Paper

A Gradient Ascent Learning Algorithm in Weight Domain for Hopfield Neural Networks

Non-member	Qi Ping CAO	(Tateyama Systems Institute)
Non-member	Zheng TANG	(Toyama University)
Student Member	Rong Long WANG	(Toyama University)

This paper presents a gradient ascent learning algorithm in weight domain for solving local minimum problem of a Hopfield network. The learning algorithm has two phases, the time domain phase and the weight domain phase. The former seeks a minimum of the energy function by updating states in time domain of a Hopfield network, and the latter intentionally increases the energy of the network by modifying weights in a gradient ascent direction of the energy in weight domain after the network update stabilization in time domain is achieved. The two phases are repeated until a condition, for example the energy function E=0 is satisfied. The learning algorithm is applied to a two-neuron Hopfield network and an N-queen problem, extensive simulations are performed and its effectiveness is confirmed.

Keywords: Hopfield neural networks, gradient ascent learning, Local minimum, Weight domain, Time domain

1. Introduction

The time domain behavior of a Hopfield neural network to decrease a well-defined energy function [1,2] has been applied to many constrained optimization problems and has shown potential for solving such problems efficiently [3-5]. Unfortunately, since the energy function of a Hopfield network has many local minima, practical limitations exist: performance is not good with a Hopfield neural network, and performance becomes poorer with larger problems, typically larger than 10 cities when applied to the traveling salesman problem [6, 7]. Furthermore, there is not an effective method to help the network escape from the local minima. The performance may be improved by some sophisticated architectures, such as the Boltzmann machine [7, 8]. The Boltzmann machine uses noise to "shake" the network state out of a local minimum [9]. However, the Boltzmann machine is very slow because of the need for extensive averaging over stochastic variable [10]. Besides annealing, sharpening (T becomes smaller), for example deterministic annealing [11], [12], and adding chaotic noise [13], [14] are also powerful methods for solving the local minimum problems. However, the former takes a very long time due to the meticulous decrease of temperature parameters [13], [14]. The latter doesn't give a general solution to the local minimum problem due to its "control difficult problem" of the dynamics [15].

In this paper, we present a gradient ascent learning algorithm in weight domain for solving the local minimum problem of a Hopfield network. The learning algorithm has two phases, the time domain phase and the weight domain phase. The former seeks a minimum of the energy function by updating the states of the Hopfield network in time domain, and the latter intentionally increases the energy of the network by modifying weights in a gradient ascent direction of the energy in weight domain after the network update stabilization in time domain is achieved. The two phases are repeated until a condition, for example the energy function E=0 is satisfied. The learning algorithm is applied to a two-neuron Hopfield network and an N-queen problem, extensive simulations are performed and its effectiveness is confirmed.

2. Hopfield Network Behavior In Time Domain

A Hopfield network is constructed by connecting a large number of simple processing elements (neurons) to each other [1, 2]. In general, the *ith* processing element is described by two variables: its total input denoted by x_i and its output denoted by y_i. The output is usually related to the input by a simple nondecreasing monotonic output function $f(x_i)$. The function is normally designed to limit the possible values of y_i to the range 0 to 1: hence the function will be nonlinear. For simplicity, $f(x_i)$ is frequently a sigmoid function of the form

$$y_i = f(x_i) = \frac{1}{1 + e^{(-x_i/T)}} \quad \cdots\cdots\cdots\cdots\cdots\cdots (1)$$

where T is a parameter called temperature parameter. The output of the *ith* neuron is fed to the input of the jth neuron by a connection of strength (usually called the weight) W_{ij}. In addition, each has an offset bias (usually called the threshold) of h_i fed to its input. The dynamic behavior of the network, consisting N neurons, is described by the following system of N nonlinear differential equations.

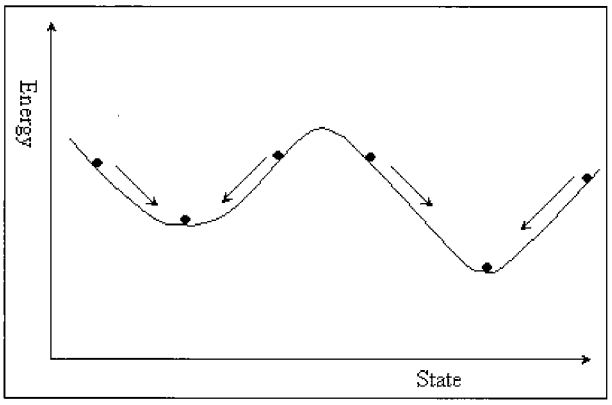


Fig. 1. Conceptual figure of energy function and dynamic of a Hopfield network.

$$\frac{dx_i}{dt} = -\frac{x_i}{\tau} + \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}y_i y_j + h_i \quad \cdots\cdots\cdots (2)$$

where $i = 1, 2, \cdots, N$, and τ is a constant. The elements of x_i are updated either asynchronously (the element to be updated being selected randomly) or synchronously, Hopfield proved that motion equation (Eq.(2)) for a network with symmetric connections ($W_{ij} = W_{ji}$) always lead to a convergence to a local minimum of the quantity (energy function) [1, 2].

