Complex Systems 1 (1987) 995–1019

A Mean Field Theory Learning Algorithm for Neural Networks

Carsten Peterson
James R. Anderson
Microelectronics and Computer Technology Corporation,
3500 West Balcones Center Drive, Austin, TX 78759-6509, USA

Abstract. Based on the Boltzmann Machine concept, we derive a learning algorithm in which time-consuming stochastic measurements of correlations are replaced by solutions to deterministic mean field theory equations. The method is applied to the XOR (exclusive-or), encoder, and line symmetry problems with substantial success. We observe speedup factors ranging from 10 to 30 for these applications and a significantly better learning performance in general.

1. Motivation and results

1.1 Background

Neural Network models are presently subject to intense studies [1,2,7,10]. Most attention is being paid to pattern completion problems. Network architectures and learning algorithms are here the dominating themes. Common ingredients of all models are a set of binary valued neurons $S_i = \pm 1$ which are interconnected with synaptic strengths T_{ij}, where T_{ij} represents the strength of the connection between the output of the i^{th} neuron and the input of the j^{th} neuron and $T_{ii} = 0$. In typical applications, a subset of the neurons are designated as *inputs* and the remainder are used to indicate the *output*.

By clamping the neurons to certain patterns, $S_i = S_i^\alpha$, the synaptic strengths adapt according to different learning algorithms. For patterns with first-order internal constraints, one has the Hebb rule [4], where for each pattern α the synapses are modified according to

$$\Delta T_{ij} \propto \langle S_i^\alpha S_j^\alpha \rangle \tag{1.1}$$

where $\langle\rangle$ denotes a time average.

In the case in which one has higher-order constraints, as in parity patterns, the situation is more complicated. Extra, so-called *hidden units* are then needed to capture or to build an internal representation of the pattern. In this case, equation (1.1) is not adequate; for the different patterns,

the hidden units have no particular values. For this reason, more elaborate learning algorithms have been developed. Most popular and powerful are the *Back-propagation Scheme* [10] and the *Boltzmann Machine* (BM) [1]. The latter determines T_{ij} for a given set of patterns by a global search over a large solution space. With its *simulated annealing* [8] relaxation technique, BM is particularly well suited for avoiding local minima. This feature, on the other hand, makes BM time consuming; not only does the stochastic annealing algorithm involve measurements at many successive temperatures, but the measurements themselves require many sweeps. Developments or approximations that would speed up this algorithm are in demand.

1.2 Objectives

In this work, we define and apply a *mean field theory* (MFT) approximation to the statistical mechanics system that is defined by the BM algorithm. The nondeterministic nature of the latter is then replaced by a set of deterministic equations. At each temperature, the solutions of these equations represent the average values of corresponding quantities computed from extensive (and expensive) sampling in the BM. It is obvious that if this approximation turns out to be a good one, substantial CPU time savings are possible. Also, these mean field theory equations are inherently parallel. Thus, simulations can take immediate and full advantage of a parallel processor.

1.3 Results

We develop and apply the MFT approximation for the Boltzmann Machine. This approximation is only strictly valid in the limit of infinite numbers of degrees of freedom. The systematic errors that occur when applying it to finite system sizes can be controlled and essentially canceled out in our applications. We find, when applying the method to the *XOR* [2], *encoder* [1], and *line symmetry* [2] problems, that we gain a factor 10–30 in computing time with respect to the original Boltzmann Machine. This means that for these problems, the learning times are of the same order of magnitude as in the Back-propagation approach. In contrast to the latter, it also allows for a more general network architecture and it naturally parallelizes. Furthermore, it in general gives rise to a higher learning quality than the Boltzmann Machine. This feature arises because the latter requires an unrealistically large number of samples for a reliable performance.

This paper is organized as follows. In section 2, we review the basics of the Boltzmann Machine. A derivation and evaluation of the mean field theory approximation can be found in section 3, and its applications to the problems mentioned above are covered in section 4. Finally, section 5 contains a very brief summary and outlook.

2. The Boltzmann Machine revisited

The Boltzmann Machine is based on the Hopfield energy function [6]

$$E(\vec{S}) = -\frac{1}{2}\sum_{i,j=1}^{N} T_{ij}S_iS_j + \sum_i I_iS_i \tag{2.1}$$

where the I_i are the neuron thresholds.[1] The last term in equation (2.1) can be eliminated by introducing an extra neuron S_0, which is permanently in a $+1$ state with $T_{0i} = T_{i0} = -I_i$. The energy then takes the simpler form

$$E(\vec{S}) = -\frac{1}{2}\sum_{i,j=0}^{N} T_{ij}S_iS_j. \tag{2.2}$$

In a Hopfield network, learning takes place with equation (1.1), which corresponds to differentiating equation (2.2) with respect to T_{ij}. With a given set of T_{ij} and a particular starting configuration $\vec{S}^s$, the system relaxes to a local energy minima with the step function updating rule

$$S_i = \begin{cases} +1 & \text{if } \sum_j T_{ij}S_j > 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.3}$$

which follows from differentiating equation (2.2) with respect to S_i along with the fact that $T_{ij} = T_{ji}$ and $T_{ii} = 0$.

As mentioned in the introduction, equation (1.1) is not appropriate when hidden units are included, since their values for different patterns are unknown. In the Boltzmann Machine, the strategy is to determine the hidden unit values for a given set of patterns by looking for a global minimum to

$$E(\vec{S}) = -\frac{1}{2}\sum_{i,j=0}^{N+h} T_{ij}S_iS_j \tag{2.4}$$

where h is the number of hidden units. The simulated annealing technique [8] is used to avoid local minima.

The major steps in the BM are the following:

1. **Clamping Phase.** The input and output units are clamped to the corresponding values of the pattern to be learned, and for a sequence of decreasing temperatures $T_n, T_{n-1}, \ldots, T_0$, the network of equation (2.4) is allowed to relax according to the Boltzmann distribution

   $$P(\vec{S}) \propto e^{-E(\vec{S})/T} \tag{2.5}$$

   where $P(\vec{S})$ denotes the probability that the state $\vec{S}$ will occur given the temperature T. Typically, the initial state $\vec{S}^s$ of the network is chosen at random. At each temperature, the network relaxes for an amount of time[2] determined by an *annealing schedule*. At $T = T_0$, statistics are collected for the correlations

[1] Throughout this paper, the notation $\vec{S} = (S_1, \ldots, S_i, \ldots, S_N)$ is used to describe a state of the network.

[2] We define time in terms of *sweeps* of the network. A *sweep* consists of allowing each unclamped unit to update its value once.

$$\rho_{ij} = \langle S_i S_j \rangle. \tag{2.6}$$

The relaxation at each temperature is performed by updating unclamped units according to the heatbath algorithm [1]

$$P(S_i \to 1) = \left[1 + \exp\left(\sum_j T_{ij} S_j / T\right)\right]^{-1}. \tag{2.7}$$

2. **Free-running phase.** The same procedure as in step 1, but this time only the input units are clamped. Again, correlations

$$\rho'_{ij} = \langle S_i S_j \rangle \tag{2.8}$$

are measured at $T = T_0$.

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

$$\Delta T_{ij} = \eta(\rho_{ij} - \rho'_{ij}) \tag{2.9}$$

where η is a learning parameter.

Steps 1, 2, and 3 are then repeated until no more changes in T_{ij} take place.

