An analysis of the Hopfield Model of Associative Memory by means of a Genetic Algorithm

Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-01 Japan
akira-i@is.aist-nara.ac.jp

Keijiro Araki
Graduate School of Information Science
and Electrical Engineering, Kyusyu University
6-1 Kasuga-koen, Kasuga, Fukuoka, 816 Japan
araki@c.csce.kyusyu-u.ac.jp

Abstract – We apply evolutionary algorithms to Hopfield's neural network model of associative memory. This paper describes a genetic algorithm using *ternary* chromosomes which successfully evolves both random and over-loaded Hebbian synaptic weights to function as an associative memory. The goal of this study is to shed new light on the analysis for understanding the model.

1 Introduction

Associative memory is a dynamical system which has a number of stable states with a domain of attraction around them [1]. If the system starts at any state in the domain of attraction, it will converge to the stable state. In 1982, Hopfield proposed a fully connected neural network model of associative memory [2]. Many discussions have been made since then, nevertheless some issues remain open and have not been explored. We apply genetic algorithms [3, 4] to the model to shed new light on its analysis from different aspects. We are interested, among others, in the studies made by Sompolinsky [5] and Derrida, Gardner & Zippelius [6]. As will be described later in detail, they analytically studied the properties of the Hopfield network by modifying the synaptic weights between neurons. To be more specific, Sompolinsky gave small perturbations η_{ij}'s to the synaptic weights J_{ij}'s, i.e.,

$$J_{ij} + \eta_{ij}.$$

On the other hand, Derrida et al. pruned some of the connections instead, i.e.,

$$c_{ij} \times J_{ij},$$

where c_{ij}'s take the value of either 0 or 1. We made similar simulations, with difference being that they *randomly* selected the connections to be modified while we *adaptively* select the connections using evolutionary computation's techniques.

The remainder of this paper is organized as follows. Since we modify only synaptic weights within framework of the Hopfield model, the model is described more fully in the following section. Other evolutionary approaches to the Hopfield model will be described in the third section. Our genetic algorithm is somewhat differnet from the standard one and these are addressed in the fourth section. After presenting experimental results and discussion, the final section describe the concluding remarks.

2 The Hopfield Model

The Hopfield model consists of N neurons and N^2 synapses. Each neuron can be in one of two states ± 1. The network is to memorize p bipolar patterns $\xi^\nu = (\xi_1^\nu, \cdots, \xi_N^\nu)$ $(\nu = 1, \cdots, p)$ in a distributed way among neurons as equilibrium states. We call these states *memorized patterns* to distinguish from other states. Hopfield employed a discrete-time, asynchronous update scheme. Namely, at most one neuron

updates its state at a time according to the sign of weighted sum of all other neurons' states, i.e.,

$$s_i(t+1) = f\left(\sum_{j\neq i}^{N} w_{ij}\, s_j(t)\right),$$

where $s_i(t)$ is a state of i-th neuron at time t, w_{ij} is a synaptic weight from neuron j to neuron i, and $f(z) = 1$ if $z \geq 0$ and -1 otherwise.

Then an input out of 0the memorized patterns which is given small noise within the size of basin of attraction, should relax to the memorized pattern after several steps of update. If we do not give any noise to the input, on the other hand, all neuron states should remain unchanged from the start, and the patterns are said to be memorized as *fixed points*. These behaviors of the collective states of the individual neurons are characterized by the synaptic weights. When these synaptic weights are determined appropriately, the network stores some number of patterns as fixed points. Hopfield specified w_{ij}'s by the Hebbian rule [7], i.e.,

$$w_{ij} = \frac{1}{N}\sum_{\nu=1}^{p} \xi_i^\nu \xi_j^\nu \ (i \neq j),\ w_{ii} = 0.$$

Then giving one of the memorized patterns, possibly including a few errors, to the network as an initial state will result in the stable state after updating.

The network has the upper limit on the number of patterns which can be stored and recalled associatively. Hopfield suggested by using computer simulation that the maximum number p of the patterns to be stored in a network with N neurons is $0.15N$, if a small error in recalling is allowed. Later, this was theoretically calculated by Amit, Gutfreund, & Sompolinsky [8] by using *replica* method. They showed that the storage capacity is $p = 0.138N$. In 1987, McEliece, Posner, Rodemick &Venkatesh [10] proved that when

$$p < N/4\ln N$$

holds, the Hopfield's model will recall the memory without error.