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}y_i y_j - \sum_{i=1}^{N} h_i y_i \quad \cdots\cdots\cdots (3)$$

The theory shows that a mathematical quantify E, which might be thought of as the *energy*, decreases during the change in neural state with time (i.e. the updating in time domain) described by Eq.(1). Started in any initial state, the system will move in a general *downhill* direction of the E function, reach a state in which E is a local minimum or a global minimum, and stop changing with time.

Thus, seeking a minimum of the energy function is analogous to seeking a minimum in a mountainous terrain. Figure 1 depicts a two-dimensional version of such terrain. The energy function is reflected in the height of the graph. Each position on the terrain corresponds to a possible state of the network, and the network moves towards a minimum position. In this graph, a local minimum and a global minimum are depicted. If the state of the network is changed, then a corresponding change is made in the horizontal coordinate position on the graph. This change in turn results in a movement downhill toward one of the minima.

The initial state of the network may be thought of as the position of a skier who has dropped randomly onto mountainous terrain. The updating procedure in time domain moves the skier downhill until he gets to the bottom of the most accessible valley. This valley may be at a local or a global minimum.

Although we may visualize this search in two or three dimensions, there are actually as many dimensions as there are processing units. A Hopfield network's convergence procedure in time domain seeks a minimum in such a multi-dimensional mountainous terrain. However, there is no effective method to help the network reach the global minimum from a local minimum.

3. Hopfield Network Learning In Weight Domain

As mentioned above, the original Hopfield network's updating in time domain may lead a convergence to either a local minimum or a global minimum. However there is not an efficient way for the network to reach the global minimum from a local minimum. We propose a gradient ascent learning method of solving the local minimum problem of a Hopfield neural network. Figure 2 shows a flowchart of the learning algorithm. In the flowchart, Phase I (state update phase) performs the state update of a Hopfield neural network in time domain in which a Hopfield neural network seeks new state, a minimum of the energy function. Then result of the updating is evaluated. If a condition for the end of learning is satisfied, then stop; otherwise, go to Phase II (weight update phase). Phase II uses the new steady state and increases the energy by changing the weights of the Hopfield network. Then Phase I is re-performed using the new weights and the new state. Thus, the two phases are repeated until a condition, for example the energy function $E = 0$ is satisfied. The detail algorithm will be described in section 4.

In order to describe the learning algorithm, we use a two-dimensional graph (Fig. 3) of energy function with a local minimum and a global minimum. The energy function value is reflected in the height of the graph. Each position on the energy terrain corresponds to a possible state of the network. For example, if the network is initialized onto the mountainous terrain A, the updating procedure in time domain moves towards a

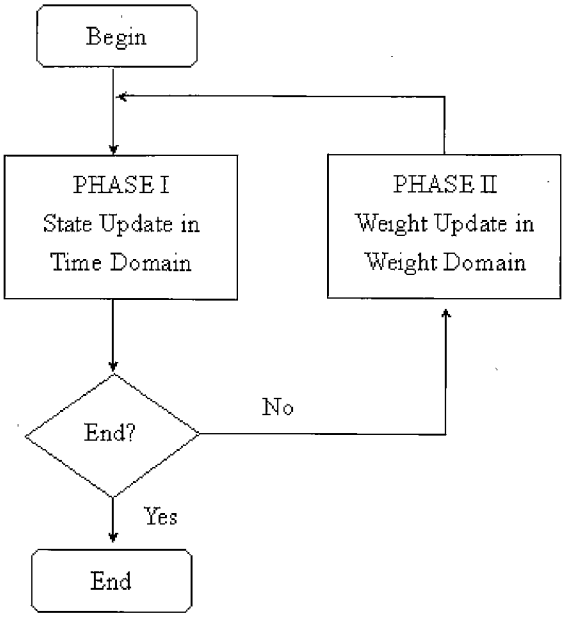


Fig. 2. Flowchart of the learning algorithm.

minimum position and reaches a steady state B (Fig. 3(a)).

