Neural Networks, Vol. 2, pp. 475–494, 1989
Printed in the USA. All rights reserved.

0893-6080/89 $3.00 + .00

ORIGINAL CONTRIBUTION

Explorations of the Mean Field Theory Learning Algorithm

CARSTEN PETERSON* AND ERIC HARTMAN

Microelectronics and Computer Technology Corporation
Austin, Texas

(*Received 2 December 1988; revised and accepted 20 March 1989*)

Abstract—*The mean field theory (MFT) learning algorithm is elaborated and explored with respect to a variety of tasks. MFT is benchmarked against the back-propagation learning algorithm (BP) on two different feature recognition problems: two-dimensional mirror symmetry and multidimensional statistical pattern classification. We find that while the two algorithms are very similar with respect to generalization properties, MFT normally requires a substantially smaller number of training epochs than BP. Since the MFT model is bidirectional, rather than feed-forward, its use can be extended naturally from purely functional mappings to a content addressable memory. A network with* N *visible and* N *hidden units can store up to approximately* 4N *patterns with good content-addressability. We stress an implementational advantage for MFT: it is natural for VLSI circuitry.*

Keywords—Neural network, Bidirectional, Generalization, Content addressable memory, Mean field theory, Learning algorithm.

1. INTRODUCTION

1.1. Background

Adaptive learning procedures for massively-parallel networks of primitive computing elements are presently subject to intense investigation. One such widely used learning algorithm is the back-propagation procedure (BP) (Rumelhart, Hinton, & Williams, 1986). In this algorithm, a set of input patterns are propagated in a feed-forward manner through a multilayer network. For each pattern, the resulting output signal is compared with a desired value and the error is recorded. The connection strengths, or weights, are then adjusted such that the error measure is minimized via gradient-descent. This algorithm falls into the category of supervised learning.

Another example of a supervised learning algorithm is the Boltzmann machine (BZ) (Ackley, Hinton, & Sejnowski, 1985). In this algorithm, learning takes place via two phases. First, the network is run with both the input and the output units clamped. Co-occurrency probabilities are measured after the system has reached a global energy minimum. This procedure is then repeated with only the input units clamped. The weights are then adjusted according to gradient-descent in an information-theoretical measure. In order to reach a global energy minimum in each phase of this process, it is necessary to use the very time-consuming simulated annealing method. This, combined with measuring the co-occurrence statistics, makes the Boltzmann machine one or two orders of magnitude slower than BP. Consequently, there has been relatively little experimentation with the Boltzmann machine.

In Peterson and Anderson (1987), it was convincingly demonstrated that the stochastic simulated annealing process in the Boltzmann machine can be replaced by a set of deterministic equations in the so-called mean field theory approximation. This Mean Field Theory Learning Algorithm (MFT) typically provides a substantial speed-up over the Boltzmann machine.

In this work we compare BP to MFT with respect to a variety of tasks. These algorithms differ in a number of ways. Their basic ingredients are very different:

- Error Measure. BP uses a sum over squared errors whereas MFT uses an asymmetric information-theoretic error measure.[1]

Acknowledgements—We have benefited from discussions with George Barna, Jim Keeler, Teuvo Kohonen, Vipin Kumar and Gale Martin. Also we would like to thank Jim Keeler for valuable suggestions on the manuscript.

* Present address and address for reprint requests: Department of Theoretical Physics, University of Lund, Solvegatan 14A, S-22362 Lund, Sweden.

[1] Recently a back-propagation algorithm based on a similar measure has been proposed (Baum & Wilczek, 1988; Hopfield, 1987).

- Notion of Final State. The final state in BP is reached when the signals have propagated in a feed-forward manner through the network to the output units. This is in contrast to MFT where no such intrinsic direction exists. The connections are bidirectional, and the system settles to steady state as does a physics system with many degrees of freedom.
- Locality of Updating. In BP, the error is propagated backwards from the output units and the weights are updated successively. In MFT, the situation is very different; the local correlations measured in the two phases are the seeds for updating the corresponding weights.

MFT has more potential than BP regarding range of applicability and hardware implementation:

- Applicability. With its strict partitioning of visible units into input units and output units, BP is designed for feature recognition.[2] MFT (or the Boltzmann machine) is not limited to this paradigm; visible units are not forced to adopt a fixed input or output functionality, and the algorithm can therefore be used as a *content addressable memory* (CAM) as well as being used as a pure feature recognizer. In addition, MFT networks (without training) can be used for combinatorial optimization problems (Hopfield & Tank, 1985; Peterson & Anderson, 1988; Peterson & Söderberg, 1989).
- VLSI implementation. The MFT equations represent steady state solutions to RC-equations of the corresponding electrical circuit (Hopfield & Tank, 1985). This fact, together with the local nature of the weight updating, facilitates a VLSI implementation.

1.2. Motivation and Results

The BP algorithm has been used extensively with encouraging results in applications ranging from small toy problems such as parity and encoder problems (Rumelhart, Hinton, & Williams, 1986) to more realistic ones such as mapping written text to phonemes (Rosenberg & Sejnowski, 1987), classifying sonar signals (Gorman & Sejnowski, 1988), predicting the structure of proteins (Qian & Sejnowski, 1988), and playing backgammon (Tesauro & Sejnowski, 1988). The MFT algorithm is a more recent development and has not yet been explored to the same extent. Given MFT's VLSI potential and inherent parallelism, it is important to investigate its potential for feature recognition problems beyond the testbed of toy problems in the original work (Peterson & Anderson, 1987). MFT's potential as a content addressable memory should also be explored. The objectives of this work are thus twofold: First, we compare the performance of BP versus MFT with respect to learning and generalization properties for two feature recognition problems: two-dimensional mirror symmetry and multidimensional statistical pattern classification (two overlapping Gaussians). We find the two algorithms approximately equal in generalization power. For the mirror symmetry problem, the number of training epochs required for MFT is substantially less than that required for BP. For the overlapping Gaussians problem, an appropriate variant of the algorithms removes the difference in their learning times. Both algorithms essentially achieve the theoretical Bayesian limit on this problem. We find this particularly impressive since, due to the statistical nature of the problem, inconsistent training is unavoidable.

Second, we explore MFT as a content addressable memory. This is a somewhat novel approach to CAM. In the Hopfield model for CAM (Hopfield, 1982), N visible units are used to store N-bit patterns. Our approach is very different. A layer of N hidden units are used to build *an internal representation of the stored N-bit patterns*. We are able in this way to store approximately $4N$ patterns with reasonable content-addressability, which should be compared to the loading factor $0.1N$ for the Hopfield model.

This paper is organized as follows: In section 2 we briefly review the ingredients of the MFT algorithm. Section 3 contains a comparison of BP and MFT performance for the mirror symmetry and statistical pattern classification problems. The novel approach to CAM is described in section 4, and section 5 contains a brief summary and outlook.

2. MEAN FIELD THEORY LEARNING

2.1. The Boltzmann Machine

The Boltzmann machine (Ackley, Hinton, & Sejnowski, 1985) is a learning algorithm for systems with or without hidden units. The dynamics are based on the Hopfield energy function (Hopfield, 1982)[3]

$$E(\vec{S}) = -\frac{1}{2}\sum_{i,j=1}^{N} T_{ij}S_iS_j \tag{1}$$

where S_i is the state of unit i, either 1 or -1,[4] and T_{ij} is the weight between units i and j. The sums run over both visible (input/output) and hidden units.

[2] Throughout this paper, we are referring to standard backpropagation, not to recurrent backpropagation (Pineda, 1987; Rumelhart et al., 1986) or other variants.

[3] Throughout this paper, the notation $\vec{S} = (S_1, \ldots, S_i, \ldots, S_N)$ is used.

[4] In Hopfield (1982), -1 and 1 are used, whereas in Ackley et al. (1985), 0 and 1 are used. Changing to 0 and 1 is not essential in the development of the Boltzmann machine.

The model learns by making an internal representation of its environment. The learning procedure changes weights so as to minimize the distance between two probability distributions, as measured by the so-called G-function

$$G = \sum_{\alpha} P_{\alpha} \log \frac{P_{\alpha}}{P'_{\alpha}} \tag{2}$$

where P_{α} is the probability that the visible units are collectively in state α when their states are determined by the environment. P_{α} represents the *desired* probabilities for these states. The corresponding probabilities when the network runs freely are denoted P'_{α}. G is zero if and only if the distributions are identical; otherwise it is positive. A slightly modified version of eqn (2) can be defined for the case when the visible units are partitioned into input (α) and output (β) units

$$\tilde{G} = \sum_{\alpha} Q_{\alpha} \sum_{\beta} P_{\beta|\alpha} \log \frac{P_{\beta|\alpha}}{P'_{\beta|\alpha}} \tag{3}$$

where Q_{α} is the probability of the state α over the input units and $P_{\beta|\alpha}$ the probability that the output units are in a state β given an input state α. Again, both Q_{α} and $P_{\beta|\alpha}$ are determined by the environment. $P'_{\beta|\alpha}$ is the probability that the system is in an output state β when the input units are clamped in the state α. Again, $\tilde{G}$ is positive and is zero if $P_{\beta|\alpha} = P'_{\beta|\alpha}$. The Boltzmann machine recipe for changing T_{ij} such that G or $\tilde{G}$ is minimized is as follows:

1. *Clamping Phase.* The values of the input and output units of the network are clamped to a training pattern, and for a sequence of decreasing temperatures $T_n, T_{n-1}, \ldots, T_0$, the network of eqn (1) is allowed to relax according to the Boltzmann distribution

$$P(\vec{S}) \propto e^{-E(\vec{S})/T} \tag{4}$$

where $P(\vec{S})$ denotes the probability that the state $\vec{S}$ will occur given the temperature T. At $T = T_0$ statistics are collected for the correlations

$$p_{ij} = \langle S_i S_j \rangle. \tag{5}$$

Relaxation at each temperature is performed by updating unclamped units according to the heatbath algorithm (Ackley et al., 1985)

$$P(S_i \longrightarrow 1) = \left[1 + \exp\left(\sum_j T_{ij} S_j / T\right)\right]^{-1}. \tag{6}$$

2. *Free Running Phase.* The same procedure as in Step 1 is used, but this time the network runs freely (G) or with only the input units clamped ($\tilde{G}$). Correlations

$$p'_{ij} = \langle S_i S_j \rangle \tag{7}$$

are again measured at $T = T_0$.

3. *Updating.* After each pattern has been processed through Steps 1 and 2, the weights are updated according to

$$\Delta T_{ij} = \eta (p_{ij} - p'_{ij}) \tag{8}$$

where η is the learning rate parameter. Equation (8) corresponds to gradient descent in $G(\tilde{G})$ (Ackley et al., 1985). Steps 1, 2, and 3 are repeated until no more changes in T_{ij} take place.

2.2. The Mean Field Theory Approximation

The mean field theory approximation is a well known technique in physics, particularly for spin-systems (Glauber, 1963). Extensive studies of the applicability of this approximation and refinements thereof has been made for spin-glass systems (Mezard, Parisi, & Virasoro, 1987), which are closely related to bidirectional neural network models. Here we limit ourselves to its crudest form, *the naive mean field theory approximation*, for which a derivation and discussion can be found in the appendices of Peterson and Anderson (1987) and Peterson and Anderson (1988). Here we briefly list the key points and equations.

