Random Perturbations to Hebbian Synapses of Associative Memory using a Genetic Algorithm

Akira Imada[1] and Keijiro Araki[2]

[1] 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

[2] Department of Computer Science and Computer Engineering
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 model of associative memory. Previously we reported that a genetic algorithm using *ternary* chromosomes evolves the Hebb-rule associative memory to enhance its storage capacity by pruning some connections. This paper describes a genetic algorithm using *real-encoded* chromosomes which successfully evolves over-loaded Hebbian synaptic weights to function as an associative memory. The goal of this study is to shed new light on the analysis of the Hopfield model, which also enables us to use the model as more challenging test suite for evolutionary computations.

1 Introduction

In the field of evolutionary computations, a lot of complicated functions have been proposed as test suites for comparing and evaluating the effectiveness of various evolutionary computations [1, 2, 3, 4, 5, 6, 7, 8]. However as Mülenbein [9] pointed out:

> *But are such problems typical applications? We have not encountered such a problem.*

these functions are far from real problems. We know the number of their local optima, values, locations etc. This makes us think of using the Hopfield model of associative memory as a test function of evolutionary computations. The model per se is very simple, and have favorable properties as a test function of real-valued parameter optimization problems. Firstly, as we will describe later in detail, for a fixed set of patterns to be stored, there exist almost infinite number of combinations of synaptic weights which give the network a function of associative memory (multi modality). Secondly, the model has a global optimum in the sense of Gardner [10]. Finally, there is a trade off between the storage capacity and error correcting capability (multi-objectivity). However, we have no knowledge

such as the exact distribution of these local optima. In fact, many problems concerning the model are still open, despite a lot of discussions have been made since Hopfield proposed the model. For example, Sompolinsky [11] studied an effect of small perturbations of Hebbian synaptic weights. Namely, he modified the weights by

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

where J_{ij}'s are Hebbian synaptic weights and η_{ij}'s are small normal random variables. His analysis was only under the condition that the number of stored pattern was less than the storage capacity, and modification were made *randomly*. What Sompolinsky showed is the robustness of the Hopfield model against small synaptic noises. But what will happen if we make the experiment for more patterns than the capacity? In this paper, we experiment it with patterns more than the saturation level, and we modify synaptic weights *adaptively* using a genetic algorithm. The genetic algorithm uses a continuous representation instead of traditional binary representation. We believe the study shed new light to the analysis of the model, as well as give a more challenging test suite to evolutionary computations.

The remainder of this paper is organized as follows. The other evolutionary approaches to the Hopfield model will be described in the following section. Since we only modify synaptic weights strictly maintaining the framework of Hopfield model, the model is described more fully in the third section. Our genetic algorithm is not 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 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 [12], [13], 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.

Previously, we successfully evolved both random weight matrices and over-loaded Hebbian weight matrices to function as associative memories, using ternary chromosomes [14, 15]. The chromosomes of those implementations were made up mostly 1 but a few components were 0 or -1. Optimization was made both by pruning some synaptic connections and by balancing the number of excitatory/inhibitory synapses with these 0 and -1, respectively. That is, the modified synaptic weights were obtained as follows

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

or

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

where c_{ij}'s are components of the chromosome and R_{ij}'s are Random synaptic weights.

The simulations were made with 49 neurons as in this paper. The random matrix evolved to store eventually 7 patterns as fixed points. And starting with overloaded Hebbian matrix which learned 16 patterns it was evolved to stabilize all these 16 patterns as fixed points. Note that this size of Hebbian matrix would store at most 8 patterns.

3 Hoppfield Model of Associative Memory

Associative memory is a dynamical system which has a number of stable states with a domain of attraction around them [16]. Applying a vector $\boldsymbol{x}_0$ as the initial state of the dynamical system results in the state trajectory $\{\boldsymbol{x}_t\}_{t\geq 0}$. If we start the system with $\boldsymbol{x}_0 = \boldsymbol{u} + \boldsymbol{\delta u}$ in the vicinity of some fixed point $\boldsymbol{u}$, and the state trajectory converges to $\boldsymbol{u}$, i.e., $\lim_{t\to\infty} \boldsymbol{x}_t = \boldsymbol{u}$, then we can regard the system as an associative memory.