The storage capacity depends strongly on how synaptic weights are specified. The specification of the synaptic weights is conventionally referred to as *learning*. Other learning schemes instead of Hebbian rule have been proposed for their increased storage capacity. The *pseud-inverse matrix* method by Kohonen & Ruohonen [11] and *spectral algorithm* by Pancha & Venkatesh [12] are the examples, and the capacity is enlarged to $p = 0.5N$ and $p = N$, respectively. Both are extensions of the Hebbian learning. In this paper, we exploring it by means of a genetic algorithm.

Then, what is the ultimate capacity when only the learning scheme is explored within the Hopfield's framework? Generally the storage capacity is traded off against basin size. Gardner [13] studied the optimal storage capacity as a function of the size of basin of attraction. She showed that as basin size tends to zero the ultimate capacity will be

$$p = 2N,$$

proposing an algorithm to obtain the weight matrix.

The storage capacity also depends on other aspects of neural networks. For example, we can increase the storage capacity exponentially instead of proportionally by modifying the architecture of the network. Chiueh & Goodman [14] proposed a new architecture in which storage capacity scales exponentially with the number of neurons N, and show that 2^N might be the ultimate storage capacity.

From the discussions above, it does not seem to be interesting any more to study the Hopfield model of associative memory. However, there are almost infinite number of solutions in the synaptic connection's space. Although a modest amount of them have been reported already, they were all found heuristically. One reason why we are interested in the Hopfield's model is to obtain the insight into the overall distribution of these solutions, for instance.

3 Related Works

Since the late 1980's, genetic algorithms have been used extensively to search the optimal set of synaptic weights of neural networks and/or their optimal architectures (see [15, 16], and references quoted therein).

However all of these researches were for layered neural networks, and applications of genetic algorithms to Hopfield networks remain few so far.

In this paper, individual matrices $\{w_{ij}\}$'s are produced by modifying an ancestor matrix $\{W_{ij}\}$ with chromosomes c_{ij}'s as follows:

$$w_{ij} = c_{ij} \times J_{ij}.$$

The chromosome is made up of mostly 1 but includes a few of 0 and -1, and the evolution works on these chromosomes. The other approach which uses the real-coded chromosomes has been reported [17]. The modification is made by

$$w_{ij} = W_{ij} + \eta_{ij},$$

where η_{ij} is a real-valued allele of the chromosome. They search the optimal combinations of synaptic weights in the N^2-dimensional space of w_{ij}, as a real-valued parameter optimization problem.

In fact, the Hopfield model has favorable properties as a test function of real-valued parameter optimization problems. Firstly, the model is multi-modal in the sense that for a fixed set of patterns to be stored, there exist almost infinite number of combinations of synaptic weights which give networks a function of associative memory. Secondly, the model has a global optimum in the sense of Gardner [13]. Finally, the model is also multi-objective since there is a trade off between the storage capacity and error correcting capability.

These researches are both sides of one coin. To use the model as a test function, the model must be understood more clearly.

4 GA Implementation

In this simulation, a set of p random bipolar patterns ξ^ν $(\nu = 1, 2, \cdots, p)$ are produced before each run. The goal of the genetic algorithm is to find a weight matrix which stores all these patterns as fixed points. A weight matrix $\{W_{ij}\}$ is also pre-determined as an ancestor, which is either of the following two forms: (1) over-loaded Hebbian synaptic weights and (2) Real-valued random synaptic weights. This matrix remains fixed during evolution. Then a population of chromosomes are initialized at the beginning of a run. Each chromosome is a N^2-dimensional vector whose components are either 1, 0, or -1. These are called as *alleles*. We denote the chromosome as

$$(c_{11}, c_{12}, \cdots, c_{1N}, c_{21}, \cdots\cdots c_{NN})$$

In each generation, a population of chromosomes make ancestor's copies by multiplying each allele to its corresponding component of the ancestor matrix. Namely,

$$w_{ij}^k = c_{ij}^k \times W_{ij}$$

where w_{ij}^k is ij-component of the copied matrix and c_{ij}^k is $(iN+j)$-th allele of the k-th chromosome. The superscript k denote the individual number in the population. Since the rate of alleles which take value 0 or -1 in a chromosome is small, these matrices are only slightly different from the ancestor.

