ESD-TR-87-281

2

DTIC FILE COPY

Technical Report
805

AD-A197 976

The Boltzmann Machine: A Survey and Generalization

M.D. Eggers

8 July 1988

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

Prepared for the Department of the Air Force
under Electronic Systems Division Contract F19628-85-C-0002.

88 8 15 116

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology with the support of the Department of the Air Force under Contract F19628-85-C-0002.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Hugh L. Southall

Hugh L. Southall, Lt. Col., USAF
Chief, ESD Lincoln Laboratory Project Office

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

THE BOLTZMANN MACHINE: A SURVEY AND GENERALIZATION

M.D. EGGERS
Group 93

TECHNICAL REPORT 805

8 JULY 1988

LEXINGTON MASSACHUSETTS

ABSTRACT

A tutorial is presented describing a general machine learning theory which spawns a class of energy minimizing machines useful in model identification, optimization, and associative memory. Special realizations of the theory include the Boltzmann machine and the Hopfield neural network. The theory is reinforced by appendices addressing particular facets of the machine, ranging from gradient descent to simulated annealing.

The treatment is systematic, beginning with the description of the energy function. A defining relationship is established between the energy function and the optimal solution. Following, both classical and new learning algorithms are presented (directing the adaption of the free parameters) for numerically minimizing such function to yield the optimal solution. Finally, both computational burden and performance are assessed for several small-scale applications to date.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

THE BOLTZMANN MACHINE

A Survey and Generalization

1. INTRODUCTION

Man is forever developing new approaches to solving the increasing problems which confront our modern society. Some of the simpler problems can be formulated in conventional mathematical constructs, with accompanying direct solutions. However, for many of the problems of real practical interest, direct solutions are often time consuming and suffer from ill-conditioning. The recent trend to solving problems of practical interest which often require computation efforts that increase exponentially with the size of the problem (NP-complete class problems) involves producing a timely estimate of the solution. That is, a solution is sought which is the best, given finite resources. Such resources include computation time, complexity, and the amount of beforehand (*a priori*) information.

Specifically, the type of problems applicable to the solution techniques presented herein are those upon which an objective or cost function can be mathematically defined to represent the merit of an estimated solution. Hence, the cost of a particular solution is a quantitative measure of how well the estimated solution conforms to the desired problem goals, subject to finite resources. As will be demonstrated, this cost function approach is applicable to a broad class of problems.

1.1 ORIGINATION

The Boltzmann machine, originated by Hinton and Sejnowski [1], is a computational algorithm for solving problems having objectives represented by a cost function. The machine iteratively alters the free parameters according to a prescribed algorithm until the cost function is minimized, corresponding to finding the optimal solution (optimal in terms of the cost function.) The iterative manner in which the system seeks the optimal solution is the distinguishing characteristic of the Boltzmann machine, and most influential in the naming.

The machine is named after Ludwig Boltzmann who discovered the fundamental law governing the equilibrium state of a gas. The general principal derived from Boltzmann's work and exploited in the present context is simply that a network consisting of a large number of units, with each unit interacting with neighboring units in a restricted region, will approach a canonical distribution at equilibrium (steady state) given by [2,3]

$$p(\bar{s}) = \frac{e^{-\beta E(\bar{s})}}{\int_{\text{all states}} e^{-\beta E(\bar{s})} d\bar{s}} \tag{1}$$

where $E(\bar{s})$ is the energy of the system in the global state $\bar{s}$.

The compatibility of Boltzmann's contributions to iterative problem solving becomes clear once several analogies are made. First, the energy function is actually the cost function of an estimated solution. Furthermore, any estimated solution is represented in the machine by a particular global state $\bar{s}$, where $\bar{s}$ is a vector containing each of the unit values. A second analogy concerns the interpretation of the Boltzmann distribution (1). Clearly the distribution favors the lower energy states at equilibrium. That is, as the process reaches equilibrium, the most probable global state configurations will be those of low energy. By analogy, when the Boltzmann machine through a prescribed iteration process reaches steady state (equilibrium), the most probable solutions (most probable global states) are those of low cost (low energy) or equivalently those near optimal. This overall analogy between numerous molecules approaching equilibrium and a machine comprised of a large number of interacting processing units reaching steady state is diagrammed in Figure 1.

A premier consequence of applying Boltzmann's result in thermodynamics to iterative problem solving is the resulting algorithm relies only on information from neighboring units for the determination of the free parameters. For example in Figure 1, learning the connection strength w_{ji} only requires knowing the outputs of processing units s_i and s_j. Hence the iteration process involves numerous *local* adaptions which can be performed in parallel to seek the optimal solution by minimizing the *global* cost function.

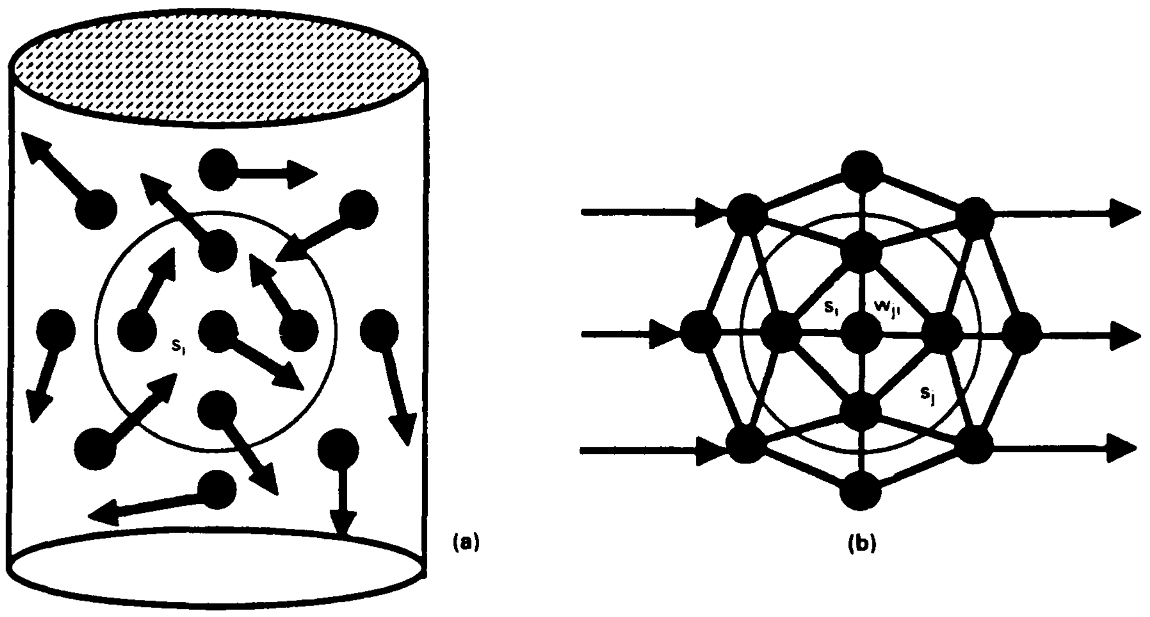


Figure 1. Boltzmann analogy: (a) gas molecules reaching equilbrium: molecule i in state s_i governed by neighboring potential and kinetic energy interactions. (b) processing units in machine reaching steady state: unit i in state s_i governed by neighboring correlated activity interactions.

1.2 ORGANIZATION

Towards a clear presentation, the organization of the material is easily understood with the diagram in Figure 2. The treatment begins with the description and role played by the cost or energy function. This function is generic to all Boltzmann machine applications and hence provides a common foundation. The next section defines the general machine as being composed of processing units and interconnections as illustrated. Now depending upon the application, the connection strengths are specified either by an iterative learning process or an assignment rule given a-priori. The iterative learning is described in the Machine Learning section. Here a general learning framework is presented with special cases yielding the original Boltzmann machine and other versions employed in optimization and associative memory. Following the learning algorithms, applications and implementations are discussed in the Machine Operation section for model identification, optimization, and associative memory. Finally, the conclusion provides an overall assessment of the machine.

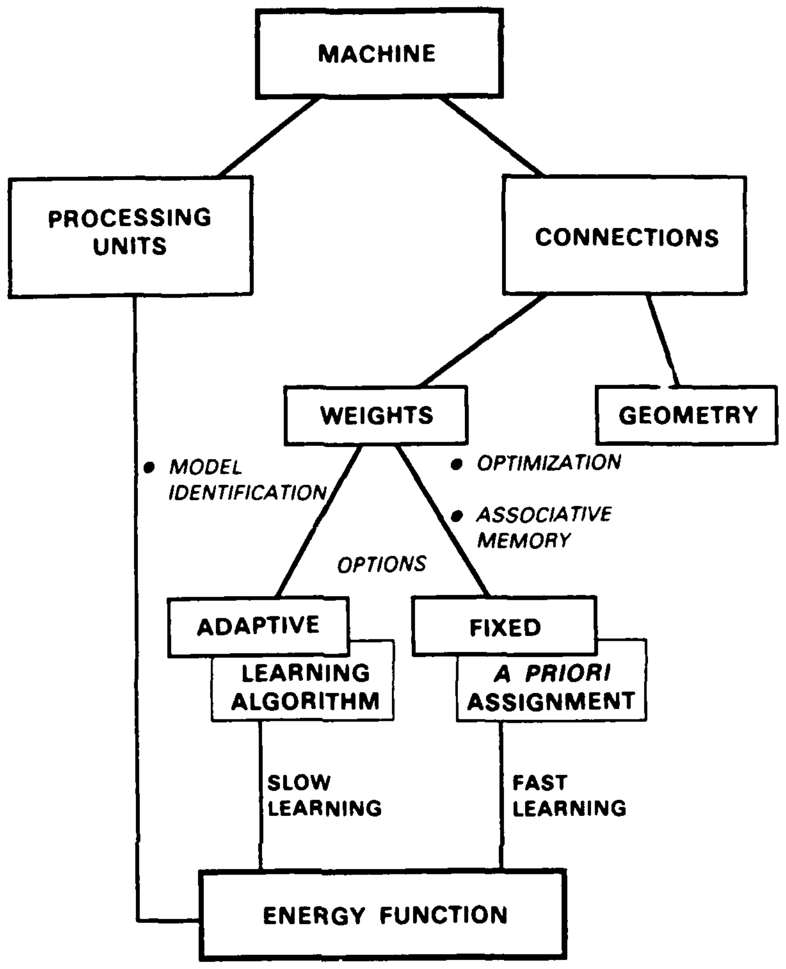


Figure 2. Learning machine organization.

2. MACHINE ENERGY FUNCTION

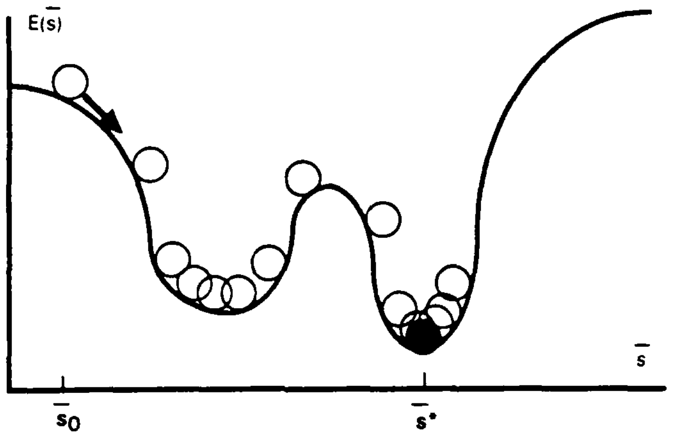
The ideal objective of the Boltzmann machine is to find the optimal solution to a given problem. The purpose of this section is to relate the meaning of optimal to the cost or energy function. Furthermore the energy function is examined for each of the mentioned applications. Thus with both the energy function specification and the relationship to the optimal solution, the true meaning of the optimal solution becomes apparent for each of the problem solving applications addressed. Additional application information follows in Section 5.