In 1982, Hopfield proposed his fully connected neural network model of associative memory [17]. The model consists of N neurons and N^2 synapses. Each neuron can be in one of two states ± 1, and p bipolar patterns $\xi^\nu = (\xi_1^\nu, \cdots, \xi_N^\nu)$ $(\nu = 1, \cdots, p)$ are memorized as equilibrium states. He employed a discrete-time, asynchronous update scheme. Namely, at most one neuron updates its state at a time, according to

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

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.

The behaviors of the collective states of the individual neurons are characterized by the synaptic weights. When these synaptic weights are determined appropriately, networks store some number of patterns as fixed points. Hopfield specified w_{ij}'s by the Hebbian rule [18], 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 certain time steps. 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 et al. [19] by using *replica* method. They showed that the storage capacity is $p = 0.14N$. In 1987, McEliece et al. [20] 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 et al. [21] and *spectral algorithm* by Pancha et al. [22] are the examples, and the capacity is enlarged to $p = 0.5N$ and $p = N$, respectively. Both are extensions of the Hebbian learning.

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[10] studied the optimal storage capacity as a function of 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.

4 GA Implementation

In this simulation, a set of random bipolar patterns ξ^ν $(\nu = 1, 2, \cdots, p)$ are produced before each run. The goal of the evolutionary programming 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. 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 real-valued vector. Its component is called as allele. We denote the chromosome as

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

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

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

where w_{ij}^k is ij-component of the copied matrix, η_{ij}^k is $(iN + j)$-th allele of the chromosome, and the superscript k denotes the individual number in the population. The chromosomes are modified through crossover and mutation operation. The fitness values of the corresponding phenotype are evaluated. According to the fitness values, individuals of the next generation are selected, using a $(\mu+\lambda)$-strategy in ES 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: In the first generation, all chromosomes are initialized so that each allele value is Gaussian random variables with mean 0 and standard deviation σ.

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. This $m^\nu(t)$ is referred to as *overlap*. 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. We evaluate the fitness value of each network by further averaging the $\langle m^\mu \rangle$ 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 fitness less than 1 includes many other cases.

Selection: We use the truncation-selection [9], i.e., two parent chromosomes are chosen *randomly* from the best T% of the population.

Recombination: Discrete Recombination [9] which is similar to *uniform crossover* [23] in binary chromosomes, operates on alleles of the 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 is made by randomly picking up an allele with probability p_m, and replacing it with uniformly random number selected from $[-\delta, \delta]$.

5 Experiment

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 standard deviation σ in initializing allele values in the first generation is set to 0.01. The mutation probability p_m and the mutation range δ are set to 0.05 and 0.1, respectively. 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

In this experiment, we start the genetic algorithm with a Hebbian weight matrix which learned a set of patterns more than capacity. The networks of

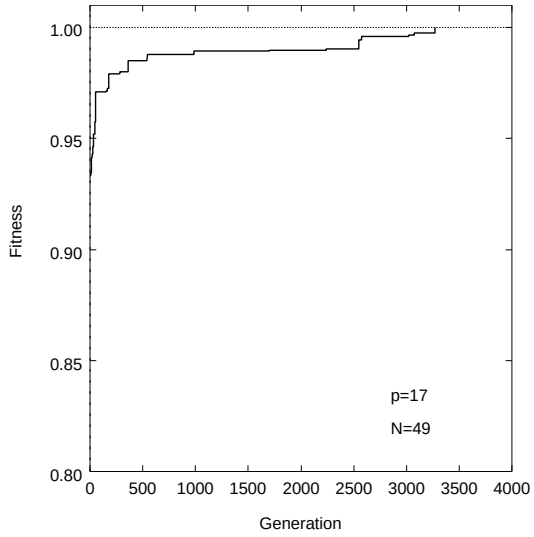

Fig. 1. Fitness vs. generation

this type were extensively investigated by Amit et al. [19, 24]. As the number of patterns increases, the correlation of these patterns can not be neglected. when the number exceeds the capacity, some memory states are shifted to their neighborhood, and spurious attractors around memory states merge into spin-glas attractors.