The chromosomes are modified through crossover and mutation operation. Fitness values of corresponding phenotype are evaluated. According to the fitness values, individuals of the next generation are selected from both parents and offspring, i.e., a $(\mu + \lambda)$-strategy in evolution strategy's [18] terminology. The cycle of reconstructing the new population with better individuals and restarting the search is repeated until a perfect solution is found or a maximum number of generation has been reached. Specific details of the operations are as follows.

Initialization: The N^2 alleles in each chromosome are initialized mostly to be 1. Allele 1, 0 and -1 are produced with the probability of $p_{c=1}$, $p_{c=0}$ and $p_{c=-1}$, respectively, such that

$$p_{c=1} >> p_{c=0} \approx p_{c=-1}.$$

Fitness evaluation: When one of memorized patterns ξ^ν is given to the network as an initial state, the state of neurons varies from time to time afterwards (unless ξ^ν is a fixed point). In order for

the network to function as associative memory, the instantaneous network state must be similar to the input pattern. The similarity is defined by,

$$m^\nu(t) = \frac{1}{N}\sum_{i=1}^{N} \xi_i^\nu s_i^\nu(t),$$

where $s_i^\nu(t)$ is the state of neuron i at time t. Clearly it is related to Hamming distance between the two states. This is conventionally referred to as *overlap*. The value ranges from -1 to 1. The quality of retrieval of memorized patterns are represented by $\langle m^\mu \rangle$, a temporal average of $m^\nu(t)$ over certain time interval t_0. Note that we can obtain an information about the distribution of attractors with these p parameters $\langle m^\mu \rangle$. We evaluate the fitness value of each network by further averaging the $\langle m^\mu \rangle$'s over all memorized patterns. Namely, the fitness f is

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

In this paper, t_0 is set to $2N$, twice the number of neurons. Note that the fitness 1 implies all the p patterns are stored as fixed points, while all other cases have a fitness less than 1.

Selection: We use the truncation-selection [19], i.e., two parent chromosomes are chosen *randomly* from the best T% of the population.

Recombination: Recombination is made with uniform crossover [20] which operates on alleles of the two selected parents' chromosomes, i.e., two parents $(u_1, \cdots, u_n)$ and $(v_1, \cdots, v_n)$ produce an offspring $(w_1, \cdots, w_n)$ such that w_i is either u_i or v_i with equal probability.

Mutation: Mutation cyclically rotates value of alleles which are selected with the probability p_m.

$$(1) \to (-1), \quad (-1) \to (0), \quad (0) \to (1).$$

5 Experiments

The parameters we used in the genetic algorithm are as follows. The population number is 256. This is just because of our computer resources. The alleles 1, 0 and -1 in each chromosome are produced with the ratio of $68 : 1 : 1$. The mutation probability p_m is set to 0.01. In truncation selection, T is chosen to be 40%. Searching procedure is iterated until 12000 generation unless perfect solution is not found.

These were determined mainly on the basis of trial and error.

6 Results and Discussion

6.1 Start with Over-Loaded Hebbian Synaptic Weights

In this experiment, we start the genetic algorithm with a Hebbian weight matrix which learned a set of patterns more than the capacity of the Hebbian learning rule. The networks of this type were analytically investigated by Amit, Gutfreund and Sompolinsky [8, 9]. As the number of patterns increases, the correlation of these patterns can not be neglected. When the number exceeds the capacity, some fixed point attractors are shifted to their neighborhood, and other (spurious) attractors around memory states are merged into spin-glass attractors, i.e., attractors whose temporal average overlap is much lower than 1.

Let us take an example. We made a network with 49 neurons learn 16 random patterns ($p = 16, N = 49$). Note that the capacity of this network size is around 8 patterns. Although each input of the memorized patterns to the network resulted in the convergence to stable attractors, most of them are not memorized states. These attractors, in general, can be studied by the temporal evolution of Hamming distance of instantaneous network state from the input pattern.