Central to the problem solving applications of the Boltzmann machine is the energy function. This function $E(\bar{s})$ is simply a measure of how well a particular solution $\bar{s}$ conforms to the objectives of the desired solution. Typically the function is chosen with lowest energy designating best conformance. Moreover since any machine solution $\bar{s}$ can be expressed as a function of the machines free parameters, the energy function is clearly dependent upon such parameters.

Now the energy function is related to the optimal solution simply by definition. A solution is defined optimal if the solution satisfies the desired objectives (represented mathematically by the energy function) better than any other solution the machine can generate. Therefore the objective of the Boltzmann machine can be viewed as minimizing the energy function with respect to the variable machine parameters. Under suitable conditions the minimization will generate a set of machine parameters which will yield a solution with lowest energy, corresponding to best conformance.

The machine seeks the optimal solution by searching the free parameter space in an iterative fashion, diagrammed in Figure 3. Unlike most other minimization techniques, provisions are provided in the Boltzmann machine to avoid local minima in the operation phase (following learning). In effect these provisions allow for occasional uphill moves, where the energy actually increases to avoid local minima.

Certainly specifying an energy function which mathematically incorporates the desired objectives of the problem solution can be challenging. The selection is often governed by intuition and tractability. Fortunately for the applications addressed, tractable energy expressions have been constructed with much intuitive appeal as demonstrated.



94420-3

Figure 3. Conceptual diagram of optimization search.

Beginning with model identification, the goal is to build a model (by specifying the free parameters of the model) to replicate the implicit behavior in an observed process. Specifically in the Boltzmann machine context, the machine is configured to learn the underlying constraints of an observed random process. That is, given training data comprised of realizations and corresponding probabilities, the machine iteratively alters the free parameters to ultimately form a model which generates similar realizations as those observed.

If the stochastic data is represented by a probability distribution $p(\bar{v})$ over the quantities observed, and the machine is represented by a probability distribution $q(\bar{v})$ over the corresponding units, then an information theoretic measure of discrepancy between the data and model is [4]

$$G(p, q) = -\int_V p(\bar{v}) \log \frac{q(\bar{v})}{p(\bar{v})} d\bar{v} \quad . \qquad (2)$$

Clearly this distance measure is zero if the model generates realizations with the exact distribution found in the observed data. Consequently, for model identification, G is to be minimized with respect to the free parameters available in the model. And the optimal solution corresponds to the set of parameters which yield a model distribution closest in measure (2) to the stochastic data distribution.

In optimization, the goal is to solve a given problem by minimizing or maximizing a cost function of several variables which incorporates the objectives of the desired solution. Typically the cost function, serving as the energy function for the Boltzmann machine, is comprised of optimization terms and limiting terms. The limiting terms serve to penalize solutions outside the boundary conditions. For example from a computer design context, where numerous circuits must be partitioned on several chips, the optimization term may be the number of signals crossing the chip boundaries. The limiting term may represent the imbalance from the partitioning, for it is desirable to divide the circuits about equally amongst the chips. Thus the optimal solution minimizes the number of signals crossing chip boundaries (and hence minimizes propagation delay) while simultaneously avoiding imbalance. Such a cost function for the optimal partitioning of N circuits between two chips is given by Kirkpatrick, Gelatt and Vecchi [5]

$$E(\bar{s}) = \sum_{i>j} \left(\lambda - \frac{a_{ij}}{2}\right) s_i s_j \tag{3}$$

where

a_{ij} = number of signals passing between circuits i and j

$$s_i = \begin{cases} 1 & \text{circuit } i \text{ is placed on chip 1} \\ -1 & \text{circuit } j \text{ is placed on chip 2} \end{cases}$$

λ = imbalance coefficient selected by user.

Another example of optimization is the traveling salesman problem where a route is planned such that every city is reached at least once. Here the cost function is simply the total distance traveled, and the optimal solution corresponds to the shortest route.

The final application is associative memory, where the goal is to retrieve memory vectors given an input vector representing a partial or incomplete que. The cost function is fabricated such that the memories $\{\bar{z}_i\}_{i=1}^{M}$ are local minima, as diagrammed in Figure 4. A specific cost function utilized in the Hopfield associative memory is [6]

$$E(\bar{s}) = -\frac{1}{2}\sum_i \sum_j T_{ij}\, s_i\, s_j \tag{4}$$

where

$$T_{ij} = \sum_l \left(2z_l^i - 1\right)\left(2z_l^j - 1\right) \tag{5}$$

and the optimal solution corresponds to the memory closest to the key pattern $\bar{s}_o$.

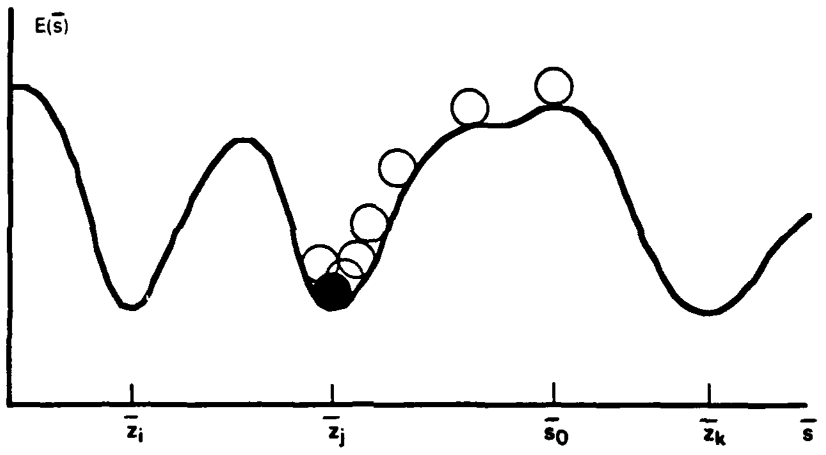


Figure 4. Energy function shown with minima (memories) at system states $\{\bar{z}_i\}_{i=1}^{M}$. The process is initiated with que $\bar{s}_o$, and then follows the energy landscape in the direction of decreasing energy until a local minima is found.

94420-4

These are just a few examples of the cost functions employed in optimal problem solving. Reiterating the key concept, the optimal solution is defined as the solution minimizing the cost (energy) function. The following section describes the machine architecture which serves as a vehicle for minimizing such function.

3. MACHINE DEFINITION

In general, parallel computation machines such as the Boltzmann machine consist of processing units and interconnections (see Figure 2). A general architecture is presented which reduces to the Boltzmann machine with appropriate restrictions. This section describes the processing units, then follows with the interconnections. The general system is referred to as simply the machine, while the specific realization by Hinton and Sejnowski [1] is appropriately termed the Boltzmann machine.

The processing units are grouped into visible and hidden units. The visible units serve as the interface between the machine and the environment and are typically divided into input and output units. The hidden units are additional units necessary in learning more complex relationships found in the training data, which can not be modeled by simple pairwise correlations amongst visible units. In effect, the hidden units form a representation internal to the machine, thought to capture significant underlying features in the data.

The units can be represented in many ways. An input/output format is shown below with the visible units represented by a vector comprised of both input and output vectors.

The second representation shown in Figure 6, simply collects all units into a single vector which represents the global state $\bar{s}$ of the machine. The values of the units are typically binary although analog values can be utilized.

The interconnections with designated strengths serve as the communication links between the processing units. Now depending on the complexity of the machine, the interconnections may range from simple pairwise connections (per Figures 5 and 6) to M-ultple connections, where each combination of M processing units are coupled together with an associated connection strength. The connection geometries together with the notation are displayed in Figure 7. Specifically for the original Boltzmann machine[1], the connections are pairwise, symmetric ($w_{ij} = w_{ji}$), and real valued.

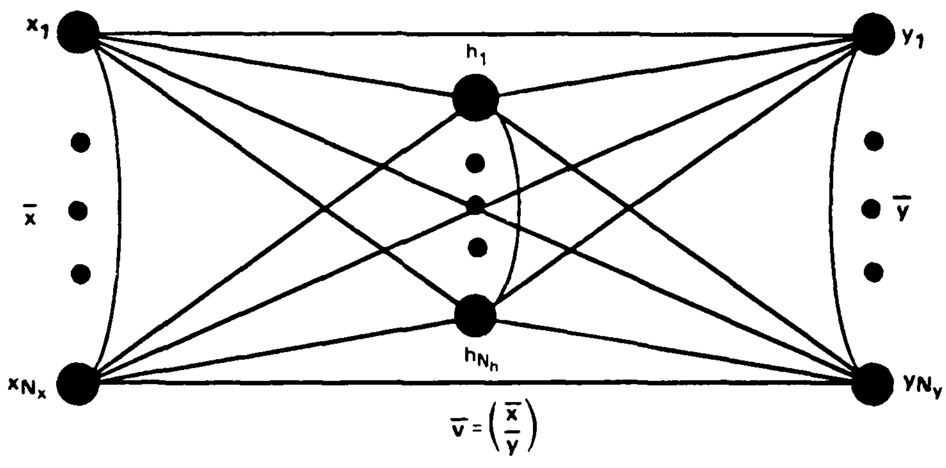


Figure 5. Input/output representation.

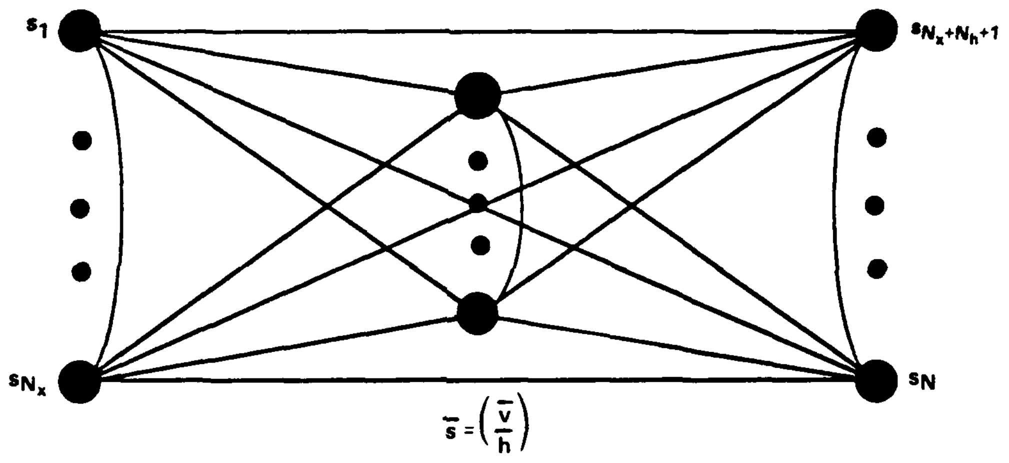

s_1
$s_{N_x+N_h+1}$
s_{N_x}
s_N
$\bar{s} = \begin{pmatrix} \bar{v} \\ \bar{h} \end{pmatrix}$
94420-6

Figure 6. Global state representation.

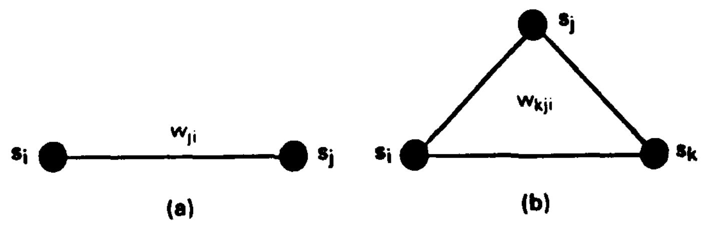


Figure 7. Interconnection geometries: (a) pairwise, (b) 3-tuple.

On a more subliminal level, the interconnections of variable strength actually represent implicit knowledge acquired during the learning phase. The following section explains how the training is conducted, or equivalently, how knowledge is acquired from the data giving rise to values for the connection strengths. Consequently, the machine is specified by the processing units and the interconnection geometry and strengths.

