting" that can occur due to poor registration. In order to ameliorate these problems, we convolve our difference images with a square indicator function. Thus the pixel value in the smoothed image is given by

$$D_{ij}^{\text{smoothed}} = \sum_{lm} k(i-l, j-m) D_{lm} \tag{2}$$

where $k(l,m) = 1$ when $|l| < r$ and $|m| < r$ and $k(l,m) = 0$ otherwise. We can adjust the amount of smoothing by varying the radius, r, of k. We can also approximate Gaussian smoothing by repeated convolution with k. Note also that this smoothing can be applied to the classifications given by the texture classifiers.

4.2 Pixel Intensity Histograms

The amounts of smoothing and thresholding needed for accurate detection of variations within an image can be suggested by examining histograms of the pixel values of a given image. We consider several histograms in our work, including histograms of the preprocessed, differenced, and smoothed images. In the preprocessed images, one typically has a smooth distribution of pixel values which are nearly identical for both images while difference images typically have a bimodal distribution. Smoothing over difference images results in a main peak in the pixel histogram corresponding to zero difference and minor peaks in the tails corresponding to more interesting pixels (See Figure 1).

5 Adaptive Thresholding

At the heart of our environmental change detection algorithm is an adaptive threshold which uses information from both the difference image and the texture classifier to filter out uninteresting regions of the images.

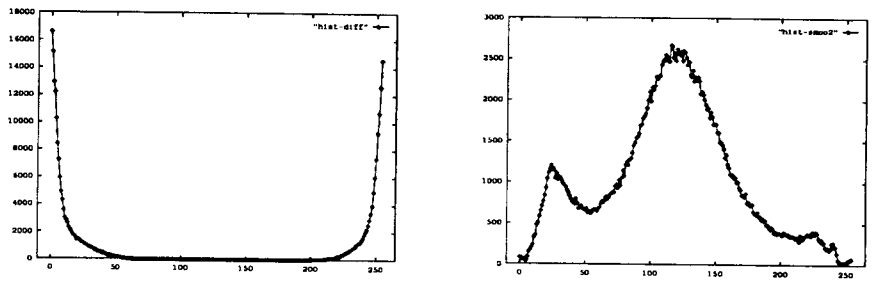


Figure 1: The left graph shows the pixel histogram of the difference image. The right graph shows the pixel histogram of the smoothed difference image. Useful information is contained in the tails of the histogram on the right.

5.1 Algorithm for Detecting Differences Between Images

Heuristically, the decision criterion for flagging a windowed region as being interesting can be stated as follows:

- If the average pixel intensity (API) of the difference image is very low, the difference is not significant/interesting.
- If the API is high and the classifications from the different images are different, the difference is significant/interesting.
- If the API is high but the classifications are identical, the difference is not significant.

We can improve on the algorithm by including a sensor fusion center (neural net based) that will learn when the three inputs are significant and when they are not. Thus we can make our thresholding nonlinear and more robust.
Ideally, we could say that for all pixels $d_{ij} \in D$, a level of intensity greater than zero indicates a change in the scene being imaged. However, due to natural variations in the imaged objects or terrain a certain level of pixel variability is expected. It is therefore necessary to identify an optimal threshold to determine whether a pixel value in the difference image is significant. We determine these values from images where known changes have been located and quantified.
Because it is unlikely that every region of the image will have the same optimal threshold, we use a neural network approach to identify various classes of regions from a given corpus of images for which different optimal thresholds can be determined. The neural networks were used to determine which "terrains" in the difference image are interesting and which are not. Once the neural networks are trained, they are used to determine what terrain class

a particular region belongs to. With this information, we can use a specialized threshold to determine whether an observed variation in the images is of significance. The advantage to this approach is that the system is more sensitive where small variations are important and less sensitive where they are not, resulting in more changes being detected and less "false alarms", or changes that are detected which have no importance.

5.2 Pattern Classifiers

In this section, we consider two image classifiers designed to identify terrain/texture class in subregions of the images: The KNN algorithm and the RCE algorithm. The training input to these algorithms are hand-labelled subimages of a fixed size. We refer to these subimages as data vectors.
We note here that there exist other neural network algorithms which could also be applied to the task of terrain classification.

