Evolution of Associative Memory using Diploid Chromosomes

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 genetic algorithms to Hopfield model of associative memory. Previously, we successfully evolved random synaptic weights with using *haploid* chromosomes to create attractors at the exact positions of a set of given patterns. We also reported that the genetic algorithm can enlarge the storage capacity of the Hebb rule associative memory. In this paper we present that the Hebb rule associative memory can also be evolved using *diploid* chromosomes.

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, it will converge to the stable state. In 1982, Hopfield [2] proposed a fully connected neural network model of associative memory in which we can store information by distributing it among neurons, and we can recall it from its noisy and/or partial input.

The dynamical behavior of the neuron states strongly depends on synaptic weights between neurons. The synaptic weight from neuron j to neuron i is denoted as w_{ij}. Hopfield used the Hebbian rule [3] to prescribe them. The Hebbian synaptic weights are conventionally denoted as J_{ij}. So far many researchers have modified the Hebbian synaptic weights, and observed the behaviors. For example, Derrida et al. [4] pruned some of the Hebbian synaptic weights, as

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

where $c_{ij} \in \{0, 1\}$. Here we explore it based on this scheme by genetic algorithms.

Previously, we successfully evolved random synaptic weights using *haploid* chromosomes to create attractors at the exact positions of a set of given patterns [5].

We also reported that the genetic algorithm enlarges the storage capacity of the Hebb rule associative memory [6].

In this paper, we use *diploid* chromosomes instead. The motivation to use the diploidy evolution is its pressure to be homozygous. Hillis [7] suggested that the number of heterozygous site decreases as generation proceeds because the descendant of heterozygotes tend to miss recessive lethal. Mitchell [8], in her introductory book of genetic algorithm, paraphrased this reason as follows. If a crucial comparison is at a heterozygous position in its chromosome, then it can be lost under a crossover, whereas crucial comparisons at homozygous position cannot be lost under crossover.

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 with maintaining the framework of Hopfield model, the model is described more fully in the third section. Our genetic algorithm implementation is addressed in the fourth section. After presenting experimental results and discussion, the final section describe the concluding remarks.

2 Related Works

In the classical analysis, Sompolinsky [9] also modifies Hebbian synaptic weights by adding small perturbations on them. Namely, the modified synapses are:

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

where η_{ij} are small Gaussian variables of mean 0.

We implemented this scheme by evolutionary algorithms, where continuous genes are used. The difference is that Sompolinsky modified synaptic weights to show the robustness of the Hopfield model for synaptic noise, while we modify them to enlarge the capacity or create memory. In this implementation, we take the Hopfield model as a continuous parameter optimization problem. The perturbations are given on the synaptic weights in two different ways. One is by chromosome, and the other is by mutation. The former exploits a promising region, that is, search points are restricted within certain domain around one point. On the other hand, the latter explores new regions, namely, search points may wander all over search space. It is beyond the scope of this paper to describe this more precisely. Please refer to [10, 11, 12].

3 Hopfield Model of Associative Memory

In 1982, Hopfield proposed a neural network model of associative memory [2]. 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

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

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 [3], 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. [13] by using *replica* method. They showed that the storage capacity is $p = 0.14N$. In 1987, McEliece et al. [14] 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. [15] and *spectral algorithm* by Personnaz et al. [16] are the examples, and the capacity is enlarged to $p = N$. 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 [17] 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 p random bipolar patterns ξ^ν $(\nu = 1, 2, \cdots, p)$ are produced before each run. Usually we produce patterns exceeding the storage

capacity of the Hebb rule associative memory. Then a network with N neurons learns the patterns using the Hebbian rule. Since number of patterns exceeds the capacity, the network does not function as associative memory due to the full of spurious attractors. The goal of the genetic algorithm is to store all the p patterns as fixed points.

Previously we used haploid chromosomes to solve the same problems. Here, we use diploid chromosomes. Chromosomes are comprized of mostly 1 but includes a few of 0 and -1. A pair of alleles of the same position in two chromosomes of a individual, represents the two components of the original matrix in the symmetric position. Thus, each pair modifies components of the original weight matrix, in the position of upper right triangle and lower left triangle, respectively. To be more specific, assume the diploid chromosome is denoted as

$$\{c^1_1, c^1_2, \cdots, c^1_{N(N-1)/2}\}$$

and

$$\{c^2_1, c^2_2, \cdots, c^2_{N(N-1)/2}\}.$$

Then, for $i = 1, \cdots, N-1$, and $j = i+1, \cdots, N$

$$w'_{ij} = w_{ij} \cdot c^1_k,$$

and for $j = 1, \cdots, N-1$, and $i = j+1, \cdots, N$

$$w'_{ji} = w_{ji} \cdot c^2_k,$$

where

$$k = \frac{(2N-i)(i+1)}{2} - 2N + j.$$

The original Hebbian weight matrix is symmetric. Thus, the phenotype will be more or less asymmetric depending on the heterozygosity of the genotype. If the two alleles of the same position in the diploid chromosome are identical (i.e., homozygous), then corresponding components of the phenotype remains symmetric. If they are different (i.e., heterozygous), it will introduce asymmetry in the weight matrix. The more homozygous positions, the more symmetric. A completely homozygous genotype would produce a symmetric phenotype.