4. MACHINE LEARNING

The process of specifying the interconnection strengths (weights) is referred to as learning. The learning procedure depends upon the application. For some applications such as optimization and associative memory the learning is immediate, whereby weights are fixed *a priori* along with the energy function. However, for model identification the learning is slow, wherein the weights are iteratively adapted until the machine reaches equilibrium (steady-state.) This section treats slow learning, in which a learning law governing the adaption of the connection weights is derived from minimizing the learning cost function. Subsequently, the learning law is implemented numerically by simulated annealing.

4.1 THEORETICAL LEARNING FORMULA

The objective of the machine in the model identification context is to adjust the free parameters (weights) such that the machine models the observed realizations (data) in the best possible manner. The best or optimal model is defined as the model that minimizes the discrepancy measure between the model and data. Thus the discrepancy measure serves as the cost function. Consequently the objective of learning can be stated as adjusting the weights to minimize the cost function, thereby yielding a model which replicates the behavior implicit in the observed realizations of a stochastic process.

Assume the observed data has a distribution $p(\bar{v})$ over some accessible (visible) units $\bar{v}$. Now $p(\bar{v})$ may be known explicitly, but most often sample realizations of $p(\bar{v})$ are given in the input/output form with an associated weighting $p(\bar{v}_i)$ as shown

$$\bar{v}_1 = \begin{pmatrix} \bar{x}_1 \\ \bar{y}_1 \end{pmatrix} \Leftrightarrow p(\bar{v}_1)$$

$$\bar{v}_2 = \begin{pmatrix} \bar{x}_2 \\ \bar{y}_2 \end{pmatrix} \Leftrightarrow p(\bar{v}_2)$$

$$\vdots$$

$$\bar{v}_L = \begin{pmatrix} \bar{x}_L \\ \bar{y}_L \end{pmatrix} \Leftrightarrow p(\bar{v}_L)$$

The set $V = \{\bar{v}_i\}_{i=1}^{L}$ is the training data. Likewise the machine is represented by a probability distribution $q(\bar{v})$, also over the visible units. Now with the data and model characterized by $p(\bar{v})$ and $q(\bar{v})$ respectively, an information theoretic measure of distribution similarity is given by

$$F(q, p) = \int_V p(\bar{v}) \log \frac{q(\bar{v})}{p(\bar{v})}\, d\bar{v} \tag{6}$$

and can be viewed as a measure of relative entropy, measuring the average uncertainty associated with $p(\bar{v})$ relative to $q(\bar{v})$. Therefore maximizing (6) corresponds to maximum entropy learning. Alternatively, if the cost function is defined as the negative of the relative entropy, then the earlier expression (2) measuring the discrepancy between the model and data results. Consequently, the optimal model solution is defined as the model maximizing the relative entropy or equivalently, minimizing the cost function.

The minimization of the learning cost function (2) with respect to the connection weights is conducted by a gradient descent algorithm. The general method of gradient descent is described in Appendix 9.1. Essentially the algorithm yields a weight update rule given by

$$w_l(n+1) = w_l(n) + \Delta w_l(n) \tag{7}$$

$$\Delta w_l(n) = -\eta \frac{\partial G(\bar{w})}{\partial w_l}\bigg|_{\bar{w}(n)} \tag{8}$$

which converges to local minima, given an appropriately slow learning rate η. Consequently for minimization via gradient descent, the gradient of the cost function with respect to the connection weights must be obtained. A general derivation applicable to the Boltzmann machine as well as other energy minimizing machines is contained in Appendix 9.2. The generalization follows from employing the Gibbs distribution [(Boltzmann distribution (1) being special case)] for the machine's visible and hidden units, given by

$$q(\bar{s}) = \frac{1}{z} e^{-U(\bar{s})} \tag{9}$$

where

$$U(\bar{s}) = \sum_{l} w_l U_l(\bar{s}) \tag{10}$$

$$z = \int_{S} e^{-U(\bar{s})} d\bar{s} \quad . \tag{11}$$

Now $U(\bar{s})$ is the global potential function of the system, being a weighted superposition of the local potentials $U_l(\bar{s})$. Each $U_l(\bar{s})$ represents the local potential of the M-tuple units coupled together with a common interconnection strength w_l. Hence the local potentials are dependent only upon a small region of units. (For example for pairwise connections $U_l(\bar{s}) \propto w_{ji} s_i s_j$, and for third order connections $U_l(\bar{s}) \propto w_{kji} s_i s_j s_k$.) Also notice the global potential function is often named the energy function. However in the treatment presented, the energy function is maintained synonymous with the cost function. Thus for optimization and associative memory applications, the energy, cost and global potential potential functions are one and the same. Yet for model identification, the energy function (2) is the learning cost function and differs from the global potential function (10) as will be demonstrated.

The Gibbs distribution is both intuitively pleasing and tractable. Intuitive since the distribution only requires the units be governed by activity within a surrounding neighborhood, or precisely, the units comprise a Markov random field [7]. In neural modeling, this requirement simply states that the activity of a neuron is determined by a restricted neuronal region about such neuron. While in image processing, a given pixel is assumed correlated with other pixels confined within a neighborhood about the given location. The tractability follows from the *global* potential function being linear in the generalized connection weights w_l, resulting in a *local* learning algorithm. That is, the quantities necessary to adapt a connection strength to achieve global minimization are within proximity of the specific weight itself.

With the Gibbs distribution employed over the machine units, the resulting partial derivative of the cost function with respect to the generalized connection weight w'_l is

$$\frac{\partial G}{\partial w'_l} = E_l^+ - E_l^- \tag{12}$$

where

$$E_l^- \equiv E_{q(\bar{v},\, \bar{h})}\left\{U_l(\bar{v},\, \bar{h})\right\} = \int\limits_H \int\limits_V U_l(\bar{v},\, \bar{h}) q(\bar{v},\, \bar{h}) d\bar{v} d\bar{h} \tag{13}$$

is the expected value of the local potential $U_l(\bar{v},\, \bar{h})$ with respect to the full Gibbs machine distribution and where

$$E_l^+ \equiv E_{q(\bar{h}\,/\,\bar{v})p(\bar{v})}\left\{U_l(\bar{v},\, \bar{h})\right\} = \int\limits_V \left(\int\limits_H U_l(\bar{v},\, \bar{h}) q(\bar{h}\,/\,\bar{v}) d\bar{h} \right) p(\bar{v}) d\bar{v} \tag{14}$$

is the expected value of the local potential $U_l(\bar{v},\, \bar{h})$ with respect to the conditional Gibbs distribution $q(\bar{h}\;/\;v)$ and the true data distribution $p(\bar{v})$. Consequently a gradient descent learning algorithm (7) for adapting the generalized connection weight w'_l is obtained upon substituting (12), (13), (14) into (8) yielding

$$\Delta w'_l(n) = -\,\eta\left(E_l^+ - F_l^-\right). \tag{15}$$

Substituting the specifics of the Boltzmann machine including symmetric pairwise connections with analog values, binary units $(s_k \in \{0,\, 1\})$, and a global potential function

$$U(\bar{s}) = -\frac{1}{T} \sum_{i} \sum_{\substack{j \\ i \le j}} w'_{ji}\, s_i\, s_j \tag{16}$$

with local potential,

$$U_l(\bar{s}) = -\frac{1}{T} s_i\, s_j \tag{17}$$

the gradient (12) reduces to

$$\frac{\partial G}{\partial w'_l} = -\frac{1}{T}\left(E_{q(\bar{h}/\bar{v})p(\bar{v})}\{s_i s_j\} - E_{q(\bar{v},\bar{h})}\{s_i s_j\}\right)$$

$$= -\frac{1}{T}\left(p_{ij}^{+} - p_{ij}^{-}\right) \tag{18}$$

where

$$p_{ij}^{+} \equiv E_{q(\bar{h}/\bar{v})p(\bar{v})}\{s_i s_j\} = \sum_{k=1}^{L}\left(\sum_{H} s_i s_j q(\bar{h}/\bar{v}_k)\right)p(\bar{v}_k) \tag{19}$$

$$p_{ij}^{-} \equiv E_{q(\bar{v},\bar{h})}\{s_i s_j\} = \sum_{S} s_i s_j q(\bar{s}) \quad . \tag{20}$$

Moreover, p_{ij}^{+} represents the average probability over L training patterns that the *ith* and *jth* units are on when running the machine with the visible units fixed to the training patterns (termed clamped learning mode). While p_{ij}^{-} is the average probability that the *ith* and *jth* units are on when running the machine without any units fixed (termed free-running learning mode.) Hence the minimization of the Boltzmann machine learning cost function requires matching the average probabilities that the coupled units are firing together under clamped and free learning modes.

And in general, the theoretical learning formula involves calculating the differences of expected values of local potentials with respect to full and conditional Gibbs distributions.

4.2 NUMERICAL LEARNING ALGORITHM

To compute the adaptive weights according to the learning algorithm derived (15), expected values of functions dependent upon the unit values $\bar{s}$ must be calculated under the Gibbs distribution. Tractable closed form expressions seldom exist. As a consequence, numerical approaches are sought to compute these expected values and hence the weight updates $\Delta w'_l$.

A Monte Carlo algorithm developed by Metropolis et.al. [9] for calculating these quantities is described in Appendix 9.3. Basically, the expected values are approximated by

$$E_{q(\vec{s})}\{U_l(\vec{s})\} \cong \frac{1}{R} \sum_{m=1}^{R} U_l\,(\vec{s}_m) \tag{21}$$

where the global states $\vec{s}_m$ are selected with a probability proportional to the Gibbs distribution $q(\vec{s})$. The Metropolis algorithm was later extended by Kirkpatrick, Gelatt, and Vecchi [5] and renamed simulated annealing (see Appendix 9.4). The contribution consisted of varying the parameter which controls the selection of the states in the algorithm, namely the temperature T (63) according to a prescribed schedule analogous to chemical annealing. The variation in temperature offers a good compromise between the state selection behavior at the extremes. For at high temperatures, the global states selected for the expected value calculations (56) are randomly chosen with uniform distribution, which presents a problem since the Gibbs distribution places most of the mass over few states with low energy. On the other hand at very low temperatures, the global states selected are within a local minima region, and hence the region of low energy states may never be sampled. This sampling behavior is demonstrated in Figure 8.

A further contribution of Kirkpatrick was the realization that the Metropolis algorithm equipped with the varying temperatures (annealing schedule) actually yields a process which tends to converge to states which are deep minima of the global potential function used in the Gibbs distribution. Thus from an optimization context, a global minimum or maximum can be found by merely substituting the function to be optimized (termed the energy function) for the global potential function in the Gibbs distribution. Furthermore Geman and Geman [11] proved that under sufficiently slow annealing schedules (64), the process converges in probability to the global minimum.

Now with the Metropolis algorithm as the foundation, the simulated annealing procedure for calculating the expected values under the Gibbs distribution can be described with the following definitions.

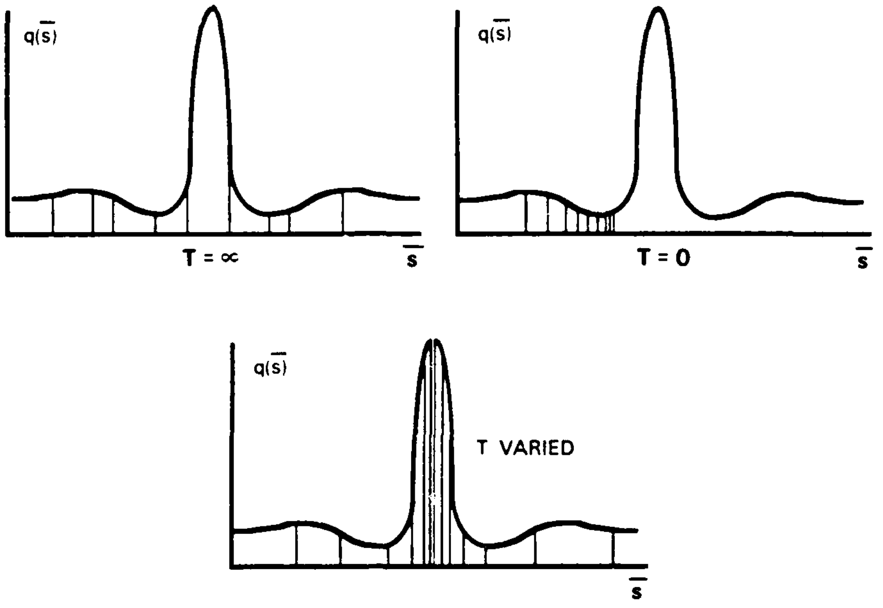


Figure 8. Sampling behavior dependent upon temperature T.

