International Conference on Power Control and Optimization, Bali, Indonesia, 1-3, June 2009

ENERGY RELAXATION FOR HOPFIELD NETWORK WITH THE NEW LEARNING RULE

SARATHA SATHASIVAM

School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia
e-mail: saratha@cs.usm.my

Abstract.

In this paper, the time for energy relaxation for Little-Hopfield neural network using the new activation rule is shown to be better than the relaxation time using Hebbian learning. However, this should be so given the characteristics of the activation function and show through computer simulations that this is indeed so. In this paper, it has been proven that the new learning rule has a higher capacity than Hebb rule by computer simulations.

Keywords: Little-Hopfield neural networks, program clauses, energy landscape, satisfiability

1. INTRODUCTION

A Little-Hopfield neural network [1,2] minimizes a Lyapunov function, also known as the energy function due to obvious similarities with a physical spin network. Thus, it is useful as a content addressable memory or an analog computer for solving combinatorial-type optimization problems because it always evolves in the direction that leads to lower network energy. This implies that if a combinatorial optimization problem can be formulated as minimizing the network energy, then the network can be used to find optimal (or suboptimal) solution by letting the network evolve freely.

Wan Abdullah [3] proposed a method of doing logic program on a Hopfield network. Optimization of logical inconsistency is carried out by the network after the connection strengths are defined from the logic program; the network relaxes to neural states which are models (i.e. viable logical interpretations) for the corresponding logic program. Using this method as basis, the energy landscape of a Little-Hopfield neural network programmed with program clauses is proven to be rather flat [4,5]. This is supported by the very good agreement with computer simulation results for corresponding network relaxation. According to Altenberg [6], the greater the ruggedness of the landscape, the complexity of the problem will increase. In our problem, the ease of the network programmed with program clauses to find solutions in the solution space, is proved by the flat energy landscape.

This paper is organized as follows. In section 2, the outline of the Little-Hopfield model is given and in section 3; logic programming on a neural network focused on the Hopfield model is described. In section 4, Hebb and the new learning rule are been discussed. This is followed by section 5, where fitness landscapes are discussed. In section 6, theory implementation of the both learning rules are been discussed. Meanwhile, section 7 contains discussion regarding the results obtained from computer simulations. Finally concluding remarks regarding this work occupy the last section.

2. THE LITTLE-HOPFIELD MODEL

The Hopfield model [7,8] is a standard model for associative memory. The Little dynamics is asynchronous, with each neuron updating their state deterministically. The system consists of N formal neurons, each of which is described by an Ising variable [9] $S_i(t),(i=1,2,....N)$. Neurons then are bipolar, $S_i \in \{-1,1\}$, obeying the dynamics $S_i \rightarrow \mathrm{sgn}(h_i)$, where the field, $h_i = \sum_j J_{ij}^{(2)} V_j + J_i^{(1)}$, i and j running over all neurons N, $J_{ij}^{(2)}$ is the synaptic strength from neuron j to neuron i, and $-J_i$ is the threshold of neuron i.

Restricting the connections to be symmetric and zero-diagonal, $J_{ij}^{(2)} = J_{ji}^{(2)}$, $J_{ii}^{(2)} = 0$, allows one to write a Lyapunov or energy function,

$$E = -\frac{1}{2}\sum_i \sum_j J_{ij}^{(2)} S_i S_j - \sum_i J_i^{(1)} S_i \qquad (1)$$

which monotone decreases with the dynamics.

The two-connection model can be generalized to include higher order connections. This modifies the "field" to be

$$h_i = ... + \sum_j \sum_k J_{ijk}^{(3)} S_j S_k + \sum_j J_{ij}^{(2)} S_j + J_i^{(1)} \qquad (2)$$

where "....." denotes still higher orders, and an energy function can be written as follows:

$$E = ... - \frac{1}{3}\sum_i \sum_j \sum_k J_{ijk}^{(3)} S_i S_j S_k - \frac{1}{2}\sum_i \sum_j J_{ij}^{(2)} S_i S_j - \sum_i J_i^{(1)} S_i \qquad (3)$$

provided that $J^{(3)}_{ijk} = J^{(3)}_{[ijk]}$ for *i, j, k* distinct, with [...] denoting permutations in cyclic order, and $J^{(3)}_{ijk} = 0$ for any *i, j, k* equal, and that similar symmetry requirements are satisfied for higher order connections. The updating rule maintains