5.2.1 The KNN Algorithm

The K Nearest Neighbor (KNN) algorithm [Duda and Hart, 1973] functions by finding the nearest K vectors from our previously labelled data vectors to a new data vector for which the terrain class is unknown. The classification for the new data vector is given by the majority class of the K nearest neighbors. The distance metric that is used in this algorithm is not essential and for high dimensional spaces an l_1-norm is generally sufficient as well as being faster to calculate than most other norms.

5.2.2 The RCE Algorithm

The Reduced Coulomb Energy (RCE) algorithm [Reilly et al., 1982] creates networks of neurons with bounded activity function given by

$$n_i(\vec{x}) = 1 - \Theta(||\vec{x} - \vec{m}_i||_2 - t_i) \tag{3}$$

where $\Theta(\cdot)$ is a step function. Thus the activity of RCE neuron i is 1 if the input is within a distance t_i of $\vec{m}_i$ and 0 otherwise. Classification of a given input is determined by choosing the class of memories which has the largest total output. In its simplest version, the RCE algorithm builds a network in the following manner. For each memory in the data set:

1) If the classification is correct, make no changes.

2) If the network activity is zero (no classification), add a new neuron to the network using the new memory as the center and set the neuron's threshold equal to the distance to the nearest memory of a different class.

3) If the classification is incorrect or confused,

a) Shrink the thresholds of the neurons which were responsible for the error.

b) Pass the memory through the network again.

This process is repeated until the network stops changing. Given enough resources, this algorithm can cover arbitrarily complex boundaries between classes for a deterministic classification problem.

6 Application

A version of the algorithm described in the preceding sections was implemented on real satellite images and the results are presented below. From Figures 2, 3, 4 and 5, it can be see that our algorithm can correctly select the regions of a photographed area which have changed.
This research is continuing. Further results will be presented at the conference on different images and more elaborate classification algorithms.

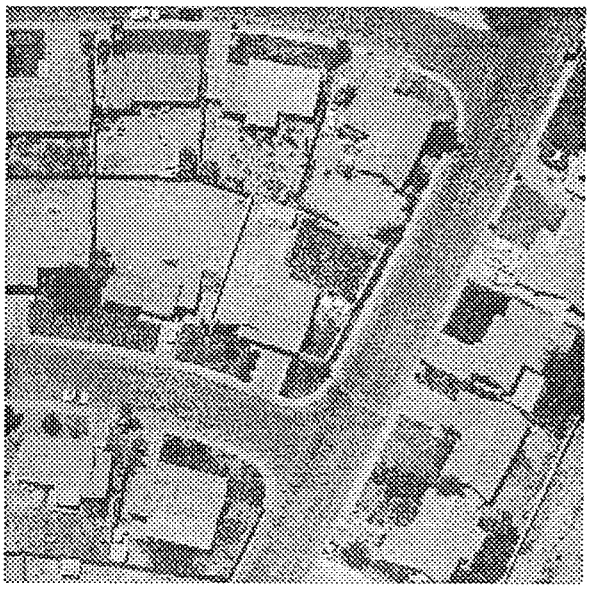


Figure 2: A small region of a real satellite photograph.

References

[Duda and Hart, 1973] Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. John Wiley, New York.

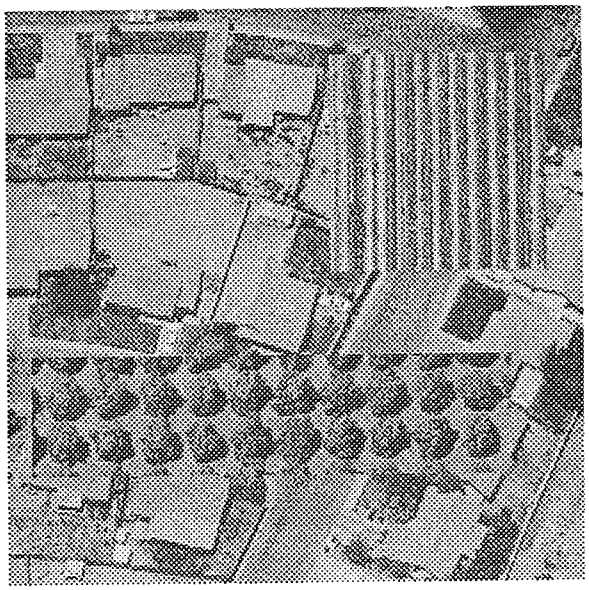


Figure 3: This image shows the same as the previous figure slightly offset to simulate registration error and with an orchard and field "planted" where houses and streets exist in the original image.