Probe: — A probe consists of first perturbing a single unit chosen at random by a small amount, and then accepting or rejecting the new system state depending upon the change in energy. Essentially this entails the first portion of the Metropolis algorithm (step 1-4 in Appendix 9.3). The distinction from the Metropolis algorithm is now the temperature T is actually a function of the iteration number. Also the new state acceptance rule utilized by Hinton [1] for binary units is slightly different from step 4 of the Metropolis algorithm. In fact because of the binary state restriction, the perturbation always entails setting the randomly chosen unit to unity with a corresponding acceptance probability

$$P\left(s_i = 1\right) = \frac{1}{1 + e^{-\Delta E \;/KT}} . \tag{22}$$

For both methods, the acceptance rules (63) and (22) are easily implemented with a uniform random number generator operating in the range [0, 1]. Defining x as the random number, the general acceptance rule (63) reduces to:

$$\Delta E \le 0 \Rightarrow \text{ accept perturbation } \bar{s} + \Delta \bar{s}_i \tag{23}$$

$$\Delta E > 0 \Rightarrow \text{ accept perturbation } \bar{s} + \Delta \bar{s}_i \Leftrightarrow x \le e^{-\Delta E / KT}$$

while the binary acceptance rule (22) reduces to

$$\text{accept perturbation } s_i = 1 \Leftrightarrow x \le \frac{1}{1 + e^{-\Delta E / KT}} . \tag{24}$$

Iteration: — An iteration consists of enough probes such that each unit is probed once on the average. Typically for N processing units, an iteration consists of N probes at a fixed temperature.

Run: — A run consists of several iterations with the temperature of each iteration specified according to the annealing schedule.

Annealing Schedule: — The annealing schedule is a list of temperatures as a function of the iteration number. The temperatures typically begin high and gradually decrease to a minimum value, per Figure 9.

The process begins with the system in a random state position. With the first temperature of the annealing schedule the units are probed enough to complete an iteration. The temperature is then adjusted according to the annealing schedule and the process is repeated until the lowest temperature is reached. At the lowest temperature, several iterations are conducted to gather the desired low energy states $\bar{s}_i$ for the expected value calculations (21). Consequently, simulated annealing is a nested iteration process, guided by the selection of the annealing schedule.

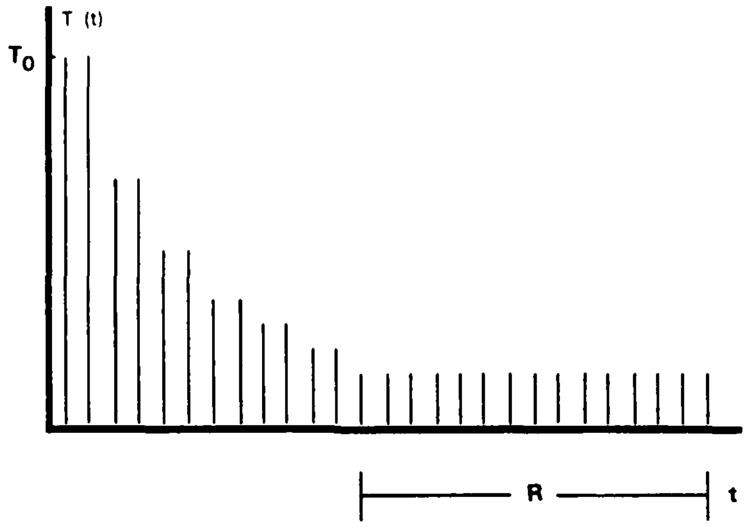


Figure 9. Annealing schedule with several iterations (R) at lowest temperature to allow statistics to be collected at equilibrium.

4.3 LEARNING PROCEDURE

The machine learning procedure based on simulated annealing is now presented. The learning procedure is shown to consist of two modes, free running and clamped. These modes are defined and the net computational burden of learning is examined.

Recall the learning formula governing the adaption of the generalized connection weight w'_l (15).

$$\Delta w'_l(n) = -\ \eta\left(E_l^{\ +} - E_l^{\ -}\right)$$

The quantity E_l^- (13) is calculated directly by the simulated annealing procedure described previously. This mode of learning is termed free-running, since given an initial global state $\bar{s}_0$, the system wanders through a variety of states allowing all units to be perturbed. Specifically for the Boltzmann machine

$$w'_l = w_{ji} \tag{25}$$

$$E_l^- = -\frac{1}{T}\, p_{ij}^- \tag{26}$$

$$p_{ij}^- = \sum_S s_i s_j\, q(\bar{s})$$

$$\cong \frac{1}{R} \sum_{m=1}^{R} s_i^{m'} s_j^{m'}$$

$$= \frac{1}{R} N_{ij}^- \tag{27}$$

where N_{ij}^- = number of occurrences in which $s_i = s_j = 1$ in the last R iterations of the annealing schedule (free running mode). Hence p_{ij}^- is simply the average probability that the *ith* and *jth* units are on at equilibrium, where the temperature remains low and fixed.

The second quantity E_l^+ is calculated also by the simulated annealing procedure, but in two stages. These stages represent the clamped mode of learning. The dual stage process arises because only the inner integral in (14), with the conditional Gibbs distribution $q(\bar{h}\ /\ \bar{v})$, can be evaluated by simulated annealing. The outer integral involves a possibly non-Gibbsian distribution $p(\bar{v})$ over the data, and hence is not suitable for simulated annealing.

The first stage in the clamped mode of learning involves applying simulated annealing to the system with the input and output units clamped (held fixed). Thus L simulated annealings are required, one for each training pattern, where only the hidden units are perturbed. Upon calculating the inner integral for each of the L training patterns applied, the second stage performs a weighted average using $p(\bar{v})$ as the weighting function to yield the resultant E_l^+. Specifically for the Boltzmann machine

$$E_l^+ = -\frac{1}{T}\, p_{ij}^+ \tag{28}$$

$$p_{ij}^+ = \sum_{k=1}^{L} \left(\sum_H s_i s_j\, q(\bar{h}\ /\ \bar{v}_k) \right) p(\bar{v}_k)$$

$$\cong \sum_{k=1}^{L} \left(\frac{1}{R} \sum_{m=1}^{R} s_i^{m'} s_j^{m'} \right) p(\bar{v}_k)$$

$$= \sum_{k=1}^{L} \left(\frac{1}{R} N_{ij}^{+}(k) \right) p(\bar{v}_k) \qquad (29)$$

where $N_{ij}^{+}(k)$ = number of occurrences in which $s_i = s_j = 1$ in the last R iterations of the annealing schedule with the system clamped to visible pattern $\bar{v}_k$. Hence, p_{ij}^{+} is simply the average probability over all L training patterns that the *ith* and *jth* units are on at equilibrium.

Consequently, the learning procedure consists of alternating between the clamped and free running learning modes, each mode utilizing simulated annealing. Depending upon how the averages for E_l^{+} and E_l^{-} are calculated, various gradient descent algorithms result. The true gradient descent algorithm (7), (8) results

$$\Delta w_l(n) = -\eta \left[-\frac{1}{T} \left\{ \sum_{k=1}^{L} \left(\frac{1}{R} N_{ij}^{+}(k) \right) p(\bar{v}_k) - \frac{1}{R} N_{ij}^{-} \right\} \right] \qquad (30)$$

requiring $L + 1$ simulated annealings, while the averaging proposed by Hinton results in a variation of gradient descent given by

$$\Delta w_l(n) = -\eta \left[-\frac{1}{T} \left\{ \sum_{k=1}^{L} \left(\frac{1}{R} N_{ij}^{+}(k) \right) p(\bar{v}_k) - \frac{1}{L} \sum_{k=1}^{L} \frac{1}{R} N_{ij}^{-}(k) \right\} \right] \qquad (31)$$

requiring $2L$ simulated annealings (since the free running statistics are also calculated over L simulated annealings indexed by k). Another variation is achieved by updating the weights following the presentation of a single training pattern $\bar{v}_k$, according to

$$\Delta w_l(n) \Big|_{\bar{v}_k} = -\eta \left[-\frac{1}{T} \left\{ \frac{N_{ij}^{+}(k)}{R} p(\bar{v}_k) - \frac{N_{ij}^{-}(k)}{LR} \right\} \right] . \qquad (32)$$

Finally, a compromise algorithm can be utilized

$$\Delta w'_l(n) = -\eta\left[-\frac{1}{T}\left\{\sum_{k=1}^{L(n)}\left(\frac{1}{R}N_{ij}^{+}(k)\right)p(\bar{v}_k) - \frac{1}{L}\sum_{k=1}^{L(n)}\frac{1}{R}N_{ij}^{-}(k)\right\}\right] \tag{33}$$

where *L(n)* is varied in an exponential manner (shown in Figure 10) to achieve a compromise between true gradient descent (30), requiring weight updates after all L patterns, and (32) requiring adaptions following each pattern presentation [12].

Towards examining the computational burden, the weight update rule (31) is implemented and diagrammed in Figure 11.

With the following symbol definitions, the computational burden can be assessed

N_v	=	No. visible units
N_h	=	No. hidden units
N	=	No. units $(N_v + N_h)$
N_w	=	No. distinct connection weights [Boltzmann $N_w = \frac{N}{2}(N-1)$]
L	=	No. training data patterns
N_s	=	No. learning sweeps (No. adaptions/connection weight)
N_a	=	No. iterations/annealing schedule
R	=	No. iterations used for collecting statistics
N_i	=	No. probes/iteration (Boltzmann $N_i^+ = N_h$, $N_i^- = N$)
N_t	=	No. averages computed by (15) for learning process.