If the updating performed by equation (2.7) in steps 1 and 2 is instead performed with the step function updating rule in equation (3), the system is likely to get caught in a local minima, which could give rise to erroneous learning. With the annealing prescription on the other hand, the global minimum is more likely to be reached.

Before moving on to the mean field theory treatment of the annealing process in the Boltzmann Machine, we will make two important comments and clarifications on the learning process described above.

Annealing Schedule. The efficiency of the hill-climbing property of equations (2.5, 2.7) depends not only on the temperatures T used, but it is rather the ratios $E(\vec{S})/T$ that set the fluctuation scales (i.e. the likelihood of uphill moves). In the Boltzmann Machine, the same annealing schedule is normally used for the entire learning process. This rigidity does not fully exploit the virtue of the algorithm. The reason for this is that the energy changes as learning takes place, since the T_{ij}'s in equation (2.4) are changing.[3] Hence, the annealing schedule $T_n, T_{n-1}, \ldots, T_0$ should be adjusted in a adaptive manner during the learning phase. It turns out that in our applications, the effects from such a fine-tuning are negligible.

[3]Typically, T_{ij}'s are initialized to small random values. Thus, as learning takes place, the T_{ij}'s grow in magnitude.

Correlations. Our description of the BM learning algorithm above differs from the original [1] and subsequent works [2] on one subtle but important point. The learning is accomplished by measuring correlations ρ_{ij} (see equation (2.6)) rather than cooccurences p_{ij}. In the latter case, one only assigns positive increments to p_{ij} when either (a) both of the units i and j are on at the same time [1], or (b) both are identical [2]. By expanding these cooccurrence measurements to correlations, one also captures negative increments, i.e., one assigns negative correlations in situations where two units are anticorrelated. (Note that the correlations ρ_{ij} and ρ'_{ij} are not probabilities since they range from -1 to $+1$). This generalization improves the learning properties of the algorithm, as indicated in reference [2]. The correlation measure has the effect of doubling the value of ΔT_{ij} that would be produced by equation (2.9) using the cooccurence measure instead of the correlations, as in reference [2].[4] This effectively doubles the learning rate η.

3. The mean field theory equations

3.1 Derivations

The statistical weight (discrete probability) for a state in a particular configuration $\vec{S} = (S_1, \ldots, S_i, \ldots, S_N)$ at a temperature T is given by the Boltzmann distribution (see equation (2.5)). From equation (2.5), one computes the average of a state dependent function $F(\vec{S})$ by

$$\langle F(\vec{S}) \rangle = \frac{1}{Z} \sum_{\vec{S}} F(\vec{S}) e^{-E(\vec{S})/T} \tag{3.1}$$

where Z is the so-called partition function

$$Z = \sum_{\vec{S}} e^{-E(\vec{S})/T} \tag{3.2}$$

and the summations $\sum_{\vec{S}}$ run over all possible neuron configurations.

It is clear that configurations with small values for $E(\vec{S})$ will dominate. The standard procedure to compute $\langle F(\vec{S}) \rangle$ in equation (3.1) is with Monte-Carlo sampling techniques. Is there any way of estimating $\langle F(\vec{S}) \rangle$ along these lines without performing Monte-Carlo simulations? It is certainly not fruitful to search for minima of $E(\vec{S})$ since then the T-dependence disappears and we are back to a local minima search problem. Instead, let us manipulate the summations in equations (3.1, 3.2).

A sum over $S = \pm 1$ can be replaced by an integral over continuous variables U and V as follows:

[4] Note that $\rho_{ij} = p_{ij} - q_{ij}$ where q_{ij} is a measure of the anticorrelated states and $q_{ij} = 1 - p_{ij}$. Then, $\rho_{ij} - \rho'_{ij} = 2(p_{ij} - p'_{ij})$.

$$\sum_{S=\pm 1} f(S) = \sum_{S=\pm 1} \int_{-\infty}^{\infty} dV f(V)\delta(S-V). \tag{3.3}$$

Using the δ-function representation

$$\delta(x) = \frac{1}{2\pi i}\int_{-i\infty}^{i\infty} dy e^{xy} \tag{3.4}$$

where the integral over y runs along the imaginary axis, one obtains

$$\begin{aligned}\sum_{S=\pm 1} f(S) &= \frac{1}{2\pi i}\sum_{S=\pm 1}\int_{-\infty}^{\infty} dV \int_{-i\infty}^{i\infty} dU\, f(V)e^{U(S-V)} \\ &= \frac{1}{\pi i}\int_{-\infty}^{\infty} dV \int_{-i\infty}^{i\infty} dU\, f(V)e^{-UV+\log(\cosh U)}.\end{aligned} \tag{3.5}$$

Generalizing to our case of N neuron variables S_i and letting $f(\vec{S}) = exp(-E(\vec{S})/T)$, one obtains

$$\begin{aligned}Z = \sum_{\vec{S}} e^{-E(\vec{S})/T} &= \sum_{S_1=\pm 1}\cdots\sum_{S_i=\pm 1}\cdots\sum_{S_N=\pm 1} e^{-E(\vec{S})/T} \\ &= c\prod_i \int_{-\infty}^{\infty} dV_i \int_{-i\infty}^{i\infty} dU_i\, e^{-E'(\vec{V},\vec{U},T)}\end{aligned} \tag{3.6}$$

where c is a normalization constant and the *effective energy* is given by

$$E'(\vec{V},\vec{U},T) = E(\vec{V})/T + \sum_i [U_iV_i - \log(\cosh U_i)]. \tag{3.7}$$

The saddlepoints of Z are determined by the simultaneous stationarity of $E'(\vec{V},\vec{U},T)$ in both of the *mean field variables* U_i and V_i:

$$\frac{\partial E'(\vec{V},\vec{U},T)}{\partial U_i} = V_i - \tanh U_i = 0 \tag{3.8}$$

$$\frac{\partial E'(\vec{V},\vec{U},T)}{\partial V_i} = \frac{1}{T}\frac{\partial E(\vec{V})}{\partial V_i} + U_i = 0. \tag{3.9}$$

Since E' is real for real U_i and V_i, the solutions to these equations are in general real.

For the neural network of equation (2.4) one gets from these equations

$$V_i = \tanh\left(\sum_j T_{ij}V_j/T\right) \tag{3.10}$$

where the neuron variables S_i have been replaced through equation (3.9) by the mean field variables V_i. Thus, the non-zero temperature behavior of the network in equation (2.4) with the step function updating rule of equation (3) is emulated by a sigmoid updating rule (see figure 1). An important property of the effective energy function $E'(\vec{V},\vec{U},T)$ is that it has a smoother landscape than $E(\vec{S})$ due to the extra terms. Hence, the probability of getting stuck in a local minima decreases.