[Reilly et al., 1982] Reilly, D. L., Cooper, L. N., and Elbaum, C. (1982). A neural model for category learning. *Biological Cybernetics*, 45:35–41.

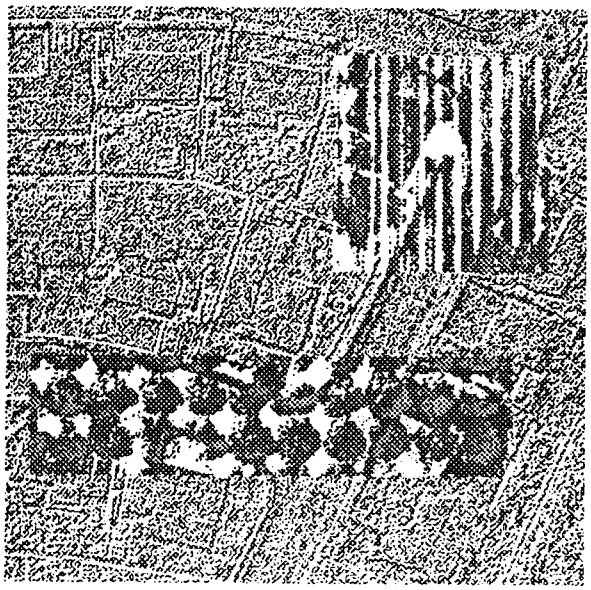


Figure 4: The difference image.

Figure 5: The regions labelled "interesting" by our algorithm.

NEURAL-NETWORK BASED CLASSIFICATION OF LASER-DOPPLER FLOWMETRY SIGNALS

N. G. Panagiotidis, A. Delopoulos and S. D. Kollias
National Technical University of Athens
Department of Electrical and Computer Engineering
Computer Science Division
Zografou 15773, Athens, Greece

Abstract: Laser Doppler flowmetry is a most advantageous technique for non-invasive patient monitoring. Based on the Doppler principle, signals corresponding to blood flow are generated, and metrics corresponding to healthy vs. patient samples are extracted. A neural-network based classifier for these metrics is proposed. The signals are initially filtered, and transformed into the frequency domain through third-order correlation and bispectrum estimation. The pictorial representation of the correlations is subsequently routed into a neural network based MLP classifier, which is described in detail. Finally, experimental results demonstrating the efficiency of the proposed scheme are presented.

INTRODUCTION

Laser-Doppler flowmetry (LDF) is a noninvasive method for semi-quantitative assessment of microcirculation currently applied in the fields of angiology, cardiology, vascular surgery neurology and physiology [1]. Its easy handling lead to its widespread clinical use in acquiring relevant information on the microcirculation. LDF appears to offer substantial advantages over other methods in the measurement of cutaneous blood flow. Studies have shown that it is not only highly sensitive and responsive to regional blood perfusion,

0-7803-2026-3/94 $4.00 © 1994 IEEE

but also versatile and easy to use for continuous noninvasive patient monitoring.

In principle, LDF is an optical technique for estimation of micro-circulation, based on the Doppler principle. When the laser beam is directed toward the issue. reflection transmission and absorption occur. Laser light backscattered from moving particles, such as red cells, is shifted in frequency according to the Doppler principle, while radiation backscattered from non-moving structures remains at the same frequency. Even though Laser-Doppler flowmeters are easy to use, sources of variation need to be known and taken into consideration.

An interesting aspect in the processing of LD signals is the extraction of appropriate parameters and the classification of signals to categories, e.g. corresponding to healthy and patient samples. In this paper, we propose a classification scheme with bispectrum analysis for extracting useful features of the LDF signal, and neural networks for classification of the extracted information.