The total number of probes for the general two mode learning procedure is

$$N_p = N_s L N_a (N_h + N) . \tag{34}$$

In addition, the total number of averages computed by (21) for the expected value calculations in (18) are

$$N_t = 2N_s N_w . \tag{35}$$

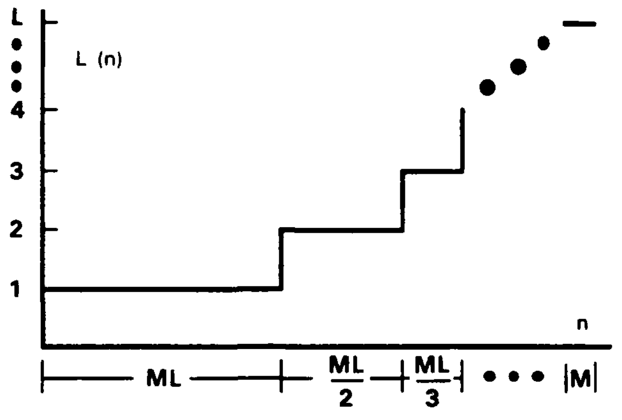
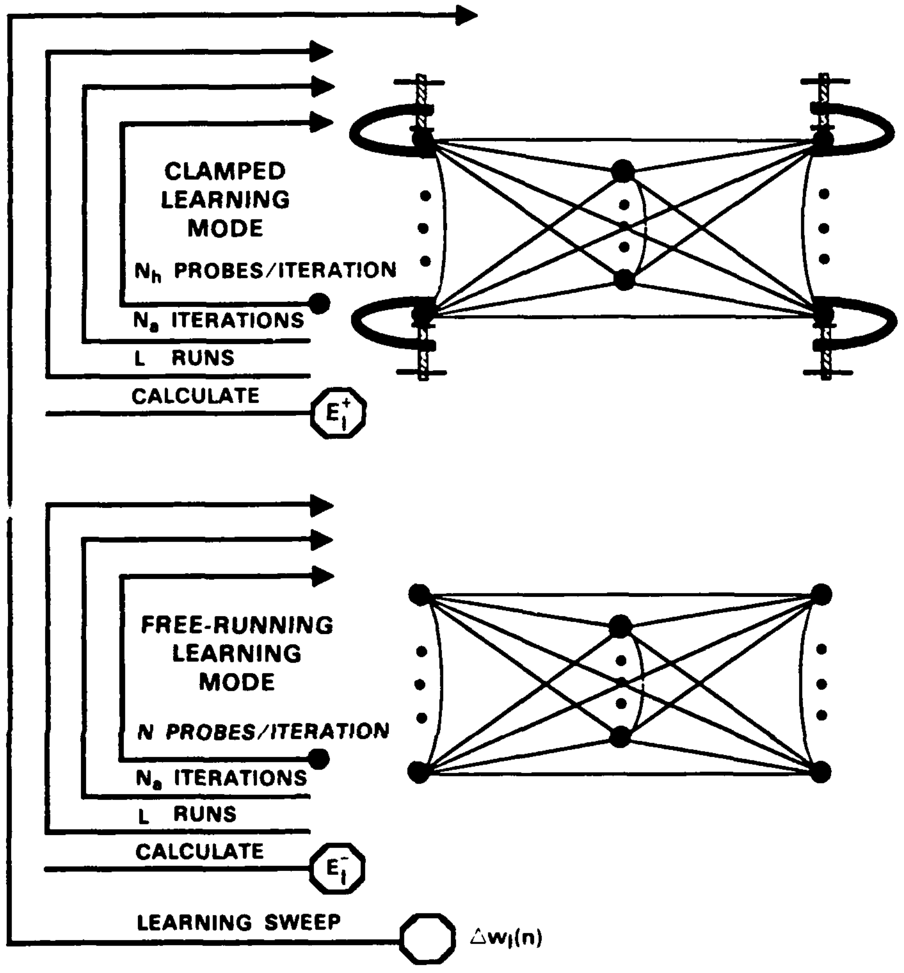


Figure 10. Pattern presentation method for compromise algorithm.

Specifically for the Boltzmann machine with symmetric weights and $N_s \cong N^2$ (from published examples) the complete learning process requires approximately N^4 averages and a number of probes exceeding N^3. And realizing that each probe consists of four steps of the Metropolis algorithm, the computational burden is extensive.

In summary, the machine learning procedure for model identification requires adjusting the connection strengths to minimize the discrepancy measure between the data and model. The particular learning algorithm to achieve this minimization is given by (15), where the expected values of local potential functions (13) and (14) must be calculated. The average local potentials are alternately numerically computed by simulated annealing. In the free running mode, E_l^- is calculated by simulated annealing with all units being perturbed. While in the clamped learning mode, E_l^+ is also calculated by simulated annealing, but with the visible units clamped, and hence only the hidden units being perturbed. Together $2L$ simulated annealings are performed to provide the gradient with respect to the connection weights. Consequently this process requiring $2L$ annealings (actually a modified gradient descent algorithm) must be conducted each time the connection weights are adapted; certainly an extraordinary computational effort.



94420-11

Figure 11. Machine learning procedure (nested iteration processes and architectures shown).

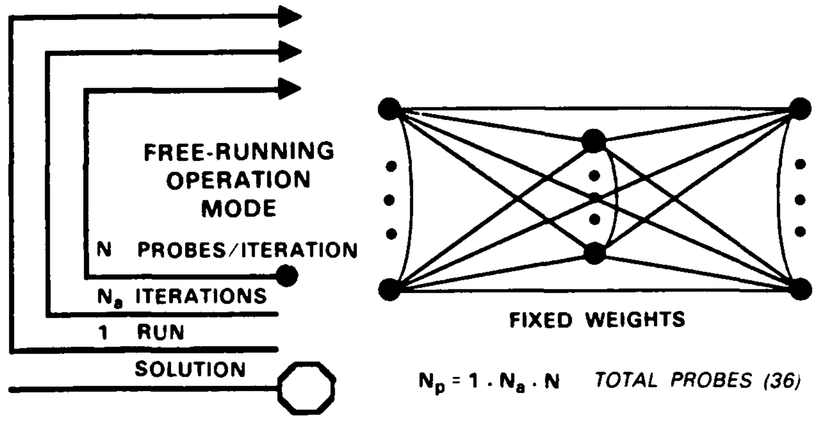
5. MACHINE OPERATION

Once the machine is trained, either by slow iterative learning (Section 4) or fast *a priori* assignment, the connection weights are fixed and the machine is ready for operation. The mode of machine operation depends upon the application. Again as in slow learning, two modes of operation are discussed, free-running and clamped. The machine operation applications are presented below in the context of these defined modes. Finally, both digital computer and optical implementations of the machine are reviewed.

The free-running mode of operation simply performs simulated annealing according to a prescribed annealing schedule allowing all units to be perturbed. In the process, the cost (energy) function expressed in terms of the now fixed connection weights is minimized. Hence when the machine reaches equilibrium at the lowest temperature, the final global state of the machine is of low energy.

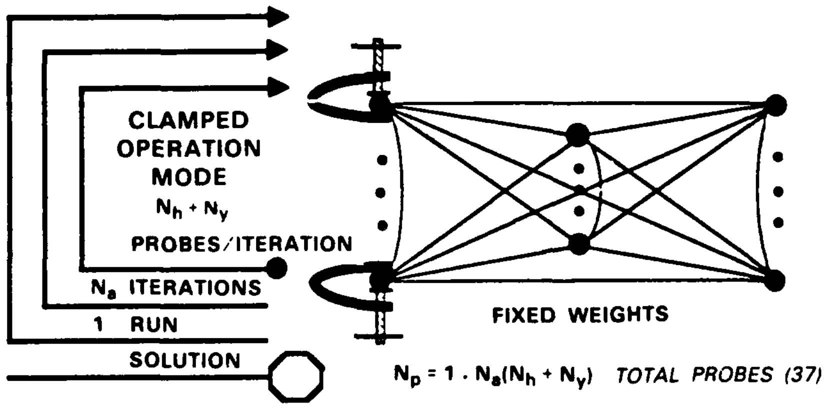
Now recall from the machine formulation, the lowest energy state corresponds to the optimal solution. Therefore if the annealing schedule decreases the temperature sufficiently slow according to Geman's result [11], the optimal solution (global minimum) is found. However good solutions (local minima) are often found with much faster annealing schedules. Therefore, the quality of the solution obtained at equilibrium as well as the computational load of the free-running mode is dependent upon the annealing schedule. The operation is diagrammed in Figure 12.

The clamped mode of application also performs simulated annealing, but here the inputs are clamped, thus only the hidden and output units are perturbed by the annealing. This process can be viewed as a conditional minimization, in which the energy function minimization is conditioned upon the input. Again the annealing schedule is chosen according to the desired quality of the solution. The clamped operation along with the computational load is shown in Figure 13.


FREE-RUNNING OPERATION MODE
N PROBES/ITERATION
N_a ITERATIONS
1 RUN
SOLUTION
FIXED WEIGHTS
$N_p = 1 \cdot N_a \cdot N$ TOTAL PROBES (36)

94420-12

Figure 12. Free-running operation mode.


CLAMPED OPERATION MODE
$N_h + N_y$ PROBES/ITERATION
N_a ITERATIONS
1 RUN
SOLUTION
FIXED WEIGHTS
$N_p = 1 \cdot N_a(N_h + N_y)$ TOTAL PROBES (37)

94420-13

Figure 13. Clamped operation mode.

5.1 APPLICATIONS

Model Identification: — A simple application demonstrating the learning capability and free-running operation of the Boltzmann machine is the encoder problem. The application originates from communication theory where a reduced representation is sought for the information transmitted. The training consists of placing identical patterns across the input and output. Through training, the hidden units are to discover a representation (code) for the presented data.

Specifically for the 4-2-4 encoder [13], the input and output of the machine consists of four units each. As the 4 patterns are presented across the visible units, the two hidden units must develop a binary code to represent the data. With a few modifications to the algorithm, the Boltzmann machine learns a proper coding after an average of 110 learning sweeps. The annealing schedule consisted of 20 iterations ($N_a = 20$), from a maximum temperature of 20 to a minimum 10. The machine can also be operated in the clamped mode, such that when excited by an input pattern, a code is generated by the hidden units and verified by observing the output units.

Another model identification application entails the processing of SAR images. The intent being to generate and classify SAR pattern textures from training patterns. Luttrell [8] proposed a hybrid Gibbs machine allowing the incorporation of *a priori* information. Essentially, a portion of the weights are specified *a priori*, while the remaining are noncommittal. Thus the learning process only adapts the noncommittal weights, thereby reducing the extensive learning computations. Unfortunately only simple two dimensional random telegraph signal patterns were used to demonstrate the machine. Consequently the approach is difficult to carefully assess.

Optimization: — Most of the optimization applications simply view the machine as a vehicle for implementing simulated annealing. Recall from Section 2, the optimization problem involves finding the extremum of a cost function of several independent variables. Here the machine is operated in the free-running mode with the connection weights and energy function determined with the problem formulation. The result is a machine which converges to the optimal solution (global extremum) of the cost function given an adequate annealing schedule.

The first application presented is image restoration. Geman and Geman [11] rigorously formulated an image restoration algorithm based upon simulated annealing, and hence implementable by the machine. The optimal solution is defined as the maximum *a posteriori* estimate (MAP) of the original image given the degraded image. The degradation examined included blurring, nonlinear deformations, and multiplicative additive noise.

The energy function used is the posterior distribution of the image, generated from the known prior distribution and degradation model. In theory, the annealing process generates a sequence of images that converge in probability to the global maximum of the *a posteriori* distribution (MAP estimate.) In practice, the theoretical annealing schedule is too slow. However good image restoration results are reported for various degradations with moderate annealing schedules.

The next optimization application involves the partitioning problem in computer design, studied by Kirkpatrick *et. al.* [5] when originating simulating annealing. Recall from the example in Section 2, the problem involves partitioning M circuits amongst two chips such that the number of signals crossing the chips is minimized, while also avoiding any significant imbalance. The solution best *satisfying the constraints* is defined optimal, resulting in fewer pins and higher speed.

The specific problem involves partitioning the IBM 370 microprocessor onto two chips. The energy function which captures these optimization constraints is given by (3). The annealing schedule begins with $T_0 = 10$ and cools slowly with exponential decay

$$T(t) = \left(\frac{T_1}{T_0}\right)^t T_0 . \tag{38}$$

As expected, the solution improves as the temperature is decreased. For example at $T = 100$ (random search) the partitioning required 6000 pins. On the other extreme, $T = 0$ (gradient descent) the partitioning required 700 pins. Finally for the exponential cooling (38), the process reached equilibrium at a partitioning with 500 pins. Also in a computer design context, both placement and wiring optimization problems were conducted and good results reported.