Algorithms based on equation (3.10) are, of course, still deterministic. Equation (3.10) can be solved iteratively:

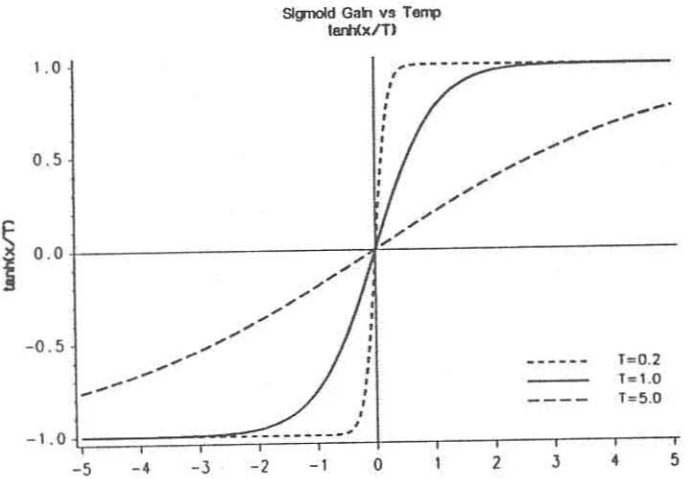

Figure 1: Sigmoid gain functions of equation (3.10) for different temperatures T. The step function updating rule of equation (3) corresponds to $T \to 0$.

$$V_i^{new} = \tanh\left(\sum_j T_{ij} V_j^{old}/T\right). \tag{3.11}$$

One can use either local or global time steps, asynchronous or synchronous updating respectively. In most applications, the asynchronous method seems to be advantageous.

Under appropriate existence and stability conditions (see e.g. [5], chapter 9), an equation of the form $f(x) = 0$ can be solved by numerical solution of

$$\frac{dx}{dt} = f(x). \tag{3.12}$$

Solving equation (3.9) in this way for the neural network of equation (2.4), substituting for V_j from equation (3.8), and making a change of variables $U_i \to U_i/T$, one gets

$$\frac{dU_i}{dt} = -U_i + \sum_j T_{ij} \tanh(U_j/T) \tag{3.13}$$

which are identical to the RC equations for a electrical circuit of interconnected amplifiers and capacitors with capacitances C and time constants τ set to one, and interconnection conductances T_{ij}. Similar equations were used in reference [7] to provide a neural network solution to the traveling salesman problem.

An alternate and more simplified derivation of equation (3.10) based on probabilistic concepts can be found in the appendix. The derivation above, however, has the nice feature of illuminating the fact that the stochastic hill-climbing property of non-zero temperature Monte Carlo can be cast into a deterministic procedure in a smoother energy landscape; rather than climbing steep hills, one takes them away. That this technique is a mean field theory approximation is clear from equations (A.8, A.10).

So far, we have computed $V_i = \langle S_i \rangle$. What we really need for the BM algorithm of equations (2.6–2.9) are the correlations $V_{ij} = \langle S_i S_j \rangle$. Again, these can be obtained by formal manipulations of the partition function along the the same lines as above or with the probabilistic approach described in the appendix. One gets

$$V_{ij} = \frac{1}{2}\left[\tanh\left(\sum_k T_{jk} V_{ik}/T\right) + \tanh\left(\sum_k T_{ik} V_{jk}/T\right)\right]. \tag{3.14}$$

This set of equations can also be solved by the same iterative technique as used for V_i in equation (3.11). One now has a system of $N \times N$ rather than N equations. This fact, together with the experience that larger systems of equations in general tend to converge slower, has motivated us to make one further approximation in our application studies. We approximate V_{ij} with the factorization:

$$V_{ij} = V_i V_j \tag{3.15}$$

3.2 Validity of the approximation

How good an approximation is the mean field theory expression of equation (6) together with the factorization assumption of equation (3.15) for our applications? The MFT derivation basically involves a replacement of a discrete sum with a continuous integral. Thus, the approximation should be exact for $N \to \infty$ where N is the number of degrees of freedom. Let us investigate how good the approximation is when computing ρ'_{ij} in equation (2.8) for the XOR problem. The XOR (exclusive-or) problem is the one of computing parity out of two binary digits. Thus, the patterns of the input-output mapping are[5]

$$\begin{array}{cc} 00 & 0 \\ 01 & 1 \\ 10 & 1 \\ 11 & 0 \end{array} \tag{3.16}$$

where the first two columns are the input units and the third column is the output unit. As is well known, this problem requires the presence of hidden units (see figure 2) [10].

[5]Throughout this paper, we use ± 1 in the calculations, rather than the 0, 1 representation of patterns.

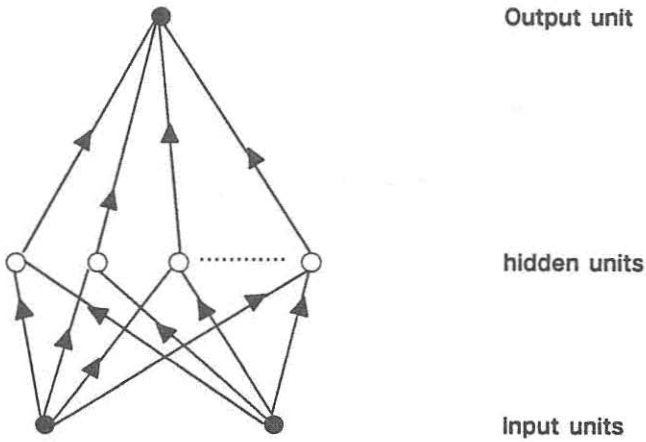

Figure 2: Neural network for the XOR problem with one layer of hidden units.

In the free phase of the Boltzmann Machine, the input units are clamped. When computing ρ'_{ij}, two different situations are encountered; one either computes $\langle S_i S_j \rangle$ or $\langle S_i \rangle S_j$, depending on whether S_j is clamped or not. We have compared $\langle S_i \rangle$ with V_i and $\langle S_i S_j \rangle$ with $V_{ij} = V_i V_j$ respectively for the free-running case with random choice of T_{ij}. In figure 3, we show the average values for the output unit $\langle S_{out} \rangle$ as a function of the number of sweeps used for measurements at the final annealing temperature $T = T_0$. Also shown is the mean field theory prediction, which is based on the same annealing schedule as for the Boltzmann Machine but with only one iteration for each temperature including $T = T_0$. Thus, $N_{sweep} = 1$ for the MFT value. For further details on annealing schedules, architectures, and T_{ij} values, we refer to the figure caption.

Two conclusions stand out from this figure. One is that the mean field theory is a very good approximation even for relatively small systems (in this case, 5 dynamical units). The second point regards the behavior of the Boltzmann Machine as a function of N_{sweep}. One expects substantial fluctuations in measured quantities around expected values for small or moderate N_{sweep}, but with decreasing errors. That the errors are decreasing is evident from figure 3. However, the approach to asymptotia has systematic features rather than being random. The reason for this is that it takes a large number of sweeps to thermalize at $T = T_0$. From the figure, we estimate that O(100–1000) sweeps seems appropriate if one wants a performance compatible with the mean field theory approximation. In figure 4, we depict the same result for the hidden unit S_1^H. The same conclusion can be drawn for

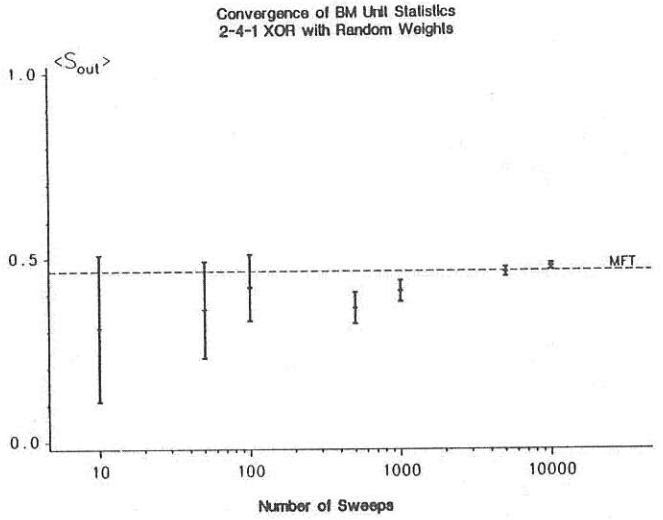

Figure 3: $\langle S_{out} \rangle$ and V_{out} from the BM and MFT respectively as functions of N_{sweep}. A one-layer network with four hidden units was used as in [2]. Random values in the range $[-2.5, 2.5]$ were used for T_{ij}. The annealing schedule used was $T = 50, 49, \ldots, 1$ with 10 sweeps/T for BM and 1 sweep/T for MFT. N_{sweep} refers time at the final temperature.