The exact form of eqn (4) reads

$$P(\vec{S}) = \frac{e^{-E(\vec{S})/T}}{Z} \tag{9}$$

where the partition function is given by

$$Z = \sum_{\vec{S}} e^{-E(\vec{S})/T}. \tag{10}$$

The summation over all possible neuron configurations $\vec{S} = (S_1, \ldots, S_N)$ is computationally explosive with problem size. Replacing this discrete sum with a multidimensional integral yields

$$Z = C \int_{-\infty}^{\infty} \left[\prod_{i=1}^{N} dV_i\right] e^{-F(\vec{V},T)/T} \tag{11}$$

where $V_i = \langle S_i \rangle$ are the mean field variables and the free energy is given by

$$F(\vec{V}, T) = E(\vec{V}) + T \sum_{i=1}^{N} \left[(1 + V_i)\frac{1}{2}\log(1 + V_i) + (1 - V_i)\frac{1}{2}\log(1 - V_i)\right]. \tag{12}$$

The saddlepoints of eqn (10) are given by the mean field theory equations

$$V_i = \tanh\left(\sum_{j=1}^{N} T_{ij} V_j / T\right) \tag{13}$$

which represent steady state solutions to the RC-equations (Hopfield, 1984)

$$\frac{dU_i}{dt} = -U_i + \tanh\left(\sum_{j=1}^{N} T_{ij} V_j\right) \tag{14}$$

$$V_i = U_i / T \tag{15}$$

used by Hopfield and Tank (1985) for the TSP-problem. A straightforward iteration of eqn (13) gives

$$V_i(t + \Delta t) = \tanh\left(\sum_j^N T_{ij} V_j(t)/T\right) \tag{16}$$

and similarly for eqns (14, 15).

As is shown in Appendix B, the partition function of eqn (10) can be approximated by the value of the integrand at the saddlepoint solution ($\vec{V} = \vec{V}_0$) given by eqn (13).

$$Z \approx C e^{-F(\vec{V}_0, T)/T}. \tag{17}$$

This naive mean field theory approximation is expected to increase in accuracy as the system size N grows. However, Peterson and Anderson (1987) demonstrated that the approximation works amazingly well in the learning algorithm context even for relatively small systems (O(10) neurons). This observation is at first glance in sharp contrast with what is known from studying the dynamics of spin glasses, where this crude method gives relatively inaccurate results (Mezard et al., 1987). One must keep in mind, however, that in the learning algorithm context, relatively small discrepancies in the settling process are very likely averaged out.

The benefits of using this approximation when annealing are obvious: the CPU-demanding stochastic process is replaced by the iterative solutions to a set of coupled nonlinear equations. The smooth sigmoid function of eqn (13) "fakes" a system of binary neurons at a temperature T.

2.3. Mean Field Theory Learning

With this mean field theory approximation, the Boltzmann machine procedure above takes the following form:

Clamping phase. The stochastic unit updating of eqn (6) is replaced by solving (13), and the correlations p_{ij} are now given by

$$p_{ij} = V_i V_j. \tag{18}$$

Here (and in the next equation) we make the simplifying assumption that the true correlations V_{ij} factorize. This approximation holds very well in all the cases we have encountered so far.

Free phase. Similarly, in the free phase

$$p'_{ij} = V_i V_j. \tag{19}$$

As discussed above, the definition of the free phase can vary. In most applications of the Boltzmann machine and mean field theory to date, *free* has meant clamping only the input units (in the clamped phase, both input and output units are clamped). This is the variant used in our feature recognition studies. Alternative modes are discussed in section 4 in the context of content-addressable memory applications.

Weight updating rule. This is the same as in eqn (8), which is easier to see in the mean field theory approximation. Since P_α is independent of T_{ij}, the derivative of G with respect to the latter is given by

$$\frac{\partial G}{\partial T_{ij}} = -\frac{\partial}{\partial T_{ij}} \sum_\alpha P_\alpha \log P'_\alpha \tag{20}$$

and correspondingly for $\tilde{G}$

$$\frac{\partial \tilde{G}}{\partial T_{ij}} = -\frac{\partial}{\partial T_{ij}} \sum_\alpha Q_\alpha \sum_\beta P_{\beta|\alpha} \log P'_{\beta|\alpha} \tag{21}$$

where

$$P'_\alpha = \sum_{\vec{S} \in \{\vec{S}\}^{(io)}} e^{-E(\vec{S})/T} \Big/ \sum_{\vec{S}} e^{-E(\vec{S})/T} \tag{22}$$

and

$$P'_{\beta|\alpha} = \sum_{\vec{S} \in \{\vec{S}\}^{(io)}} e^{-E(\vec{S})/T} \Big/ \sum_{\vec{S} \in \{\vec{S}\}^{(i)}} e^{-E(\vec{S})/T}. \tag{23}$$

In eqns (22) and (23), $\{\vec{S}\}^{(i)}$ *and* $\{\vec{S}\}^{(io)}$ denote sets of $\vec{S}$ with the input and input/output units clamped, respectively. Again using the mean field theory approximation, the sums can be rewritten as integrals which in turn can be approximated by the saddlepoint values of the integrands. One gets for $\log P'_\alpha$ and $\log P'_{\beta|\alpha}$

$$\log P'_\alpha = (F - F^{(io)})/T \tag{24}$$

and

$$\log P'_{\beta|\alpha} = (F^{(i)} - F^{(io)})/T \tag{25}$$

where the free energies F, $F^{(i)}$, and $F^{(io)}$ correspond to the conditions of no units clamped, input units clamped, and input/output units clamped, respectively. With a uniform P_α (or Q_α) distribution, that is, with all environmental patterns presented with equal frequencies, eqns (22) and (23) yield (8). With these expressions for $\log P'_\alpha$ and $\log P'_{\beta/\alpha}$, explicit expressions for G and $\tilde{G}$ (eqns (2) and (3)) are easily obtained.

2.4. Choice of Parameters

To use the MFT algorithm, one needs to specify a few parameters: initial T_{ij} values, annealing schedule, learning rate, and weight updating frequency. Also, it is sometimes useful to fix the size of the weight changes. Furthermore, when using MFT, BP, and other such algorithms, there are two subtle but important issues that arise: $[-1, 1]$ versus $[0, 1]$ representation, and endpoint versus midpoint success criterion in the testing phase.

2.4.1. *Initial* T_{ij} *values.* The network is initialized to random T_{ij} values in the range $[-a, a]$. The choice of a is related to the choice of final temperature (see below), because $\langle T_{ij} \rangle$ sets the scale of the energy in eqn (1) and therefore of the temperature T for a given level of fluctuations (see eqn (4)). We use $a = 0.5$ and 1.0 throughout this work.

2.4.2 *Annealing schedule.* We use a geometric annealing schedule

$$T_0 = T_n \times k^n \tag{26}$$

where T_n is the initial temperature, T_0 is the final temperature, and $k < 1$. In principle, T_{final} should change as the learning progresses since E in (1) changes. In other words, the location of the phase transition point varies with learning (see Figure 1). We speculate that if this changing phase transition temperature were known, one could operate the algorithm at that temperature instead of annealing.

We have for simplicity chosen to use a fixed T_{final} in our applications. There is a trade-off between a slow annealing schedule (k large) with few sweeps/temperature and a fast annealing schedule (k small) with a large number of sweeps/temperature. In practice, the total number of sweeps required to achieve a certain learning quality is a constant, relatively independent of the annealing rate, provided the final temperature is chosen reasonably.

There is a more systematic way of chosing initial T_{ij} and the temperature scale.[5] For a given initial T_{ij}, compute the average $\langle \Delta E_i \rangle = \langle \Sigma_j T_{ij} V_j \rangle$ for the hidden units and set T_{initial} equal to this value. Then anneal to $T_{\text{final}} = \frac{1}{3} T_{\text{initial}}$. With this choice of $T_{\text{final}} \approx \langle \Delta E_i \rangle$, to good approximation $\Delta E/T = O(1)$. This process of choosing T_{final} is then repeated for every learning cycle. However, we have stuck to the recipe above since no quantitative improvements were achieved in our applications with this more elaborate method.

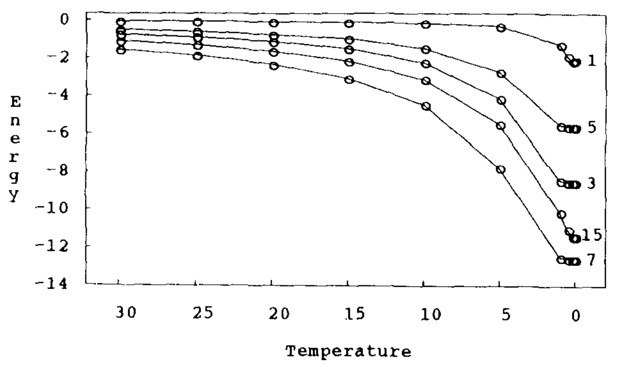

FIGURE 1. $E(T)$ for various learning passes for the 2-4-1 XOR problem.

[5] This was first suggested in Prager, Harrison, and Fallside (1986) in the context of the Boltzmann machine.

2.4.3. *Learning rate.* Ideally, one should adapt the learning rate as the learning progresses. In principle, this could be done by monitoring G after each learning pass. In the Boltzmann machine, such a calculation (see eqn (2)) is very time-consuming. In the MFT approximation on the other hand, G can be computed using the saddle point approximation in (17). However, it turns out that for the relatively small problems we deal with, the approximation is not accurate enough for this purpose; the relatively modest errors occurring when solving eqn (12) get exponentiated when constructing G. (For large enough N, however, we expect it to become feasible to compute G in this way.) We have therefore for simplicity chosen the learning rate to be either constant or monotonically decreasing in our applications.

2.4.4. *Weight updating frequency.* Ideally, perhaps, each step taken in weight space would reflect the influence of the entire training set. One expects that in most cases positive and negative contributions will result in moderately sized weight changes, but with a large number of weights and training instances the contributions can fail to balance out and some weight changes can become inappropriately large. Therefore, a balance must be found between the learning rate and the number of training examples presented between weight updates. In problems with inconsistent training, many examples should be presented between weight changes (see below).

2.4.5. *"Manhattan" updating.* In eqn (8), the weights are changed according to gradient descent, that is, steps in weight space are taken along the gradient vector—each gradient component (weight change) will be of different size. If one instead updates with a fixed step size κ,

$$\Delta T_{ij} = \kappa \cdot \text{sgn}(p_{ij} - p'_{ij}) \tag{27}$$

a step is taken in a slightly different direction along a vector whose components are all of equal size. Everything about the gradient is thrown away except the knowledge of which "quadrant" it lies in; learning proceeds on a lattice. In situations where it is advisable to present many examples before taking a step in weight space, we have found this "Manhattan" updating procedure to be beneficial. We think the reason is related to the discussion above: the gradient rule, in this situation, is likely to produce weight changes which vary greatly in magnitude, and thus finding a suitable learning rate is difficult. This is not the case with the "Manhattan" updating of eqn (27), where the weight change sizes are bounded and fixed.