Again using the machine as a vehicle for implementing simulated annealing, Ticknor and Barrett [14] formulated and implemented a Boltzmann machine for the general problem of estimating a quantity given noisy measurements. Specifically, the generator model is

$$\bar{g} = \bar{H}\bar{f} + \bar{n} \tag{39}$$

where $\bar{g}(M \times 1)$ represents the measurements, $\bar{H}(M \times N)$ is a system transition operator describing how variations in $\bar{f}$ affect $\bar{g}$, $\bar{f}(N \times 1)$ is the quantity to be estimated, and $\bar{n}(M \times 1)$ represents the noise composed of unpredictable discrepancies in the measurements and system errors. Classically, solutions exist requiring the calculation of pseudoinverses. These approaches are often time consuming and suffer from ill-conditioning. The approach taken in [14] is to find an estimate of the solution, $\hat{f}$, defined optimal if $\hat{f}$ minimizes the distance measure

$$E(\bar{s}) = \left\| \bar{H}\hat{f}(\bar{s}) - \bar{g} \right\|^2 . \tag{40}$$

Thus the machine is built with (40) as the energy function. A simple example demonstrating the approach involved reconstructing rectangular functions which have been blurred, causing oscillatory behavior in the images. Using *a priori* knowledge to suppress the oscillations, good results were reported. However the major contribution is perhaps the optical implementation of the machine.

Associative Memory: — The final application utilizes the machine in an associative memory capacity. The objective is to perform memory recall given incomplete or partial information (key). The machine operates in the free running mode upon initialization with the input key pattern. At equilibrium an output pattern is generated. If the generated output is the closest memory to the initial key pattern, the solution is termed optimal.

One method of performing associative memory with the machine involves creating an energy function such that the memories to be retrieved $\{\bar{z}_i\}_{i=1}^{M}$ are the local minima. Since the machine is desired to reach equilibrium at the closest local minima (memory), the provisions of the machine that enable escaping from local minima are not needed. Rather, the machine is operated at temperature $T = 0$, thereby reducing the machine to a gradient descent algorithm converging to local minima (see Appendix 9.1). Although the machine with $T = 0$ converges to local minima, the optimal solution is not guaranteed, since the search is always in the direction of non increasing energy (see Figure 14).

Additional consequences of this method for machine based associative memory become apparent once the energy function is specified. In short, formulating an energy function with local minima corresponding to given arbitrary memories is difficult. As an example, consider the Hopfield model [6] with the energy function reformulated in a linear algebra framework

$$E(\bar{s}) = -\left\|\bar{H}\bar{s}\right\|^2 \qquad (41)$$

where

$$\bar{H} = \sum_{i=1}^{M} \bar{z}_i \, \bar{z}_i^* - M\bar{I} \quad . \qquad (42)$$

Now McEliece, Posner, Rodemich and Venkatesh [15] have shown that if the number of memory vectors to be stored is significantly less than the dimension of the memory vector $(M < N / 4 \log N)$ then all memories can be recalled and are eigenvectors of the connection matrix H. Therefore the memories actually produce local minima in the energy function as desired. However other local minima may be introduced resulting in spurious memories, even if the memories are kept within capacity (see Appendix 9.5).

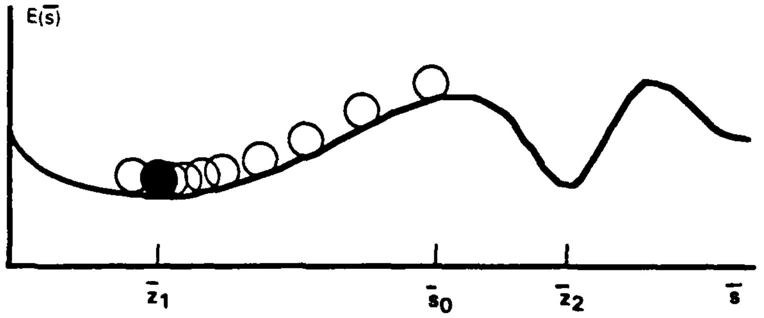


Figure 14. Search diagrammed beginning at $\bar{s}_0$ (key) and proceeding in the direction of decreasing E, and consequently not converging to the closest local memory $\bar{z}_2$.

Consequently associative memory in the sense of Hopfield can certainly be implemented by the Boltzmann machine. In fact, the Boltzmann machine operating in the free-running mode at temperature $T = 0$ and initialized with the key pattern reduces to the Hopfield associative memory. Even though the optimal solution is unlikely to be generated, due to the nonincreasing directional search and possible spurious memories, the method often provides fair solutions in acceptable times. And utilizing the full capacity of the machine, other approaches may be developed to counter these limitations.

5.2 IMPLEMENTATIONS

Since the kernel of the machine algorithm is the Metropolis probing, whereby the units are sequentially perturbed, digital computer implementation is quite natural. In fact the theory supporting the machine entails probing the units sequentially at random according to an annealing schedule. Unfortunately the computational burden is extensive.

The net computational burden can be appreciated from viewing Table 1, along with the figures provided for the amount of computation required for machine learning (34), (35) and operation (36), (37). The slow learning requires approximately N^3 probes and N^4 averages, while the operation mode entails N probes. Moreover, realizing that each probe demands at most 3 random number generations, matrix operations, comparison and storage, the current digital computer implementations by simulation are feasible only for small scale problems requiring tens of processing units. Due to this large computational burden, especially during learning, algorithm modifications and alternative implementations are being pursued.

The computational burden is significantly reduced by using a modified algorithm allowing parallel probing. That is, all N units are probed simultaneously, rather than sequential as the theory suggests. Here the energy gap calculation (step 3) for each unit ignores the current state of the other units and relies instead on the most recent global state. Although the original theory does not support this parallel extension, often in practice the empirical results do provide justification.

TABLE 1.
Probe Computations

(1)	s_i	— Random Unit Selection •RNG
(2)	$s_i + \Delta s$	— Random Perturbation •RNG •Addition
(3)	ΔE	— Energy Gap Calculation $(s_i \rightarrow s_i + \Delta s \Rightarrow E \rightarrow E + \Delta E)$ •Matrix Multiplication •Matrix Addition
(4)	Perturbation Acceptance Decision (23) or (24)	•RNG •Exponentiation •Comparison

An optical implementation utilizing the parallel machine modification has been introduced by Tichnor and Barrett [14]. The matrix operations are conducted by incoherent optical techniques. Also, the random numbers required are generated optically by detecting the light intensity from a moving speckle pattern. Compared to the digital computer, a significant gain in speed is achieved at the expense of lower dynamic range in the optical calculation of ΔE .

6. CONCLUSION

In principle. the Boltzmann machine operating with an appropriate (yet perhaps infeasible) annealing schedule will provide the optimal solution (global extremum) to a problem expressed mathematically in terms of a cost function. Although the theory supports the achievement of the optimal solution, rarely are the assumptions supporting the theory maintained. From applications, the machine with much faster annealing schedules often produces solutions which are not optimal, yet satisfactory. Therefore the machine is perhaps best assessed empirically in the context of the particular applications. A summary of such assessment is displayed in Table 2.

With regards to model identification, the machine learns the underlying constraints of the data from training examples. Thus no *a priori* knowledge or modeling assumptions are necessary. In addition the distributed representation of the knowledge over the machine interconnections offers resistance to minor damage. However with no *a priori* information, everything must be learned, resulting in extremely slow learning. In fact the minimization of the objective function for learning requires numerous simulated annealings for a single adaption of the connection weights. Furthermore the complete learning procedure is actually a gradient descent algorithm, requiring the numerous simulated annealings just to compute the gradient. Hence the entire extensive learning process is a local minimization, and thus unlikely to reach the optimal solution for model identification.

Utilizing the machine for optimization applications requires very slow annealing schedules to reach the optimal solution. And certainly mapping the optimization problem onto the machine by specifying both the cost function and connection weights can be challenging. However if the mapping can be accomplished, the Boltzmann machine provides a solution which on the average increases in quality as the system is cooled according to the annealing schedule.

Finally, the associative memory application of the Boltzmann machine suffers from spurious memories. This together with the process always searching in the direction of decreasing cost (at temperature $T = 0$) prevents the machine from always converging to the closest memory. However alternative associative memories can be constructed and implemented using the full capabilities of the Boltzmann machine to avoid some of these issues.

TABLE 2
Machine Applications Characteristics

Application	Objective	Approach	Cost Function	Training	Disadvantages	Advantages
Model Identification	Identify Model To Replicate Implicit Behavior Found In Stochastic Training Data	Minimize Cost Function with Respect to Connection Weights	Measures Discrepancy Between Model and Training Data Probability Distributions	Connection Weights Iteratively Adapted To Minimize Cost Function	• Slow • Local Minimization of Cost Function	• Robust (Tolerant to Unit Failure) • Tractable
Optimization	Find Optimal Solution to Problem Given Finite Resources and Constraints	Globally Minimize Cost Function With Respect to Global State Configuration	Measures Discrepancy Between Any Solution and the Desired Solution Goals	Connection Weights Fixed *a priori* with Cost Function Specification	• Slow Annealing Schedules • Cost Function and Weights Specified *a priori*	• Progressive Solution (Average Quality of Solution Increases)
Associative Memory	Find Closest Memory to Given Input Pattern (Key)	Locally Minimize Cost Function with Respect to Global State	Measures Discrepancy Between Specific Pattern and a Memory	Connection Weights Fixed *a priori* with Cost Function	• Spurious Memories • Closest Memory not Guaranteed	• Distributive Representation

Consequently given current computer technology and the assessment derived from preliminary examples, the optimization category of application is perhaps most promising for the Boltzmann machine. The idea of iteratively generating a solution which converges to the optimal upon following a prescribed annealing schedule is quite pleasing. In practice, moderate annealing schedules often lead to solutions which are near optimal in acceptaole time.

Further research on the machine is likely to be focused on reducing the excessive processing time. Included would be algorithm modifications as well as new implementation techniques. By increasing the complexity of the interconnection strengths within the general machine formalism presented, the learning times can be reduced at the expense of complexity. Also VLSI and optically based machine implementations together are likely to significantly reduce the excessive processing times experienced by current digital simulation.

In all, the Boltzmann machine is a computational algorithm (motivated from both classical statistical mechanics and information theory) for finding the optimal solution to a problem having objectives represented by a cost function. And it remains to be seen if the Boltzmann machine, or the generalized machine, can be scaled to solve useful problems in acceptable times.

7. ACKNOWLEDGMENT

I wish to thank Tom Goblick for careful reading of the manuscript, and Terje Vold for the much appreciated comments.

Also special thanks to S. Luttrell who realized and documented [8] the Gibbs distribution generalization of the Boltzmann machine. For it is through his initial efforts, the machine learning generalization became apparent and exploited in the present context.

8. REFERENCES

[1] G.E. Hinton and T.J. Sejnowski, Optimal perceptual inference. *Proceedings of the IEEE Computer Society on Computer Vision and Pattern Recognition*, pp. 448-453, June 1983.

[2] L. Boltzmann, "Further investigations on the thermal equilibrium of gas molecules, *Proceedings Imperial Academy of Science Vienna*, 1872.

[3] E. Cohen, *The Boltzmann Equation: Theory and Application*, Springer-Verlag, New York, 1973.

[4] S. Kullback, *Information Theory and Statistics*, New York, Wiley, 1959.

[5] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, Optimization by simulated annealing, *Science*, Vol. 220, pp 671-679, May 1983.

[6] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proc. Natl. Acad. of Sci.*, USA, Vol. 19, pp. 2554-2558, April 1982.

[7] R.L. Dobrushin, "The description of a random field by means of conditional probabilities and conditions of regularity", *Theory Prob. Appl.*, Vol.13, pp. 197-224, 1968.

[8] S.P. Luttrell, *The Implications of Boltzmann-type Machines for SAR Data Processing: A Preliminary Survey*, Royal Signals and Radar Establishment Memorandum 35315, June 1985.

[9] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, " Equation of state calculations for fast computing machines." Journal of Chemical Physics, Vol. 6, pp. 1087-1092, June 1953.

[10] V. Cerny, "A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm," preprint, *Inst. Phys. & Biophys.*, Comenius Univ., Bratislava, 1983.