BISPECTRUM ANALYSIS

As an initial step for the LDF biomedical signals are subjected to the following preprocessing :

First, the original signal is decomposed into three components, consisting of the trend ($\leq 20mH$), component 2 ($\approx 20mH - 800mH$) and component 3 ($\geq 800mH$). This step has proved more useful for the preprocessing of the signals and particularly bispectral analysis. A FIR low-pass Hamming filter (25-taps) was used for the detection of the trend, which allows attenuation of the artifacts or abrupt and brief changes in the signals. The second component is obtained through subtraction of the trend from the original signal and additional low-pass filtering. The effect of linear phase delay is subtracted from the resulting signal.

Let $x(t)$ be a real two-dimensional signal with support $S = [0...N-1] \times [0...N-1]$. Its triple correlation is defined as ,

$$x_3(\tau_1,\tau_2) = \frac{1}{N^2}\sum_S x(t)x(t+\tau_1)x(t+\tau_2)$$

where τ_1,τ_2 are defined in $S' = [-(N-1),...,(N-1)] \times [-(N-1),...,(N-1)]$

In general, we can move indistinguishably from the signal domain to the triple correlation domain without loss of information or, in other words, we can distinguish two signals by comparing their triple correlations.

Third-order signal correlations and their Fourier transforms i.e. the corresponding bispectra are higher-order statistics with two important properties [2].

→ In contrast to second order correlations, triple correlations of *deterministic* signals have a one-to-one correspondence with the original signal (except of a shift ambiguity).

→ Third-order-correlations of zero-mean non-skewed noise (such as Gaussian or linear and symmetrically distributed) are zero in the mean, and furthermore, they tend to zero w.p. 1 as the size of the available data record tends to infinity.

The first property generally yields a complete description of the signal, based on its triple-correlation. On the other hand, the second property can be used under certain conditions, to improve SNR in applications where the signal under consideration is corrupted by non-skewed additive noise. Based on their properties, third-order correlations can be very advantageous for image recognition, leading to invariant representation of the input images with respect to scale, rotation and translation.

The bispectrum $X_3(u,v)$ of a signal $x(t)$ is computed as

$$X_3(u,v) = X(u)X(v)X(-u-v)$$

where $X(u,v)$ is the Fourier transform of $x(t)$. As a consequence, $X_3(u,v)$ can be computed as the triple product of FFTs using fast software or hardware implementations.

The final step of pre-processing consists of computing the absolute values of the resulting bispectra. Sample plots of these values are shown in (Figure 1, Figure 2, Figure 3). The first one corresponds to a signal obtained from a healthy volunteer, whereas the other two to signals obtained from patients suffering from arterial occlusion.

The volunteers bispectra appear to have frequency components coupled to a certain pair of frequencies. On the contrary, the patients bispectra do not include such regular structures and tend to have several mutually coupled frequencies. In this paper, we use a neural network architecture to classify the LD-images, based on the aforementioned observations.

PROPOSED NEURAL CLASSIFIER

Multilayer perceptrons have been widely examined in the neural network field, as a tool for signal classification, based on the extraction of appropriate features from signals [4]. Error-feedback supervised learning algorithms, such as backpropagation, are generally used to train a multilayer feed-forward neural network. A crucial aspect concerning the network performance is generalisation i.e. the ability of a network to classify correctly input data which were not included in its training set. Good generalisation is a result of appropriate network design; a small number of interconnection weights (i.e. free parameters during training) should generally be used for this purpose, and any a-priori knowledge about the problem should be included in the network architecture. Consequently, structured networks of small size are likely to have better generalisation. Our architecture consists of a multilayer feed-forward perceptron, whose inputs are described below.

The LD signals bi-spectra are processed as grayscale images. Since the size of the images is quite large, we chose to decompose them into images of lower size, using a multiresolution decomposition scheme described below.

Let x_0 denote an $N \times N$ image representation. Using appropriate reconstruction FIR filters $h_l(n)$ and $h_h(n)$, where $h_l(n)$ generally is a low-pass and $h_h(n)$ a high-pass filter, we can split the image into four $(N/2 \times N/2)$ images. Applying for example the low-pass filter $h_l(n)$ in the horizontal and then vertical direction of the original image (we consider the separable case for simplicity) we get the *approximation* image at the lower resolution level $j=-1$ denoted as

$$x_{-1}^{LL}(m,n) = \sum_{k=1}^{N}\sum_{l=1}^{N} h_l(2m-k)h_l(2n-l)x_0(k,l)$$

By applying all other possible combinations of the above FIR filters, we get three lower resolution *detail* images, denoted as $x_{-1}^{LH}, x_{-1}^{HL}, x_{-1}^{HH}$. Moreover, if the above procedure is successively applied to the approximation images, we have a *multiresolution approximation* of the original image, providing images of continuously decreasing size.

