(4)

Technical Document 1333
August 1988

An Adaptive Hull-to-Emitter Correlator

C. E. Priebe
D. J. Marchette

ADMINISTRATIVE INFORMATION

Information in this report originally was prepared by members of the Architecture and Applied Research Branch (NOSC Code 421) to be presented as a professional paper and published in the proceedings for the Data Fusion Symposium - 1988 held at Johns Hopkins University, Baltimore, Maryland, 17 to 19 May 1988.

Released by
V. J. Monteleon, Head
Architecture and Applied Research Branch

Under authority of
J. A. Salzmann, Jr., Head
Information Systems Division

LO

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA199912

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NOSC Technical Document 1333	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center — 6b. OFFICE SYMBOL (if applicable) Code 421	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) San Diego, CA 92152-5000	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Ocean Systems Center — 8b. OFFICE SYMBOL (if applicable) Block Programs	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	AGENCY ACCESSION NO.
62232N	RC32S11	CDB2	DN306 242

11. TITLE (include Security Classification)
AN ADAPTIVE HULL-TO-EMITTER CORRELATOR

12. PERSONAL AUTHOR(S)
C. E. Priebe and D. J. Marchette

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM TO	August 1988	13

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
command control
data fusion

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper explains a modification of a statistical technique that retains the distributed nature of an Adaptive Network System, while allowing the network to learn to approximate the probability distribution of the data.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL C. E. Priebe	22b. TELEPHONE (include Area Code) (619) 553-4048 — 22c. OFFICE SYMBOL Code 421

DD FORM 1473, 84 JAN

83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

An Adaptive Hull-to-Emitter Correlator

Carey Priebe
David Marchette

Command and Control Department
Code 421
Naval Ocean Systems Center
San Diego, Ca. 92152

INTRODUCTION

One of the functions of a data fusion system involves identifying the platforms reported in Electronic Intelligence ~~(Elint)~~ data. These reports contain a set of parameters measured from the platform's radar. The assignment of a name to this radar report is called Hull-to-Emitter Correlation (hultec).

In [4] the authors described an Adaptive Network System (ANS) (popularly refered to as "neural network") approach to this problem. The particular paradigm used was a backpropagation network. This system classifies the data into linearly separable classes, using a gradient descent, least mean squared error algorithm to determine the separating hyperplanes. The network returns a value for each hull that gives a measure of its distance from a point in the teaching set (actually it's a measure of the distance from the separating hyperplane). This approach compared favorably to traditional statistical techniques. One problem with this approach is that the output of the network, though it ranges continuously from zero to one, has no relationship to probability. Although the network gives a ranking for the hulls it has learned, the value associated with any one hull gives little information. It is only the value relative to the other hulls that can be used in further decision making.

This paper deals with a modification of a statistical technique that retains the distributed nature of the ANS, while allowing the network to learn to approximate the probability distribution of the data. This not only allows the network the potential of better performance, but it allows other systems to use the output of the network in a probablistic sense.

THE HULTEC PROBLEM

For the purposes of this paper, a report is a set of parameters measured from a radar signal: PRI (pulse repetition interval), Scan (scan rate), RF (frequency), etc. The hultec problem is to determine the emitter of origin for the report, based on these measured parameters. By its nature, this is a difficult problem. Errors in measurement are bound to occur, so the system requires a measure of fault tolerance. Some sensors report different parameters than others, requiring the system to be able to handle missing data. The parameter spaces of several emitters may overlap, so the system must respond with a ranked list of emitters, rather than a single emitter.

The main problem with the backpropagation approach investigated in [4] is that it attempts to assign a single hull to the report. Often, as mentioned above, the distributions of the hulls are not disjoint. It is therefor not always possible or desirable to assign a report to a single hull. Instead, one would like to provide a list of the possible hulls with some measure of the liklihood of classification associated with each hull. One way to do this is to model the probability density function associated with each hull. In other words, the reports define a sample space for each of the classes (hulls), and the reports associated with each class define a probability distribution for that class over the sample space.

An improvement over the ANS approach described in [4] is one which can learn the probability density functions for the classes. This is the approach taken by the network described below. The output from the net is the value of the probability density functions for all the hulls evaluated on the report. The value output for a given hull, in contrast to that given by the backpropagation scheme, has a meaning in an absolute sense as well as relative to the values output by other hulls. The network can indicate how certain it is that a given input stimulus belongs to a given class, thereby indicating, for example, that the second place choice for classification is very high indeed and should not be disregarded. This gives more information than a winner-take-all approach, and this information could be used in conjunction with other information (such as a priori information, data from other sensors, etc.) to improve overall performance.

