Variations on the Boltzmann machine

1989 J. Phys. A: Math. Gen. 22 5143

(http://iopscience.iop.org/0305-4470/22/23/018)

J. Phys. A: Math. Gen. **22** (1989) 5143-5151. Printed in the UK

Variations on the Boltzmann machine

György Barna†‡ and Kimmo Kaski‡

† Central Research Institute for Physics, Hungarian Academy of Sciences, PO Box 49, H-1525 Budapest, Hungary

‡ Tampere University of Technology, Microelectronics Laboratory, PO Box 527, SF-33101 Tampere, Finland

Received 25 May 1989

Abstract. The performance of several neural-network-like models for pattern recognition tasks is analysed. A comparison based on recognition of random points in a multi-dimensional space is made between backpropagation, learning vector quantisation and three variations of Boltzmann machine models. The ordinary Boltzmann machine is found to perform well although at the expense of being time consuming. Replacement of the Monte Carlo method in the Boltzmann machine by a direct formula promises faster evaluation without significant loss of precision.

1. Introduction

Models of neural networks (NNS) belong to a computing paradigm which can be seen as an alternative or counterpart to the conventional sequential (von Neumann) computing. Although NNS date back as far as the 1940s, the recent interest among physicists resulted from the finding of some probabilistic models, like the Hopfield network [1] and the Boltzmann machine (BM) [2] analogous to statistical mechanical systems, e.g. spin glasses. The realisation of this connection has also made it possible to exploit the powerful methods of statistical physics.

A NNS is an 'intelligent' system which mimics some of the structures and functions of natural neural networks. It consists of simple processing units, 'neurons' that are connected to form a network. Each unit performs a weighted sum of all its inputs and applies a non-linear (sigmoid or threshold) function to be delivered to a single output. The difference from conventional computing devices manifests itself in that NNS are not programmed, but the connecting weights between units are changed adaptively, according to the environment. Although the threshold logic and Boolean logic networks are functionally the same, NNS have several advantages over conventional von Neumann computers. They are inherently parallel, adaptive and fault-tolerant architectures. From the pattern classification point of view, NNS can be considered using non-parametric methods. They can form arbitrarily complex decision surfaces, and this can be done adaptively, without any *a priori* knowledge about the nature of the problem.

There are several neural-network-like models which can be used for solving pattern recognition tasks. Huang and Lippmann [3] made a comparison among the backpropagation (BP) [4] and some conventional Bayesian classifiers. Barna, Chrisley and Kohonen [5] have presented a comparison among BP, BM and learning vector quantisation (LVQ) [6]. In the latter, the BM emerged as the method of highest precision, though it proved to be the most time consuming algorithm. Replacement of the Monte Carlo

0305-4470/89/235143+09$02.50

(MC) method in the BM by a direct formula promises faster evaluation without significant loss of precision [7].

In this paper we compare the performance of various NN algorithms in a statistical pattern recognition task. Probabilistic networks have been emphasised by investigating variations of the Boltzmann machine. We have developed a new deterministic variation of the previously known stochastic Boltzmann machine and deterministic mean-field theory Boltzmann machine (MFB) [7]. Our deterministic Boltzmann machine (DBM) differs from MFB in that it uses a two-step method instead of iteration.

In the experiments the BM and the MFB performed best and the DBM was fairly close to them. The performance of BP and LVQ is good only at lower dimensions. The mean-field Boltzmann machine is found to be faster than the ordinary Boltzmann machine but not with the magnitude given in [7]. Also the DBM is shown to be faster than the MFB. The speed of backpropagation and learning vector quantisation were not compared with the variations of the Boltzmann machine but both of them are known to be fast algorithms.

2. The Boltzmann machine

The Boltzmann machine is a probabilistic network consisting of binary units (s_i) which are fully connected by symmetrical bidirectional links ($w_{ij} = w_{ji}$). The behaviour of the network is governed by an energy (Liapunov) function:

$$E = -\sum_{i<j} w_{ij}s_is_j + \sum_i b_is_i \qquad s_i \in \{0, 1\},\ w_{ij} \in \mathbb{R}^1,\ i = 1, \ldots, n \tag{1}$$

where the second term can be absorbed into the first term by a link to an additional unit being always in state 1 ($w_{i,n+1} \equiv b_i$, $s_{n+1} \equiv 1$). In physics this equation stands for a spin-glass Hamiltonian, with the first term describing the interactions between spins ($s_i = +1$ or -1) and the second an external magnetic field. In NN language, however, the first term describes connection weights between units and the second term acts as a threshold. The energy function can form a very complex landscape with many local extrema.

