Genetic Algorithm as a Learning Method in Associative Memory

Akira Imada
akira-i@is.aist-nara.ac.jp

Keijiro Araki
araki@is.aist-nara.ac.jp

Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-01 Japan
phone: +81-7437-2-5083

abstract

We simulated an associative memory with mutually connected neural network, and successfully made the connection matrix learn some binary patterns only by means of genetic algorithm. Although the memory capacity is about 12% of number of neurons, the fact that it was made without any learning algorithm like Hebbian rule is very interesting. The structure of connection matrices we obtained is quite different from that of Hopfield network. Therefore we conclude that the genetic algorithm has its own ability to learn some patterns, and we could use it as a new learning method of associative memory.

Our overall goal for this research is two fold. One is to know if we can use genetic algorithm as a more effective learning method than that proposed so far, and another is to understand this learning mechanism of genetic algorithm.

1 Introduction

Associative memory has a limit in capacity depending on the learning rule in memorizing patterns with. Since Kohonen [1] proposed some learning rules in 1977, many ideas have been reported. Hopfield [2], for example, proposed

his model with using Hebbian rule [3]. His associative memory system has the memory capacity of about 15% of number of neurons. And there have been many other approaches to enlarge the capacity. They have achieved it by modifying the weight matrices ([4], for example), with using different dynamic system ([5], for example), and so on.

More recently, genetic algorithms shed light to the research of neural networks (see [6]), especially for the determination of connection weights ([7, 8], for example) and structures ([9, 10], for example). And many successful works have been published ([11, 12], for example). But most of them were for the layered neural networks.

Recetly Fujita et al. [13], however, reported the evolution of associative memory with mutually connected neural networks in which they took into account the physical structure of the neural networks.

We also reported an application of genetic algorithm to mutually connected neural network. As we will describe it later, our genetic algorithm is much simpler than the one they used. In that report, we showed that the memory capacity of connection matrix which learned some patterns by Hebbian rule can be enlarged up to about 30% of number of neurons, i.e. about twice as much as the one in Hopfield network.

In this report, on the other hand, we applied genetic algorithm to the connection matrices generated randomly instead. Neural networks with randomly generated connection matrices can not retrieve any given patterns of course. But after applying genetic algorithm to them, we were able to make the network memorize some patterns. Although the number of patterns is about 80% of that of Hopfield network, it is interesting considering the fact that we do not use any learning rules.

In the following sections, we will attempt to provide the method to evolve the connection matrices and the results of our simulations that show our genetic algorithm could be a new learning method for associative memory.

2 Method

One of the features of our genetic algorithm is that the initial connection matrix is remained fixed over the time of evolution, be it learned matrix or randomly determined one. Our chromosomes have a fixed length of $(number\ of\ neurons)^2$, and their allele values are chosen randomly from either -1, 0, or 1, in which 0 implies to prune the connection and -1 to reverse the role of enforcement or suppression of the weight.

The genetic algorithm we used in this paper is basically the same as previous one [14], so please refer to it for further precise description. However, we have two different points as follows.

One is the initial matrix. In the simulation here, each element of initial matrix is chosen randomly from $\{-1, 1\}$. Hence the matrix does not memorize any patterns at the beginning. And the matrix does not include any zeros, and are not symmetrical at the start of genetic algorithm.

And another difference is the method of evaluating fitness. In the previous simulation with Hopfield network [14], we evaluated fitness at one point of the time of update. If we do it here in the same way, we will obtain the time sequences of fluctuating fitness which have a high value at that point of time.

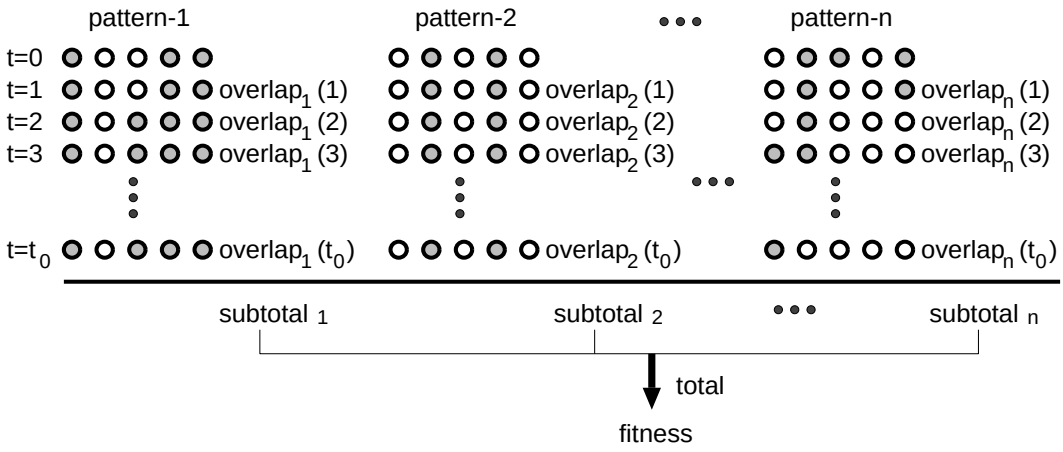

Figure 1: **Overview of our Fitness Calculation**

To solve the problem, we summed up all the overlappings at each time of update not more than certain time t_0. That is, our fitness f is