Before being mated, uniform crossover is operated on the diploid chromosome pair within each individual, which produces a single *gamete*. The length of the gamete is same as each diploid chromosome ($= n(n-1)/2$). Each of these gametes is paired with another gametes from different parent to form a single diploid offspring. The selection is made according to the fitness value of the parent. That is, for each parent of the best 40% of the population, the other parent is selected randomly from the same best 40% of the population, prohibiting self-mating. This process is repeated until the rest of the best 40% of the population are replaced with the offspring.

Offspring is mutated by cyclically rotating alleles which are selected with the probability p_m.

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

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.

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 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

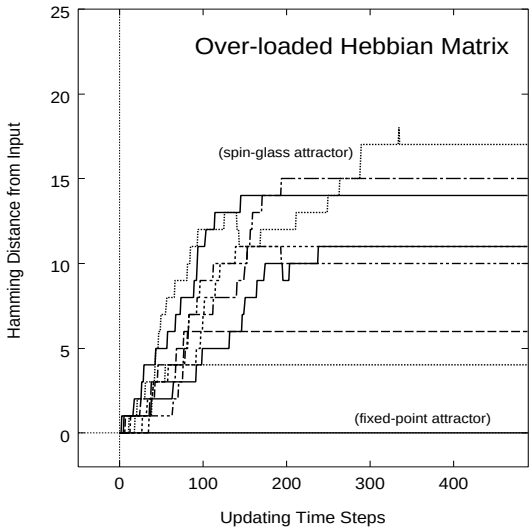

Fig. 1. Trajectories to various attractors.

We started the genetic algorithm with an over-loaded Hebbian matrix. In this case, each input of a given set of patterns will converge to some attractor ranging

from fixed-point attractors to spinglass attractors. These behaviors are visualized by plotting the Hamming distance d_H between the input and instantaneous states of the network against updating time steps. Examples of the trajectories are shown in Figure 1. The goal of the genetic algorithm is to find a solution which stores a given set of patterns as fixed point attractors, where an initial input of the patterns remains unchanged from the beginning.

For a set of 13 patterns, started with the over-loaded matrix used to draw Figure 1, a matrix emerged at generation 4791 which creates attractors at exactly the same position of all the given patterns. In Figure 2, we show the fitness of the best individual, together with the average fitness of the evolution. We have not obtained the emergence of higher capacity yet. Note that the genetic algorithm with haploid chromosomes founnd the solution which stored a maximum of 16 patterns as fixed points [6].

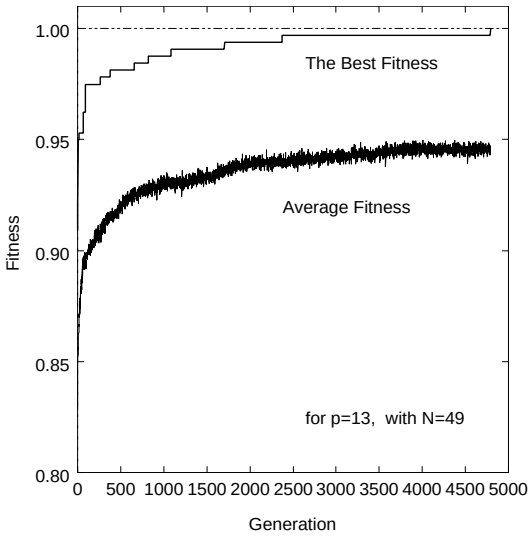

Fig. 2. Fitness vs. generation

Clearly the evolution proceeds with varying individual matrices' degree of symmetry and zero ratio. We investigated the time expansion of these two ratio of the above evolution. The degree-of-symmetry is defined by

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

following after Krauth et al. [18]. The results are shown in Figure 3. The zero ratio does not change so drastically, which is almost the same result as evolution with haploid chromosomes. The degree of symmetry decreased from the value 1 rapidly, and fluctuate around the value between 0.7 to 0.8, which is slightly higher than in haploid case.