The idea behind the BM approach is to shape the energy surface by changing w_{ij} and thereby get the desired stable states to correspond to learned patterns. This task is realised stochastically by decreasing the energy of the system with the simulated annealing method [8]. This Monte Carlo approach is used to avoid high-energy local minima in the energy landscape.

The probability of the network being in state α is determined by the energy function (see equation (1)), according to the Boltzmann distribution:

$$P_\alpha = \frac{\exp(-E_\alpha/T)}{\Sigma_\beta \exp(-E_\beta/T)}. \tag{2}$$

Thus the lower the energy E_α of the network, the more probable is the state α. In (2), T is analogous to the temperature of a physical system. Now if the temperature T decreases, a low-energy state becomes more probable, i.e. the distribution becomes narrower and energy barriers between local energy minima become relatively higher. Considering all the units but s_i fixed, (2) yields the probability of s_i being 1:

$$P(s_i = 1) = \frac{1}{1+\exp(-\Delta E_i/T)} \qquad \Delta E_i = \sum_{j \neq i} w_{ij}s_j. \tag{3}$$

The BM can be formulated to solve pattern recognition tasks by clamping a subset of the units, namely input units, to the input pattern and expecting that another subset, namely the output units, changes to the desired pattern which labels the class to which the input pattern belongs. The information-theoretical Kullbach-Rényi measure is used to describe the difference between the desired probability P_α and the actual probability P'_α:

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

The derivative of the measure G gives the learning rule:

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T}(\langle s_i s_j \rangle - \langle s_i s_j \rangle') \tag{4}$$

where $\langle s_i s_j \rangle$ and $\langle s_i s_j \rangle'$ are the desired and the actual mean values of $s_i s_j$, respectively. These means in fact describe correlations between units i and j and they can be related to the probabilities that both of them are simultaneously 1. The first term in (4) describes the Hebbian kind of learning while the second term describes 'unlearning'. Statistics for these mean values are collected after lowering the temperature T to a given final value T_0 and allowing the network to equilibrate. The purpose of this simulated 'cooling' [8] in small steps from a high initial temperature to a low temperature (T_n, T_{n-1}, ..., T_0) is to avoid high local energy minima during the learning and also the recognition phase. This is realised with the Metropolis Monte Carlo method, which sets the fluctuation scales of the system and thus allows 'hill climbing' in the energy landscape. The simulated annealing method is known to perform well in optimisation problems.

3. The mean-field Boltzmann machine

The Boltzmann machine algorithm is stochastic in nature and thus brings up some drawbacks. For example, cooling down the system and also collecting statistics for the mean values in (4) takes a considerable amount of computing time. Therefore, it is worthwhile to seek ways to substitute the MC method by some other, hopefully analytical, method. From statistical mechanics we know that there is, indeed, an analytical but approximative mean-field theory (MFT) approach to solving properties of stochastic systems. The basic idea behind MFT is to suppress fluctuations by replacing the fluctuating s_i values by their averages. One important justification for searching this kind of deterministic method is its inherent parallelism, which would allow one to take full advantage of parallel processors.

Peterson and Anderson [7] devised the MFB by replacing in (4) the mean value of the product $s_i s_j$ by the product of their individual mean values:

$$\langle s_i s_j \rangle \cong \langle s_i \rangle \langle s_j \rangle.$$

Each $\langle s_i \rangle$ is computed from the following system of equations:

$$\langle s_i \rangle = \frac{1}{1 + \exp(-\langle \Delta E_i \rangle / T)} \qquad \langle \Delta E_i \rangle = \sum_{j \neq i} w_{ij} \langle s_j \rangle \qquad i = 1, \ldots, n.$$

Let us consider $\langle s_i \rangle$ as a variable and start from an arbitrary set of initial values $s_i^{(0)}$. Then we can write the following recursion equation:

$$s_i^{(t+1)} := \frac{1}{1+\exp(-\Delta E_i^{(t)}/T)} \qquad s_i^{(t)} \in (0,1) \qquad i = 1, \ldots, n \tag{5}$$

which can reach a stable fixed point.