The resulting low-resolution (LR) approximation image is used as input to the classifier. Furthermore, in order to exploit useful information included in the detail images, we extract from them several features, especially the number of pixels with non-zero values at each detail level. These pixels generally correspond to non-zero frequency couples in the original image content.

The LR images are fed to the first hidden layer MLP, which is of a receptive field type, while the extracted features as well as the output of the first layer are subsequently fed to a second hidden

layer. The output of the second layer is fed to the final layer, the output of which constitutes the result of our classifier.

After training with data obtained both from signals corresponding to healthy persons and patients, our classifier was fed with bispectra obtained by LD-signals. The results were most satisfactory,including a correct classification rate of 93%. Sample bispectra that were successfully classified are shown in (Figure 4, Figure 5).

Further research and experiments are currently performed using extended data sets, as well as refinements to the pre-processing methodology and fine-tuning of the proposed neural network classifier architecture.

ACKNOWLEDGEMENTS

Part of the research presented in this paper was done for the BIOMED-1 project *"Laser Doppler Flowmetry for Microcirculation Monitoring"*, 1993-1995.

REFERENCES

[1] Oberg, P.A. , *"Laser Doppler Flowmetry"*, Biomed Eng. , 18:125-163, 1990.

[2] Delopoulos A., Tirakis A., Kollias S., *"Invariant Image Classification using Triple Correlation Based Neural Networks "*, IEEE Trans. Neural Networks, May 1994.

[3] Mallat S. *"A theory for multi-resolution signal decomposition : the wavelets representation "*, IEEE Trans PAMI vol 11, pp 674-692, 1989.

[4] Mendel J. *" Tutorial in higher-order statistics (spectra) in signal processing and systems theory : theoretical results and some applications "*, IEEE proc. vol. 79, pp. 278-305, 1991.

[5] Tirakis A. "*Optimal Filter Banks for Multiresolution Image Analysis* ", Ph. D. Thesis, N.T.U.A. , 1994.

[6] Pao Y. H., "*Adaptive Pattern Recognition and Neural Networks* ", Addison-Wesley, 1989.

SAMPLE LD-plots

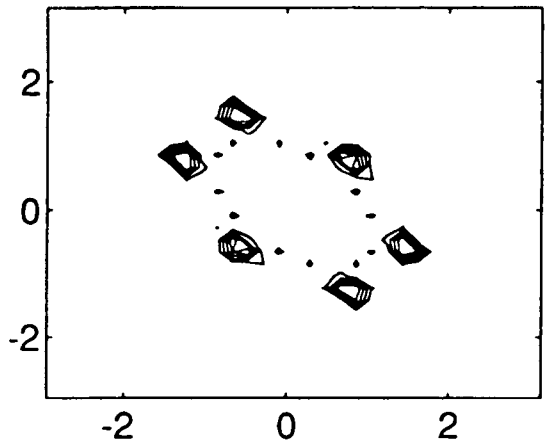


Figure 1 (Volunteer/Healthy)

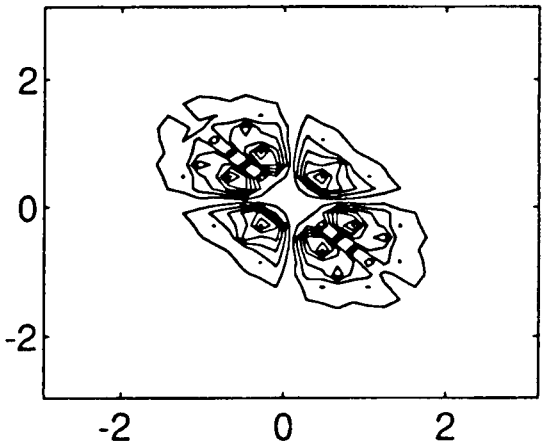


Figure 2 (Volunteer/Patient)