2.4.6. $[-1, 1]$ *versus* $[0, 1]$ *representation.* In section 2 we used $[-1, 1]$ representation for the neurons. With a linear transformation, the whole formalism could trivially be redone for $[0, 1]$ representation with one important difference: in the $[-1, 1]$ case, both "on-on" and "off-off" correlations are counted as positive correlations in the learning rule of (8), but in the $[0, 1]$ case, only "on-on" correlations are counted (Alspector & Allen, 1987; Peterson & Anderson, 1987). For this reason we expect faster learning for both BZ and MFT when using the $[-1, 1]$ alternative. For BP, one also expects the $[-1, 1]$ representation to allow faster learning since, like BZ and MFT, this algorithm is unable to modify weights on connections from input units that are set to zero (Stornetta & Huberman, 1987). With respect to generalization power, the situation could very well be the opposite. In cases where two neurons are undecided, that is, have values near 0.0 and 0.5, respectively, no learning takes place in the $[-1, 1]$ case, whereas this "uncertainty" is emphasized with a positive correlation in the $[0, 1]$ case. In other words, one expects less "stiff" learning in the latter case and hence perhaps better generalization.

A separate issue also affects learning times: in BP, the weight update rule is proportional to the derivative $g'(x)$ of the gain function $g(x) = \tanh(x)$. This derivative is maximum at the midpoint and falls off to zero at the endpoints. Since the derivative factor causes weights to change more slowly as the unit's value moves away from the midpoint,[6] a longer learning time can be expected for BP than for MFT.[7]

2.4.7. *Endpoint versus midpoint success criterion.* As a success criterion in the learning process, a value fairly close to the target is typically demanded of both BP and MFT (e.g., $|V_i| > 0.8$ in $[-1, 1]$ representation); we call this an endpoint criterion. When testing for generalization, the question arises whether this same endpoint criterion should be used or just a midpoint criterion: V_i on the correct side of 0. It turns out that the performance of BP is sensitive to this choice while MFT is insensitive to it. When trained with endpoints as targets, MFT output units tend to take on values near the endpoints during generalization testing,[8] while BP outputs often take on intermediate values. This difference between the algorithms is very likely due to the feed-forward vs. feed-back dynamics. For either algorithm, by using a different gain during generalization than during learning, the degree of approach to the endpoints can be tuned.

2.5. Solving the MFT Equations

In Peterson & Anderson (1987) and in the applications presented in this work, asynchronous unit updating with one iteration per temperature is used. This gives convergence with a 3-digit criteria. To obtain solutions with 6-digit accuracy requires a few more sweeps/temperature (4–10 depending on problem size). These extra sweeps have very little impact on the learning process since the induced errors are of both signs and are averaged out when taking the difference in eqn (8).

We have also investigated how well the algorithm performs with synchronous updating. On the average, the system takes a factor 1.5 longer to converge (see Figure 2) than with asynchronous updating.

Thus, if one were to use synchronous updating and increase the number of iterations from 1 to 2 per temperature, the resulting learning curve should be identical to the one in the asynchronous case.[9]

The slight degradation observed when going from asynchronous to synchronous updating in the learning algorithm is very encouraging. It means that the inherent parallelism of the method can be fully exploited in Single Instruction Multiple Data (SIMD) architectures like CRAY and the Connection Machine (Blelloch &— Rosenberg, 1987). Also, it makes

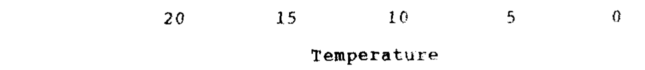

FIGURE 2. Convergence time for the 2-4-1 XOR problem with 6 digits accuracy in the free phase for synchronous versus asynchronous updating. Note that the convergence time peaks near the phase transition around $T = 1$ for both cases.

[6] Driving a BP unit's value to the endpoints for a given input is thus like driving a nail into wood: the farther it's driven, the harder it becomes to move it (in either direction).

[7] Notice, however, that if Manhattan updating is used (see section 2.4.5) the derivative factor does not come into play (see section 3.2).

[8] This is not meant to imply that MFT cannot learn analog values.

[9] This is in contrast to when the same equations were used to solve the graph bisection problem in Peterson and Anderson (1988a) where an order of magnitude difference in convergence times was observed. The origin of the different behaviors is that in the graph bisection problem one has $T_{ij} = 0$ or 1. Hence the system is more frustrated which makes it more unstable.

algorithms of this category suitable for optical implementations (Peterson & Redfield, 1988; Peterson, Redfield, Keeler, & Hartman, 1989), where synchronous updating is natural. It should be pointed out that the original Boltzmann machine with stochastic updating assumes asynchronous updating and is therefore not suitable for SIMD.

3. GENERALIZATION

The term "generalization" refers to the response of a network, after some amount of training, to novel (unlearned) inputs.[10] There are at least two different ways to test generalization:

1. Continuous learning. The training set covers the entire input space. Each time the network is presented with a training pattern it is first tested for generalization on that pattern. In this mode of operation, the distinction between learning and generalization is blurred.
2. Fixed training set. After learning a training set consisting of a fixed subset of the total input space, the network is tested for generalization on patterns it has not seen before.

We have investigated the generalization properties of MFT and BP using the two-dimensional mirror symmetry problem (Sejnowski, Kienker, & Hinton, 1986) and a statistical pattern classification task consisting of two multidimensional heavily overlapping Gaussians (Kohonen, Barna, & Chrisley, 1988). The mirror symmetry problem requires detecting which one of three possible axes of symmetry is present in a $N \times N$ pixel (binary) input (see Figure 3). The overlapping Gaussians problem consists of correctly assigning input patterns to one of two overlapping classes (see Figure 4). The statistical nature of this problem makes it particularly challenging as it necessarily involves inconsistent training.

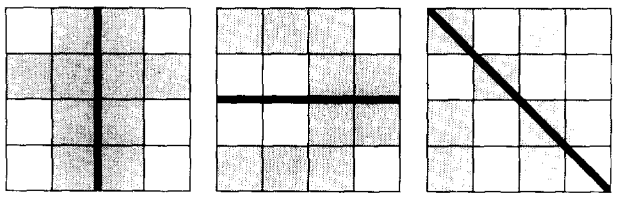
FIGURE 3. The two-dimensional mirror symmetry problem.

[10] Without an assumed interpretation ("intended model") for a syntax, there is no basis for judging generalization to be correct or incorrect (see Denker et al. (1987) for an extended discussion). We are implicitly adopting the interepretations inherent in the problem descriptions as the basis for judging the correctness of generalizations.

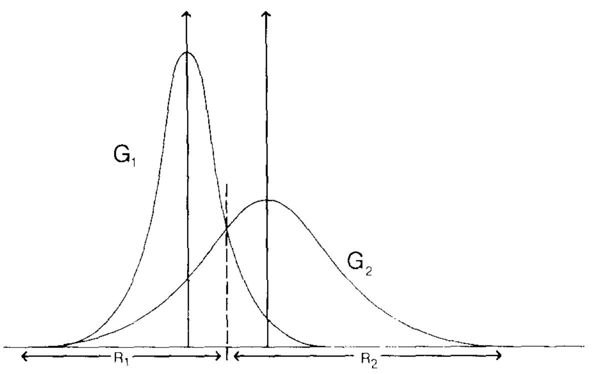

FIGURE 4. A one dimensional example of two overlapping Gaussian distributions. The non-statistical limit of the problem consists of two delta-functions located at the center of the Gaussians. Areas of misclassification are indicated.

We feel that these problems are different and difficult enough to represent suitable benchmarks. The mirror symmetry problem is characterized by a second-order predicate (Minsky & Papert, 1969; Sejnowski et al., 1986) and a very large number of possible input patterns. The overlapping Gaussians problem is an artificial abstraction of the statistical nature of many natural signal (e.g., speech) processing tasks.

3.1. The Mirror Symmetry Problem

For this problem we used the architecture of Sejnowski et al. (1986): $N \times N$ input units, one layer of 12 hidden units, and 3 output units (one for each axis of symmetry). Our experiments were performed with two problem sizes: 4×4 and 10×10.[11] In both continuous and fixed training set experiments, the weights were updated after every 5 pattern presentations. Optimal parameters were sought for each algorithm for each size problem; the same parameters were used for continuous and fixed training set runs. The parameters used are shown in Appendix A.

Only input patterns with exactly 1 of the 3 possible axes of symmetry were included. There are $\approx 1.5 \times 10^3$ such patterns in the 4×4 case, and $\approx 3.7 \times 10^{16}$ in the 10×10 case.

3.1.1. *Continuous learning.* We begin by comparing MFT with the BZ results of Sejnowski et al. (1986) on the 4×4 and 10×10 mirror symmetry problems.

[11] In order to compare our MFT results with the Boltzmann machine results of Sejnowski et al. (1986), the 3 output units in the MFT networks were interconnected. This is not (symmetrically) possible in a BP network. Tests indicated that these connections did not play an important role in network performances.

The comparisons are shown in Figure 5. As can be seen, the relative performance of the algorithms is consistent with the results of Peterson and Anderson (1987): the MFT algorithm learns faster and better than BZ.

Next we turn to comparing MFT with BP. Figure 6 shows the performance of the two algorithms for the 4 × 4 and 10 × 10 cases using the endpoint criterion for generalization tests, and in Figure 7 MFT and BP are compared for 4 × 4 using the midpoint criterion.

A few observations can be made from Figures 6 and 7: First, as discussed above, the relative performance of BP improves significantly with the midpoint criterion, whereas MFT is virtually unaffected. Second, even with use of the midpoint criterion, BP (using [0, 1]) lags behind MFT (using [− 1, 1]). Note finally that any difference in performance between the algorithms decreases as learning progresses.

Since continuous learning blurs the distinction between learning and generalization, we now turn to fixed training set experiment.

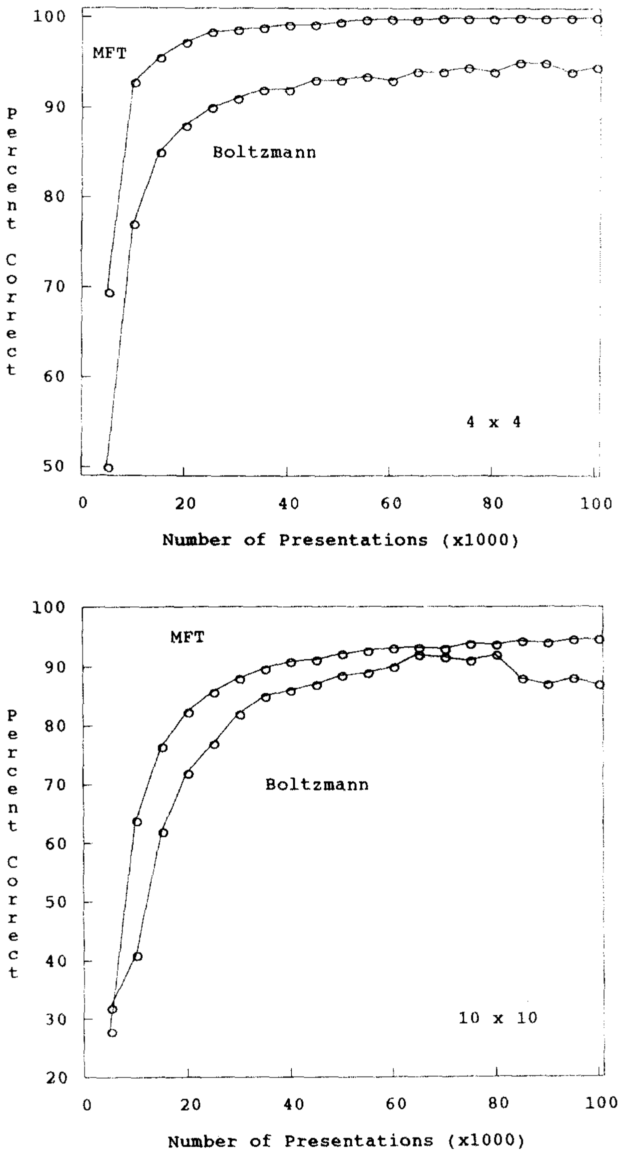

FIGURE 5. Learning curves for the 4 × 4 and 10 × 10 mirror symmetry problems with 12 hidden units for MFT and Boltzmann machine, respectively. Parameters used in the simulations are found in Appendix A. For the MFT algorithm, [− 1, 1] representation was used. Boltzmann machine curves are from Sejnowski et al. (1986).