Because the energy terrain is determined by various parameters, such as the weights and the thresholds of the network. We can change the weights and the thresholds to increase the energy at the point B so as to fill up the local minimum valley and finally drive the point B out of the valley. Here, suppose that a vector W correspond to the weights and the thresholds of the Hopfield network. Since for the weight vector W, the updating requires the weight change to be in the positive gradient direction, we take

$$\Delta\vec{W} = \varepsilon \bigtriangledown E(\vec{W}) \quad \cdots\cdots (4)$$

where ε is a positive constant and $\bigtriangledown E$ is the gradient of energy function E with respect to the weight W. If the constant ε is small enough, the energy is increased. Applying Eq.(3) to Eq.(4), we then obtain:

$$\Delta W_{ij} = p\frac{\partial E}{\partial w_{ij}} = -py_iy_j \quad \cdots\cdots (5)$$

$$\Delta h_{ij} = q\frac{\partial E}{\partial h_i} = -qy_i \quad \cdots\cdots (6)$$

where p and q are positive constants. y_i, y_j correspond to the state of B.

Now we show that after we change the weights and the thresholds according to Eq.(5) and Eq.(6), point B will be on the slope of a valley. Suppose y_{Bi} represent the state of point B, y_{Pi} represent the state of any point P of energy terrain, then the change of energy in point P by the learning rule (Eq.(5) and Eq.(6)) will be:

$$\begin{aligned}\Delta E_P = &\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}y_{Pi}y_{Pj} + \sum_{i=1}^{N} h_iy_{Pi} \\ &-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(w_{ij} - py_{Bi}y_{Bj})y_{Pi}y_{Pj} \\ &-\sum_{i=1}^{N}(h_i - qy_{Bi})y_{Pi} \\ =&\frac{p}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(y_{Bi}y_{Pi})(y_{Bj}y_{Pj}) \\ &+q\sum_{i=1}^{N}(y_{Bi}y_{Pi}) \quad \cdots\cdots (7)\end{aligned}$$

Because point B is a minimum of energy function and the output of neuron in point B is at or near 0 or 1 [2], from Eq.(7), we can know easily that the change of energy is largest when point P is at the same point as point B, and the larger the difference of state between point P and point B has, the smaller energy changes in point P. Thus, we can see that the valley will be filled up in a most effective way. In general, point B may become a point on the slope of the valley.

Thus, the previous steady state B becomes a point on the slop of a valley (B'). From Eq.(5) we can see that the modification of weights is symmetric, and therefore the new weight will be kept symmetric. Thus, after updating of the Hopfield network with the new weights, the point B' goes down the slope of the valley and reach a new steady state C (Fig. 3(b)). Thus, the repeats of the updating in time domain and the learning in weight

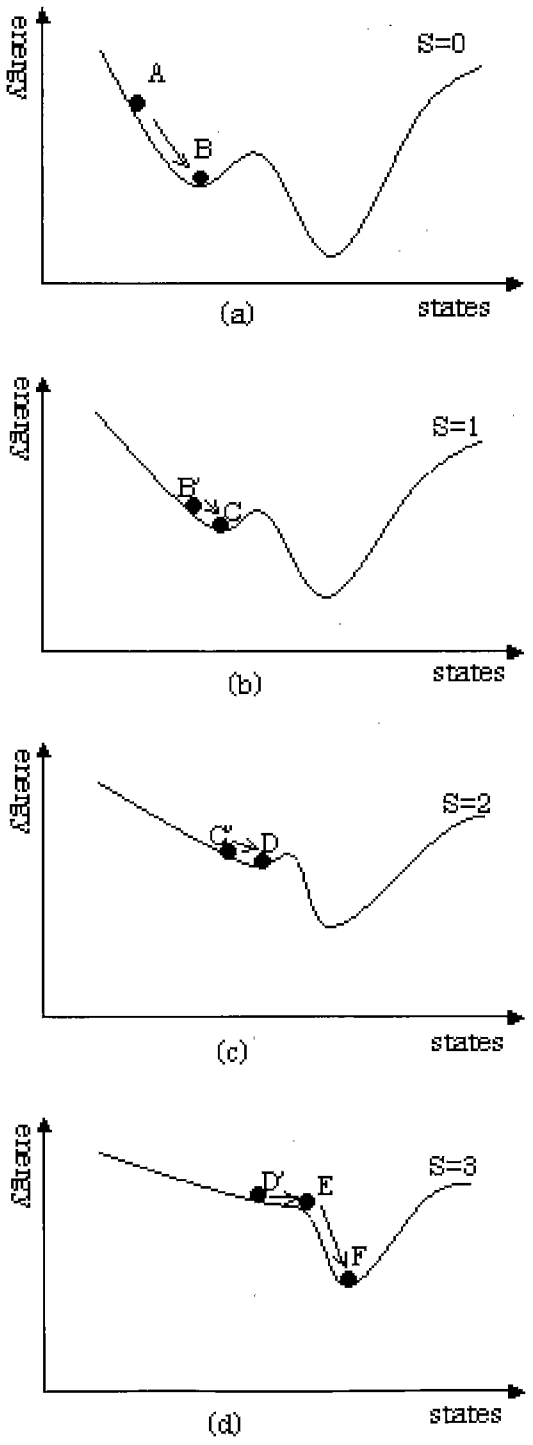


Fig. 3. Relation between energy and state update during the learning process of a Hopfield network with two stable states.

domain may result in a movement out of a local minimum (Figs. 3(c) and (d)).