For certain tasks Peterson and Anderson found this method at least one magnitude faster than the ordinary Boltzmann machine. In addition they could reduce the number of necessary samples. They have studied the exclusive-or (XOR), encoder and the line symmetry problems which are considered as non-trivial for a neural network model. The XOR (or two-dimensional parity) problem consists of classifying the two bit input patterns as to whether or not they are identical. In the encoder problem the input-output pattern pairs should be associated together, and for the line symmetry problem the task is to recognise whether the input patterns are symmetrical or not.

4. The deterministic Boltzmann machine

As mentioned above, the deterministic approach, as employed in the MFB, promises a speed-up in comparison with the stochastic approach but without significant loss of accuracy. This gives us the idea of searching for other, deterministic algorithms, more suitable for applications and hardware implementations than the BM. On the other hand, the number of cooling steps and the amount of statistics in BM yield only approximations to the accurate result. Also, the MFB itself is an approximation. Hence, from the theoretical point of view it is interesting to study how far one can keep approximating the Boltzmann machine without losing too much of its inherent capabilitities.

Next we will propose a new method, called the deterministic Boltzmann machine (DBM) which is based on a simple minimum search principle so that the largest possible step towards the energy minimum is made. This is somewhat reminiscent of the steepest-descent minimisation method. Our method is deterministic like the MFB, so it can offer some speed-up with hopefully small loss of accuracy as compared with the ordinary BM. Another distinction is that while the mean-field Boltzmann machine iterates towards a fixed point, which is somehow related to the energy minimum, the deterministic Boltzmann machine uses another, direct method to get close to that energy minimum. It is a two-step process with discrete and continuous s_i values in the first and second step, respectively.

In the first step the units have three values: $s_i \in \{0, 0.5, 1\}$. The additional value 0.5 is introduced for starting purposes. Thus the network is initiated from a homogeneous state:

$$s_i^{(0)} := 0.5 \qquad i = 1, \ldots, n$$

and for every iteration the energy E decreases by updating only one unit at a time:

$$s_i^{(t+1)} := s_i^{(t)} \qquad i = 1, \ldots, k-1, k+1, \ldots, n;$$

$$s_k^{(t+1)} := \begin{cases} 1 & \text{if } \Delta E_k^{(t)} > 0 \\ 0 & \text{if } \Delta E_k^{(t)} < 0 \end{cases}$$

$$\Delta E_k^{(t)} = \sum_{j \neq k} w_{kj} s_j^{(t)}.$$

The updated k unit is always the one with which the largest decrease in the energy function can be achieved:

$$k : |\Delta E_k^{(t)}| = \max_{i \in A} \{|\Delta E_i^{(t)}|\}.$$

Here A stands for the set of possible energy changes, which would decrease E:

$$A = \{i | s_i^{(t)} = 0.5 \text{ and } \Delta E_i^{(t)} \neq 0 \text{ or } s_i^{(t)} = 1 \text{ and } \Delta E_i^{(t)} < 0 \text{ or } s_i^{(t)} = 0 \text{ and } \Delta E_i^{(t)} > 0\}.$$

The iteration continues until the energy keeps decreasing. The algorithm does not specify how many times a unit will be updated before the set A becomes empty. However, in our experiments updatings are practically always performed just once (a second update was very rare).

The first step of the DBM serves the purpose of getting close to the energy minimum as fast as possible. The second step recalls the MFT idea by obtaining mean values for the s_i of the network. But as we start from a state of the network which is already close the the energy minimum it seems sufficient to use (5) only once. This method has two advantages over the MFB. First, it is not an iterative but a two-step algorithm. Second, it avoids heavy matrix-vector multiplications. In fact, the DBM can be realised using only additive operations and the sigmoid function.

For the BM and the MFB the desired values of the output units are 0 and 1. However, this did not prove to be suitable for the DBM. It is more appropriate to set the desired values closer to the possible values which the network can achieve with the given weights. For a particular input pattern, the possible minimum and maximum values of the energy change ΔE_i for unit i can be determined as

$$\Delta E_i^+ = \sum_{j \neq i} \max\{0, w_{ij}\}$$

$$\Delta E_i^- = \sum_{j \neq i} \min\{0, w_{ij}\} \qquad \text{so that } \Delta E_i^- \leq \Delta E_i \leq \Delta E_i^+.$$

For any change of other units, the value of unit i can be bound:

$$\frac{1}{1+\exp(-\Delta E_i^-/T)} \leq s_i \leq \frac{1}{1+\exp(-\Delta E_i^+/T)}.$$