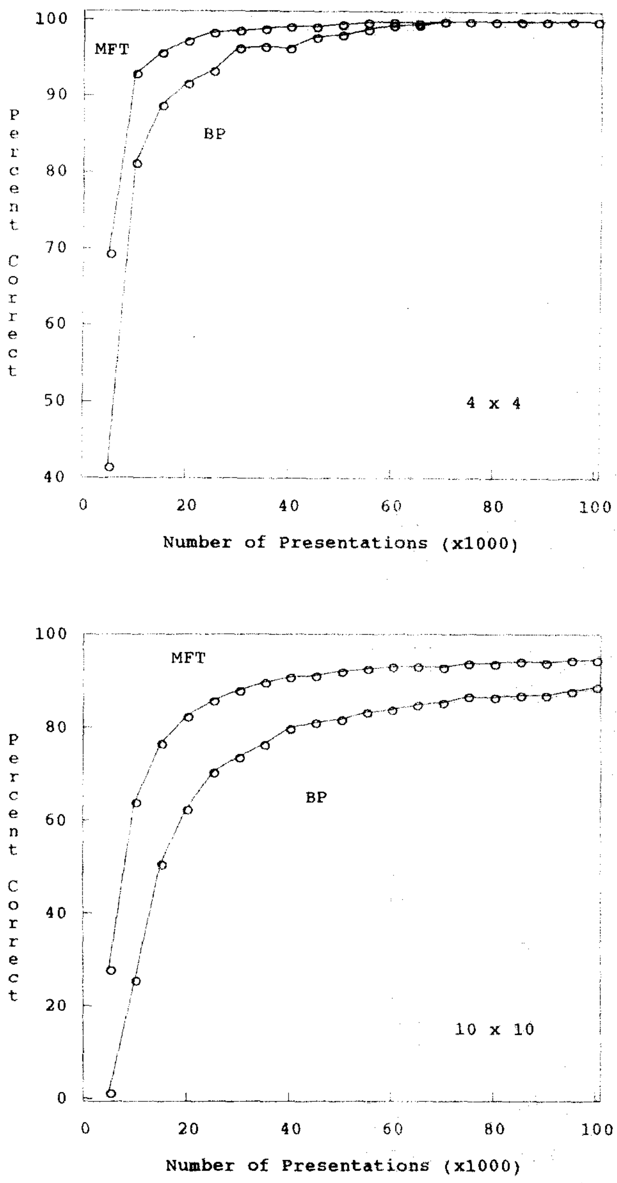

FIGURE 6. Learning curves using the *endpoint* criterion for the 4 × 4 and 10 × 10 mirror symmetry problems with 12 hidden units for MFT and BP. For MFT, [− 1, 1] representation was used; for BP, [0, 1]. Parameters used in the simulations are in Appendix A.

3.1.2. *Fixed training set learning.* In an attempt to thoroughly explore how the choice of representation

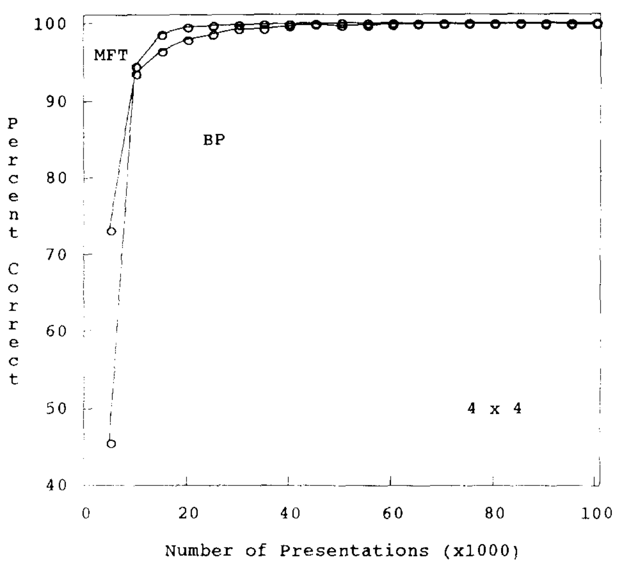

FIGURE 7. Learning curves using the *midpoint* criterion for the 4 × 4 mirror symmetry problem with 12 hidden units for MFT and BP. For MFT, [−1, 1] representation was used; for BP, [0, 1]. Parameters used in the simulations are in Appendix A.

affects the algorithms on this problem, MFT and BP networks for the 4 × 4 problem were trained on six different sets of 100 patterns. In addition to the pattern sets varying with respect to [0, 1] vs. [−1, 1], they varied with respect to the average number of bits that were on (see Table 1). Each pattern set was generated randomly without duplicates and was used for both learning algorithms. Each training task was repeated 10 times with different initial conditions. For testing generalization, test sets of 100 unique random patterns were generated without intersecting the training sets, and each network was tested on the appropriate test set after correctly learning 100% of its training set. The midpoint criterion was used throughout generalization testing. The results are summarized in Table 1.

From Table 1 we can draw the following observations and conclusions:

1. MFT learns faster than BP, as expected from the discussion in section 2.4.6. The two algorithms do equally well on generalization on [0, 1], and MFT does somewhat better on [−1, 1].[12]

TABLE 1
Training and Generalization Performance of MFT and BP for the 4 × 4 Mirror Symmetry Problem for Six Different Training Sets, Each of Size 100

		A1		B1		C1	
		epochs	genlz	epochs	genlz	epochs	genlz
MFT	−1,1	44	63	34	62	36	63
	0,1	51	68	68	67	54	70
BP	−1,1	80	51	113	54	55	57
	0,1	299	69	266	67	281	69
		A2		**B2**		**C2**	
		epochs	genlz	epochs	genlz	epochs	genlz
MFT	−1,1	30	55	27	56	36	69
	0,1	39	57	42	57	44	69
BP	−1,1	210	46	195	45	88	62
	0,1	234	59	216	59	289	72
		Avg A		**Avg B**		**Avg C**	
		epochs	genlz	epochs	genlz	epochs	genlz
MFT	−1,1	37	59	31	59	36	66
	0,1	45	63	55	62	49	70
BP	−1,1	145	49	154	50	72	60
	0,1	267	64	241	63	285	71

In a given training set, the [−1, 1] and [0, 1] patterns differed only in whether 0 or −1 was used for bits which were "off." The six different training sets were created as follows:
A1 and A2 were generated with each bit having a 0.4 probability of being on.
B1 and B2 were the complements of the patterns in sets A1 and A2, respectively.
C1 and C2 were generated with each bit having a 0.5 probability of being on.
Generalization pattern sets were also of size 100. A single generalization set (with the same average number of on bits) was used to test training sets A1 and A2; these three sets were non-intersecting. Similarly for sets B1 and B2, and for sets C1 and C2. Numbers of epochs and generalization percentages shown each represent the median value of 10 different runs.

2. For both algorithms, learning is faster with [− 1, 1] than [0, 1], also as expected from the discussion in section 2.4.6.
3. For both algorithms generalization is more powerful with [0, 1] than [− 1, 1], consistent with the same discussion.[13]
4. MFT appears less sensitive to the representation choice than BP.[14]

Other experiments, not reported here in detail, confirm the picture that emerges from Table 1. These include experiments with the clumps problem (Denker et al., 1986) and mirror symmetry experiments with fixed training sets of size 200 and 500.

3.2. A Statistical Pattern Recognition Problem

Most neural network applications to date have been in the area of non-statistical problems. However, many natural problems entail noisy and inconsistent training. The success of the neural network technology will therefore be judged largely according to its ability to deal with statistical problems. Kohonen et al. (1988) benchmarked the backpropagation, Boltzmann machine, and Learning Vector Quantization algorithms for testbeds consisting of heavily overlapping Gaussian distributions with dimensionality ranging from 2 to 8 (see Figure 4).

The potential difficulty in this problem lies in the presence of inconsistent training: for inputs where the two Gaussians overlap, the training examples map the same inputs to both of two different outputs. This is in contrast to the mirror symmetry problem discussed above and the delta function limit in Figure 4 where the same training output is consistently associated with a given input.

In Kohonen, Barna, and Chrisley (1988), the neural network algorithms generally produced good results, with the Boltzmann machine performing very close to the theoretical Bayesian limit.

We have compared the performance of BP and MFT with the theoretical limit using three of the testbeds of Kohonen et al. (1988). The first case consists of two overlapping Gaussians (G_1, G_2) in 8 dimensions, centered around (0, 0, 0, . . . , 0) and (2.32, 0, 0, . . . , 0) with standard deviations σ equal to 1 and 2, respectively. At the theoretical minimum (Bayesian limit), one has (Duda & Hart, 1973; Kohonen et al. 1988)

$$P_{\text{error}} = \int_{R_1} G_2 + \int_{R_2} G_1 \tag{28}$$

where R_1 and R_2 are chosen such that P_{error} is minimized. In Figure 4, the optimal choice of R_1 and R_2 is illustrated in two dimensions. For the above Gaussians, $P_{\text{error}} = 0.062$; that is, a 93.8% success rate is the maximum achievable. In the second case, the difficulty of the problem is increased by using identical means for the two classes: G_2 is shifted to (0, 0, 0, . . . , 0). The maximum success rate for this case is 91.0%. In the third and most difficult case, there are only two input dimensions instead of eight, and again the two classes have identical means (of (0, 0)). The maximum success rate for this case is only 73.6%.

The details of our MFT and BP simulations can be found in Appendix A. Here we list a few additional technicalities that are important when comparing our results to those of Kohonen et al. (1988) or when aiming for peak performance. In our experiments, both algorithms used [0, 1] units, the midpoint was used as the correctness criterion (see section 2.4.7), and in all cases there were 8 hidden units and 1 output unit.[15]

- Architecture. In order to fully explore the capabilities of the neural network algorithms in this application, we used two different architectures: fully connected (full) and layer-to-layer connected (layer). In the fully connected architecture, all connections except input-input connections were present. (In BP, the hidden units cannot be symmetrically interconnected, so they were connected asymmetrically: imagining the hidden units in a row, each hidden unit was connected to all hidden units to its right.) In the layer-to-layer architecture, only input-hidden and hidden-output connections were present.
- Encoding of input values. As in Kohonen, Barna, & Chrisley (1988), we used two alternatives for encoding the D-dimensional input data: D continuous units (cont) and $D \times 20$ digitized (binary) units (dig). In the latter case, each continuous input was subdivided into 20 subranges and a local representation was used: an input pattern consisted of exactly one unit on in each of the D sets of 20 units.
- Parameters. The learning rates and other parameters are found in Appendix A. In Table 2, (std) indicates that the standard gradient following rule

[12] Varying the gain for BP in the [− 1, 1] case (fixed for a given run) did not improve generalization.
[13] An opposite result is reported in Ahmad and Tesauro (1988) for the "majority" problem.
[14] Varying the gain for BP in the [− 1, 1] case (fixed for a given run) did not improve generalization.
[15] No change in performance was observed in trials using 2 output units.