Another important advantage that a probability density estimator gives over a winner-take-all classifier is the ability to detect unknown emitters. When the output is low for all the emitters known by the system, it is likely (though not certain) that the report comes from an emitter that is unknown to the system. The ability to recognize new emitters can be as important as the correct classification of known emitters, and a system that can continue to cluster reports from this new emitter is of great value to a data fusion system.

KERNEL ESTIMATORS

The problem of probability density estimation has been attacked by many different methods. One of the more successful is the technique of kernel estimation [5]. Consider a one dimensional problem, with a single class. The goal is to aproximate the probability density function for the teaching data, which consists of a collection of points drawn from an unknown distribution. In its simplest form, the kernel estimator involves assigning to each of the teaching points a gaussian distribution, the kernel, with mean equal to the input point and a fixed variance, called the window width. This produces, as a partitioning of the input space, a Voronoi diagram [1], the optimal nearest neighbor partitioning. These gaussians are summed, weighted by the window width and the number of teaching points. Therefor, the kernel estimator with kernel K is defined by

$$f(x) = \frac{1}{nh} \Sigma K \left(\frac{x - X_i}{h} \right) \qquad (1)$$

where h is the window width. Distributions other than the normal may be used for the kernel K, but for the purposes of this paper, the kernel will always be a normal distribution.

The ANS architecture proposed, an implementation of the kernel estimator, assigns a node in the hidden layer for each gaussian (see Figure 1). The node is connected to the input layer, with the weights set to the mean of the gaussian. These connections do a subtraction, rather than the traditional multiplication of their signal, and this signal is then summed by the node and the kernel is applied. In the case of a single dimensional input the node retains a variance, while in the multidimensional case, the node may retain a covariance matrix, or it may keep a single variance.

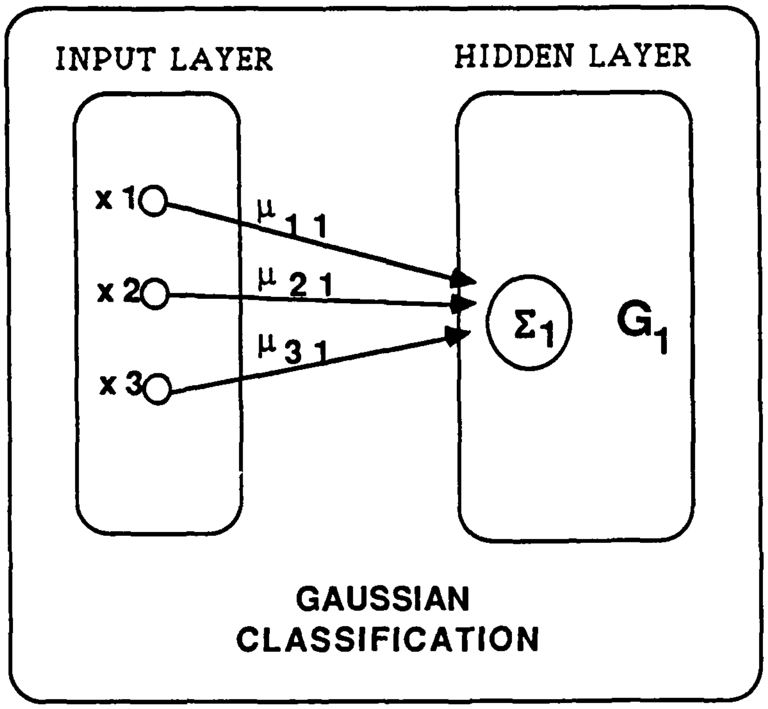

FIGURE 1

Representation of Gaussians

The gaussians are presented with the n-dimensional input from the input layer having n nodes and independently compute their respective activation values (gaussian distances) from this input. The activation value for these gaussian nodes in the hidden layer can be obtained via the following formula:

$$G_j(x) = \frac{\exp(-0.5[(\underline{x} - \underline{\mu})^t . \Sigma^{-1} . (\underline{x} - \underline{\mu})])}{(2\pi)^{(d/2)} * |\Sigma|^{(1/2)}} \qquad (2)$$

Here $G_j(\underline{x})$ is the activation value of the jth gaussian node G_j when presented with vector input $\underline{x}$. Σ is the covariance matrix for gaussian node G_j, while $\underline{\mu}$ is the vector-valued mean for this gaussian. d is the dimensionality of this particular gaussian and is equated with the dimensionality of the input layer, i.e. the number of parameters in the report. Since the components of the mean $\underline{\mu}$ are represented as a node's input weights, these weights can be thought of as shifting the origin for the node. A gaussian function is then applied to the inputs of the node.