In Figure 1, we show some trajectories from the examples when each input out of the 16 learned patterns is presented to the network. Note that changes are at most one at a time because of synchronous

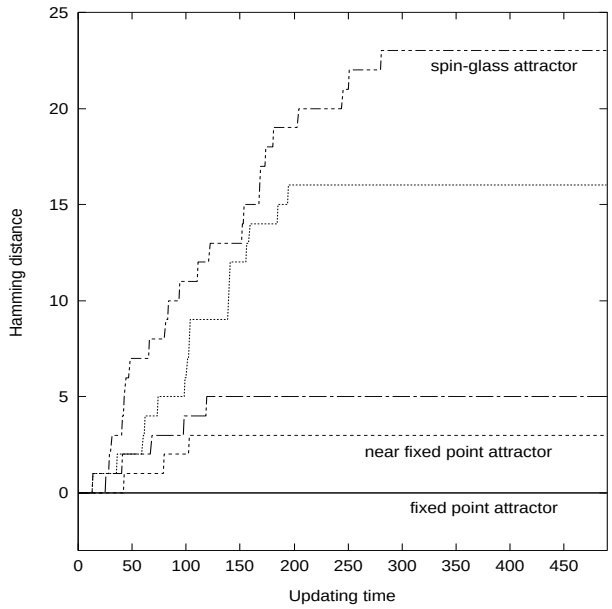

Figure 1: **Temporal evolution of Hamming distance of instantaneous network state from input pattern: starting with Hebbian weights.**

update. We observed that Hamming distances of the 16 stable states from their corresponding input were 0, 0, 0, 0, 1, 2, 3, 3, 5, 6, 9, 11, 16, 17, 21 and 23. Namely, the four are fixed-point attractors and the rest are distributed from near-fixed-point attractors to spin-glass-attractors.

We use this weight matrix as the ancestral matrix for our genetic algorithm. Surprisingly, we obtained a evolved matrix which store all these 16 patterns perfectly as fixed points after 10795 generations. We show the evolution of the best fitness in each generation in Figure 2. In the figure, we also show the representative samples for p=15 and p=17. While we obtained the perfect solutions more easily for less than 16 patterns, we were not able to observe the success for 17 patterns after 30 runs with different random number seed. The network in the figure which had learned 17 patterns evolved to finally store 14 patterns as fixed points.

6.2 Start with Random Synaptic Weights

If we further increase the number of patterns p, the synaptic weights J_{ij}'s become random variables with the exception of symmetry $J_{ij} = J_{ji}$ [9]. This is equivalent to the Sherrington-Kirkpatrick Model [21], and includes a full of spin-glass attractors. In 1986, Toulouse, Dehaene & Changeux [22] and Nadal, Touluse, Changeux & Dahaene [23] modified purely random synaptic weights instead of Hebbian weights by adding synapses small perturbations η_{ij}'s which were determined using Hebbian rule.

$$w_{ij} = R_{ij} + \eta_{ij}.$$

This is the same as Sompolinsky's modification mentioned in the first section. These methods succeeded in strengthening the valleys near patterns to be stored and eliminating many other attractors. These are more or less close to *the Lamarckian evolution of random synaptic weights* addressed by Imada & Araki [17]. In this paper we explore this by pruning connections instead of perturbing them. Namely,

$$w_{ij} = c_{ij} \times R_{ij}.$$

We started with asymmetric random synaptic weights. In this case, any input pattern results in chaotic trajectory. In Figure 3, we show two examples of such trajectories taken from the experiment with $p = 7$ which we describe bellow.

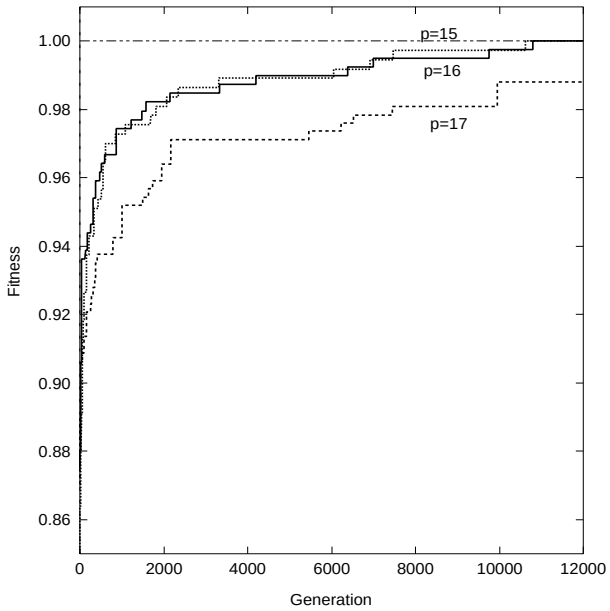

Figure 2: **The best fitness of each generation: starting with Hebbian weights.**