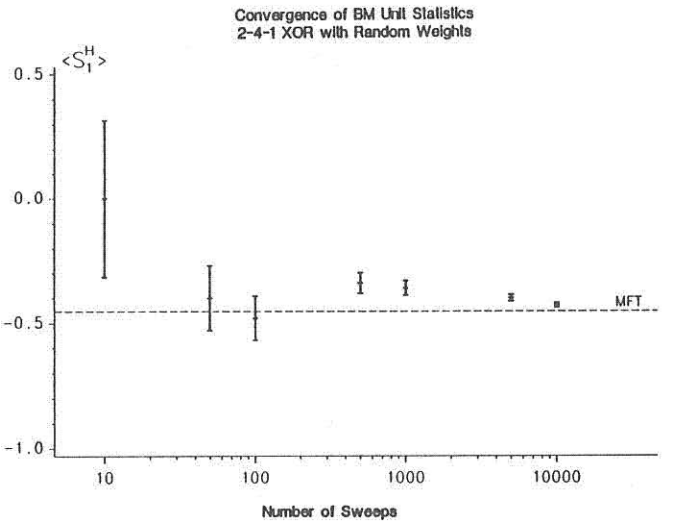

Figure 4: $\langle S_1^H \rangle$ and V_1^H from the BM and MFT respectively as functions of N_{sweep}. For details on architecture, annealing schedule, and T_{ij} values, see figure 3.

the correlation $\langle S_1^H S_{out} \rangle$ (see figure 5).

All averages and correlations show the same features as in the examples above. In figure 6, we summarize our findings by showing the average deviation Δ between the Boltzmann Machine statistics and the mean field theory results,

$$\Delta = \frac{1}{N} \sum_{i>j} \left| \rho'_{ij}(BM) - \rho'_{ij}(MFT) \right| \tag{3.17}$$

again as a function of N_{sweep}. From this figure, it is clear that even for a large number of sweeps there is small but systematic deviation between the Boltzmann Machine and the mean field theory. It is interesting to study how this discrepancy varies with the number of degrees of freedom (in our case, the number of hidden units, n_H). In figure 7, we show the average of Δ for 100 different random T_{ij}, $\langle \Delta \rangle$, as a function of the number of hidden units. It is clear that the discrepancy decreases with n_H. As discussed above, this phenomenon is expected.

In summary, we find the mean field theory approximation to be extremely good even for a relatively small number of degrees of freedom. As expected, the approximation improves with the number of degrees of freedom. Furthermore, using the Boltzmann Machine without allowing for ample thermalization might provide erroneous results.

Being convinced about the efficiency of the mean field approximation, we now move on to learning applications.

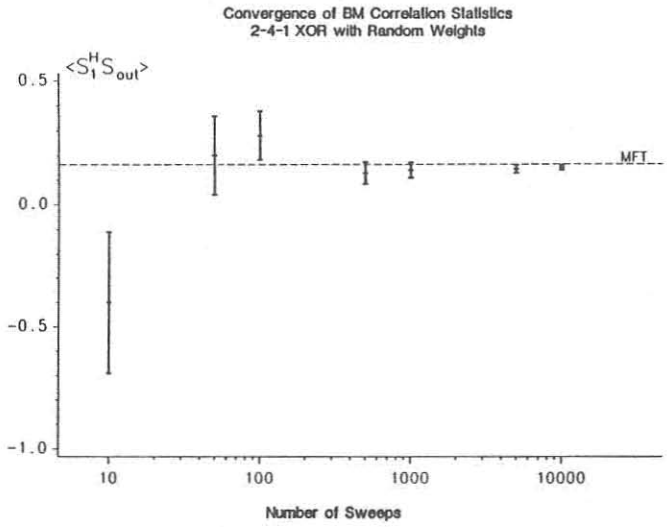

Figure 5: $\langle S_1^H S_{out} \rangle$ and $V_1^H V_{out}$ from the BM and MFT respectively as functions of N_{sweep}. For details on architecture, annealing schedule, and T_{ij} values, see figure 3.

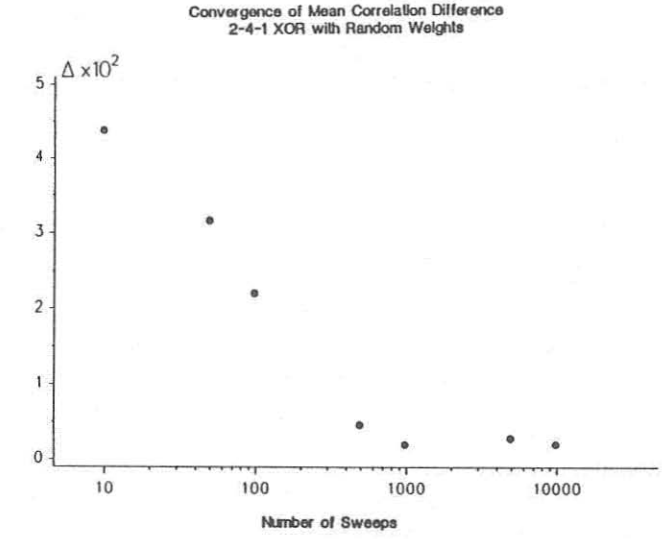

Figure 6: Δ as defined in equation (3.17) as a function of N_{sweep}. For details on architecture, annealing schedule, and T_{ij} values, see figure 3.

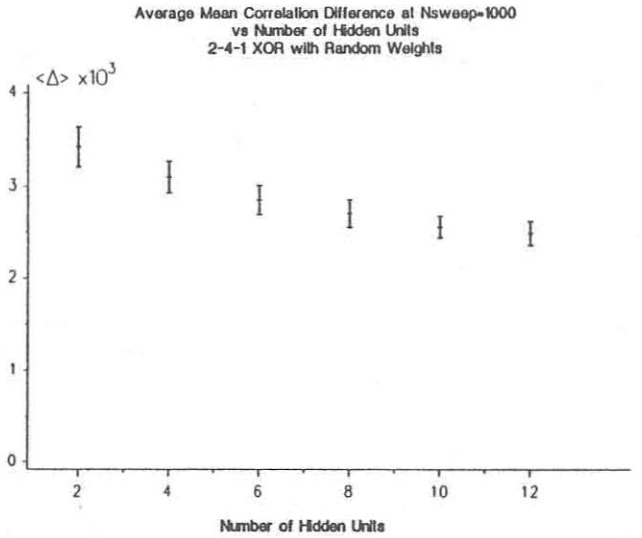

Figure 7: $\langle\Delta\rangle$ (Δ defined in equation (3.17)) as a function of the number of hidden units n_H. Annealing schedules are as in figure 3. For the Boltzmann Machine $N_{\text{sweep}} = 1000$ was used. The statistics are based on 100 different random sets of T_{ij}.