TABLE 2
Peak Performance and Number of Patterns Required for the Statistical Pattern Classification Problems (see text)

Case 1: Theoretical maximum = 93.8%

			MFT		BP		BZ*	
Input	Learning	Connections	%	patts	%	pats	%	patts
dig	man	full	93.2	140k	93.2	120k	93.3	?
dig	std	full	92.9	130k				
dig	man	layer	92.6	80k				
dig	std	layer	90.5	140k				
cont	man	layer			92.2	320k		
cont	man	full			92.1	390k		

Case 2: Theoretical maximum = 91.0%

			MFT		BP		BZ*	
input	learning	connections	%	patts	%	patts	%	patts
dig	man	full	90.0	170k	90.7	160k	90.6	?

Case 3: Theoretical maximum = 73.6%

			MFT		BP*		BZ*	
input	learning	connections	%	patts	%	patts	%	patts
dig	man	full	73.3	60k			73.5	?
cont	std	layer			73.7	?		

Percentages in asterisked columns are taken from Kohonen et al. (1988). Percentages from the present study are with respect to the preceding 10,000 patterns.

was used in updating the weights. "Manhattan" learning (man) was described in section 2.4.5.

In Table 2 we show the peak performance and number of training patterns required for some of the options described above. Using the optimal options, both MFT and BP essentially reach the theoretical limit.

Table 2 also illustrates another effect of Manhattan updating: in this case the derivative factor does not affect weight change sizes in BP, and hence learning proceeds as rapidly as with MFT (see the discussion in section 2.4.6).

In Table 2, data in asterisked columns was taken from Kohonen et al. (1988). Not shown in Table 2 is the performance reported in Kohonen et al. (1988) for BP in cases 1 and 2: 88.7% and 81.1% respectively. The origin of those low values was that only layer-to-layer connections were used with the continuous input option. Also, "Manhattan" updating was not used.[16] (BP did well on case 3 (low input dimensionality) in Kohonen et al. (1988), as is shown in Table 2.) It is clear from Table 2 that both MFT and BP are quite successful regardless of architectural and encoding details. Performance using the optimal techniques is very impressive given that this problem is non-trivial.

[16] We acknowledge Teuvo Kohonen and Gyorgy Barna for kindly communicating the details of their simulations to us.

4. A CONTENT ADDRESSABLE MEMORY WITH MEAN FIELD THEORY LEARNING

4.1. Content Addressable Memory versus Feature Recognition

Our applications so far have been in the area of feature recognition where a functional mapping from input to output units takes place (see Figure 8a). Both bidirectional (e.g., MFT) and feed-forward (e.g., back propagation) algorithms are suited to this paradigm. A feature recognizer can be viewed as a spe-

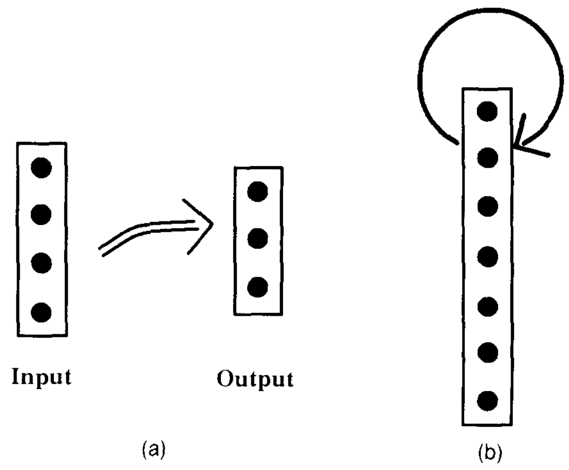

FIGURE 8. (a) A feature recognizer (b) A content addressable memory.

cial case of content addressable memory. Here, the visible units are not partitioned into input units and output units (see Figure 8b). Hence, while bidirectional (e.g., MFT) networks are appropriate for CAM, backpropagation is not (see footnote 2).

In discussing CAM, it is useful to distinguish three functional possibilities. Assume that a network has learned (stored) M N-bit patterns:

1. *Error-correction-CAM.* The network is initialized (but not permanently clamped) to a noisy version of a pattern, that is, some bits are changed at random, unknown positions. This initial state evolves to the stored pattern (the network moves to the closest attractor), correcting the random errors.

The remaining two functional possibilities correspond respectively to training and generalization patterns in feature recognition:

2. *Partial-contents-CAM.* In feature recognition, if the network is clamped to the input portion of a training pattern, the output is retrieved. Partial-contents-CAM generalizes this by allowing any subset of visible units to be clamped as input units. In contrast to error-correction-CAM, there is perhaps nothing to be gained by using a neural network for partial-contents-CAM, since conventional computer hardware can be constructed to perform this type of "table-lookup" retrieval in parallel (Kohonen, 1987).

3. *Schemata-completion.* In feature recognition generalization testing, the input units are clamped to a novel pattern. Schemata-completion generalizes this by allowing any subset of visible units to be clamped to a novel pattern. This case is distinguished here from partial-contents-CAM in that the patterns clamped do not occur during training, so that instead of retrieving a stored memory, generalization occurs in response to novel stimuli (Rumelhart, Smolensky, McClelland, & Hinton, 1986). Unlike the case of partial-contents-CAM, neural networks make sense for schemata-completion. Of course, partial-contents-CAM can be viewed as the trivial or default case of schemata-completion.

In this paper, by CAM we mean error-correction-CAM.

Bidirectional models, where the notion of separate input and output units is not inherent, are suitable for CAM (Anderson, 1970; Kohonen, 1988a). One such model is the Hopfield model (Hopfield, 1982).

4.2 The Hopfield Model

The architecture of the Hopfield model consists of N fully connected visible units. The storage is Hebbian (Hebb, 1949)

$$T_{ij} = \sum_{p}^{M} S_i^p S_j^p \qquad (29)$$

where M is the number of N-bit patterns ($\vec{S}^p$) to be stored. The dynamics is governed by eqn (13) with $V_i = S_i$ and $T = 0$ (binary threshold units). With the storage prescription of (29) the system has the stored patterns $\vec{S}^p$ as attractors. For uncorrelated (random) patterns, the storage capacity M_{max} is given by (Amit, Gutfreund, & Sompolinsky, 1985)

$$M_{\max} \approx 0.14 \times N \qquad (30)$$

(For correlated patterns this number is smaller.) If M exceeds $M_{\max}$, so-called spurious states appear in addition to the stored ones, causing the performance to deteriorate.

4.2.1. *Improvements on the Hopfield Model.* Various modifications of eqn (29) have been suggested to improve the storage capacity. They fall into two classes: local (Hopfield, Feinstein, & Palmer, 1983; Wallace, 1986) and non-local (Kanter & Sompolinsky, 1987). We briefly mention two modifications which are local: REM sleep (Hopfield, Feinstein, & Palmer, 1983) and the Bidirectional Perceptron Learning Algorithm (Wallace, 1986; Bruce, Canning, Forrest, Gardner, & Wallace, 1986); both are related to MFT learning.

It has been suggested that during REM sleep, mammals "dream in order to forget" as a means for removing spurious undesired memories (Crick & Mitchison, 1983). It was demonstrated in Hopfield et al. (1983) that if the storage rule of eqn (29) is supplemented by "unlearning" of the spurious states

$$\Delta T_{ij} = -\gamma \sum_{\beta}^{M_s} S_i^\beta S_j^\beta \qquad (31)$$

the CAM performance improves. In eqn (31), γ is a parameter and M_s is the number of spurious states $\vec{S}^\beta$. Subsequent work has verified the power of this method with storage capacities in the range 30%–40% as a result (Kleinfeld & Pendergraft, 1987). This "unlearning" procedure is closely related to the Boltzmann machine learning prescription of eqn (8); positive learning (29) corresponds to the clamped phase whereas "unlearning" corresponds to the free phase (Sejnowski et al., 1986). Since MFT relies on the same learning rule (8) it effects the same "pruning" of state space or "sculpting" of the energy landscape.

The Bidirectional Perception Learning Algorithm (Wallace, 1986; Bruce et al, 1986) uses the same unit updating rule as the Hopfield model, allows visible units only, and is a direct extension of the perceptron algorithm (Minsky & Papert, 1969) to bidirectional networks. The learning process goes as follows. For each pattern p and unit i, the error ε_i^p is recorded

$$\varepsilon_i^p = \frac{1}{2}\left[1 - \operatorname{sgn}\left(S_i^p \sum_j T_{ij} S_j^p\right)\right]. \qquad (32)$$

The weights are then updated according to

$$\Delta T_{ij} = \frac{1}{N}\sum_{p}(\varepsilon_i^p + \varepsilon_j^p)S_i^p S_j^p. \qquad (33)$$

This process is repeated until all errors are corrected. Storage of up to N random patterns (with negligible basins of attraction, however, Forrest, 1988) has been achieved with this method. A variant of this algorithm (Diederich & Opper, 1987; Gardner, 1987; Krauth & Mezard, 1987) produces non-negligible basins of attraction. In Appendix C, we show that these algorithms are specials cases of MFT with $T = 0$ and no hidden units.

4.3 A Content Addressable Memory with Mean Field Theory Learning

We now depart from the Hopfield content addressable memory in the sense that hidden units will be used to build up internal representations of the stored states. (The Hopfield model has only visible units.) For an N-bit memory, the architecture consists of N visible units and N hidden units and is completely connected. (The number of hidden units was chosen arbitrarily for this preliminary study.) Because of the presence of hidden units as well as very different learning and retrieval procedures (see below), analytical results such as eqn (30) and similar calculations for the bidirectional perceptron algorithm (Gardner, 1987) cannot be expected to apply.

Learning takes place by presenting M N-bit patterns to the network using the MFT learning algorithm. In the algorithm as described in previous sections, the input units were always clamped, as they were when those networks were subsequently operated. In CAM operation this is not the case, and all visible units must be trained to respond correctly when unclamped; hence a different procedure is called for. We now describe one such procedure.

4.3.1. *A MFT learning procedure for CAM.* In this procedure, the clamped phase operates as in the future recognition case—all visible units are clamped to the training pattern. In each free phase, however, instead of clamping a fixed set of visible units (the input units), one-half of the visible units are chosen at random and are clamped. In this way, no visible unit is always clamped during training, and the problems associated with clamping to units at all during the free phase are avoided as well.[17]

A natural convergence criterion for this learning process would be:

$$G < \varepsilon. \qquad (34)$$

However, as discussed in section 2, computing G in the MFT approximation does not give very accurate results for the small systems considered here. Therefore we have instead used as a criterion

$$|\Delta T_{ij}|_{\max} < \varepsilon \qquad (35)$$

but in practice learning was continued until retrieval performance effectively ceased to improve (see Table 3).