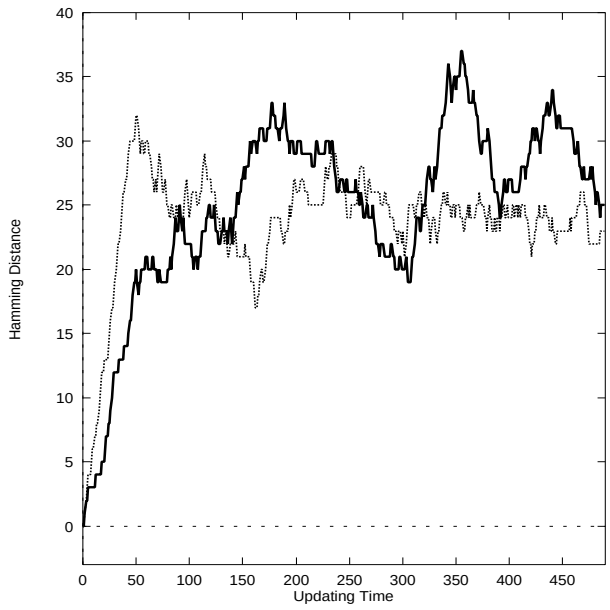

Figure 3: **Temporal evolution of Hamming distance of instantaneous network state from input pattern: starting with random weights: starting with random weights.**

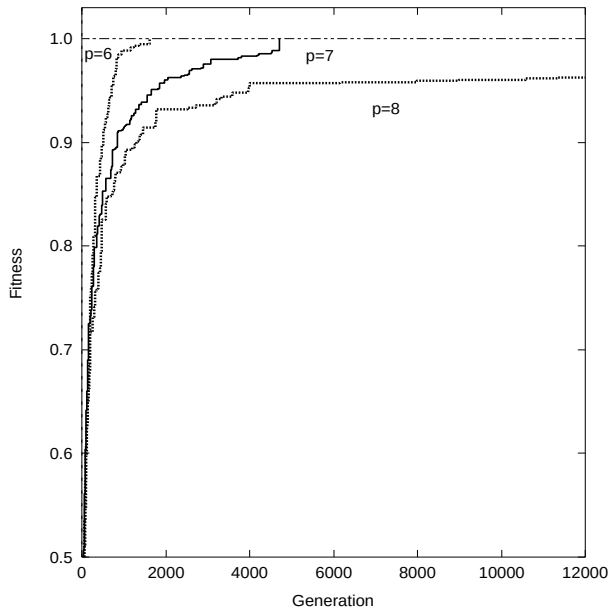

Figure 4: **The best fitness of each generation: starting with random weights.**

The experiment was made with networks of the same size as above (N=49). The goal is to evolve the synaptic weights eventually to store a set of pre-determined random patterns as fixed points. Surprisingly again, the successful networks were emerged which store a maximum of 7 pre-determined patterns perfectly as fixed points. In Figure 4 we show the results for $p = 7$. A random matrix converged to the perfect solution at generation 4719.

6.3 The Effect of Asymmetry and Dilution

6.3.1 Asymmetry of the synaptic weight matrix

Hertz, Grinstein & Solla [24] suggested in their analysis using Ising model that asymmetry of synaptic weights destabilizes spin-glass attractors. Parisi [25] also suggested that asymmetry convert the spin-glass attractors into chaotic trajectories. These analyses were made on the condition that the number of memorized patterns is less than the capacity, and the objective was to show the robustness of the network for the synaptic noises. Hence we may conjecture that the above successes in the case of starting with over-loaded Hebbian synaptic weights is due to the asymmetry introduced by the genetic algorithm.