Note that once the weights and the thresholds are changed, the energy terrain may be changed and the position of global minimum in energy terrain may also be changed. But the problem can be easily solved by performing the update using the original Hopfield network with the new state as its initial state.

Furthermore, in contrast to Boltzman mechine [9], our learning method does not attempt prevent the system getting trapped in a local minimum; rather, if the network converges to a local minimum, it aims to fill up the local minimum valley. Our algorithm proceeds in cycles, with each cycle comprising a energy minimization phase followed by a gradient ascent phase. In the energy minimization phase, the Hopfield network is updated in the normal manner until a minimum state is located. The method then enters the gradient ascent phase, during the course of which the algorithm increases the energy by modifying weights in a gradient ascent direction of energy. The cycles are repeated iteratively until a global minimum or a better one is located. One advantage of this approach is that it has no effect on the energy minimization until such time as the network converges to a stationary point. Our algorithm is a deterministic algorithm. There are no any random perturbations of the search direction. Therefore, the algorithm is very fast, very efficient and reliable compared to Boltzman mechine

4. Algorithm

The following procedure describes the proposed algorithm. Note that the Hopfield updating procedure and the gradient ascent learning procedure are all synchronous parallel procedure.

1. Set the constants p, q in Eq.(5) and (6).
2. Update the Hopfield network with original weights and thresholds until the network converges a stable state (Phase I).
3. Record the stable state.
4. Use Eq.(5) and (6) to compute the new weights and new thresholds (Phase II).

For i=1,$\cdots$,N

a. Compute the ΔW_{ij}, using Eq.(5) for $j = 1, \cdots, N$ and $j \neq i$.

b. Change the w_{ij} for $j = 1, \cdots, N$ and $j \neq i$.

$w_{ij} = w_{ij} + \Delta w_{ij}$

c. Compute the Δh_i, using Eq. (6).

d. Change the h_i.

$h_i = h_i + \Delta h_i$

5. Update the Hopfield network with the new weights and thresholds until the network reaches a stable state.
6. Because the weights and the thresholds of the Hopfield network determine the energy terrain, once the weights and the thresholds are changed, the energy terrain may be changed and the position of global minimum in energy terrain may also be changed. In order to avoid the shift of the state of the global minimum to a specific problem, reupdate the Hopfield network with original weights and thresholds until the network reaches a new stable state.
7. If the new stable state obtained from step 6 is better than the recorded stable state, then replace the recorded stable state with the new stable state obtained from step 6.
8. If a condition, for example the energy function $E = 0$ is satisfied. then terminate this procedure, otherwise use the stable state obtained from step 5 go to the step 4.

5. Simulation Results

Using the learning procedure given above, two examples are cited. The first example is a Hopfield neural network with only two neurons. The learning constants were $p = 0.001, q = 0.01$, the weights and the thresholds were $W12 = W21 = -2.0, h1 = 1.0$ and $h2 = 0.9$ and the temperature parameter $T = 0.24$. Figure. 4 is a 3-dimensional contour line figure of the energy of the network. The two horizontal axes correspond to the states of the two neurons, the vertical axis is the energy of the network. It can be seen that the relation between the energy function and the