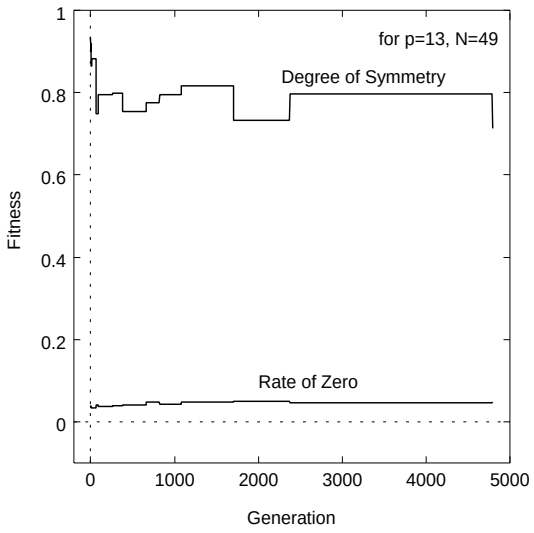

Fig. 3. Degree of symmetry of the matrix which evolved to perfect solution.

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. To see this, 1000 initial state patterns are generated by randomly picking up a memory pattern and flipping d bits. Then they are given to the network and updated the state $2N$ times (possibly it reaches a stable state). The resulting states is then compared with the original memory pattern, and the run is called a success if they match *exactly*. In Figure 4, the number of successes out of 1000 runs of the above experiments is plotted against the number of input errors d, together with the result of Gardner's optimal algorithm [17] for the same set of patterns. We can see that the network resulted from the above experiment still maintain the tolerance of noisy input, although the size of basin of attraction is smaller than the Gardner's one. This is probably because of remaining spurious attractors around the created memories.

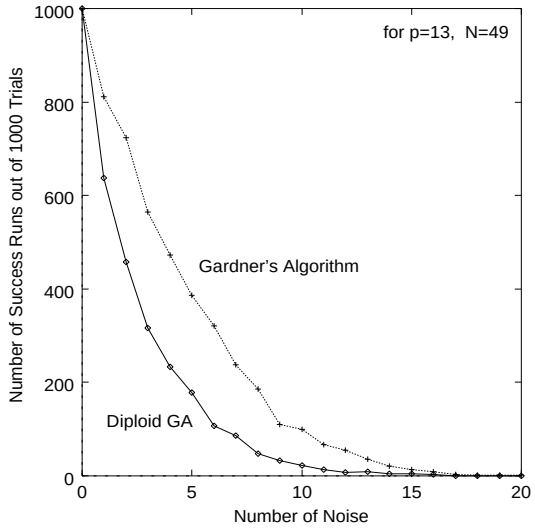

Fig. 4. Similarity of updated states with corresponding memory

7 Conclusion

We have described an application of a genetic algorithm using *diploid* chromosomes to Hopfield model of associative memory. The genetic algorithm implementation using diploid chromosomes is said to be more biologically realistic than those using haploid chromosomes [8]. However we have not obtained the network which has a higher storage capacity. The approach is still very much in its infancy. Notable difference from the haploid evolution has not been observed. As future work, we think of multiple pairs of diploid chromosomes in which each pair have different target, such as storage capacity, basin size, etc.

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. 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. D. O. Hebb (1949) The Organization of Behavior. Wiley.
4. B. Derrida, E,Gardner, A. Zippelius (1987) An Exactly solvable Asymmetric Neural Network Model. Europhysics Letter 4(2) pp167-173.
5. 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)
6. 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.
7. W. D. Hillis (1991) Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure. C. G. Langton, C. Taylar, J. D. Farmer, and S. Rasmussen eds. Artificial Life II. SFI Studies in the Science of Complexity Vol. X, Addison-Wesley
8. M. Mitchell (1996) An Introduction to Genetic Algorithms. MIT Press.
9. H. Sompolinsky (1986) Neural Network with Non-linear Synapses and Static Noise. Phys. Rev. **A34** pp2571-2574.
10. A. Imada, and K. Araki (1996) Lamarckian Evolution of Associative Memory Proceedings of IEEE International Conference on Evolutionary Computation, pp676-680.
11. A. Imada, and K. Araki (1997) Searching Real-Valued Synaptic Weights of Hopfield's Associative Memory using Evolutionary Programming. The 6th Annual Conference on Evolutionary Programming, (to appear)
12. A. Imada, and K. Araki (1997) Application of an Evolution Strategy to the Hopfield Model of Associative Memory IEEE International Conference on Evolutionary Computation, (to appear)
13. D. J. Amit, H. Gutfreund, and H. Sompolinsky (1985) Statistical Mechanics of Neural Networks near Saturation Annals. of Physics 173, pp30-67.
14. 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.
15. T. Kohonen, and M. Ruohonen (1973) Representation of Associated Data by Matrix Operators. IEEE Trans. Computers C-22(7), pp701–702.
16. L. Personnaz, I. Guyon, and G. Dreyfus (1985) Information Storage and Retrival in Spin-Glass like Neural Networks J. Physique Lett. 46 L359-L365
17. E. Gardner (1988) The Phase Space of Interactions in Neural Network Models. J. Phys A: Math Gen. 21, pp257–270.
18. 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