To see this, we investigated the time evolution of degree of symmetry of weight matrices. The degree of symmetry is defined by

$$\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}w_{ji} \cdot \left(\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^2\right)^{-1},$$

following after Krauth, Nadal & Mezard [26]. We show the results of both experiments above in Figure 5. At the beginning the Hebbian matrix, however over-loaded it may be, is totally symmetric. As evolution proceeds, the degree of symmetry decreases and asymptotically approaches to the value around 0.7 and retrieval states emerge. Our conjecture is that the asymmetry destabilized the many spin-glass attractors which existed at the beginning. However the role of asymmetry of the synaptic weights is still an open question.

On the other hand, experiments which start with random synapses show that some degree of symmetry is needed in order for the network to function as associative memory. However very little is known about how this symmetry influences the network with full of chaotic attractors.

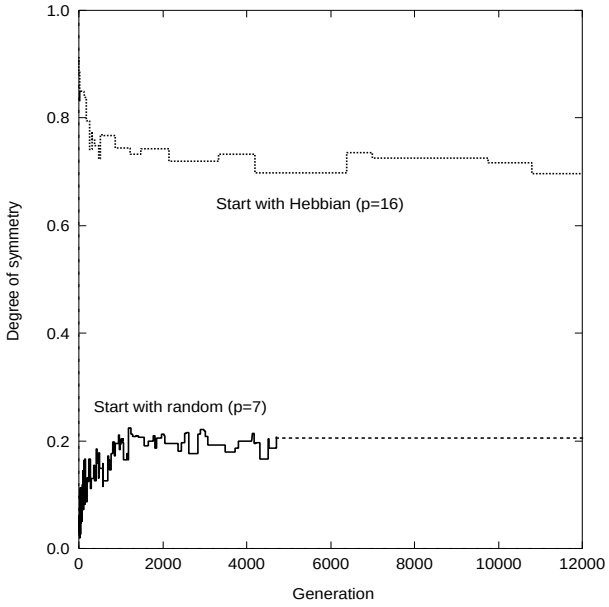

Figure 5: **Evolution of degree of symmetry.**

6.3.2 Dilution

To see the effect of dilution of synaptic weights of our experiment, we investigated the dilution rate as generation proceeds. The results for both experiments are shown in Figure 6. Here, we must note that when even number of patterns are learned by Hebbian rule, some synaptic weights are equal to zero, while in the case of odd number patterns Hebbian rule does not make any zero in weight matrix. Therfore, we also show a Hebbian example for 15 patterns as well as 16 patterns. We can see in the figure, the difference between 15 patterns and 16 patterns is only the initial difference. The behaviors in two cases are quite different however. When algorithm start with Hebbian synapses, the zero density does not increase drastically. On the other hand, when the run start with random synapses, it increases rapidly at the beginning and approach to the value of around 0.17 asymptotically. It should be interesting to note that the value is independent on the initial percentage of zero in chromosomes.

Derrida, Gardner and Zippelius [6] showed that the dynamical equation of state transition for a randomly connected network are exactly solvable if connections are extremely sparce. However it is much more difficult to analyze our experimental results, because of operations (e.g. crossover) in the genetic algorithm. In addition, we not only prune some connections but reverse excitatory/inhibitory synapses with using -1 in our chromosomes. We can see that this reverse of excitatory/inhibitory synapses plays an important role to give a network a function of associative memory. However we have not known the reason of these behaviors well so far.

6.4 Basin of Attraction

To say a network an associative memory system, the tolerance for certain amount of input noises should be shown. That is, each memorized pattern must have a certain domain of basin of attraction.

We can observe the basin size of a network as follows. One of the memorized patterns are chosen and flipped d bits at random. Then the pattern is given to the network, and temporal average of *overlaps* of the network state visited by the dynamics is calculated. This is repeated 800 times for each incremented d. In Figure 7, averaged overlap over the repetition is plotted against noisy-bit d, together with the same experiment with Gardner's weight matrix [13] for the same set of memorized patterns. Although the size of basin of attraction is much smaller than that of Gardner's optimized matrix, it still shows some