Let us take an example. We made an experiment for 17 patterns to be memorized on a network with 49 neurons. Since the storage capacity of the Hebbian synaptic weights of this size would be around 8 patterns, these 17 patterns are a little more than twice of the capacity. Each input of memorized patterns results in the convergence to the stable attractors though, most of them are not memorized states. These attractors can be studied by the temporal evolution of Hamming distance of instantaneous network state from the corresponding memorized pattern. We observed that the Hamming distances of the 17 stable states from their corresponding input were 0, 0, 0, 0, 1, 1, 1, 6, 7, 7, 8, 10, 10, 14, 15, 18 and 19. 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 an ancestor matrix for our genetic algorithm as described above. Surprisingly we obtained a evolved matrix which store all these 17 patterns as fixed points perfectly after 3273 generations. We show the fitness evolution in Figure 1.

While we obtained the perfect solutions more easily for less than 16 patterns (not

in the Figure), we have not observed the same success for 18 patterns among 30 runs with different random number seed.

As Hertz et al. [25] and Parisi [26] suggested in their analysis using Ising model, we conjectured that the above success is due to the destabilization of the spin-glass attractors by the asymmetry of synaptic weights introduced by the genetic algorithm.

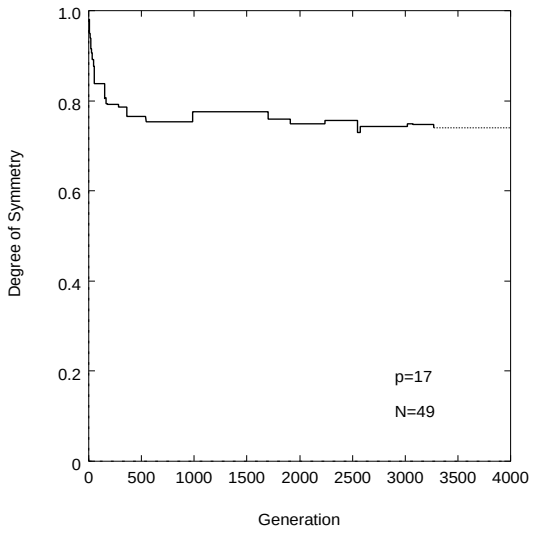

Fig. 2. Degree of symmetry of the matrix which evolved to perfect solution.

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 et al. [27]. We show the results of both experiments in Figure 2. 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 0.8 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. To say a network an associative memory system, the tolerance for certain amount of noises should be shown. That is, each memorized pattern must have a certain domain of basin of attraction.

We observed the basin size of the network of the above experiment as follows. One of the memorized patterns are chosen and flipped d bits at random. 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 3, the averaged overlap over repetition are plotted against noisy-bit d, together with the same experiment with Gardner's weight matrix 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 tolerance for noises.

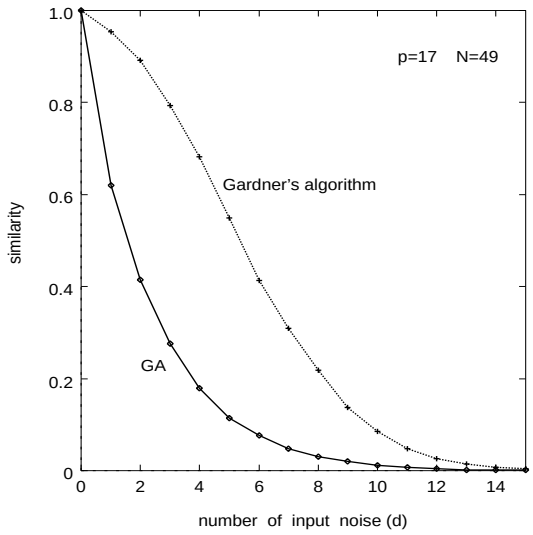

Fig. 3. Similarity of updated states with noisy input

7 Conclusion

We have described an application of genetic algorithms to Hopfield model of associative memory. We give small perturbations to Hebbian synaptic weights. This is essentially the same as Sompolinsky's model, with difference being that we use an over-saturated network and choose synapses *adaptively*, while Sompolinsky

used an unsaturated network 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 8 patterns as fixed points. We succeeded in evolving a network which learned 17 patterns by Hebbian rule to store eventually all these patterns as fixed points.