[11] S. Geman and D. Geman., "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 6, pp. 721-741, Nov. 1984.

[12] M. Eggers, T. Khuon, "Learning algorithms for the multilayer perceptron." Submitted to *IEEE Trans. Pattern Anal. Mach. Intell.*, 1988.

[13] D. H. Ackley, G.E. Hinton, T.J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, pp. 147-169, 1985.

[14] A.J. Ticknor, H.H. Barrett, "Optical implementations in Boltzmann machines," *Optical Engineering*, Vol. 26, pp. 16-21, Jan. 1987.

[15] R. McEliece, E. Posner, E. Rodemick, S. Venkatesh, *The Capacity of the Hopfield Associative Memory*, Dept. of Elec. Engr., Caltech, 1986.

[16] G. Strang, *Linear Algebra and Its Applications*, Academic Press, Florida, 1980.

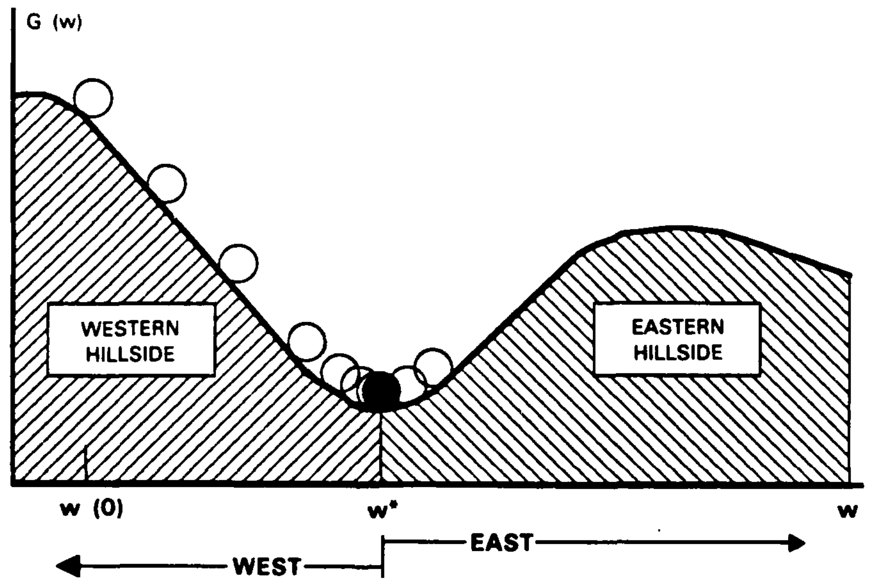
9. APPENDICES

9.1 GRADIENT DESCENT

The method of gradient descent can be intuitively formulated with the following example. Consider the single dimension case where $G(w)$ is to be minimized with respect to the independent variable w. The function G can be viewed as a landscape with hills and valleys, as shown in Figure 15.

With such interpretation, the goal of the minimization process is to travel from an initial position upon a hillside to the final destination in the valley below. From the diagram, the traveler should travel east (increasing w) when on the western hillside in order to reach the valley. Notice the slope of the western hillside is negative while the eastern hillside is positive. Hence the traveler's plan can be stated mathematically as always moving in the direction opposite to the polarity of the derivative at the present location. Thus the traveler's rule guiding the next step can be formulated in terms of the present step according to

$$w(n+1) = w(n) + \Delta w(n) \tag{43}$$



94420-15

Figure 15. Landscape interpretation of minimization.

where

$$\Delta w'(n) = - \eta \;\; \text{sign} \left(\frac{\partial G}{\partial w'} \Big|_{w'(n)} \right) \tag{44}$$

and η represents the traveling rate (dependent upon the traveler's physical condition.)

A refinement can be made by realizing the traveler should take large steps when high on the hillside far from the valley (large $|\partial G \;/\; \partial w' \;|$), and conversely take small steps when nearing the valley (small $|\partial G \;/\; \partial w' \;|$.) Mathematically, this is equivalent to making the step size proportional to the magnitude of the derivative. Consequently, the refined rule is (43) with

$$\Delta w'(n) = - \; \eta \frac{\partial G}{\partial w'} \Big|_{w'(n)} \tag{45}$$

and this minimization process is diagrammed in Figure 15.

The final extension involves expanding the landscape in many directions. Here G is now a function of several independent variables, denoted by the vector $\bar{w}'$. And the traveler simply moves in the direction of steepest descent yielding the following algorithm applied to each component.

$$w'_l(n + 1) = w'_l(n) + \Delta w'_l(n) \tag{46}$$

where

$$\Delta w'_l(n) = - \; \eta \frac{\partial G(\bar{w}')}{\partial w'_l} \Big|_{\bar{w}'(n)} . \tag{47}$$

Disadvantages of the algorithm include the inability to escape from local minima as well as excessive computation times for large dimensions.

9.2 MACHINE LEARNING FORMULA DERIVATION

The machine learning formula derivation consists of minimizing the discrepancy measure $G(p, q)$ with respect to the connection strengths. To conduct the minimization, the partial derivatives with respect to the weights must be determined. As in the formulation by Luttrell [8], a most g ɪeral approach is taken. Thus the weight w_l corresponds to the lth M-tuple of units coupled together with common connection strength w_l . (For typical pairwise connections, w_l represents the lth weight coupling two units as diagramed in Figure 7.) Since only the machine is a function of the connection weights

$$\frac{\partial G}{\partial w_l} = - \int_V p(\bar{v}) \frac{\partial}{\partial w_l} \log\ q(\bar{v}) d\bar{v} \quad . \tag{48}$$

With the addition of the hidden units, the machine is now fully characterized by the probability distribution amongst all units, visible and hidden, given by $q(\bar{v},\ \bar{h})$. Thus the machine distribution $q(\bar{v})$ over the visible units, necessary for the minimization (48), is related to the distribution over all units by

$$q(\bar{v}) = \int_H q(\bar{v},\ \bar{h}) d\bar{h}$$

$$H = \text{hidden units} \tag{49}$$

thus

$$\frac{\partial}{\partial w_l} \log\ q(\bar{v}) = \frac{1}{q(\bar{v})} \int_H \frac{\partial}{\partial w_l} q(\bar{v},\ \bar{h}) d\bar{h} \quad . \tag{50}$$

To continue, assumptions must be made about the functional form of the machine distribution $q(\bar{v},\ \bar{h})$ (such assumption gives rise to the naming of the machine i.e. Boltzmann machine, Gibbs machine, . . .). For reasons discussed in the Machine Learning section, the Gibbs distribution (generalization of Boltzmann) is an attractive choice given by

$$q(\bar{s}) = q(\bar{v},\ \bar{h}) = \frac{1}{z} e^{-\sum_l w_l U_l (\bar{v},\ \bar{h})} \tag{51}$$

and enables the calculation of (50)

$$\frac{\partial}{\partial w_l} q(\bar{v}, \bar{h}) = -\frac{1}{z} e^{-\sum_l w_l U_l(\bar{v}, \bar{h})} U_l(\bar{v}, \bar{h})$$

$$-\frac{1}{z^2} e^{-\sum_l w_l U_l(\bar{v}, \bar{h})} \frac{\partial}{\partial w_l} \int_H \int_V e^{-\sum_l w_l U_l(\bar{v}, \bar{h})} d\bar{v} d\bar{h}$$

$$\Rightarrow \frac{\partial}{\partial w_l} q(\bar{v}, \bar{h}) = q(\bar{v}, \bar{h}) \left[\int_H \int_V U_l(\bar{v}, \bar{h}) q(\bar{v}, \bar{h}) d\bar{v} d\bar{h} - U_l(\bar{v}, \bar{h}) \right] \quad . \quad (52)$$

Thus upon substitution of (52), (50) into (48)

$$\frac{\partial G(p, q)}{\partial w_l} = -\int_V p(\bar{v}) \frac{1}{q(\bar{v})} \int_H \left\{ \frac{\partial}{\partial w_l} q(\bar{v}, \bar{h}) d\bar{h} \right\} d\bar{v}$$

$$= -\int_V \int_H p(\bar{v}) \frac{q(\bar{v}, \bar{h})}{q(\bar{v})} \left[E_{q(\bar{v}, \bar{h})} \{ U_l(\bar{v}, \bar{h}) \} - U_l(\bar{v}, \bar{h}) \right] d\bar{h} d\bar{v}$$

$$= -E_{q(\bar{v}, \bar{h})} \{ U_l(\bar{v}, \bar{h}) \} + E_{q(\bar{h} / \bar{v}) p(\bar{v})} \{ U_l(\bar{v}, \bar{h}) \} \quad . \quad (53)$$

where E_f denotes expected value with respect to distribution f.

9.3 METROPOLIS ALGORITHM

Metropolis *et al.* [9] in a classical statistical mechanics setting devised a clever algorithm for numerically calculating the properties of substances consisting of interacting individual molecules. The algorithm for example can be used to calculate the average pressure of a gas at equilibrium.

However, application of the algorithm is much more general. In fact the algorithm can be viewed as a Monte Carlo integration method for calculating expected values of functions with respect to the canonical Boltzmann distribution. Specifically the algorithm is used to numerically evaluate the equation

$$E\{F(\bar{s})\} = \int_S F(\bar{s}) q(\bar{s}) d\bar{s} \tag{54}$$

where $q(\bar{s})$ is the Boltzmann distribution in the form

$$q(\bar{s}) = \frac{1}{z} e^{-E(\bar{s})/KT} \tag{55}$$

and $E(\bar{s})$ is the global potential (energy) of the system in state configuration $\bar{s}$. Since the units (whose values represent the global state) number in the hundreds, conventional numerical integration by approximating sums is not feasible. Standard Monte Carlo approaches entail approximating the integral by the sum

$$E\{F(\bar{s})\} \cong \sum_i F(\bar{s}_i) q(\bar{s}_i) = \frac{1}{z} \sum_i F(\bar{s}_i) e^{-E(\bar{s}_i)/KT} \tag{56}$$

where the global states $\bar{s}_i$ are chosen randomly with uniform probability. A particular problem arises when implementing this approach with the Boltzmann distribution, and is illustrated in Figure 16. Typically the global potential function $E(\bar{s})$ is dominated by high energy states, yielding few of low energy. Consequently, the Boltzmann distribution responsible for weighting the function $F(\bar{s})$ (56) places most of the mass over a very limited portion of the state space, therefore the random samples chosen occur with very low probability.

To counter such a problem, Metropolis realized the same calculation could be performed by selecting particular global states $\bar{s}_i'$ with probability

$$P(\bar{s}_i') = e^{-E(\bar{s}_i')/KT} \tag{57}$$

and subsequently evenly weighting the function $F(\bar{s}_i')$ yielding

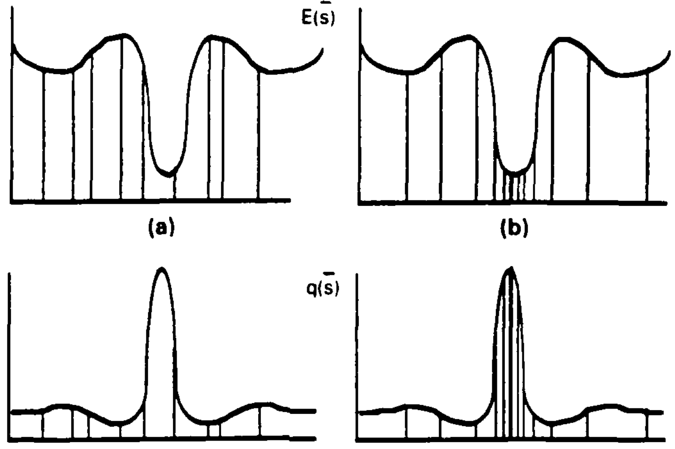


Figure 16. (a) Uniform, and (b) exponential random sampling of global states.

$$E\{F(\bar{s})\} \cong \frac{1}{R} \sum_{i=1}^{R} F(\bar{s}_i') \quad . \tag{58}$$

Thus instead of choosing at random global states with uniform probability and then weighting the function $F(\bar{s}_i)$ by the Boltzmann distribution according to (56), the Metropolis algorithm chooses at random global states with probability $e^{-E(\bar{s}_i)/KT}$ and weights the function $F(\bar{s}_i)$ uniformly. The companion diagram for the Metropolis algorithm is shown in Fig.16b.