The desired states of the output units are set as continuous functions of ΔE_i^+ or ΔE_i^- instead of the rigid 0 or 1:

$$s_i = \frac{1}{1+\exp(-\alpha \Delta E_i^-/T)} \text{ instead of } 0$$

$$s_i = \frac{1}{1+\exp(-\alpha \Delta E_i^+/T)} \text{ instead of } 1 \qquad s_i \in \{\text{output units}\}, 0 < \alpha \leq 1.$$

One should bear in mind that with finite simulation times, e.g. stopping the cooling at non-zero temperature, the Boltzmann machine is not guaranteed to converge to a global energy minimum. In this situation the Boltzmann machine learning does not rely on finding the global minimum but rather associates the learned patterns to well defined states of the system. Nevertheless, these states are likely to be strongly related to the global or near-global energy minima. This is of course the case with the DBM, although we expect it to be somewhat further away from the energy minimum. But

this is not a serious handicap as far it does not confuse the learning. In particular this turned out to be the case in problems discussed in § 5. The other reason for expecting the DBM to have good convergence properties is based on the similarity of the second step of the DBM algorithm with the MFB algorithm.

5. Experimental set-up

In this paper we have made comparison between various neural network models using the same statistical pattern recognition task as in [5]. We believe that this gives somewhat wider scope to the performance analysis of NNs than the tasks used in [7]. For completeness, we will next review this statistical pattern recognition task.

The input samples to be recognised are random points in a multidimensional space. Each class is represented by a multivariate probability distribution. In the experiments shown below, there are two classes with circular Gaussian (or normal) distribution:

$$N_i = N(\boldsymbol{m}_i, \mathbf{R}_i) \qquad \boldsymbol{m}_i \in \mathbb{R}^d,\ \mathbf{R}_i \in \mathbb{R}^{d\times d},\ i = 1, 2$$

$$\boldsymbol{m}_1 = \mathbf{0} \qquad \boldsymbol{m}_2 = (r, 0, \ldots, 0)$$

$$\mathbf{R}_i = \begin{pmatrix} \sigma_i & & 0 \\ & \ddots & \\ 0 & & \sigma_i \end{pmatrix} \qquad \sigma_1 = 1,\ \sigma_2 = 2.$$

We have considered two tasks, one with different means (difference $r = \sqrt{5.4}$ for historical reasons) and the other with the same means of the two classes ($r = 0$) called 'easy' and 'hard', respectively (see figure 1). The dimensionality is set to vary from 2 to 8 for each task so that there are 14 different problems to solve.

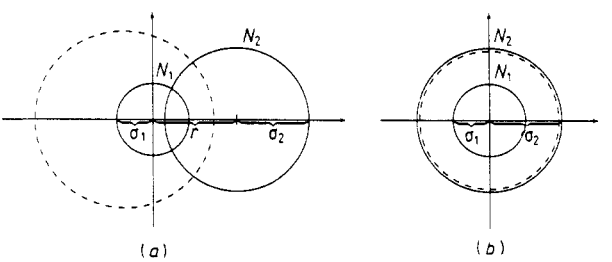


Figure 1. The representation of the two tasks on $\mathbb{R}^2$ ($d = 2$). The Gaussian probability distributions are represented by circles with radius equal to deviations σ_i. The broken circle is the decision boundary. The 'easy' task is illustrated in (*a*) and the 'hard' task in (*b*).

The BM can be considered as an input-output machine so that the set $\{s_i\}$ is divided into three parts: input, output and hidden units. The input units are always clamped by the input pattern, the hidden units are always free to change and the output units are clamped by the desired state or they are free to change during the collection of statistics for the means $\langle s_i s_j \rangle$ or $\langle s_i s_j \rangle'$, respectively. For updating the network connections w_{ij}, only the sign of the right-hand side of (4) is used:

$$\Delta w_{ij} := \Delta w \ \mathrm{sgn}(\langle s_i s_j \rangle - \langle s_i s_j \rangle').$$