If the matrix Σ is diagonal, then (2) is just the product of d independant gaussians, one for each input. This observation gives a method for coping with missing data: "drop" the appropriate gaussian from the product. In (2) this corresponds to assigning the missing input the value at the mean for that component. The system will continue to work as if the data were there. This approach gives an overly optimistic output for the report, but the ranking will be reasonable.

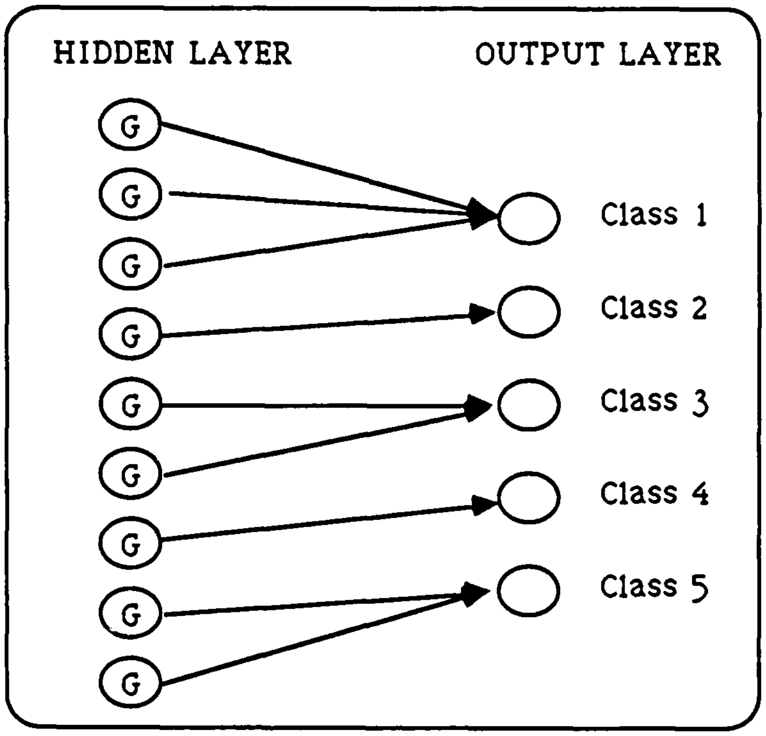

FIGURE 2

Combination of Gaussians

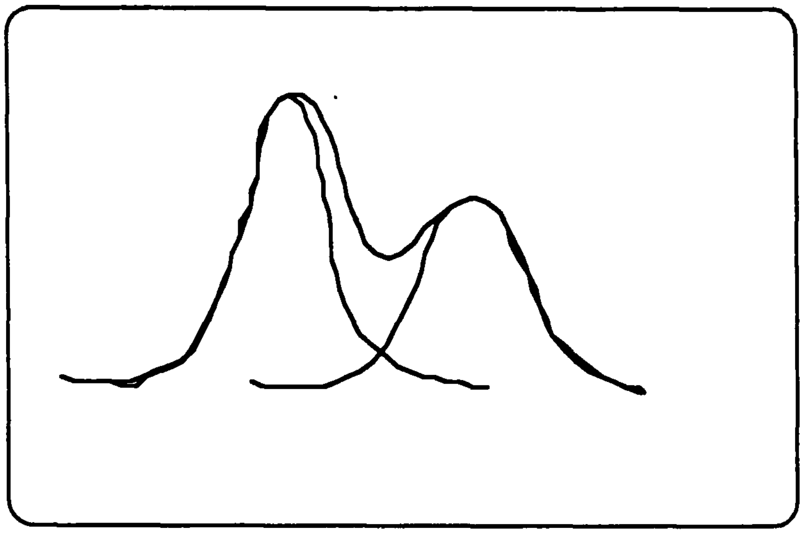

FIGURE 3

Sum of Gaussians

The outputs from the hidden layer are summed, weighted by the number of points in the node, to produce a final answer (see figure 2). There is one node on the output layer per class. If there is more than one class stored within the network, the nodes on the final layer are connected only to the nodes corresponding to their class. This method of combining gaussians to produce an overall distribution provides the system with enormous flexibility. Although the initial kernels being used are predetermined to be gaussians, many widely varying distributions can be approximated by a sum of gaussians (see Figure 3). The overall distribution for a particular class is not, therefor, assumed to be normal. In this manner the network can be used as a classifier, giving output that is meaningful in a probabilistic sense.

One difficulty with the kernel estimator is the choice of window width. A width that is too wide will produce a unimodal approximation, while a narrow choice can produce a multimodal distribution with inherent spikes. The width can be chosen experimentally, or one of the many adaptive kernel estimation algorithms may be employed (see [5]). A modification of the kernel estimator which allows the window to adapt to the data is described below.

MOVING MEAN AND VARIANCE

The kernel estimator described above is static once the initial learning is finished. One way to modify this scheme to produce a dynamic system is to allow the nodes to vary their means and covariance matrix as the data is recieved.

Only those nodes whose activation values reach some threshold defined by that node's activation function (usually the value of the gaussian at a fixed multiple of one standard deviation from the mean) are considered to be a likely category for the current input and are updated. This updating consists of moving the mean and variance of the gaussian based on the current input and some measure of the total number of inputs to the gaussian thus far. For the updating of the mean, we have

$$\mu(j+1) = \mu(j) + [1/(j+1)][I(j+1) - \mu(j)] \qquad (3)$$

where $\mu(j)$ is the (one-dimensional) mean after the j^{th} input and $I(j+1)$ is the $j+1^{st}$ input to be categorized in this gaussian. The updating of the covariance is similar. Here