$$S_i(t+1) = \mathrm{sgn}[h_i(t)] \tag{4}$$

3. LOGIC PROGRAMMING

In the simple propositional case, logic clauses take the form $A_1, A_2, \ldots\ldots, A_n \leftarrow B_1, B_2, \ldots\ldots, B_m$. , which says that (A_1 or A_2 or .... or A_n) if (B_1 and B_2 and ... and B_n); they are program clauses if $n = 1$ and $m \geq 0$: we can have rules e.g. $A \leftarrow B, C.$ saying $A \vee \neg(B \wedge C) \equiv A \vee \bar{B} \vee \bar{C}$, and assertions e.g. $D \leftarrow .$ saying that D is true.

A logic program consists of a set of program clauses and is activated by an initial goal statement. In Conjunctive Normal Form (CNF), the clauses contain one positive literal.

Basically, logic programming in Hopfield model [6] can be treated as a problem in combinatorial optimization. Therefore it can be carried out in a neural network to obtain the desired solution. Our objective is to find a set of interpretation (i.e., truth values for the atoms in the clauses which satisfy the clauses (which yields all the clauses true). In other words, the task is to find 'models' corresponding to the given logic program.

The following algorithm shows how a logic program can be done in a Hopfield network based on Wan Abdullah's method:

i) Given a logic program, translate all the clauses in the logic program into basic Boolean algebraic form.
ii) Identify a neuron to each ground neuron.
iii) Initialize all connections strengths to zero.
iv) Derive a cost function that is associated with the negation of all the clauses, such that $\frac{1}{2}(1+S_x)$ represents the logical value of a neuron *X*, where S_x is the neuron corresponding to *X*. The value of S_x is define in such a way that it carries the values of 1 if *X* is true and -1 if *X* is false. Negation (neuron *X* does not occur) is represented by $\frac{1}{2}(1-S_x)$; a conjunction logical connective is represented by multiplication whereas a disjunction connective is represented by addition.
v) Obtain the values of connection strengths by comparing the cost function with the energy, *H*.
vi) Let the neural networks evolve until minimum energy is reached. Checked whether the solution obtained is a global solution.

The applied methodology may be summarized in the following way: given an optimization problem, find the cost function that describes it, design a Hopfield network whose energy function must reach (one of) its minima at the same point in configuration space as the cost function, so that the stable configurations of the network correspond to solutions of the problem. Detail review regarding neural network logic programming is not provided in this paper, but instead refers the interested reader to Wan Abdullah's paper [3,4,5].

4. LEARNING RULE

4.1 HEBB RULE

Hebb's postulate of learning is the oldest and most famous of all learning rules; it is named in honor of the neurophysiologist Hebb [10]. According to him:

When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased.

This statement is made in a neurobiological context. We may expand and rephrase it as a two-part rule:

1) If two neurons on either side of a synapse (connection) are activated simultaneously, then the strength of that synapse is selectively increased.

2) If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.

The simplest form of Hebbian learning formula for the modification of synaptic strengths, J_{ij} when memorizing a pattern $\{\xi\}$ is described by:

$$J_{ij}(new) = \lambda \xi_i \xi_j + J_{ij}(old) \tag{5}$$

The coefficient λ can be called the rate of learning.

If learning is started from zero connection strengths or *tabula rasa*, e.g. (5) can be rewritten as:

$$J_{ij} = \lambda \xi_i \xi_j \tag{6}$$

4.2 THE NEW LEARNING RULE

Storkey [11] introduced new learning rule.

Definition: The weight matrix w_{ij} of an attractor neural network is said to follow the new learning rule if it obeys

$w_{ij}^0 = 0 \forall i, j$ and

$$w_{ij}^{\nu} = w_{ij}^{\nu-1} + \frac{1}{n}\xi_i^{\nu}\xi_j^{\nu} - \frac{1}{n}\xi_i^{\nu}h_{ji}^{\nu} - \frac{1}{n}h_{ji}^{\nu}\xi_j^{\nu} \quad (7)$$

where

$$h_{ij}^{\mu} = \sum_{k=1,k\neq i,j}^{n} w_{ik}^{\mu-1}\xi_k^{\mu} \quad (8)$$

is a form of local field at neuron i (the input to the neuron i), and ξ^{μ} is the new pattern to be learnt ($\xi_i^{\mu} = \pm 1$).