4. Performance studies of the mean field theory algorithm

We have investigated the performance of the MFT algorithm in three different applications: the XOR [2], encoder [1], and line symmetry [2] problems. These problems, while small, exhibit various types of higher-order constraints that require the use of hidden units. The results of the MFT calculations are compared with the corresponding results from the BM simulations.

4.1 Annealing schedules and learning rates

Boltzmann Machine (BM). For all the applications described below, except as noted, the following annealing schedule and learning rate were used:

$$\begin{aligned} N_{sweep}@T &= 1@30, 2@25, 4@20, 8@15, 8@10, 8@5, 16@1, 16@0.5 \\ \eta &= 2 \end{aligned} \tag{4.1}$$

For the final temperature, $T = 0.5$, all 16 sweeps were used for gathering correlation statistics. This schedule, which is identical to the one used in [2], appears to provide good results for all three applications. Any attempts to reduce the annealing time leads to degradation of learning performance, and improving the performance with a longer annealing schedule results in longer learning times.

Mean field theory (MFT). A brief exploration of the iterative techniques of equation (3.11) produced good results with the following parameter choices:

$$\begin{aligned} N_{sweep}@T &= 1@30, 1@25, 1@20, 1@15, 1@10, 1@5, 1@1, 1@0.5 \\ \eta &= 1 \end{aligned} \tag{4.2}$$

Notice that this annealing schedule is almost a factor 8 faster than the Boltzmann Machine schedule.

For both the BM and MFT algorithms, except as noted below, the T_{ij} are initialized with random values in the range $[-\eta, +\eta]$. Let us move on to the different applications in more detail.

4.2 The XOR problem

This problem consists of the four patterns in equation (16), which exhaust the combinatorial possibilities. For both the BM and MFT algorithms, we confirm previous results (see reference [2]) that an architecture with at least four hidden units seems to be needed for a good performance. We use four hidden units with limited connectivity in order to facilitate comparisons with [2]; no active connections between two hidden units and between input and output units (see figure 2). However, it should be stressed that in contrast to feedforward algorithms like Back-propagation [10], the BM and MFT algorithms are fully capable of dealing with fully connected networks. Performance studies of different degrees of connectivities will be published elsewhere [9].

As a criteria for learning performance, one normally uses the percentage of patterns that are completed (i.e. correct output produced for a given input) during training (see e.g. reference [2]). This measure is inferior, at least for small problems, as it does not indicate what portion of the input space is being learned. Therefore, for the XOR and the encoder problems, the entire input space is tested for proper completion. Thus, the entire input space is presented during each learning cycle.

In figure 8, we show the percentage of completed *input sets* as a function of the number of learning cycles performed. (An input set consists of a collection of patterns that are presented during a single learning cycle. In the case of the XOR problem, an input set consists of the entire input space.) Each data point represents the percentage of the previous 25 learning cycles (100 patterns) in which the network correctly completed the entire input set. In all of the figures in this section, the curves presented are obtained by averaging the statistics from 100 different experiments. The significant feature of figure 8 is that MFT demonstrates a higher quality of learning than BM. This does not appear to be simply a matter of MFT learning faster. For the XOR, as well as the other experiments in this section, the quality of learning exhibited by MFT seems to be asymptotically better than BM. This has been attributed to errors in the estimates of $\langle S_i S_j \rangle$ by the BM

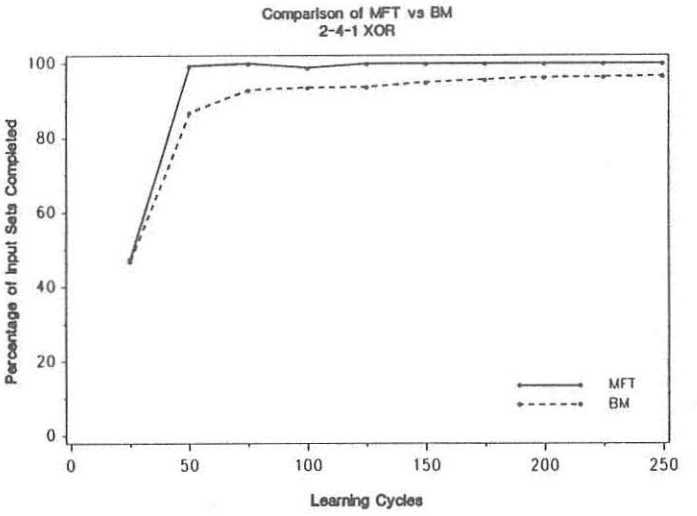

Figure 8: Percentage of completed input sets for the XOR problem as a function of learning cycles for the BM and MFT algorithms. For further details, see section 4.2.

algorithm due to the large number of sweeps at T_0 that are required to obtain accurate estimates (see section 3.2).

The curves shown in figure 8 do not take into account the difference in annealing schedules between MFT and BM. To get a better idea of the computing performance improvement offered by MFT, we show in figure 9 the percentage of completed input sets as a function of the number of sweeps performed.[6] If we consider BM to (nearly) reach its final performance value at approximately 5×10^4 sweeps while MFT does so at approximately 0.4×10^4, we can consider the MFT algorithm to achieve a factor of 10 to 15 percent improvement in execution time. Based on these curves, this appears to be a conservative claim.

A final evaluation of MFT performance is based on the notion of an experiment having completely learned the input space. Such a notion requires definition of a learning criteria. We consider the input space to be *completely learned* if the input set is correctly completed for 75 successive cycles (300 patterns). In figure 10, we show the percentage of experiments that completely learn the input space as a function of the number of sweeps. From

[6]Each sweep in both MFT and BM consists of updating each unclamped unit once during both the clamped and free-running phases. We consider an MFT sweep to be equivalent to a BM sweep as both involve the same number of updates. However, in practice, a BM sweep takes longer than an MFT sweep; both require evaluation of a similar function, but BM requires in addition a random number generation (for the update) and collection of statistics (estimation of ρ_{ij} and ρ'_{ij}).

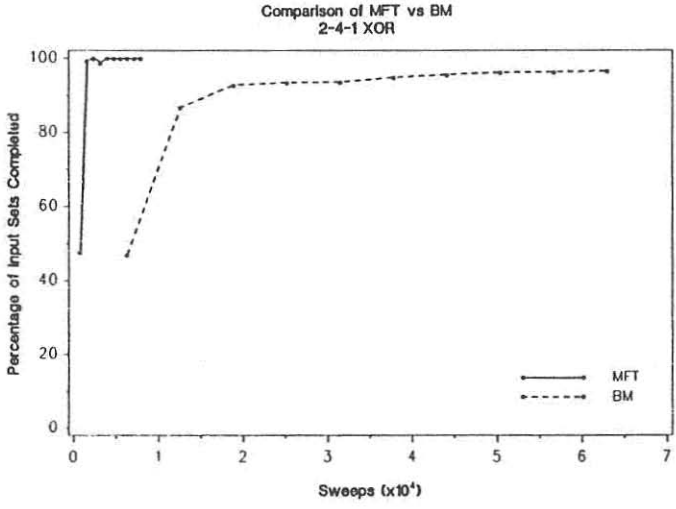

Figure 9: Percentage of completed input sets for the XOR problem as a function of N_{sweep}. The data is the same as in figure 8.