This is taking place after sufficient statistics have been collected. In our experiments we have used 10^3 input samples. The temperature parameter T and the step size Δw for changing the connection weights decrease over time with the following schedule:

$$T' := 1.3 + 0.7 \max\{0, 1 - t/130\}$$

$$\Delta w' := 0.125 \cdot 0.99^t \qquad t = 1, \ldots, 400.$$

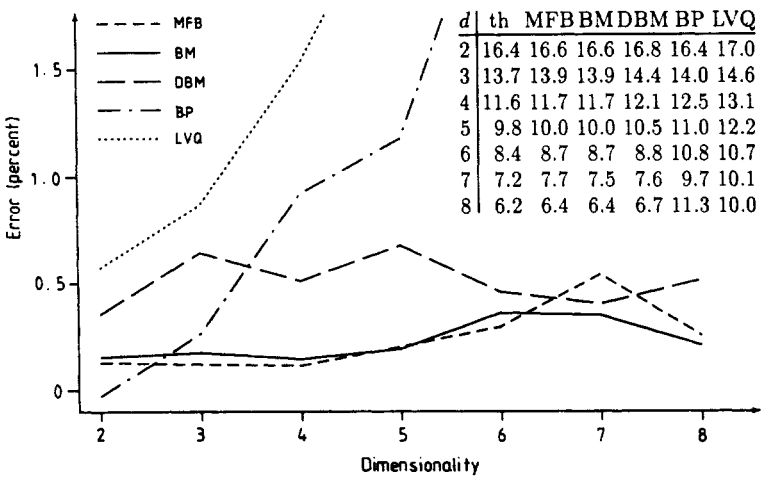
Although the BM can be formulated as having continuous input values, the binary-valued input units are preferable [5, 9]. Therefore, we have digitised their values as follows. In each dimension twenty ranges are determined. Every input unit belongs to one of the twenty ranges and all of them are set to 0 except the one, which is set to 1, that is associated with the input value of the dimension.

The network consists of 20 d input, four hidden and two output units. Initially the values are zero for the biases (b_i) and small random values for the connections (w_{ij}). The configuration is the same for all three machines.

For the BM, three-step annealing and two-step relaxation was used. In the MFB two iterations of (5) were executed. These are notably small values. However, using one magnitude slower annealing for every sample in the BM and also using one magnitude more iteration in the MFB the results were not better than those shown below. In the DBM all units are updated twice.

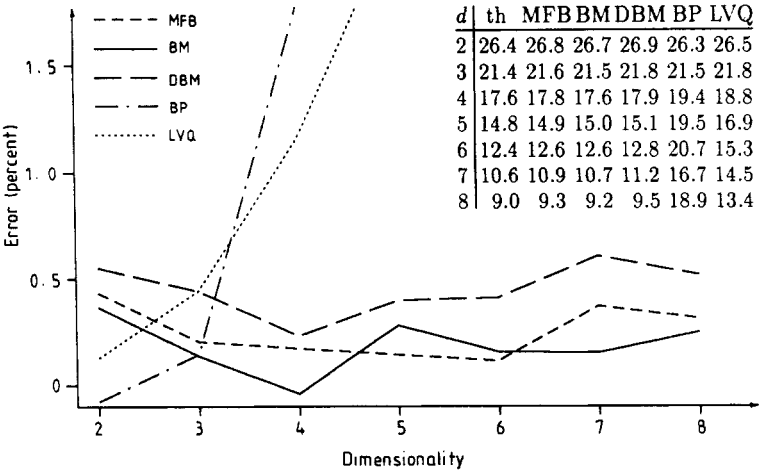
6. Experimental results

The results for the experiments are summarised in figures 2 and 3. The numerical values of these results and also the theoretical limit are given as inserts in the figures. The tasks have been defined so that the analytical calculation of the theoretical limit of recognition is possible. It is obtained by integrating the multivariate circular Gaussian distribution over an offset circle (the decision surface) which is equal to the



d	th	MFB	BM	DBM	BP	LVQ
2	16.4	16.6	16.6	16.8	16.4	17.0
3	13.7	13.9	13.9	14.4	14.0	14.6
4	11.6	11.7	11.7	12.1	12.5	13.1
5	9.8	10.0	10.0	10.5	11.0	12.2
6	8.4	8.7	8.7	8.8	10.8	10.7
7	7.2	7.7	7.5	7.6	9.7	10.1
8	6.2	6.4	6.4	6.7	11.3	10.0

Figure 2. Percentages of error from the theoretical limit for the 'easy' task. The curves are straight lines between these error values of the examined dimensions. The inserted table shows the corresponding absolute numerical values (th stands for the theoretical limit). The values for BP and LVQ are obtained from [5].



d	th	MFB	BM	DBM	BP	LVQ
2	26.4	26.8	26.7	26.9	26.3	26.5
3	21.4	21.6	21.5	21.8	21.5	21.8
4	17.6	17.8	17.6	17.9	19.4	18.8
5	14.8	14.9	15.0	15.1	19.5	16.9
6	12.4	12.6	12.6	12.8	20.7	15.3
7	10.6	10.9	10.7	11.2	16.7	14.5
8	9.0	9.3	9.2	9.5	18.9	13.4