There are different ways to study the capacity and performance of a CAM. One is to initialize the network with a stored pattern, turn on the dynamics of the network, and study the distance in terms of bit errors between the stored pattern and the final state. For a perfect CAM, the final state should of course be identical to the stored pattern. However, this kind of test does not describe the efficiency of the CAM in terms of attraction radii (error-correction) for the different stored patterns. To probe this question of content-addressability, one can initialize the network B bits in distance from a stored pattern and again measure the number of bit errors in the final state.

These schemes for investigating the CAM are typical for Hopfield models (Hopfield, 1982; Kleinfeld & Pendergraft, 1987) where all units are visible. To deal with the existence of hidden units, these procedures require modification.

Our CAM was examined as follows.[18]

1. The N visible units are clamped to the initial state, being either a stored memory or B bits in distance from a stored memory.
2. Annealing begins but is not completed. The state of the hidden units begins to approximate the learned internal representation of the stored state.
3. The visible units are released.
4. Annealing is completed. With all units now free, the network settles as a whole with the evoked representation in the hidden units helping the visible units settle into the stored state.

TABLE 3
Number of MFT epochs executed in storing *M* 32-bit patterns in a CAM network of size 2 × 32

M	Epochs
32	600
64	1100
128	1800

[17] The Boltzmann machine learning algorithm as originally proposed prescribes clamping no units at all during the free phase. This procedure yields very poor CAM performance in MFT networks. We suspect that the random clamping procedure would improve CAM learning for error-correction in the Boltzmann machine as well.

[18] This procedure was used by Touretzky and Geva (1987) in a different context and without hidden units.

Learning parameters can be found in Appendix A. In Figure 9 we show the distribution of errors in the final state for $M = 32$, 64, and 128 for $N = 32$ when the visible units are initially clamped to one of the stored states. In Figures 10 and 11 the corresponding graphs are shown for the case when the network is initially clamped $B = 2$ and $B = 4$ bits away from a stored pattern.

A precise evaluation of the loading capacity of this network (which has an arbitrarily chosen number of N hidden units) would require the gathering of extensive additional statistics. However, as a rough characterization, Figure 9 indicates that this network's loading capacity is very approximately given by

$$M_{\max}^{\text{rand}} \approx 4 \times N \tag{36}$$

where N is the number of visible units, which greatly exceeds the corresponding number for the Hopfield network of eqn (30).

Some comments:

- Storage capacity. We have not yet explored the upper limits of storage capacity. What are the fundamental limits? Except for the bound set by the semantics of the task, there are no clear absolute theoretical constraints since the hidden units and weights are analog. In practice, the limits will be set by the technologies used in the hardware implementations, for example, the precision level of the operations on the resistors representing the T_{ij}-elements.
- Learning rates. It is clear from Table 3 that while this algorithm is optimizing the number of stored patterns and their content-addressability, in its current form it is paying the price in the required number of learning cycles.

The results of this initial exploration of a CAM using MFT are very encouraging. It is clear, however, that many issues must be dealt with in future work: How does the performance scale with N? With the number of hidden units? What is the optimal architecture (number of layers, degree of connectivity, etc.)? What free phase clamping process is optimal? Can the learning rates of Table 3 be improved? How does the algorithm behave when no hidden units are used? What are the storage and retrieval properties for correlated patterns? These and other issues are currently being investigated (Hartman, 1988).

5. SUMMARY

5.1 MFT versus BP on Feature Recognition

We have benchmarked MFT against BP on two different testbeds: the two-dimensional mirror symmetry problem and a statistical pattern classification problem consisting of two multidimensional overlapping Gaussians. Our main results can be summarized as follows:

- The generalization capabilities of the two algorithms are similar. When the algorithms are used in their standard forms, MFT learns in a substantially smaller number of training epochs than BP; when "Manhattan" learning is used the learning speeds are similar.
- The performance of both algorithms on the problem of two overlapping Gaussians is particularly

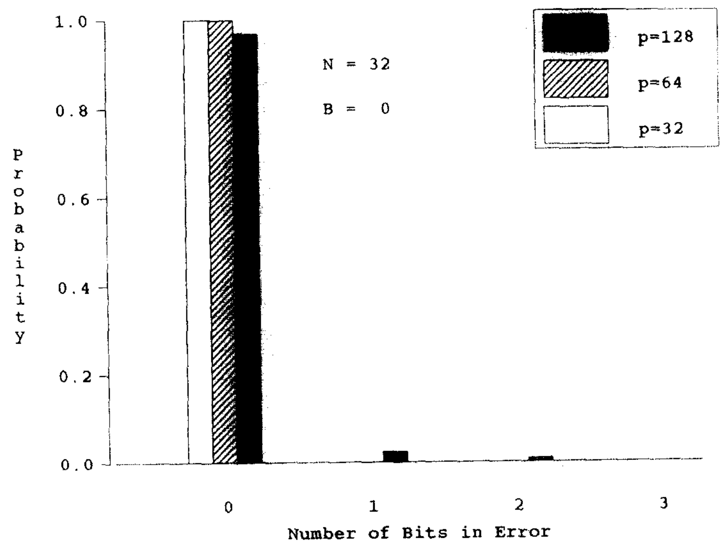

FIGURE 9. Distribution of final state errors for M = 32, 64, and 128 for N = 32 with visible units initially clamped to a stored pattern.

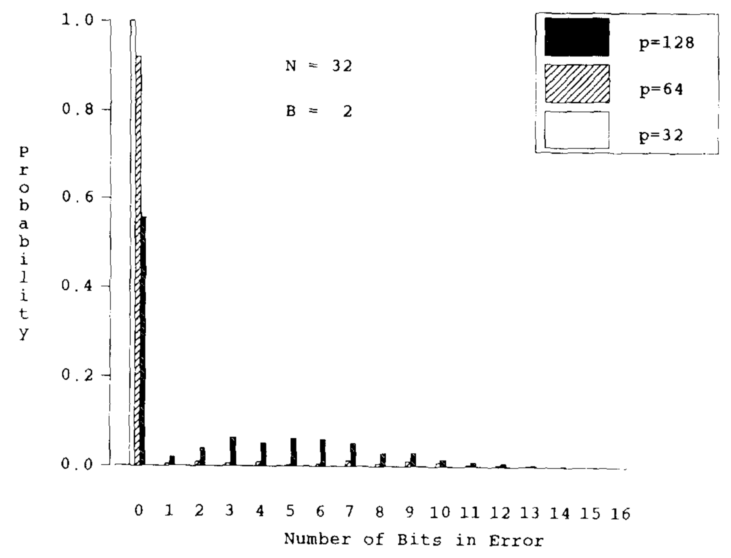

FIGURE 10. Distribution of final state errors for M = 32, 64, and 128 for N = 32 with visible units initially clamped to a state B = 2 bits in distance from a stored pattern.

impressive since it involves inconsistent training; the Bayesian theoretical limit is essentially reached!

In this context we also made the following observations:

- When it is advisable for learning to occur over many averaged training patterns, as in the overlapping Gaussians problem, it is crucial to use "Manhattan" updating, that is fixed step size weight updates, in order to prevent some weight changes from growing inappropriately large, thus preventing the optimal region of weight space from being located.
- The question of using [0, 1] or [−1, 1] representation is more than a syntactic one. For both algorithms, [0, 1] representation yields slower learning but better generalization than [−1, 1].

We have limited ourselves to comparing MFT with BP. The reason for this limitation is the popularity of BP in the research community. There also exist alternative approaches such as Learning Vector Quantization (LVQ) (Kohonen, 1988a, 1988b). For a comparison of BP, BZ, and LVQ on the overlapping Gaussians problem we refer the reader to Kohonen et al. (1988).

Our results for BP on this problem differ substantially from those of Kohonen et al. (1988). Three

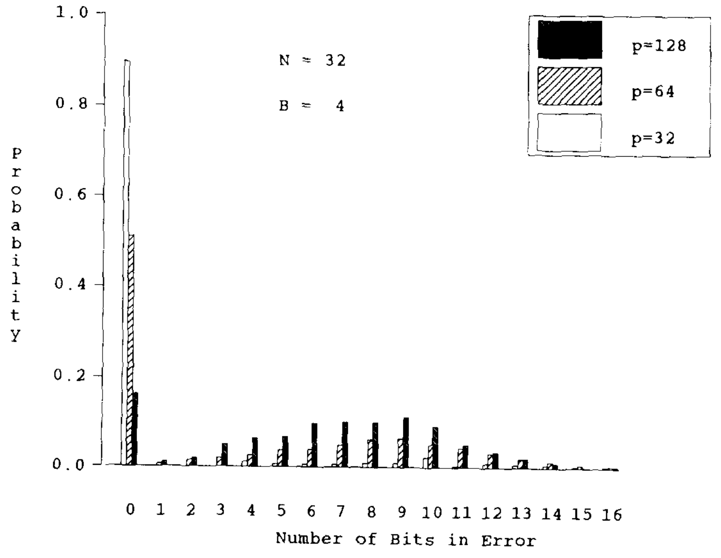

FIGURE 11. Distribution of final state errors M = 32, 64, and 128 for N = 32 with visible units initially clamped to a state B = 4 bits in distance from a stored pattern.

factors are responsible: (a) We use "Manhattan" learning, which seems to be necessary in this case; (b) we use full (feed-forward) connections; and (c) we represent each continuous input value with multiple binary units. These factors differ from Kohonen et al. (1988) which (c. Barna, personal communication, 1988)(a) used the usual gradient-following descent, (b) used strictly layered connectivity, and (c) represented each continuous input value in a single continuous unit.

One might wonder why MFT and BP behave so similarly despite their different foundations. As discussed in Hopfield (1987), they are related in the limit of high temperature (linear gain functions). Consider the BP updating rule for an output unit V_i^{out}

$$\Delta T_{ij} = \eta(V_i^{\text{target}} - V_i^{\text{out}})V_j g'(I_i) \qquad (37)$$

where $g(x)$ is the gain function (cf. $\tanh(x)$) and I_i is the input to unit i. With g' = constant eqn (13) reads

$$\Delta T_{ij} = \eta(V_i^{\text{target}}V_j - V_i^{\text{out}}V_j) \qquad (38)$$

which is closely related to eqn (8), given that "target" and "out" here roughly correspond to "clamped" and "free" phase, respectively.

To summarize our results we find MFT and BP almost comparable in performance with an edge for MFT. The MFT learning algorithm also appears more attractive to us for the following reasons:

- Future studies should focus on the issue of scaling. Intuitively, one might expect MFT learning to have better scaling properties than BP, since in MFT the effective "errors" are calculated locally after propagating activations (cf. eqns (18, 19)), whereas in BP errors are registered in the output layer and are explicitly propagated to the various links. It would not be surprising if in very large BP networks it is difficult for the error to propagate down without diffusion to change the appropriate links. It is difficult to find a good testbed for systematic scaling studies; it is important to have a "minimal" architecture for a given problem.[19] This question of minimal architecture is not decoupled from the properties of the learning algorithm. With a powerful algorithm we would expect unnecessary links to be efficiently put to zero in the learning process. Neither BP nor MFT is very powerful in this respect. A cure for the BP algorithm by adding terms to the error function has recently been suggested (D. E. Rumelhart, personal communication, 1988) and similar refinements can be made to the MFT algorithm.
- Present learning algorithms are far too crude to claim substantial similarities with biological systems. However, we feel that MFT is somewhat more biologically plausible than BP, since changes to connection strengths in MFT depend directly only on the states of the neighboring "neurons," while such changes in BP depend on explicitly propagated error values.
- MFT lends itself naturally to hardware implementations. The real promise for the neural network paradigm lies in its potential for custom-made hardware. It would facilitate the design and prototyping of such hardware if a strong common ingredient existed for different application areas: feature recognition, content-addressable memory, and optimization. The MFT algorithm has an advantage in this respect. For an MFT network, learning either content addressable memories or feature recognition problems is equally natural. And, with the learning turned off, the dynamics of MFT is identical to that of optimization problems (Hopfield & Tank, 1985; Peterson & Anderson, 1988).