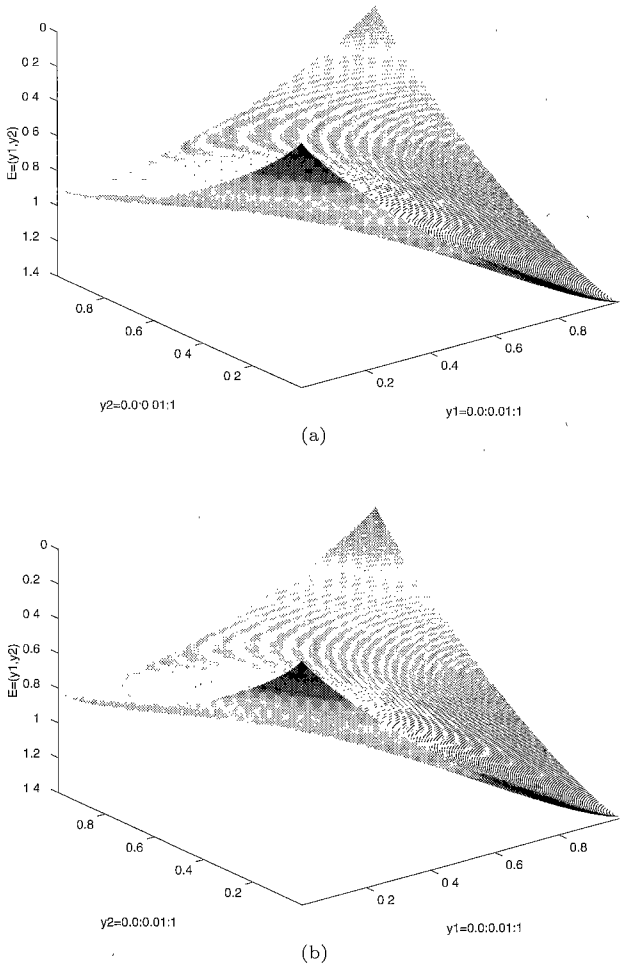


Fig. 4. 3-dimensional energy contour map for a two-neuron Hopfield network: (a) before learning, (b) after learning.

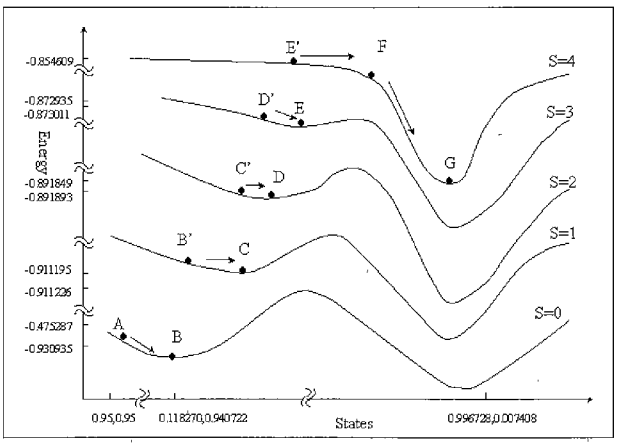


Fig.5 Simulated energy of Hopfield network and its state change during learning.

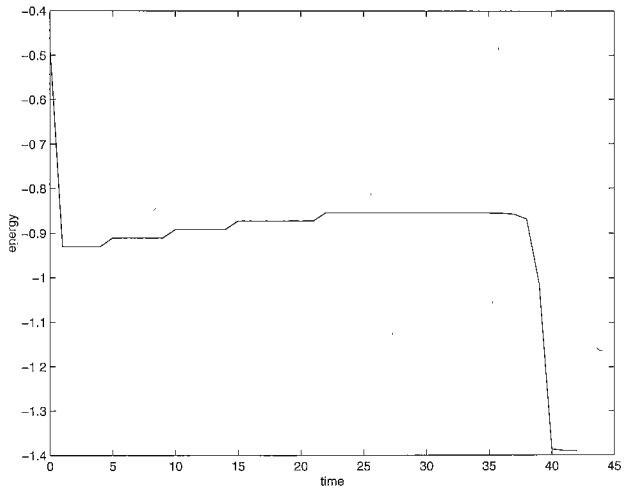
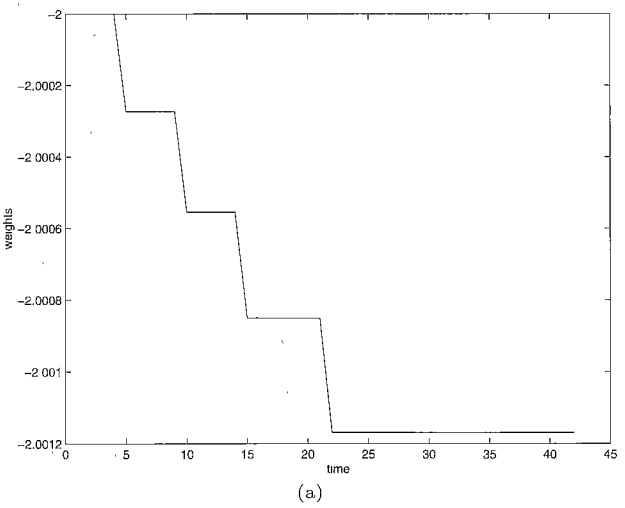
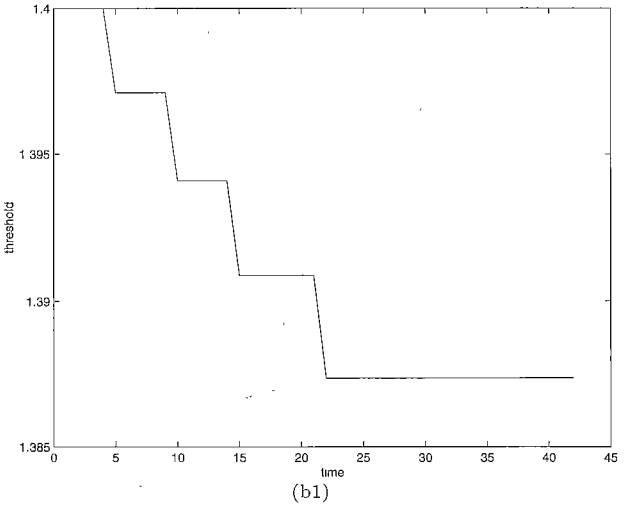