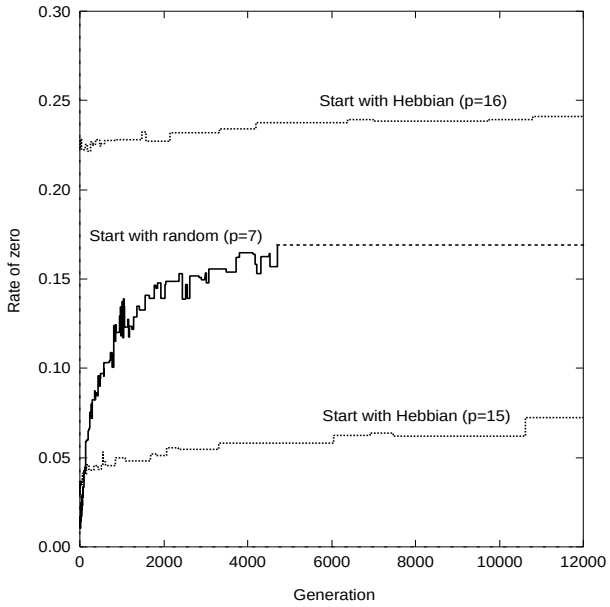

Figure 6: **Evolution of zero density.**

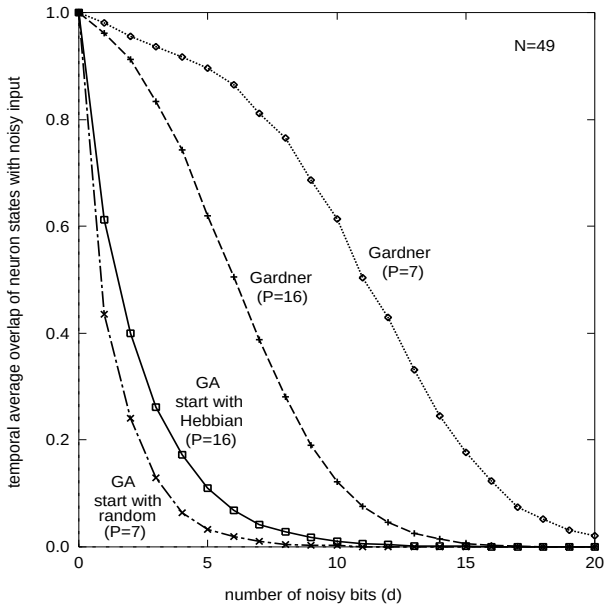

Figure 7: **Similarity of updated states with noisy input**

tolerance for noises. We conjecture that the reason of this small basin size is also due to the attractors introduced in the vicinity of fixed point attractors

7 Conclusions

We have described an application of a genetic algorithm to Hopfield model of associative memory. We adaptively prune some Hebbian synaptic connections. This is essentially the same as Derrida et al.'s model. with difference being that we use an *over-saturated* networks and choose synapses *adaptively*, while Sompolinsky used an *unsaturated* networks and chose synapses *randomly*. The simulations in this paper were made with 49 neurons. This size of Hebb-rule associative memory would store at most 7 patterns as fixed points. We succeeded in evolving a network which had learned 16 patterns by Hebbian rule to store eventually all these patterns as fixed points. Our conjecture is that this success is because asymmetry eliminated the many spin-glass attractors which had dominated at the beginning. We also succeeded in evolving a network starting with random synaptic weights to store 7 pre-determined patterns as fixed points. This is quite an amazing result since we did not use any learning method except for evolution. The genetic algorithm seems to be able to find very small valleys around the states to be memorized and strengthening them while eliminating other attractors, which is made *adaptively* and have never explored before.

Then, where have those spin-glass attractors gone after the genetic algorithm run? As we have mentioned above, the dynamics of retrieving a memorized pattern includes such as (1) fixed point attractor; (2) retrieval attractor (e.g., $m > 9.8$); (3) spin-glass-attractor; (4) chaotic trajectory; and (5) trajectories which take exceedingly long time to reach attractors. This question about the fate of these attractors will provide an interesting topic for near future work, and many other tasks are still waiting for exploration.

We believe these researches shed new light on the analysis of the Hopfield model of associative memory from the different aspect.

References

[1] J. Komlós, and R. Paturi (1988) *Convergence Results in an Associative Memory Model.* Neural Networks **1**. pp239–250.

[2] J. J. Hopfield (1982) *Neural Networks and Physical Systems with Emergent Collective Computational Abilities.* Proceedings of the National Academy of Sciences, USA. **79**. pp2554–2558.