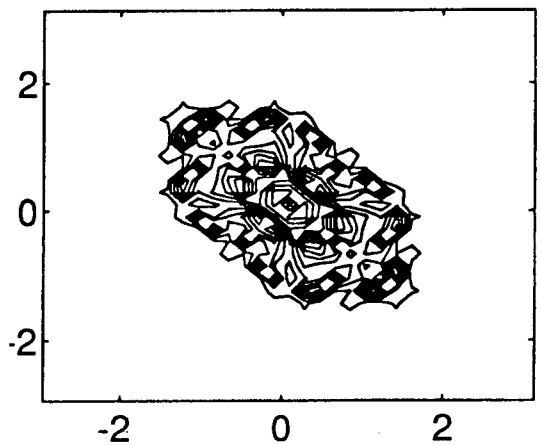


Figure 3 (Volunteer/Patient)

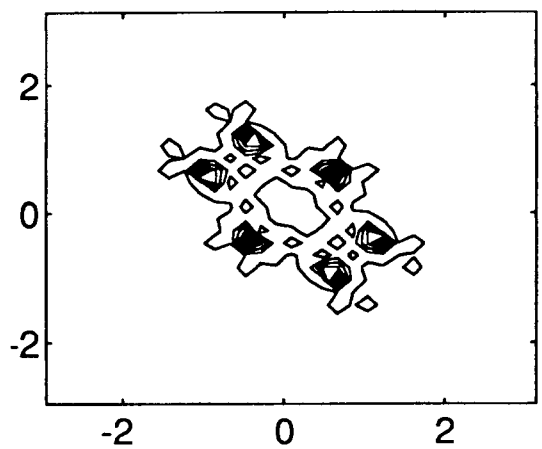


Figure 4

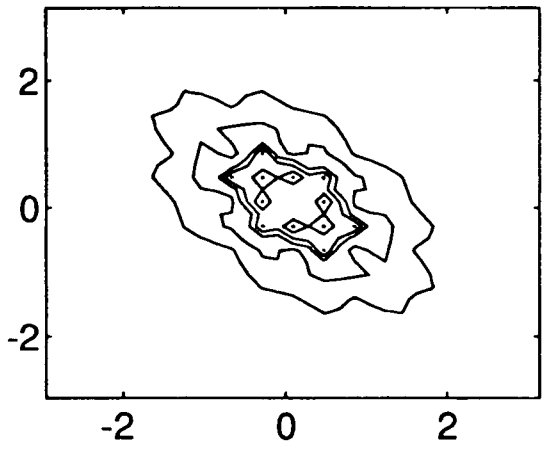


Figure 5

Conference Author Index

A

Akita, R. M. 451
Alberti, M. 555
Albesano, D. 241
Alder, M. 375
Anderson, J. S. 596
Andreano, K. J. 394
Andrews, M. 441
Attikiouzel, Y. 375
Autret, Y. 384

B

Bachmann, C. M. 394
Back, A. 146
Back, A. D. 565
Beet, S. W. 319
Bellesi, G. 309
Benidir, M. 573
Birgmeier, M. 527
Birkett, A. N. 249
Blekas, K. 163
Bors, A. G. 105
Bouras, D. P. 535
Bourlard, H. 289
Bracco, C. 573
Burrows, T. L. 117
Busch, C. 461

C

Cancelliere, R. 241
Castellanos, J. 587
Çelebi 155
Charokopos, N. 482
Chiu, M. Y. 413
Clothiaux, E. E. 394
Comley, R. 681
Comley, R. A. 671
Constantinides, A. G. 431, 473
Cook, G. D. 269

D

Delopoulos, A. 709
Desai, U. B. 88
deSilva, C. J. S. 375
Dorffner, G. 499
Dreyfus, G. 229
Dumitras, A. 606

E

Economopoulos, N. M. 482
Economou, G. -P. K. 482

F

Farrell, K. 279
Fechner, T. 187
Finton, D. J. 52
Fog, T. 616

G

Gemello, R. 241
Giles, C. L. 690
Goodman, R. M. 219
Gori, M. 309
Goubran, R. A. 249
Goutis, C. E. 482
Grumstup, P. 490