Fig.6 Details of energy change with time during learning.

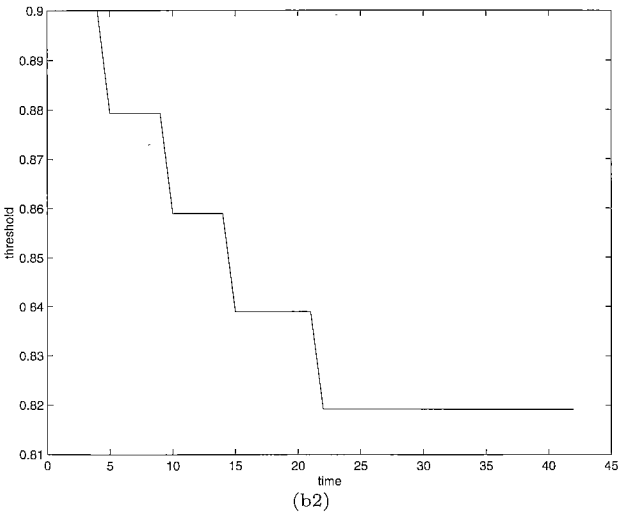
states of the two neurons becomes a shape of a saddle. As shown in Fig. 4(a), the network has a local minimum at the corners of $y_1 = 0, y_2 = 1$ and a global minimum at the corner of $y_1 = 1, y_2 = 0$. Figure. 5 shows the energy of the Hopfield network, and its state change during the learning intuitively. First, in an initial state A at time $s = 0(y_1 = 0.95, y_2 = 0.95)$, the energy takes very large value -0.475287. From the initial state, the network goes down to lower the energy as shown in the figure, and converges to -0.930935 by the energy and $y_1 = 0.118270, y_2 = 0.940722$ by the state B. Next if the learning in weight domain using the formulas (5) and (6) is performed, the energy in state $B(y_1 = 0.118270, y_2 = 0.940722)$ will go up to $E = -0.911195$ and becomes a point B' on the slope of the valley again. This is called the first learning ($s = 1$). If the network updates using the new weights and the new thresholds in time domain from B' again, it will converge to $E = -0.911226$ by the energy and $y_1 = 0.124065, y_2 = 0.932768$ by the state C. In this way, the updating in the time domain and the learning in weight domain are repeated on the Hopfield network. After the $4 - th$ learning ($s = 4$), the network goes up to the energy $E = -0.8546609$ and the state $E'(y_1 = 0.142568, y_2 = 0.902425)$. Thus,



(a)



(b1)



(b2)

Fig.7 Changes of the weight (a) and the thresholds (b1 and b2) during learning.

the local minimum valley of the energy of the network is disappeared, and the network updates to a state $G(y_1 = 0.996728, y_2 = 0.007408)$ from the state E' through F, thus resulting in an escape from the local minimum valley. The energy also decreased abruptly

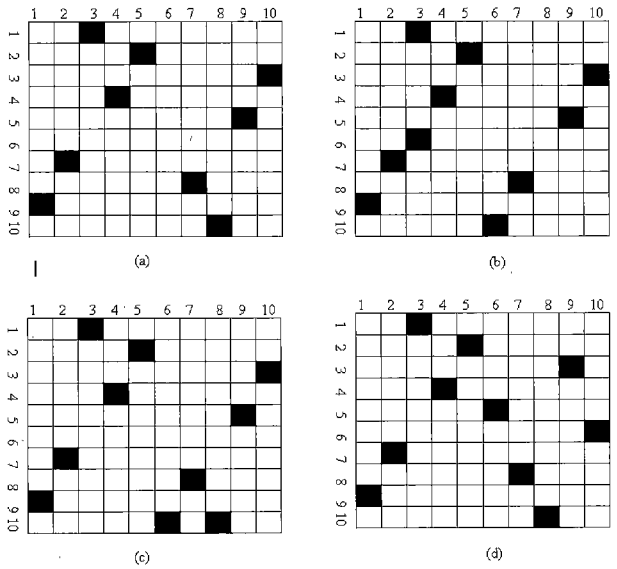


Fig.8 Simulated progressive intermediate placements during the updating in time domain and the learning in weight domain for a 10-queen problem.