Figure 3. Percentages of error from the theoretical limit for the 'hard' task. Notation is the same as in figure 2.

cumulative distribution function of the non-central χ^2 distribution. The results for backpropagation and learning vector quantisation from [5] are also included in order to make wide comparison possible. All the Boltzmann machines perform well for all the studied dimensionalities and clearly better than BP and LVQ for dimensions $d \geq 4$. This is the case for both 'easy' and 'hard' problems. The differences between the BM and MFB results are not significant. This is not surprising because in the thermodynamic limit, i.e. the number of units (degrees of freedom) goes to infinity, the mean-field theory yields exact results for the network. Although the results for the DBM differ slightly from those of the BM and the MFB, possibly the same thermodynamic limit as for the MFB does not apply.

The computation times for the BM, the MFB and the DBM (executed on a serial computer) were roughly 6, 4 and 3 time units, respectively. In [7] a speed-up factor of 10–30 was quoted. In our experiments this did not appear, which could be partly explained in terms of the nature of the tasks studied. In the experiments we have investigated corrupted data, which was not the case in [7].

In contrast to the study by Peterson and Anderson, we have not performed particular experiments to examine the number of necessary samples for learning. In all our experiments we have used 400 000 samples, which we considered to be a sufficient amount. We have not found that more samples could help any of the three Boltzmann machine methods.

An important factor of Boltzmann machine algorithms is finding 'good' parameters. It was quite easy for the BM and more difficult for the MFB and the DBM. Comparing the BM with the MFB and the DBM, its behaviour is found to be invariant for a larger part of the parameter space.

7. Conclusions

Pattern recognition tasks are known to be difficult and require a large amount of computation time. The inherent parallelism and generalisation ability of neural

networks makes them ideal for statistical pattern classification. In general they need a large amount of samples, but once trained they recognise quickly, especially if they are implemented by using parallel architectures.

The ordinary Boltzmann machine is an example of a stochastic neural network. It relies on a simulated annealing method and it is virtually capable of yielding the best possible solution to any problem. One of its drawbacks is that the stochastic behaviour and the cooling can result in extremely slow operation.

We have reviewed three versions of the Boltzmann machine and compared with two other widely used neural network classifiers. It was found that the ordinary Boltzmann machine is not necessarily as slow as one would think, and it seems to be relatively easy to configure. The mean-field Boltzmann machine demonstrates the power of a statistical mechanical approach in that it performs just as well as the ordinary Boltzmann machine. It was faster, though not by as much as claimed earlier. The performance of the deterministic Boltzmann machine is quite close to them and it is still somewhat faster than the mean-field Boltzmann machine. As regards the other two methods, the performance of both backpropagation and learning vector quantisation exhibit dependence on dimensionality. They performed well only in lower dimensions, but are known as fast algorithms.

The mean-field Boltzmann machine and determinstic Boltzmann machine are viable alternatives to the ordinary Boltzmann machine. Besides their higher speed, they are also more easily implemented in hardware, especially as parallel systems. In fact, there are several silicon implementations of neural network algorithms (even the Boltzmann machine) in progress (e.g. see [10]). They are justified by applications which require flexibility and dedicated hardware.

References

[1] Hopfield J J 1982 *Proc. Natl Acad. Sci. USA* **81** 2554
[2] Ackley D H, Hinton G E and Sejnowski T J 1985 *J. Cognitive Sci.* **9** 147
[3] Huang W Y and Lippmann R P 1987 *IEEE 1st Int. Conf. on Neural Networks* vol 4 (Piscataway, NJ: IEEE) p 485
[4] Rumelhart D E, Hinton G E and Williams R J 1986 *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (Cambridge, MA: MIT Press) p 318
[5] Barna G, Chrisley R and Kohonen T 1988 *Neural Networks* **1** sup 1 7
[6] Kohonen T 1984 *Self-organisation and associative memory* (Berlin: Springer) 2nd edn
[7] Peterson C and Anderson J R 1987 *Complex Systems* **1** 995
[8] Kirpatrick S, Gelatt C D and Vecchi M P 1983 *Science* **220** 671
[9] Prager R W, Harrson T D D and Fallside F 1987 *Comp. Speech Lang.* **1** 3
[10] Tomberg J and Kaski K 1989 *Proc. Nordic Symp. on Neural Computing* (Helsinki: Rolf Nevanlinna Institute)