5. FITNESS LANDSCAPES

In the process of finding the global minimum, corresponding to the global optimum, the neural networks might be caught in local minima. So, satisfiability is related to the ruggedness of the energy landscapes. The more rugged the energy landscape, the harder it will be to obtain good solutions. In neural networks, every configuration of S_i is represented by a point in configuration space and has an energy value associated with it, forming an energy landscape.

The energy landscape affects how easy the network can find the global minimum. Imada and Araki [10] showed that as the number of patterns to be stored increases, the task to locate one of the optimal solutions becomes difficult and this phenomenon is due to the increased ruggedness in the energy or 'fitness' landscape. This provides a useful feature of fitness landscapes and measures of their structure: the structure of a landscape can reflect how easy or difficult it is for a search algorithm to find good solutions [12].

5.1 FITNESS EVALUATION

Consider a Hopfield network consists of N bipolar neurons with patterns:

$$\xi^{\nu} = (\xi_1^{\nu},....,\xi_N^{\nu}), \qquad \nu = 1,....,p \quad (9)$$

After one of these patterns is given to the network, neuron states are updated asynchronously.

$$S_i(t+1) = \mathrm{sgn}\left(\sum_{j\neq i}^{N} J_{ij}S_j(t)\right) \quad (10)$$

where $S_i(t)$ is the state of i-th neuron at time t.

When stored patterns ξ^{ν} evolve, fitness value based on Kauffman's model is defined as [12]:

$$m^{\nu}(t) = \frac{1}{N}\sum_{i=1}^{N}\xi_i^{\nu}S_i^{\nu}(t) \quad (11)$$

$$f = \frac{1}{t_0 \cdot p}\sum_{t=1}^{t_0}\sum_{\nu=1}^{p} m^{\nu}(t) \quad (12)$$

In Imada and Araki [12], t_0 is set to $2N$, twice the number of neurons. The network is been simulated for different numbers of neurons and clauses.

6. THEORY IMPLEMENTATION

Firstly, random program clauses are generated. Then, initializing initial states for the neurons in the clauses is been carried out. Next, let the network evolve until minimum energy is reached. Test the final state obtained whether it is a stable state. If the states remain unchanged for five time steps, then consider it as stable state. Following this, calculate the corresponding final energy for the stable state. If the difference between the final energy and the global minimum energy is within tolerance value, then consider the solution as global solution. Then, analyze the energy landscapes.

We run the relaxation for 1000 trials and 100 combinations of neurons so as to reduce statistical error. The selected tolerance value is 0.001. All these values were obtained by trial and error, where tried several values as tolerance values, and selected the value which gives better performance than other values.

7. RESULTS AND DISCUSSION

In our analysis, the energy landscapes formed by network programmed by program clauses are rather flat due to the zero fitness values (differences between the fitness values of neighboring points are zero). From the graphs obtained, observed that energy landscapes for number of neurons (NN) =10, 20, 30 and 40 neurons are almost similar. Due to this, overlap between the values occurred resulting one single line around 1. So, to simplify the graph, just the results obtained for NN=40 is presented.

In the previous section, it had been argued that program clauses are always satisfiable. So, the neurons do not yield or get trapped in any sub-optimal solutions. Smoothness in the figures obtained reflects this idea. Energy landscapes for program clauses are flat because solutions always exist for program clauses, and there is the freedom of unattached literals. Figures 1, 2 and 3 illustrate the differences in fitness values obtained, where NC1, NC2 and NC3 indicate the numbers of literals per clause.

Table 1 shows the time comparison between the both methods. It can be observed that the new rule performs better. It is already known that the new learning rule has a higher capacity than the Hebb rule, and does not suffer significant capacity loss when input clauses get larger.