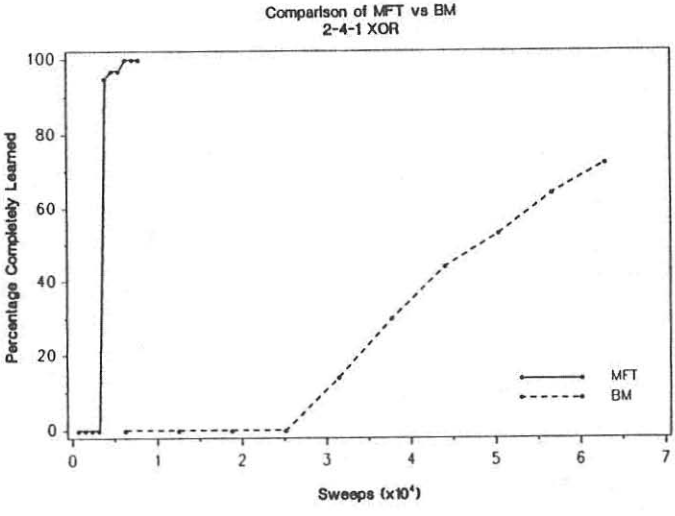

Figure 10: Percentage of XOR experiments that completely learned the input space as a function of N_{sweep}. For details of the learning criteria, see section 4.2.

these curves we see that MFT completely learns the XOR input space both faster and with a higher success rate than BM. Based on the results of the encoder problem (see figures 11 through 14), we expect that if the XOR experiments had been run for a longer number of learning cycles, the percentage completely learned by BM would approach the percentage of input sets completed by BM shown in figure 9.

4.3 The encoder problem

The encoder problem (see reference [1]) consists of the input-output mapping

$$\begin{array}{ll} 1000 & 1000 \\ 0100 & 0100 \\ 0010 & 0010 \\ 0001 & 0001 \end{array} \tag{4.3}$$

In its most difficult form (4-2-4), there are only two hidden units which must optimally encode the four patterns. Because there is no redundancy in the hidden layer, it is necessary to provide active connections between the units in the hidden layer. This allows the hidden units to "compete" for particular codes during learning. Connections are also provided between the units in the output layer. This allows lateral inhibition to develop so that the desired output unit can inhibit the other output units from being on at the same time. In addition, the T_{ij} are initialized to zero for BM and to very small random values ($[-\eta, +\eta] \times 10^{-3}$) for MFT.[7] Finally, we found it necessary to reduce the learning rates for both BM and MFT to $\eta = 1$ and $\eta = 0.5$ respectively in order to achieve good results. This has the effect of lowering the $E(\vec{S})/T$ ratio (see section 2), thereby introducing more thermal noise into the learning algorithm. This helps to resolve conflicts between encodings among the hidden units.

In figure 11, we show the percentage of completed input sets as a function of sweeps performed for the 4-2-4 encoder. We also show, in figure 12, the percentage of experiments that completely learn the input-output encoding as a function of sweeps. The final data points for these curves correspond to 500 learning cycles. Notice that for BM, the percentage completely learned shown in figure 12 asymptotically approches the percentage of input sets completed shown in figure 11. Both BM and MFT have trouble learning this problem, but the MFT learning quality as measured by percentage completely learned is nearly a factor of 3 better than BM.

[7] If the T_{ij} are not initially zero for problems with interconnected hidden units, there is an initial bias towards a certain internal representation of the encoding. Very often this leads to conflicts between hidden units that prevents learning from occuring. On the other hand, the T_{ij} are initially set to non-zero values in problems were the hidden units are not interconnected (e.g. XOR, line symmetery) to take advantage of random bias. This improves the probability of achieving a well-distributed internal representation among the hidden units. The fact that the T_{ij} are initially non-zero for MFT is a consequence of equation (3.10); all zero T_{ij} yields a solution of all zero V_i, while in BM there is enough noise to prevent this from being a problem.

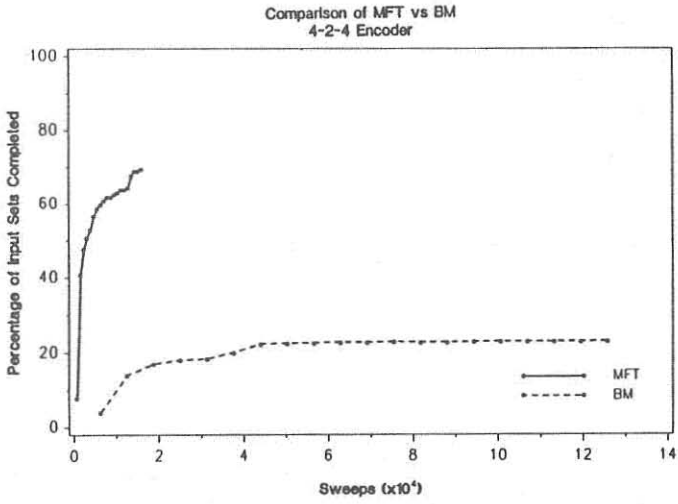

Figure 11: Percentage of input sets completed as a function of N_{sweep} for the 4-2-4 encoder.

We have also investigated the encoder problem with three hidden units (4-3-4). This configuration provides some redundancy in the hidden layer and we expect better results. Figures 13 and 14 show the percentage of input sets completed and percentage of experiments that completely learned the input space for the 4-3-4 encoder with 500 learning cycles. Again, the percentage completely learned by BM shown in figure 14 asymptotically approaches the percentage of input sets completed by BM shown in figure 13. This seems to indicate that once BM begins to complete input sets, it continues to do so in a uniform manner such that the complete learning criteria is eventually met. This will not appear to be true of BM for the line symmetry problem in the next section. For the 4-3-4 encoder, MFT easily achieves nearly perfect learning quality, providing a 2.5 factor of improvement over BM learning quality.

4.4 The line symmetry problem

This is the problem of detecting symmetry among an even number of binary units [2]. We have investigated the MFT performance for a problem consisting of six input units, six hidden units (no connections between hidden units), and one output unit to indicate presence or absence of symmetry. The symmetrical input patterns that are to be learned (i.e. produce an output of 1) consist of the following 8 patterns out of the 64 possible patterns:

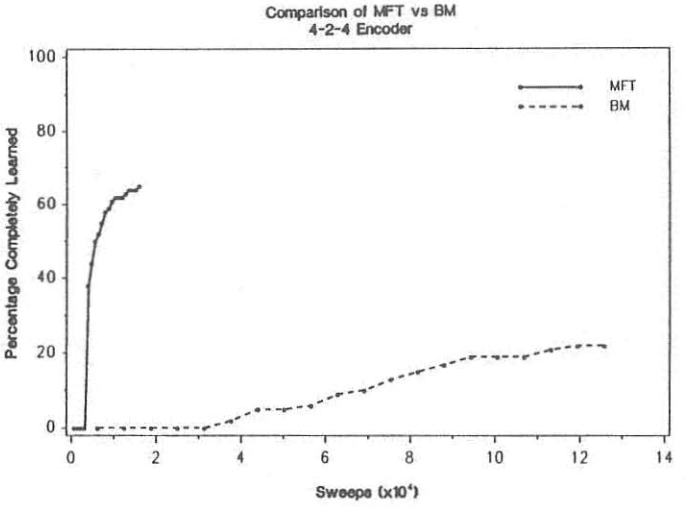

Figure 12: Percentage of 4-2-4 encoder experiments that performed complete learning of the input space as a function of N_{sweep}.