VLSI implementation. In Figure 12 we show a VLSI implementation of a fully connected network (Hopfield & Tank, 1985). The RC-equations governing its dynamics for C_i and $R_i = 1$ are given by (13). Hence the circuit of Figure 12 will converge to the MFT solutions. That is, the circuit equations are isomorphic to the MFT equations. There is no need for additional circuitry to carry out the arithmetic calculations. For learning purposes, local products of the amplifier values must be formed. It has been demonstrated that this is feasible for the original

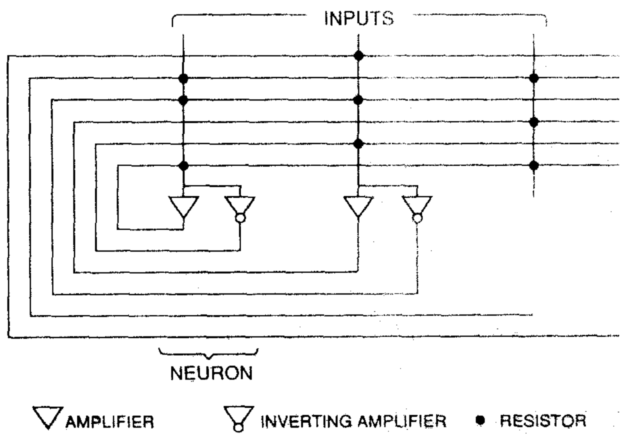

FIGURE 12. A VLSI implementation of a fully connected network.

[19] A widely cited scaling failure in neural network models is a study of the 4-4-4-4-4 encoder problem (Ballard, 1987) with the BP algorithm, where poor performance was observed as compared with the 4-4-4 problem. (We observe similar results using the MFT algorithm. We attribute this failure to the abundance of degrees of freedom compared to the minimal requirements of the problem.)

Boltzmann machine (Alspector & Allen, 1987; Alspector, Allen, Hu, & Satyanarayana, 1988). No correspondingly natural implementation exists for the BP algorithm.

Optical Implementation. Matrix multiplication, which plays a central role in neural network computations, is natural for optics. Both spatial light modulator and photorefractive crystals have been suggested as T_{ij}-masks (Anderson, 1986; Psaltis & Farhat, 1985; Soffer, Dunning, Owechko, & Marom, 1986). The latter technology is particularly appealing for implementing neural networks with local learning schemes: the T_{ij} elements are modified with holographic two-wave mixing $\Delta T_{ij} \propto V_i V_j$ (cf. eqns (18, 19)). In Peterson and Redfield (1988), MFT is implemented using a single crystal architecture. This is in contrast to work with BP where one crystal for each hidden layer has been used (Psaltis, Yu, Gu, & Lee, 1987). Both MFT and BP are implemented in a single crystal architecture in Peterson et al. (1989); the BP implementation is necessarily somewhat less natural and less straightforward than the MFT implementation.

5.2 A Content Addressable Memory with MFT Learning

Since there is no inherent distinction between input and output units in MFT, its applications are not limited to feature recognition problems. In section 4 it was convincingly demonstrated how to construct a CAM by building up internal representations of the patterns to be stored using MFT. With analog neurons and weights, no clear theoretical limit on the storage capacity exists (except that set by the semantics of the task), and indeed we were able to store N patterns with N visible and N hidden units with very good retrieval properties. At $4N$ patterns the network's storage capacity was saturated. So far, our performance tests have been limited to uncorrelated (random) patterns. It will be interesting to see how this novel architecture and algorithm for CAM performs for correlated patterns (Hartman, 1988).

REFERENCES

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, **9**, 147–169.

Ahmad, S., & Tesauro, G. (1988). Scaling and generalization in neural networks: A case study. In D. Touretzky, G. Hinton, & T. Sejnowski (Eds.), *Proceedings of the 1988 Connectionist Models Summer School, Carnegie Mellon University*. San Mateo, CA: Morgan Kaufmann.

Alspector, J., & Allen, R. B. (1987). A neuromorphic VLSI learning system. In P. Losleben (Ed.), *Proceedings of the 1987 Stanford Conference: Advanced Research in VLSI*. Cambridge: MIT Press.

Alspector, J., Allen, R. B., Hu, V., & Satyanarayana, S. (1988). Stochastic learning networks and their electronic implementation. In D. Z. Anderson (Ed.), *Neural information processing systems*. New York: American Institute of Physics.

Amit, D. J., Gutfreund, H., & Sompolinsky, H. (1985). Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, **55**, 1530–1533.

Anderson, D. Z. (1986). Coherent optical eigenstate memory. *Optics Letters*, **11**, 56–58.

Anderson, J. A. (1970). Two models for memory organization using interacting traces. *Mathematical Biosciences*, **8**, 137–160.

Ballard, D. H. (1987). Modular learning in hierarchical neural networks. (Tech. Rep. No. NSF-ITP-87-69). Santa Barbara, CA: Institute for Theoretical Physics.

Baum, E. B., & Wilczek, F. (1988). Supervised learning of probability distributions by neural networks. In D. Z. Anderson (Ed.), *Neural information processing systems*. New York: American Institute of Physics.

Blelloch, G., & Rosenberg, C. (1987). Network learning and the Connection Machine. In J. McDermott (Ed.), *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 323–326). Los Altos, CA: Morgan Kaufmann.

Bruce, A. D., Canning, A., Forrest, B., Gardner, E., & Wallace, D. J. (1986). Learning and memory properties in fully connected networks. In J. S. Denker (Ed.), *Proceedings of the Conference on Neural Networks for Computing, Snowbird, UT*. New York: American Institute of Physics.

Crick, F., & Mitchison, G. (1983). The function of dream sleep. *Nature*, **304**, 111–114.

Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L., & Hopfield, J. (1987). Large automatic learning, rule extraction and generalization. *Complex Systems*, **1**, 877–922.

Diederich, S., & Opper, M. (1987). Learning of correlated patterns in spin-glass networks by local learning rules. *Physical Review Letters*, **58**, 949–952.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Forrest, B. M. (1988). Content-addressability and learning in neural networks. *Journal of Physics A:'Math. Gen.*, **21**, 245–255.

Gardner, E. (1987). Maximum storage capacity in neural networks. *Europhysics Letters*, **4**, 481–485.

Glauber, R. J. (1963). Time-dependent statistics of the Ising model. *Journal of Mathematical Physics*, **4**, 294–307.

Gorman, R. P., & Sejnowski, T. J. (1988). Learned classifications of sonar targets using a massively-parallel network. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **36**, 1135–1140.

Hartman, E. (1988). *Content-addressable memories in neural networks using the mean field theory learning algorithm*. Unpublished dissertation proposal, University of Texas.

Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science, USA*, **79**, 2554–2558.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two state neurons. *Proceedings of the National Academy of Science, USA*, **81**, 3088–3092.

Hopfield, J. J. (1987). Learning algorithms and probability distributions in feed-forward and feed-back networks. *Proceedings of the National Academy of Science, USA*, **84**, 8429–8433.

Hopfield, J. J., & Tank, D. W. (1985). Neural computations of decisions in optimization problems. *Biological Cybernetics*, **52**, 141–152.

Hopfield, J. J., Feinstein, D. I., & Palmer, R. G. (1983). Unlearning has stabilizing effects in collective memories. *Nature*, **304**, 158–159.

Kanter, I., & Sompolinsky, H. (1987). Associative recall of memory without errors. *Physical Review*, **A35**, 380–392.

Kleinfeld, D., & Pendergraft, D. B. (1987). Unlearning increases the storage capacity of content addressable memories. *Biophysical Journal*, **51**, 47–53.

Kohonen, T. (1987). *Content-addressable memories* (2nd ed.). New York: Springer-Verlag.

Kohonen, T. (1988a). *Self-organization and associative memory* (2nd ed.). New York: Springer-Verlag.

Kohonen, T. (1988b). An introduction to neural computing. *Neural Networks*, **1**, 3–16.

Kohonen, T., Barna, G., & Chrisley, R. (1988). Statistical pattern recognition with neural networks: Benchmarking studies. *Proceedings of the IEEE Second International Conference on Neural Networks* (Vol. 1, pp. 61–68). San Diego, CA: SOS Printing.

Krauth, W., & Mezard, M. (1987). Learning algorithms with optimal stability in neural networks. *Journal of Physics A: Math. Gen.*, **20**, L745–L752.

Mezard, M., Parisi, G., & Virasoro, M. (1987). *Spin glass theory and beyond.* Singapore: World Scientific.

Minsky, M., & Papert, S. (1969). *Perceptrons.* Cambridge: MIT Press.

Peterson, C., & Anderson, J. R. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems*, **1**, 995–1019.

Peterson, C., & Anderson, J. R. (1988). Neural networks and NP-complete optimization problems: A performance study on the graph bisection problem. *Complex Systems*, **2**, 59–89.

Peterson, C., & Söderberg, B. (1989). A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, 1, 3–21.

Peterson, C. & Redfield, S. (1988). A novel optical neural network learning algorithm. In J. W. Goodman, P. Chavel, & G. Roblin (Eds.), *Proceedings of the International Conference on Optical Computing* (SPIE Vol. 963, pp. 485–496). Bellingham, WA: SPIE.

Peterson, C., Redfield, S., Keeler, J. D., & Hartman, E. (1989). *Optoelectronic implentation of multi-layer neural networks in a single photorefractive crystal.* (Tech. Rep. No. MCC-ST-146-89). Austin, Texas: Microelectronics and Computer Technology Corporation.

Pineda, F. J. (1987). Generalization of backpropagation to recurrent neural networks. *Physical Review Letters*, **18**, 2229–2232.

Psaltis, D., & Farhat, N. (1985). Optical information processing based on an associative memory model of neural nets with thresholding and feedback. *Optics Letters*, **10**, 98–100.

Psaltis, D., Yu, J., Gu, G., & Lee, H. (1987). Optical neural nets implemented with volume holograms. In *Proceedings of Topical Meeting on Optical Computing*, Technical Digest Series 1987 (Vol. 11, pp. 129–136). Washington, DC: Optical Society of America.

Prager, R. W., Harrison, T. D., & Fallside, F. (1986). Boltzmann machines for speech recognition. *Computer Speech and Language*, **1**, 3–27.

Qian, N., & Sejnowski, T. J. (1988). Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology*, **202**, 865–884.

Rosenberg, C. R., & Sejnowski, T. J. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, **1**, 145–168.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1). Cambridge: MIT Press.

Rumelhart, D. E., Smolensky, P., McClelland, J. L., & Hinton, G. E. (1986). Schemata and sequential thought processes in PDP models. In J. L. McClelland and D. E. Rumelhart (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 2). Cambridge: MIT Press.