The procedure for conducting this type of Monte Carlo integration is as follows.

(0) Choose an arbitrary global state $\bar{s}$.

(1) Select a component (single unit) of the global state vector $\bar{s}$ at random, denoted s_i.

(2) Perturb the selected unit by a small amount Δs

$$s_i \rightarrow s_i + \Delta s \tag{59}$$

where Δs is a uniform random variable in the range $(-\infty, +\infty)$.

(3) Calculate the global energy change due to the perturbation

$$s_i \to s_i + \Delta s \Rightarrow E \to E + \Delta E \tag{60}$$

where

$$\Delta E = E\left(\bar{s} + \Delta \bar{s}_i\right) - E(\bar{s}) \tag{61}$$

$$\Delta \bar{s}_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \Delta s \ . \tag{62}$$

(4) Decide to accept or reject new global state according to:

$$\Delta E \le 0 \Rightarrow \text{accept } \ \bar{s} + \Delta \bar{s}_i$$

$$\Delta E > 0 \Rightarrow \text{accept } \ \bar{s} + \Delta \bar{s}_i \ \text{ with probability } \ e^{-\Delta E / KT} \ . \tag{63}$$

(5) Continue the process by cycling back to step 2, until equilibrium is reached.

(6) Finally calculate the expected value according to (58) where $F\left(\vec{s}_i\right)$ is the function calculated at the global state $\vec{s}_i$, obtained at the ith iteration.

9.4 SIMULATED ANNEALING

Simulated annealing is an iterative procedure for determining the global minimum or maximum of a function of several independent variables. The procedure is derived from the Metropolis algorithm (Appendix 9.3).

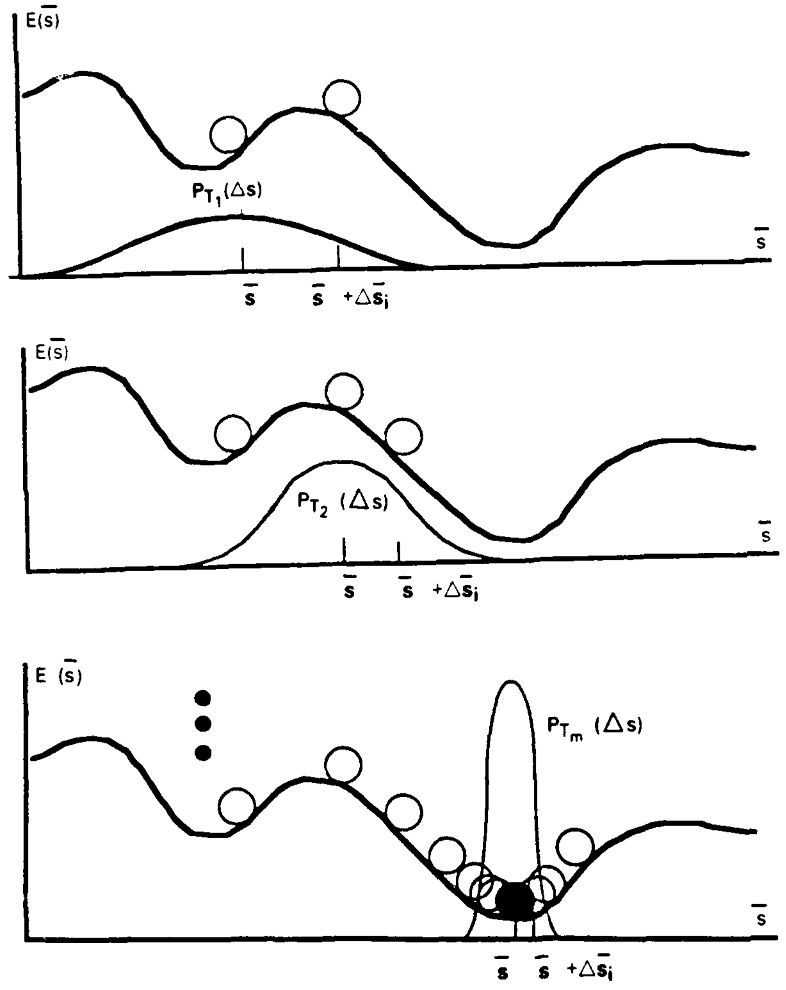
By examining the steps of the Metropolis algorithm (steps 0-5) the decision rule is seen to always accept state transitions resulting in lower energy, or equivalently, accepting downhill moves along the energy landscape. However, the distinctive feature is the occasional uphill moves allowed with a probability dependent upon the temperature T (see (63). With high temperature, the uphill moves are readily accepted. In fact with $T = \infty$ the states are chosen with uniform probability corresponding to a random walk in the state space. And at low temperature, uphill moves are rarely accepted, thereby allowing the process to reach steady state (equilibrium), yet possibly at a local minima. In the limiting case $T = 0$, the process is simply a gradient descent algorithm for finding a local minimum of the energy function. As a natural conseauence then, the Metropolis algorithm (steps 0-5) can be alternatively viewed as an algorithm which tends to converge to the minimum of any function $E(\bar{s})$ of several independent variables (see Figure 16b.)

Now Kirkpatrick *et al.* [5] and Cerny [10] realized heuristically that by carefully varying the temperature , beginning with large temperatures and gradually decreasing to lower temperatures , many local minima could be avoided and hence the probability of converging to the global minimum greatly increased. Consequently, this temperature variation (annealing) provides both local minima avoidance properties (characteristic of high temperatures), as well as favorable convergence properties (characteristic of low temperatures). The insight into the temperature variation originated from chemical annealing, where the low energy state of a material is determined by first melting the substance then gradually lowering the temperature, spending much time near the vicinity of the freezing point.

Later Geman and Geman [11] *rigorously* proved that indeed this heuristic technique guarantees convergence in probability to the global minimum if the temperature variation (annealing schedule) is properly chosen. Specifically, the conditions for global minimum convergence require the state perturbations to be Gaussian , and the annealing as a function of the iteration time must satisfy

$$T(t) \geq T_0 / \log(1 + t) . \qquad (64)$$

The simulated annealing process is conceptualized in Figure 17. Each figure represents a snapshot in the iteration process with a different temperature, showing how the perturbation generating distribution $p_T(\Delta s)$ decreases in variance to promote convergence.



94420-17

Figure 17. Simulated annealing concept with annealing schedule $T_1 > T_2 > \ldots T_M$

9.5 SPURIOUS MEMORIES

The following demonstrates that under certain conditions, the Hopfield associative memory produces local minima of the energy function at the desired memories. However other (spurious) memories are also created which often prevent convergence to the optimal solution as shown.

Given the Hopfield associative memory with M binary random memories $\{\bar{z}_i\}_{i=1}^{M}$ each of dimension N, connection matrix $H = \sum \bar{z}_i \bar{z}_i^* - M\bar{I}$, and a sufficient signal to noise ratio $M < N / 4 \log N$, then the memories are local minima of the energy function $E(\bar{s}) = -\|H\bar{s}\|^2$. This is shown by first directly expressing the *jth* output component of the connection matrix vector product with a memory input $\bar{z}_l$

$$y^j = \left(\bar{H}\bar{z}_l\right)^j = (N-1)z_l^j + \sum_{i \neq l}^{M} \sum_{k \neq j}^{N} z_i^j z_i^k z_l^k$$

$$= \text{signal} \quad + \quad \text{noise} \quad . \qquad (65)$$

Invoking the central limit theorem for sufficiently large N, the noise term is Gaussian with zero mean and variance $(M-1)(N-1)$. Thus the output becomes

$$\bar{H}\bar{z}_l = (N-1)\bar{z}_l + \bar{n} \qquad (66)$$

with a signal to noise ratio (SNR)

$$SNR \equiv \frac{Var\left((N-1)\bar{z}_l\right)}{Var(\bar{n})} = \frac{(N-1)^2}{(N-1)(M-1)} = \frac{N-1}{M-1} . \qquad (67)$$

Hence the memories are seen to be eigenvectors (with identical eigenvalues $(N-1)$) of the connection matrix for sufficiently large SNR (66). In fact in an elegant paper, McEliece *et al.* [15] have shown that the SNR must satisfy

$$SNR = \frac{N-1}{M-1} \approx \frac{N}{M} > 4 \log N \tag{68}$$

to ensure both the integrity of the eigenvector interpretation and perfect recollection of the stored memories. Now from linear algebra

$$\|\bar{H}\bar{s}\| \leq \|\bar{H}\| \|\bar{s}\| \tag{69}$$

where

$$\|\bar{H}\| \equiv \max_{\bar{s} \neq 0} \frac{\|\bar{H}\bar{s}\|}{\|\bar{s}\|} \tag{70}$$

and for symmetric connection weights the matrix norm is simply the norm of the largest eigenvalue, representing the largest amount by which any vector is amplified by the matrix-vector product [16]. Therefore since all eigenvalues corresponding to the M memories are identical for the Hopfield model, each eigenvalue (memory) produces a maxima of $\|\bar{H}\bar{s}\|$ and hence a minima of $-\|\bar{H}\bar{s}\|^2$.

Notice arbitrary (spurious) memories will be created even if the SNR is large. In addition to the M memories, there exist other eigenvectors which also produce minima in the energy function (since $M < N$). Therefore the price for accurate memory recall is low storage capacity as well as induced spurious memories $(\leq N - M)$.

As a final comment, the SNR constraint is crucial for correct memory recall. For low SNR , the noise term dominates (66) and the memories are no longer eigenvectors and hence no longer minima of the energy function. The result is a new energy function placing minima at new eigenvectors, also giving rise to spurious memories.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

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 unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Technical Report 805	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESD-TR-87-281

6a. NAME OF PERFORMING ORGANIZATION Lincoln Laboratory, MIT	6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION Electronic Systems Division
6c. ADDRESS *(City, State, and Zip Code)* P.O. Box 73 Lexington, MA 02173-0073		7b. ADDRESS *(City, State, and Zip Code)* Hanscom AFB, MA 01731
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-85-C-0002

8c. ADDRESS *(City, State, and Zip Code)*	10. SOURCE OF FUNDING NUMBERS			
Air Force Systems Command Andrews AFB Washington, DC 20334	PROGRAM ELEMENT NO. 12424F 31310F	PROJECT NO. 80	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*

The Boltzmann Machine: A Survey and Generalization

12. PERSONAL AUTHOR(S)
Mitchell D. Eggers

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Technical Report	FROM ______ TO ______	8 July 1988	66

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	neural networks ; Boltzmann machine ; Gibbs machine ; energy minimizing neural networks ; simulated annealing. [illegible]

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

A tutorial is presented describing a general machine learning theory which spawns a class of energy minimizing machines useful in model identification, optimization, and associative memory. Special realizations of the theory include the Boltzmann machine and the Hopfield neural network. The theory is reinforced by appendices addressing particular facets of the machine, ranging from gradient descent to simulated annealing.

The treatment is systematic, beginning with the description of the energy function. A defining relationship is established between the energy function and the optimal solution. Following, both classical and new learning algorithms are presented (directing the adaption of the free parameters) for numerically minimizing such function to yield the optimal solution. Finally, both computational burden and performance are assessed for several small-scale applications to date. Keywords: ;

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT.] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE *(Include Area Code)*	22c. OFFICE SYMBOL
Lt. Col. Hugh L. Southall, USAF	(617) 981-2330	ESD/TML