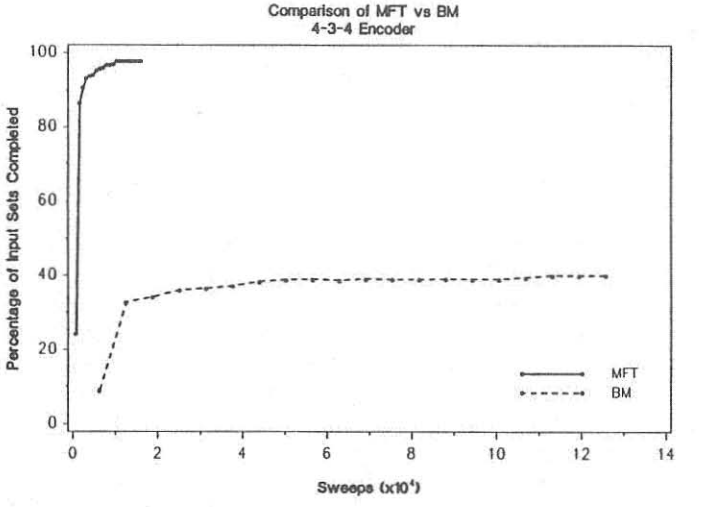

Figure 13: Percentage of input sets completed as a function of N_{sweep} for the 4-3-4 encoder.

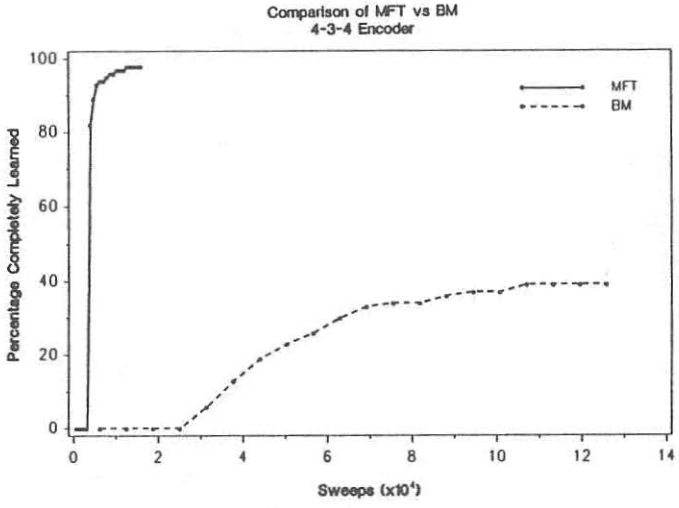

Figure 14: Percentage of 4-3-4 encoder experiments that performed complete learning of the input space as a function of N_{sweep}.

000	000
001	001
010	010
011	011
100	100
101	101
110	110
111	111

This problem differs from the previous problems in two ways:

Low signal-to-noise ratio (SNR). The network is required to identify 8 out of the 64 different patterns it is presented with, as opposed to 2 out of 4 for the XOR (the encoder is more like a completion problem than a classification problem). Thus, there are 8 *signal* patterns compared to 56 *noise* patterns. With a low SNR, learning to properly identify the symmetric patterns is difficult, because it is much easier for the network to identify the noise. For this problem, if the network identifies all patterns as asymmetric, it will be right at least 87.5% of the time. To remove this difficulty, the input space is adjusted so that symmetric patterns are presented as often as asymmetric ones. This is done by selecting a symmetric pattern with a 43% probability, otherwise select any pattern at random.[8]

[8]When we select any pattern at random with 57% probability, 12.5% (8 of 64) will be symmetric. Thus, P (symmetric) $= (0.57 \times 0.125) + 0.43 = 0.5$.

Large pattern space (generalized learning). We use this problem to explore the situation in which the size of the input space approaches a point at which it becomes intractable to present the entire input space for each learning cycle. Thus, we want to explore the the problem of generalized learning. Here we wish to learn the entire input space by training on a small random subset of the space during each learning cycle.

We have investigated generalized learning for the line symmetry problem by choosing ten input patterns at random from the distribution described above to be presented at each learning cycle. An input set consists of these ten random patterns, and these are checked for completion at each learning cycle prior to updating the T_{ij}.[9]. In figure 15, we show the percentage of input sets completed as a function of sweeps corresponding to 500 learning cycles. In contrast to corresponding curves for the XOR and encoder problems (see figures 9, 11, 13), we notice that learning in the line symmetry problem appears to be noisy. This is attributed, at least in part, to a certain amount of unlearning occuring due to training on small subsets of the input space. Again, MFT provides significant improvement in learning quality and performance. In figure 16, we show the percentage of experiments that perform complete learning (300 successive patterns completed) as a function of the number of sweeps. MFT appears to be better suited for generalized learning as nearly 90 percent of the experiments learn completely whereas less than 20 percent of the experiments learn completely with BM for 500 learning cycles.

Notice that in contrast to the encoder problems, the percentage of experiments completely learned by BM shown in figure 16 does not appear to be approaching the percentage of input sets completed by BM shown in figure 15. We conclude that, due to the noisiness of generalized learning, BM for the line symmetry problem does not exhibit the uniform approach to complete learning that was exhibited by BM in the encoder problems. On the other hand, MFT performance does not appear to degrade significantly when dealing with generalized learning.

5. Summary and outlook

We have developed, evaluated, and implemented a mean field theory learning algorithm. It is exteremely simple to use. Equation (3.10) is solved for a sequence of temperatures with appropriate units clamped. From these solutions, ρ_{ij} and ρ'_{ij} are computed from equation (3.15). The rest of the

[9]While this may look like generalization in the broadest sense, in that we may be testing a pattern that has not been trained on yet, eventually all patterns will be used for training. Thus, we cannot say that the network has generalized to patterns it has never seen before. We can say that there is generalization in learning, in that the learning done by ten particular patterns is not so specific as to cause the next ten patterns to undo previous learning. This would be the case if there was not any exploitable regularity in the input space.

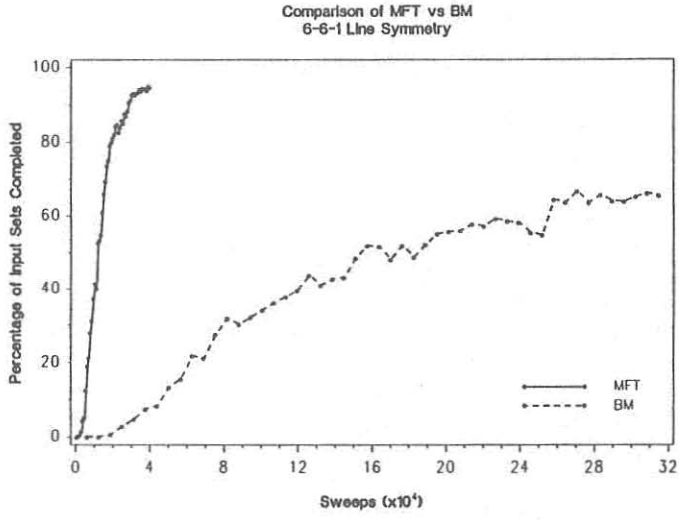

Figure 15: Percentage of input sets consisting of 10 random patterns completed as a function of N_{sweep} for the line symmetry problem.