Sejnowski, T. J., Kienker, P., & Hinton, G. E. (1986). Learning symmetry groups with hidden units: Beyond the perceptron. *Physica*, **22D**, 260–275.

Soffer, B. H., Dunning, G. J., Owechko, Y., & Marom, E. (1986). Associative holographic memories with feedback using phase conjugate mirrors. *Optics Letters*, **11**, 118–120.

Stornetta, W. S., & Huberman, B. A. (1987). An improved three-layer back-propagation algorithm. In M. Caudill & C. Butler (Eds.), *Proceedings of the IEEE First International Conference on Neural Networks.* San Diego, CA: SOS Printing.

Tesauro, G., & Sejnowski, T. J. (1988). A neural network that learns to play backgammon. In D. Z. Anderson (Ed.), *Neural information processing systems.* New York: American Institute of Physics.

Touretzky, D. S., & Geva, S. (1987). A distributed connectionist representation for concept structures. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 155–164). Hillsdale, NJ: Erlbaum.

Wallace, D. J. (1986). Memory and learning in a class of neural network models. In B. Bunk and K. H. Mutter (Eds.), *Lattice gauge theory—A challenge to large scale computing.* New York: Plenum.

APPENDIX A

In this appendix we give details of the experiments performed in this work in terms of parameter tables. A momentum term coefficient of 0.9 was used throughout for BP.

TABLE A1
Parameters Used for the Mirror Symmetry Problem (see Figures 5–7 and Table 1) Values for the Boltzmann Machine are from Sejnowski et al. (1986)

4 × 4

Model	Repr.	Learning Rate	Update Freq.	Annealing Schedule	Initial Wgt. Range
Boltzmann Machine	0,1		5	2@(40, 30, 25, 20, 16, 14, 12), 14@10	
Mean Field Theory	−1,1	$\eta = 0.05$	5	$1@40(0.7^n)$, $n = 0 \ldots 10$	±1.0
	0,1	$\eta = 0.10$	5	$1@40(0.7^n)$, $n = 0 \ldots 10$	±1.0
Back Propagation	−1,1	$\eta = 0.005$	5		±0.5
	0,1	$\eta = 0.05$	5		±1.0

10 × 10

Model	Repr.	Learning Rate	Update Freq.	Annealing Schedule	Initial Wgt. Range
Boltzmann Machine	0,1		5	2@(40, 30, 25, 20, 16, 14, 12), 14@10	
Mean Field Theory	−1,1	$\eta = 0.02$	5	$1@40(0.5^n)$, $n = 0 \ldots 4$	±0.5
Back Propagation	0,1	$\eta = 0.08$	5		±0.5

TABLE A2
Parameters Used for the Statistical Pattern Classification Problem (see Table 2)

Mean Field Theory

Input Repr.	Connectivity	Learning Type	Learning Step/Rate	Update Freq.	Initial Wgt. Range
	full	man	$\kappa = 1/n$	1000	±1.0
dig	full	std	$\eta = 0.005$	1000	±0.5
dig	layer	man	$\kappa = 1/n$	1000	±1.0
dig	layer	std	$\eta = 0.001$	100	±0.5

Back Propagation

Input Repr.	Connectivity	Learning Type	Learning Step/Rate	Update Freq.	Initial Wgt. Range
dig	full	man	$\kappa = 1/n$	1000	±1.0
cont	layer	man	$\kappa = 1/n$	1000	±1.0
cont	full	man	$\kappa = 1/n$	1000	±1.0

All experiments used [0, 1] representation; all MFT experiments used an annealing schedule of $1@40(0.7^n)$; $n = 0 \ldots 10$. Full: full connectivity, Layer: layered connectivity, Man: Manhattan updating, Std: standard gradient descent updating, Dig: digital input encoding, Cont: continuous input encoding. In the 8-dimensional cases using Manhattan updating, $\kappa = 1/n$ was initialized to $n = 8$, and n was incremented by 1 every 10,000 patterns; in the 2-dimensional case n was decreased much more rapidly. In cases using standard updating, no corresponding benefit was observed from slowly decreasing η.

TABLE A3
Parameters Used for the CAM Learning Experiments (see Figures 9–11)

Model	Repr.	Learning Rate	Update Freq.	Annealing Schedule	Initial Wgt. Range
Mean Field Theory	−1, 1	$\eta = 0.005$	epoch	$1@40(0.7^n)$, $n = 0 \ldots 10$	±1.0

APPENDIX B

For completeness we here briefly derive the mean field theory approximation of the partition function (10).[20] The partition function of eqn (10) can be rewritten as

$$Z = C \prod_i \int_{-\infty}^{\infty} dV_i e^{-Nf(\vec{V},T)/T} \qquad \text{(B1)}$$

in terms of the free energy per neuron:

$$f(\vec{V}, T) = \frac{F(\vec{V}, T)}{N}. \qquad \text{(B2)}$$

Expanding $f(\vec{V}, T)$ around a saddle point $\vec{V} = \vec{V}_0$ (solution to eqn (13)) gives

$$f(\vec{V}_0 + \delta\vec{V}, T) \approx f(\vec{V}_0, T) + \frac{1}{2}\sum_{\mu,\nu=1}^{N} \delta V_\mu \delta V_\nu D_{\mu\nu}(\vec{V}_0, T) \qquad \text{(B3)}$$

where

$$D_{\mu\nu}(\vec{V}_0, T) = \frac{\partial}{\partial V_i \partial V_j} f(\vec{V}_0, T). \qquad \text{(B4)}$$

For the free energy of our neural network model (12), (B4) takes the form

$$D_{\mu\nu}(\vec{V}_0, T) = \frac{-T_{\mu\nu}}{2N}. \qquad \text{(B5)}$$

The partition function then becomes

$$Z = Ce^{-Nf(\vec{V}_0, T)} \prod_i \int_{-\infty}^{\infty} d(\delta V_i) e^{-N\frac{1}{2}\sum \delta V_\mu \delta V_\nu D_{\mu\nu}}. \qquad \text{(B6)}$$

The integral yields terms that are linear in N such that in the limit $N \rightarrow \infty$, the first exponent dominates and we have

$$Z \approx Ce^{-Nf(\vec{V}_0, T)/T} = Ce^{-F(\vec{V}_0, T)/T}. \qquad \text{(B7)}$$

Thus, the partition function is dominated by the value of $F'(\vec{V}, T)$ at a saddle point $\vec{V} = \vec{V}_0$.

Fortunately, the functional integral of the correction factor in (B6) is Gaussian and can hence be evaluated

$$\prod_i \int_{-\infty}^{\infty} d(\delta V_i) e^{-N\frac{1}{2}\sum \delta V_\mu \delta V_\nu D_{\mu\nu}} = \frac{1}{\sqrt{\det[N \times D_{\mu\nu}/2\pi T]}} = \frac{1}{\sqrt{\det[-T_{\mu\nu}/2\pi T]}}. \qquad \text{(B8)}$$

APPENDIX C

In this appendix we show that the bidirectional perceptron learning algorithm (Bruce et al., 1986; Wallace, 1986) (see section 4.2.1) is a special case of the MFT CAM algorithm (see section 4.3). We also discuss the relation of a modified, more successful, version of this algorithm (Diederich & Opper, 1987; Gardner, 1987; Krauth & Mezard, 1987) to MFT.

To do this we define the quantity $S_i'^p$

$$S_i'^p = \text{sgn}\left(\sum_j T_{ij} S_j^p\right). \qquad \text{(C1)}$$

The error of eqn (32) can then be rewritten as

$$\varepsilon_i^p = \frac{1}{2}(1 - S_i^p S_i'^p) \qquad \text{(C2)}$$

indicating that the error for a unit i can be measured by finding its state during a "free phase" in which all units except unit i are clamped. One such free phase is required for each unit. Substituting (C2) into the weight update rule of (33) yields

$$\Delta T_{ij} = \frac{1}{2N}\sum_p (2S_i^p S_j^p - S_i'^p S_j^p - S_i^p S_j'^p). \qquad \text{(C3)}$$

Now consider the MFT CAM algorithm restricted to the condition of visible units only, all but one of which are clamped during each free phase. In this case, the change to a weight T_{ij} due to a free phase with unit i free can be written

$$\Delta T_{ij}^p = \eta(V_i^p V_j^p - V_i'^p V_j^p) \qquad \text{(C4)}$$

since unit j is clamped in the free phase just as it is in the clamped phase. Similarly, the change to the same weight T_{ij} due to a free phase with unit j free can be written

$$\Delta T_{ij}^p = \eta(V_i^p V_j^p - V_i^p V_j'^p). \qquad \text{(C5)}$$

The total change to T_{ij} is then

$$\Delta T_{ij}^p = \eta(2V_i^p V_j^p - V_i'^p V_j^p - V_i^p V_j'^p). \qquad \text{(C6)}$$

When summed over all memories and restricted to $T = 0$, (C6) yields (C3).

This establishes the result that the bidirectional perceptron algorithm is the MFT CAM algorithm restricted to (a) $T = 0$, (b) no hidden units, and (c) all units but one are clamped during each free phase.

In the above algorithm, the error is zero if the summed input has the correct sign, regardless of its magnitude. This weak learning criterion results in negligible basins of attraction (Forrest, 1988). There exists a modified algorithm (Diederich & Opper, 1987; Gardner, 1987; Krauth & Mezard, 1987) which greatly enlarges the basins of attraction (Forrest, 1988). The definition of the error is slightly modified to ensure that the summed input is at least a certain size in addition to having the correct sign:

$$\varepsilon_i^p = \frac{1}{2}\left[1 - \text{sgn}\left(\left[S_i^p \sum_j T_{ij} S_j^p\right] - B_i\right)\right] \qquad \text{(C7)}$$

where B_i is some positive number. (Forrest, 1988 uses $B = K \langle|T_{ij}|\rangle N^{\frac{1}{2}}$ where K is an adjustable parameter and $\langle|T_{ij}|\rangle$ is the average magnitude of the weights attached to unit i.) Note that B_i is involved only in the error function, not the dynamics, and does not correspond to a threshold.

How this modification relates to the MFT CAM algorithm can be understood qualitatively by observing the following. The modification ensures the development of weights such that if, after the algorithm had completed execution, the binary ($T = 0$) units were replaced by continuous valued ($T > 0$, say $T = 1$) units, those units would remain of the same sign as the units they replaced and would take on values near the endpoints (close to 1 or -1). The modification thus corresponds to replacing the midpoint learning criterion of the unmodified algorithm with an endpoint learning criterion (see discussion in section 3.1). In our MFT CAM learning simulations, we in effect used an endpoint criterion with a zero margin (corresponding to infinite B_i), thus continuing to change weights until content-addressability testing ceased to improve. Beyond this point the basins continued to become deeper without becoming any broader.

It is shown analytically in Gardner (1987) that an upper bound on storage capacity of $2N$ exists for the modified bidirectional perceptron algorithm. As mentioned in section 4.3, such calculations are not directly applicable to MFT CAM networks due to the very different training and retrieval procedures and the presence of hidden units. It will be interesting, however, to discover the extent to which such bounds are relatively independent of the training and retrieval procedures (Hartman, 1988).

[20] This Appendix is to a large extent identical to Appendix A in Peterson and Anderson (1988). It differs by a more consistent notation and with an explicit evaluation of the correction factor to the MFT approximation to Z.