$$S_{xy}(j+1) = S_{xy}(j) + [j/(j+1)] * A_{xy}(j+1) \qquad (4)$$

with

$$A_{xy}(j+1) = (x(j+1) - \mu_x(j)) * (y(j+1) - \mu_y(j)) \qquad (5)$$

Then

$$\Sigma_{xy}(j+1) = \frac{1}{j} S_{xy}(j+1) \qquad (6)$$

where we are calculating $\Sigma_{xy}(j+1)$, the x,y component of the covariance matrix Σ after the $j+1^{st}$ input clustered in this Gaussian. x(j) and y(j) are the x and y component of the input vector after the j^{th} input, and μ_x, μ_y are components of the mean. These formulas require the node to store j, the number of points in the node. This value provides the weight used in the sum of nodes computed for the final layer: j divided by the total number of points in the class. In this manner, the nodes corresponding to clusters near the mean of the distribution are weighted higher than nodes corresponding to outliers.

This approach can also be made to choose the number of nodes in the hidden layer. When a point comes in, the hidden nodes all evaluate their response to the input. If none of the nodes fire above a threshold, it is assumed that none of the existing nodes properly categorizes the input and a new node is created. Otherwise, all nodes of the appropriate class that fired above a threshold are allowed to adjust their parameters.

One of the advantages of this system is that it is dynamic. If the distribution of the data is drifting with time, the network's distribution will also drift.

The kernel method, and the moving parameters method, can be modified to allow the network to continue in an unsupervised manner. This allows the system to continue to learn, even though truth data may not be available. This is important in many Navy applications, as truth data is often hard to come by, and a system that clusters like entities may be the best that one can hope for.

The idea behind the unsupervised learning rule is that if the system has no a priori knowledge pertaining to the class of the input, it should update each class proportionaly to the likelihood that the input comes from that class. This is indicated in equations 7-10. Equations 7 and 8 are the usual update rule for the mean (note that this is written in a less efficient but conceptually simpler form than equation 3):

$$\mu_{new} = \frac{j*\mu_{old} + x}{j + 1} \qquad (7)$$

$$j = j + 1 \qquad (8)$$

Here j is the number of points, and x is the input. In the case of an unknown input, the value of the distribution is used to weight the amount of change:

$$\mu_{new} = \frac{j*\mu_{old} + f(x)*x}{j + f(x)} \qquad (9)$$

$$j = j + f(x) \qquad (10)$$

A similar rule is used for the variance. The variable j is now a floating point number rather than an integer, but the two approaches are very similar in their implementations. This is essentially the Bayesian approach described in [7].