Although our investigations have not been conclusive so far, we believe this study shed new light on the analysis of Hopfield model from new aspect, and give new test suite for evolutionary computations.

Acknowledgments

We thank Peter Davis at Advanced Telecommunication Research Institute (ATR) for providing us great insight into the dynamics of Hopfield neural networks.

References

1. K. DeJong (1975) *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Ph.D thesis, University of Michigan.
2. D. Ackley (1987) *A Connectionist Machine for Genetic Hillclimbing.* Boston: Kluwer.
3. K. Deb (1989) *Genetic Algorithms in Multimodal Function Optimization.* Master's thesis. The University of Alabama.
4. H. Mühlenbein, M. Schomish, and J. Born (1991) *The Parallel Genetic Algorithm as Function Optimizer.* Parallel Computing **17** pp619–632.
5. J. D. Schaffer, B. A. Caruana, L. J. Eshlman, and R. Das (1991) *A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization.* Proceedings of the 3rd International Conference on Genetic Algorithms. pp51–60.
6. D. Whitley, K. Mathias, S. Rana, and J. Dzubera (1995) *Building Better Test Functions.* Proceedings of the 6th International Conference on Genetic Algorithms. pp239-247.
7. S. M. Mahfoud (1995) *A. Comparison of Parallel and Sequential Niching Methods.* Proceedings of the 6th International Conference on Genetic Algorithms. pp239-247.
8. D. Schlierkamp-Voosen, and H. Mühlenbein (1996) *Adaptation of Population Sizes by Competing Subpopulation.* Proceedings of IEEE International Conference on Evolutionary Computation. pp330-335.
9. H. Mühlenbein, and D. Schlierkamp-Voosen (1995) *Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization.* Evolutionary Computation **1** (1996) pp25–49.
10. E. Gardner (1988) *The Phase Space of Interactions in Neural Network Models.* J. Phys A: Math Gen. **21**. pp257–270.
11. H. Sompolinsky (1986) *Neural Network with Non-linear Synapses and Static Noise.* Phys. Rev. **A34** pp2571-2574.
12. 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.

13. X. Yao (1993) *A Review of Evolutionary Artificial Neural Networks.* International Journal of Intelligent Systems **8**. pp539–567.
14. A. Imada, and K. Araki (1997) *Evolved Asymmetry and Dilution of Random Synaptic Weights in Hopfield Network Turn a Spin-glass Phase into Associative Memory.* The 2nd International Conference on Computational Intelligence and Neuroscience (to appear)
15. A. Imada, and K. Araki (1995) *Genetic Algorithm Enlarges the Capacity of Associative Memory.* Proceedings of the 6th International Conference on Genetic Algorithms. pp413–420.
16. J. Komlós, and R. Paturi (1988) *Convergence Results in an Associative Memory Model.* Neural Networks **1**. pp239–250.
17. 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.
18. D. O. Hebb (1949) *The Organization of Behavior.* Wiley.
19. D. J. Amit, H. Gutfreund, and H. Sompolinsky (1985) *Statistical Mechanics of Neural Networks near Saturation* Annals. of Physics 173, pp30-67.
20. 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.
21. T. Kohonen, and M. Ruohonen (1973) *Representation of Associated Data by Matrix Operators.* IEEE Trans. Computers **C-22(7)**. pp701–702.
22. 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.
23. G. Syswerda (1989) *Uniform Crossover in Genetic Algorithms.* Proceedings of the 3rd International Conference on Genetic Algorithms, 2.
24. 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.
25. 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.
26. G. Parisi (1986) *Asymmetric Neural Networks and the Process of Learning.* J. Phys. **A19** ppL675–L680.
27. 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.

This article was processed using the LaTeX macro package with LLNCS style