H

Hadjiagapis, S. 204
Hadjiprocopis, A. 681
Hansen, L. K. 42, 78, 490, 509, 616
Haralambopulu, E. 482
Hochberg, M. M. 269
Holm, S. 616
Hu, Y. H. 52
Hwang, J. N. 22

J

Johnsen, J. 490

K

Kadirkamanathan, V. 12
Kaski, K. 641
Kasper, K. 335
Katagiri, S. 259, 352
Kechriotis, G. I. 545
Kevrekidis, I. G. 596
Kitsonas, M. 204
Koller, H. 499
Kollias, S. D. 709
Komori, T. 352
Kosonocky, S. 279
Kung, S. Y. 413
Kuo, J. M. 661
Kurimo, M. 362

L

Laddad, R. R. 88
Lakkos, S. 681
Larkin, M. J. 700
Larsen, J. 42, 78
Lastrucci, L. 309
Law, I. 616
Lawrence, S. 146
Lazarescu, V. 606
Leitgeb, E. 499
Lendaris, G. G. 451
Likas, A. 163
Lim, G. 375
Lin, J. N. 126
Linneberg, C. 509
Liu, C. C. 423
Lonardi, S. 651
Luong, D. Q. 394
Lymberopoulos, D. 482

M

Madiraju, S. V. R. 423
Maglaveras, N. 518
Makrakis, D. 535
Mammone, R. 279
Mana, F. 241
Manolakos, E. S. 545
Marcos, S. 573
Martinelli, G. 32
Mascioli, F. M. F. 32
Mathiopoulos, P. T. 535
Matsuura, Y. 329
McDermott, E. 259
Meyrowitsch, J. 509
Mirghafori, N. 289
Miyazawa, H. 329
Moakes, P. A. 319
Moore, J. W. 394
Morgan, N. 289
Murgan, A. T. 606
Mylonas, S. A. 671

N

Nakano, R. 69
Nielsen, L. H. 616
Niranjan, M. 117
Nummonda, T. 441

O

Ojala, P. 641
Omlin, C. W. 690

P

Panagiotidis, N. G. 709
Pap, R. M. 451
Papageorgiou, C. 204
Pappas, C. 518
Parente, E. 98
Pattichis, C. S. 431
Paulson, O. 616
Pazos, A. 587
Pedreira, C. E. 98
Perrone, M. P. 700
Personnaz, L. 229
Pitas, I. 105
Poonacha, P. G. 88
Poopalasingam, S. 633
Prakash, S. R. 345
Principe, J. 155
Principe, J. C. 661
Procházka, A. 195

R

Rabavilas, A. 204
Rasmussen, C. E. 78
Reeves, C. R. 633
Reininger, H. 335
Renals, S. J. 269
Rico-Martínez, R. 596
Rios, J. 587
Robinson, A. J. 177, 269
Roussel-Ragot, P. 229

S

Saarinen, J. 641
Saeks, R. E. 451
Salamon, P. 509
Seibert, F. 461
Sekhar, C. C. 345
Skinner, T. E. 329
Smith, P. 681
Soda, G. 309
Solaiman, B. 384
Sørensen, J. A. 171
Sperduti, A. 651
Spiliopoulou, M. 482
Spriopoulos, C. 482
Stafylopatis, A. 163
Stamkopoulos, T. 518
Starita, A. 651
Stathaki, T. 473
Steele, N. C. 633
Stefanis, C. 204
Strintzis, M. 518
Svarer, C. 78, 509, 616
Sys, V. 195

T

Tanger, R. 187
Taur, J. S. 413
Terman, D. 136
Thomas, C. R. 451
Torkkola, K. 299
Tsoi, A. C. 146, 565
Tufts, D. W. 61

U

Ueda, N. 69
Unbehauen, R. 126
Urbani, D. 229
Uzunoglu, N. 204

V

Van Hulle, M. M. 3
Ventouras, E. 204
von Spreckelsen, S. 490

W

Wan, E. A. 146
Wang, C. J. 22
Wang, D. 136, 624
Waterhouse, S. R. 177
Wen, W. 209
Willett, D. 461
Wilson, E. 61
Wolf, D. 335
Wong, Y. F. 404
Wüst, H. 335

Y

Yegnanarayana, B. 345
Yokoo, M. 209

Z

Zafra, J. L. 587
Zeng, Z. 219