The unsupervised learning rule allows the system to continue to learn when truth data is unavailable, producing clusters of "like" reports. A typical use of this rule would be to first teach the system on known inputs using the supervised learning rule, then allow the system to monitor data, modifying its distributions using the unsupervised learning rule.

RESULTS

The kernel estimator was compared against CARI (Computer Assisted Radar Identification) [3], a classification program based on standard statistical techniques, on a data set consisting of 4019 points taken from 63 hulls. The parameters reported are PRI and Scan. The two systems were cross-validated using this data set: each system is taught on 4018 reports and tested on the missing report. If the hull with the largest value on the output corresponds to the hull from the missing report, the system is said to be correct. If the second highest output corresponds to the correct hull, the system gets a second place, and if the third highest is correct the system gets a third place. This is repeated until the system has been tested on all the reports in the data set.

The results are tabulated below. Note that the kernel estimator is slightly better on this data set. Preliminary indications are that the adaptive mean and covariance approach can in some cases improve on the kernel estimator, but in general it requires more points to construct its estimate.

Method	Correct	Second	Third
CARI	79%	16%	3%
Kernel	82%	12%	3%

These implementations were run on a PC AT compatible computer. It should be noted that the backpropagation algorithm described in [4] could not be cross-validated on this data set, even using a Cray, due to the prohibitive processing requirements of the learning algorithm. Although this is irrelevant for the finished system (after teaching), it is an important consideration for a system intended to be dynamic, and hence continually learning.

CONCLUSIONS

The classification scheme described herein developes a statistical model of the input entering the system. In addition, any a priori knowledge about the distribution of the incoming data can be incorporated into the system. This yields a powerful, dynamic classification tool possessing the advantages inherent in network methodologies and yet is readily understood via statistical analysis. The system can operate in both a supervised and unsupervised mode, and can handle missing as well as noisy data.

The gaussian nodes which are integral to the system are independent of one another. This allows each node to independently perform its processing based on its inputs and its activation function. In keeping with traditional neural network schemes, this yields an easily parallelizable system. Each node can be thought of as an individual, independent, primitive processor. The nodes in various layers of the net are then linked through weighted connections to nodes in the previous layer. Thus the processing time in a hardware implementation is independent of the number of nodes required to estimate the density, and it is independent of the number of hulls in the system.

Another aspect of the system's dynamic nature is the adaptive network size. Within hardware constraints, gaussian nodes can be allocated dynamically, as dictated by the statistics of the incoming data. Under certain learning conditions, many gaussian nodes may be necessary to capture the salient features of the data. At other times, the actual distributions being modeled may resemble unimodal gaussian functions, thereby requiring very few gaussian nodes.

It may be useful to consider a more mature system in which information entering the system at different times from multiple reports can be used to strengthen the classification process. A temporal classification system employing the gaussian scheme described herein has been developed [6]. While one report may not give enough information for a classification, a track of several reports may give more information. Another application of this approach would be to look at the signal itself rather than measured parameters of the signal, and classify it based on the temporal nature of the signal.

REFERENCES

1. Aggerwal, Chazelle & Guibas, "Parallel Computational Geometry", (1985), *Proc. of the 26th IEEE Symp. on the Foundations of Comp. Sci.*, 468-477.

2. Chan, Golub & LeVeque, "Algorithms for computing the sample variance: analysis and recommendations", (1983), *The American Statistician*, (37) 242-247.

3. Kramer, "A Survey of New Techniques for processing Electronic Intelligence", (1987), *1987 Tri-Service Data Fusion Symposium*, Vol. II, 1-16.

4. Marchette & Priebe, "An Application of Neural Nets to a Data Fusion Problem", (1987), *1987 Tri-Service Data Fusion Symposium*, Vol. I, 230-235.

5. Silverman, Density Estimation , Chapman and Hall, 1986.

6. Sung & Priebe, "Temporal Information Processing", (1988), Submitted to *1988 Int'l Conf. on Neural Networks*.

7. Titterington, "Updating a Diagnostic System using Unconfirmed Cases", (1976), *Appl. Statistics*, 25, 238-247.