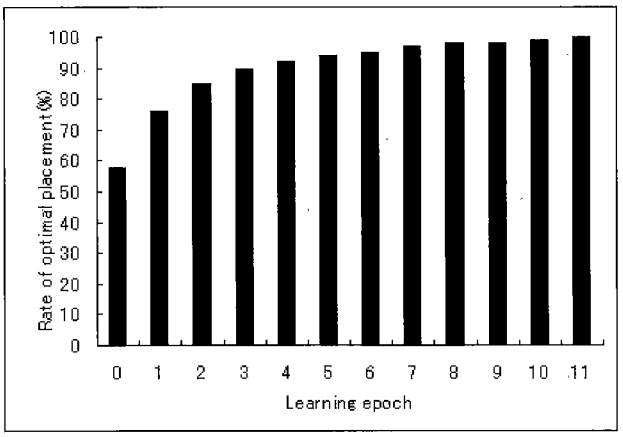


Fig.9 Histogram of the rate of the optimal placements during learning

from $E = -0.8546609(E')$ to $E = -1.389895(G)$ after a small decrease from $E = -0.8546609(E')$ to $E = -0.868840(F)$. The details of the energy change during the learning are shown in Fig. 6. Figure. 7 shows the changes of the weights and the thresholds during the whole learning process. Figure. 4(b) shows the 3-dimensional contour line figure of the energy of the network after learning.

By the above simulation results, the valley (the local minimum) of energy is extinguished, namely, is understood that the learning method proposed in this paper is effective in making a Hopfield network escape from a valley (a local minimum). Furthermore, according to Fig. 5 and 6, by updating in time domain after the $4-th$ learning, the energy decreases from E' to F gently first and falls in the minimum G from F abruptly. This is because the energy has an extraordinary loose hill as shown in Fig. 4(b). These simulation results are in agreement with our learning algorithm. Furthermore, the information such as the structure of the energy and the feature may be acquired from the energy change during the learning by this learning method.

Table 1. The rate of optimal solution to N-Queen problems

Queens	50	100	150	200	300	400	500
Takefuji [17]	86	98	96	93	85	69	67
Maximum NN [18]	78	99	95	95	95	87	86
Proposed Method	100	100	100	100	100	100	100

In the second example, the learning algorithm was applied to an N-queen problem which is a well-known constraint satisfaction problem. The task is given a standard chessboard and N chess queens, to place them on the board so that no queen is on the line of attack of any other queen. The problem can be solved by constructing an appropriate energy function and minimizing the energy function to zero ($E = 0$) using an $N \times N$ two-dimensional Hopfield neural network [16]. A simulation on a 10-queen problem was carried out using the proposed algorithm. The simulation results that illustrate a typical progressive intermediate placement during the updating in time domain and the learning in weight domain are shown in Fig. 8. Initially the Hopfield network converged to a time independent state (Fig. 8(a)). It is obviously not an optimal placement to the problem. After the 1st, 2nd, and 3rd learning, the network found the placements of Fig. 8(b) and (c) and finally the placement of Fig. 8(d), an optimal placement. Furthermore, to see how the learning was being made of optimal placements, we generated 100 initial placements randomly for a 30-queen problem and performed the updating in time domain and the learning in weight domain. Figure. 9 shows a histogram of the rate of the optimal placements. We also found from the simulation results that the learning has led to 100% optimal placement while the traditional Hopfield network provided only about 57% optimal placement. For larger problems such as 50, 100, 150, 200, 300, 400 and 500 queens problems, the algorithm does learn perfectly as compared with the other existing neural network methods [17], [18] in synchronous parallel computation model as illustrated in Table 1.

6. Conclusions

By increasing the energy function of a Hopfield network in weight domain intentionally, we have given a comprehensive description of the way in which a Hopfield network gets out of a steady state which may be either a local minimum or a global minimum. We have also derived the analytic expressions and shown how to use them to real application problems. With the learning algorithm, it is possible to solve the local minimum problem of a Hopfield network. Simulations were performed a two-neuron Hopfield network and an N-queen problem up to 500 queens and produced 100% optimal solutions.