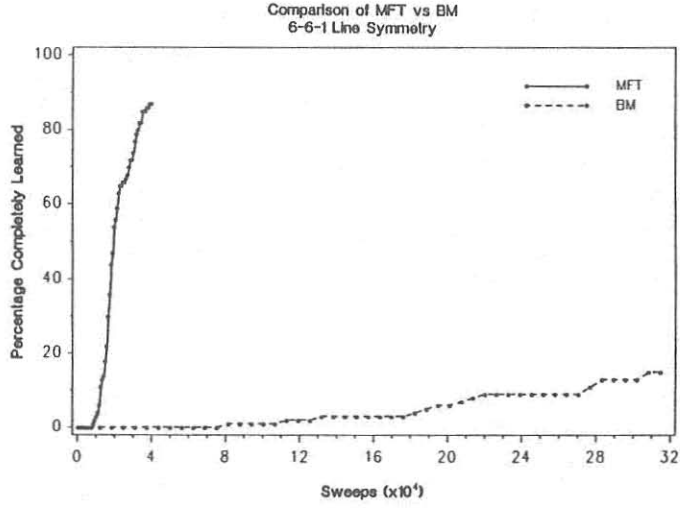

Figure 16: Percentage of line symmetry experiments that performed complete generalized learning of the input space (see section 4.4) as a function of N_{sweep}.

algorithm follows from the Boltzmann Machine updating rule of equation (2.9). The algorithm is inherently parallel.

Testing the algorithm on the XOR, encoder and line symmetry problems gives very encouraging results. Not only are speedup factors in the range 10–30 observed as compared to the Boltzmann Machine, but the quality of the learning is significantly improved. Exploratory investigations [9] on scaling up the problems to larger sizes tentatively indicate that the promising features survive.

The next step is to investigate what the generalization properties are of this algorithm and to apply it to realistic problems.

Appendix A.

Here we present an alternate derivation of the mean field theory equations following closely the Markov approach of reference [3].

The Boltzmann distribution of equation (2.5) is valid for equilibrium configurations. Let us define a time-dependent distribution $\hat{P}(S_i,t)$ that also holds for non-equilibrium situations. In a Markov process, it obeys the following master equation:

$$\begin{aligned}\frac{d\hat{P}(S_i,t)}{dt} &= -\sum_i W_i(S_i \to -S_i,t)\hat{P}(S_i,t) \\ &\quad +\sum_i W_i(-S_i \to S_i,t)\hat{P}(-S_i,t)\end{aligned} \tag{A.1}$$

where $W_i(\pm S_i \to \mp S_i,t)$ is the transition probability.

At equilibrium, $d/dt\hat{P}(S_i,t) = 0$ and $\hat{P}(S_i,t) = P(S_i)$, so one obtains from equation (A.1):

$$W_i(S_i \to -S_i,t)P(S_i) = W_i(-S_i \to S_i,t)P(-S_i) \tag{A.2}$$

Substituting the Boltzmann distribution (see equation (2.5)) for $P(S_i)$, one gets from equation (A.2):

$$\frac{W_i(S_i \to -S_i,t)}{W_i(-S_i \to S_i,t)} = \frac{\exp(-S_i\sum_j T_{ij}S_j/T)}{\exp(S_i\sum_j T_{ij}S_j/T)} \tag{A.3}$$

Thus, the transition probability can be rewritten as:

$$\begin{aligned}W_i(S_i \to -S_i,t) &= \frac{1}{2\tau}\left(1-\tanh\left(S_i\sum_j T_{ij}S_j/T\right)\right) \\ &= \frac{1}{2\tau}\left(1-S_i\tanh\left(\sum_j T_{ij}S_j/T\right)\right)\end{aligned} \tag{A.4}$$

where τ is a proportionality constant and the second line follows from the fact that $S_i = \pm 1$ and that $\tanh(x)$ is odd in x. The average of S_i is defined by

$$\langle S_i \rangle = 1/\hat{Z} \sum_{\vec{S}} S_i \hat{P}(S_i, t) \tag{A.5}$$

where

$$\hat{Z} = \sum_{\vec{S}} \hat{P}(S_i, t) \tag{A.6}$$

Noting that $\hat{Z} = 1$ and ignoring the proportionality constant τ, we compute $d/dt\langle S_i \rangle$ from equations (A.1), (A.4), and (A.5):

$$\begin{aligned}
\frac{d\langle S_i \rangle}{dt} &= \sum_{\vec{S}} S_i \frac{d\hat{P}(S_i, t)}{dt} \\
&= \sum_{\vec{S}} S_i \left[-\sum_j W_j(S_j \to -S_j, t)\hat{P}(S_j, t) \right. \\
&\qquad \left. + \sum_j W_j(-S_j \to S_j, t)\hat{P}(-S_j, t) \right] \\
&= -\left[\langle S_i \rangle - \tanh\left(\sum_j T_{ij} \langle S_j \rangle / T \right) \right].
\end{aligned} \tag{A.7}$$

The last equality follows from the fact that only the $j = i$ term contributes to the summation in the second line and we have made the mean field approximation

$$\left\langle \tanh\left(\sum_j T_{ij} S_j / T \right) \right\rangle = \tanh\left(\sum_j T_{ij} \langle S_j \rangle / T \right). \tag{A.8}$$

Similarly, for $\langle S_i S_j \rangle$ one gets:

$$\begin{aligned}
\frac{d\langle S_i S_j \rangle}{dt} &= \sum_{\vec{S}} S_i S_j \frac{d\hat{P}(S_i, t)}{dt} \\
&= \sum_{\vec{S}} S_i S_j \left[-\sum_k W_k(S_k \to -S_k, t)\hat{P}(S_k, t) \right. \\
&\qquad \left. + \sum_k W_k(-S_k \to S_k, t)\hat{P}(-S_k, t) \right] \\
&= -\left[2\langle S_i S_j \rangle - \tanh\left(\sum_k T_{ik} \langle S_i S_k \rangle / T \right) \right. \\
&\qquad \left. - \tanh\left(\sum_k T_{jk} \langle S_j S_k \rangle / T \right) \right].
\end{aligned} \tag{A.9}$$

At equilibrium, equations (7) and (9) are identical to equations (3.10) and (3.14) with the identifications

$$V_i = \langle S_i \rangle, \tag{A.10}$$

$$V_{ij} = \langle S_i S_j \rangle. \tag{A.11}$$

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, **9:1** (1985) 147–169.

[2] Joshua Alspector and Robert B. Allen, "A Neuromorphic VLSI Learning System," in *Advanced Research in VLSI: Proceedings of the 1987 Stanford Conference*, edited by Paul Losleben, (MIT Press, 1987).

[3] Roy J. Glauber, "Time-Dependent Statistics of the Ising Model," *Journal of Mathematical Physics*, **4:2** (1963) 294–307.

[4] D. O. Hebb, *The Organization of Behavior*, (John Wiley & Sons, 1949).

[5] Morris W. Hirsch and Stephen Smale, *Differential Equations, Dynamical Systems, and Linear Algebra*, (Academic Press, 1974).

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

[7] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, **52** (1985) 141–152.

[8] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, **220:4598** (1983) 671–680.

[9] Carsten Peterson and James R. Anderson, work in progress.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, (MIT Press, 1986) 318–362.