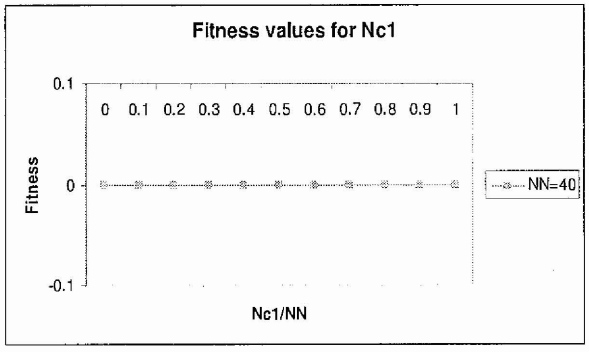

Figure 1: Fitness value for NC1

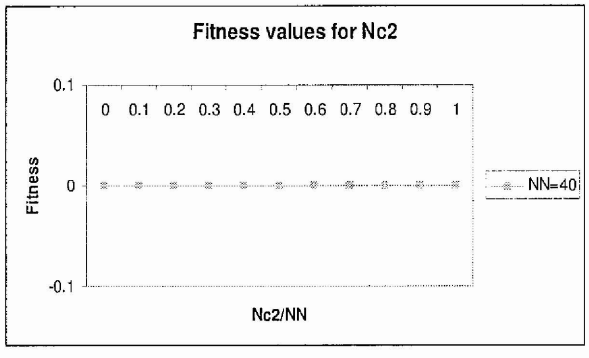

Figure 2: Fitness value for NC2

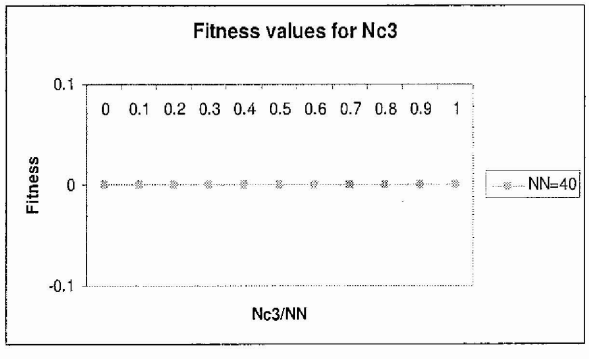

Figure 3: Fitness value for NC3

8. CONCLUSION

A unified approach for proving that the new rule performs better than the Hebb rule when the input clauses get larger is presented in this paper. Smoothness in the figures obtained reflects that the solutions never stuck in any local minima values. Besides that, the new learning rule has a higher capacity than the Hebb rule, and does not suffer significant capacity loss when input clauses get larger. This theory is supported by the very good agreement of the time comparison between the both methods. As a conclusion the new learning rule performs better than Hebb rule in doing logic programming in Hopfield neural network.

Table 1: Time comparison for energy relaxation between Hebbian Rule and the New Rule

NC1/NN	CPU TIME (second)	
	HEBB RULE	NEW RULE
0	0	0
0.1	3	1
0.2	8	3
0.3	13	5
0.4	19	7
0.5	23	9
0.6	27	12
0.7	31	15
0.8	36	19
0.9	41	23
1.0	45	26

ACKNOWLEDGEMENTS

This research was supported by FRGS grant offered by Ministry of Higher Education and short term grant by Universiti Sains Malaysia.

REFERENCES

[1] J.J. Hopfield. "Neural Networks and Physical Systems with Emergent Collective Computational abilities", *Proc. Natl. Acad. Sci. USA*, 79, 1982, 2554-2558.

[2] Little, W.A. *Math. Biosci.* 19, 1974, 101-120.

[3] W.A.T. Wan Abdullah. "Logic programming on a neural network". *Int. J. Intelligent Sys. 7*, 1982, 513-519.

[4] Saratha Sathasivam. & Wan Abdullah, W.A.T. *Flatness of the Energy Landscapes of Horn Clauses*, MATEMATIKA, 23(2), 2008, pp 47-156.

[5] Saratha Sathasivam. & Wan Abdullah, W.A.T. *Logic Learning in the Hopfield Networks*, Modern Applied Science, 2(3), 2008, pp 57-62.

[6] Altenberg, Lee. *Handbook of Evolutionary Computation*: Oxford University Press, 1997.

[7] Haykin, S. *Neural Network: A Comprehensive Foundation.* New York: Macmillan, 1999.

[8] J.J. Hopfield. "Neural computation of decisions in optimization problems", *Biol. Cybern., 52*, 1985, 141-152.

[9] W.A.T. Wan Abdullah. "The logic of neural networks". *Physics Letters A. 176*, 1993, 202-206.

[10] Hebb, D.O. (1949). *The Organization of Behavior.* New York: John Wiley.

[11] Storkey, A.J. (1997). Increasing the capacity of the Hopfield network without sacrificing functionality. In W. Gerstner& A. Germond & M. Hastler & J. Nicoud (Eds.), *ICANN97: Lecture Notes In Computer Science 1327*, pp 451-456, Berlin: Springer.

[12] Imada, A. & Araki, K. What does the landscape of a Hopfield associative memory look like? *Proceedings of the 7th Annual Conference on Evolutionary Programming*, Springer Verlag, Lecture notes in Computer Science (in press). 1997.