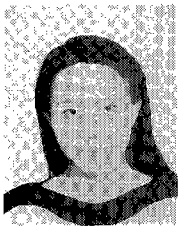
7. Acknowledgment

This work was supported in part by the grant-in-aid for Science Research of the Ministry of Education, Sci-

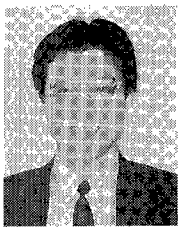
ence and Culture of Japan under Grant (C)(2)12680392.
(Manuscript received May 24, 2001, revised November 1, 2001)

References

(1) J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proc. Natl. Acad. Sci. USA, vol. 79, pp.2554-2558, 1982.
(2) J. J. Hopfiled, "Neuron with graded response have collective computational properties like those of two-state neurons," Proc. Natl. Acad. Sci. USA, vol. 81, pp.3088-3092, 1984.
(3) J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," Bio. Cybern., no. 52, pp.141-152, 1985.
(4) J. J. Hopfield and D. W. Tank, "Computing with neural circuits: a model," Science, no. 233, pp.625-633, 1986
(5) J. M. Zurada, Introduction to Artificial Neural Systems, West Publishing Company, 1992.
(6) G. V. Wilson, and G. S. Pawley, "On stability of the traveling salesman problem algorithm of Hopfield and Tank," Bio. Cybern., no. 58, pp.63-70, 1988.
(7) S. Kirkpafick, Jr. C. D. Glaff, and M. P. Vecchi, "Optimization by simulated annealing," Science, no. 220, pp.650-671, 1983.
(8) D. H. Ackley, Q. E. Hinton and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," Cognitive Science, no. 9, pp.147-169, 1985.
(9) Q. E. Hinton and T. J. Sejnowski, "Learning and relearning in Boltzmann Machines," in Parallel Distributed Processing, MIT Press, 1986, pp.282-317.
(10) J. Hertz, A. Krogh and R. G. Palmer, Introduction to the Theory of Neural Computation, Addison Wesley Publishing Company, 1991
(11)] A. L. Yuille, "Generalized deformable models, statistical physics, and matching problems", Neural Computation 2, pp.1-24, 1990.
(12) P. D. Simic, "Statistical mechanics as the underlying theory of 'elastic' and 'neural' optimizations", Network 1, pp.89-103, 1990.
(13) S. Yoshizawa, S. Doki, S. Okuma, "Deterministic annealing method with equidistortion principle for vector quantization", T. IEE Japan, vol.119-C, no.8/9, pp.1010-1017, 1999.
(14) J. Tani, "Proposal of chaotic steepest descent method for neural networks and analysis of their dynamics", IEICE A, vol.J74-A, no.8, pp.1028-1215, 1991.
(15) K. Aihara, Applied Chaos, Science Publishing, 1994.
(16) Y. Fujitake, Neural Network Parallel Computing, Kluwer Academic Publishers, 1992, Chapter 1
(17) Y. Takefuji, "Neural network parallel computing," Kluwer Academic Publishers, 1992.
(18) Y. Takenaka, N. Funabiki and S. Nishikawa, "A proposal of competition resolution methods on the maximum neuron model through N-Queens problem" J. IPSJ vol.38, no.11, pp.2142-2148, 1997.



Qi Ping Cao (Non-member) received a B.S. degree from Zhejiang University, Zhejiang, China and an M.S. degree from Beijing University of Posts and Telecommunications, Beijing, China in 1983 and 1986, respectively. From 1987 to 1989, she was an Assistant Professor in the Institute of Microelectronics at Shanghai Jiaotong University, Shanghai, China. From 1989 to 1990, she was a Research Student in Nagoya University, Nagoya, Japan. From 1990 to 2000, she was a Senior Engineer in Sanwa Newtech Inc., Japan. In 2000, she joined Tateyama Systems Institute, Japan. Her current research interests include multiple-valued logic, neural networks, and optimizations.



Zheng Tang (Non-member) received a B.S. degree from Zhejiang University, Zhejiang, China in 1982 and an M.S. degree and a D.E. degree from Tsinghua University, Beijing, China in 1984 and 1988, respectively. From 1988 to 1989, he was an Instructor in the Institute of Microelectronics at Tsinhua University. From 1990 to 1999, he was an Associate Professor in the Department of Electrical and Electronic Engineering, Miyazaki University, Miyazaki, Japan. In 2000, he joined Toyama University, Toyama, Japan, where he is currently a Professor in the Department of Intellectual Information Systems. His current research interests include intellectual information technology, neural networks, and optimizations.

Rong Long Wang (Student Member) received a B.S. degree from Hangzhe teacher's college, Zhejiang, China and an M.S. degree from Liaoning University, Liaoning, China in 1987 and 1990, respectively. Since 1990 he has been a lecturer in Benxi University, Liaoning, China. Now he is working toward the Ph.D degree at Toyama University, Japan. His main research interests are neural networks and optimization problems.