[3] Holland, J., *Adaptation in Natural and Artificial Systems.* The University of Michigan Press (1975)

[4] Goldberg, D., *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley (1989)

[5] H. Sompolinsky (1986) *Neural Network with Non-linear Synapses and Static Noise.* Phys. Rev. **A34** pp2571-2574.

[6] B. Derrida, E,Gardner, A. Zippelius (1987) *An Exactly solvable Asymmetric Neural Network Model* Europhysics Letter 4(2) pp167-173.

[7] D. O. Hebb (1949) *The Organization of Behavior.* Wiley.

[8] D. J. Amit, H. Gutfreund, and H. Sompolinsky (1985) *Statistical Mechanics of Neural Networks near Saturation* Annals. of Physics 173, pp30-67.

[9] D. J. Amit, H. Gutfreund, and H. Sompolinsky (1985) *Storing Infinite Number of Patterns in a Spin-glass Model of Neural Networks.* Physical Review Letters **55**. pp1530-1533.

[10] R. J. McEliced, E. C. Posner, E. R. Rodemick, and S. S. Venkatesh (1987) *The Capacity of the Hopfield Associative Memory.* IEEE Trans. Information Theory **IT-33**. pp461-482.

[11] T. Kohonen, and M. Ruohonen (1973) *Representation of Associated Data by Matrix Operators.* IEEE Trans. Computers **C-22(7)**. pp701–702.

[12] G. Pancha, S. S. Venkatesh (1993) *Feature and Memory-Selective Error correction in Neural Associative Memory.* in M. H. Hassoun eds. Associative Neural Memories: Theory and Implementation. Oxford University Press. pp225-238.

[13] E. Gardner (1988) *The Phase Space of Interactions in Neural Network Models.* J. Phys A: Math Gen. **21**. pp257–270.

[14] T. D. Chiueh, and R. M. Goodman (1991) *"Recurrent Correlation Associative Memories."* IEEE Trans. Neural Networks 2(2), pp275–284.

[15] J. D. Schaffer, D. Whitley, and L. J. Eshelman (1992) *Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art.* Proceedings of the Workshop on Combinations of Genetic Algorithms and Neural Networks. pp1–37.

[16] X. Yao (1993) *A Review of Evolutionary Artificial Neural Networks.* International Journal of Intelligent Systems **8**. pp539–567.

[17] A. Imada, and K. Araki (1996) *Lamarckian Evolution of Associative Memory* Proceedings of IEEE International Conference on Evolutionary Computation pp676–680.

[18] I. Rechenberg (1965) *"Cybernetic Solution path of an Experimental Problem."* Royal Aircraft Establishment, Library Translation, 1122 (Hants UK).

[19] H. Mühlenbein, and D. Schlierkamp-Voosen (1995) *Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization.* Evolutionary Computation **1** (1996) pp25–49.

[20] G. Syswerda (1989) *Uniform Crossover in Genetic Algorithms.* Proceedings of the 3rd International Conference on Genetic Algorithms, pp2–9.

[21] S. Kirkpatrick, and D. Sherrington (1978) *Infinite-ranged Models of Spinglasses.* Physical Review B17(11) pp4384-4403.

[22] G. Toulouse, S. Dehaene, J. P. Changeux (1986) *Spinglass Model of Learning by Selection.* Proceedings National Academy of Science(USA)83 p1695–1698.

[23] J. P. Nadal, G. Toulouse, J. P. Changeux, and S. Dahaene (1986) *Networks of Formal Neurons and Memory Palimpsests.* Europhysics Letter 1(10) pp535–542.

[24] J. A. Hertz, G. Grinstein, and S. A. Solla (1987) *Irreversible Spin Glasses an Neural Networks.* in L. N. van Hemmen and I. Morgenstern eds. Heidelberg Colloquium on Glassy Dynamics. Lecture Notes in Physics No.275 Springer-Verlag, pp538–546.

[25] G. Parisi (1986) *Asymmetric Neural Networks and the Process of Learning.* J. Phys. **A19** ppL675–L680.

[26] W. Krauth, J.-P. Nadal, and M.Mezard (1988) *The Roles of Stability and Symmetry in the Dynamics of Neural Networks.* J. Phys. A: Math. Gen. **21** pp2995–3011.