$$f = \frac{1}{p \cdot (t_0 - 1) \cdot 49} \sum_{\mu=1}^{p} \sum_{t=2}^{t_0} \sum_{j=1}^{49} \xi_j^\mu s_j^\mu(t),$$

where $(\xi_1^\mu, \xi_2^\mu, \xi_3^\mu, \cdots, \xi_{49}^\mu)$ is the μ-th pattern, and $s_i^\mu(t)$ is the corresponding state of update of i-th neuron at time t. We show this in Figure 1. The time t_0 has to be an appropriate value. If it is much shorter than the value, the fitness tends to be high only before that point, and if it is much longer than the value, it becomes difficult to find individuals which can make the patterns to be fixed points. Here t_0 was two times of the number of neurons,

i.e. 98, which we determined after some trial and error tests.

In this situation, if we obtained the matrix of fitness 1.000, it implies that all the initial patterns are fixed points for the obtained weight matrix. For fitness less than 1.000, on the other hand, there are many possibilities. Some patterns could be fixed points but not all. Let us take a fictitious example to give a brief explanation. Both $(1.000 + 1.000 + 1.000 + 0.200)/4$ and $(0.800+0.800+0.8000+0.800)/4$ would show the same fitness value for four initially given patterns. Therefore we must check each cases individually. But any way, high fitness cases have a general tendency that many alleles are overlapped with the initially given pattern.

The other parameters in this paper are same as our another report [14].

3 Results and Analysis

In Figure 2, we showed some examples of fitness vs. generation for 4, 6, 7, 8, and 12 patterns, where t_0 was two times of the number of neurons.

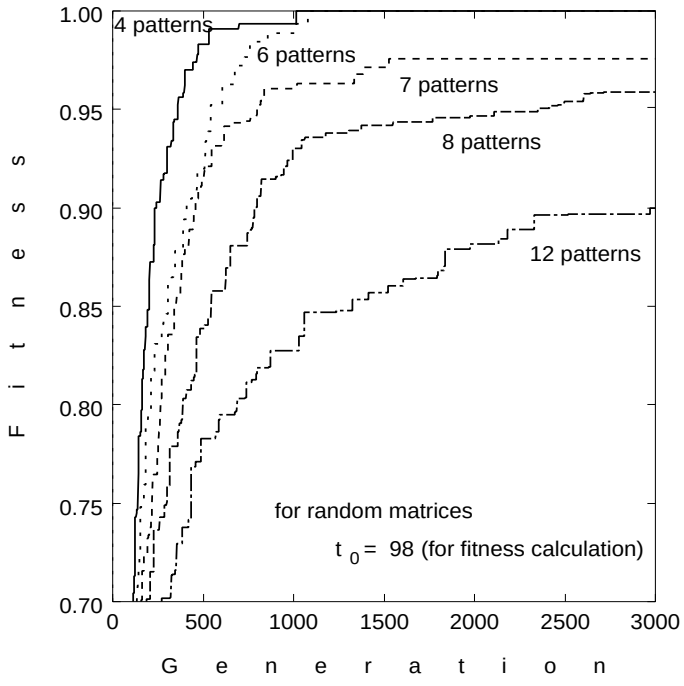

Figure 2: **Fitness vs. Generation**

In Table 1, we showed fitness values and numbers of fixed points obtained

at 3000th generation with increasing number of initial patterns. In the case of 4, 5, and 6 patterns, we obtained the fitness value of 1.000 within 3000 generations, which implies that all the initial patterns are fixed points for the obtained weight matrix as described above. So far, however, we have not obtained the fitness value of 1.000 for the patterns more than six.

Table 1: Maximum Number of Fixed Points

patterns	4	5	6	7	8	12
fitness	1.000	1.000	1.000	0.976	0.959	0.900
fixed-point	4	5	6	3	2	1

In Hopfield network, connection matrix is symmetric and the elements are not zero except for diagonal elements. Our matrices, on the other hand, do not contain any zero including the diagonal elements. In creating individuals, however, some elements could become zero by being multiplied by zero. These zeros are generated at the rate of 0.02 in the beginning, and later the rates vary according to selection pressures during the evolution. And the degree of symmetry of our matrix is almost zero from the beginning.
In Figure 3 and Figure 4, we showed behaviors of degree of symmetry of the connection matrices obtained and rate of zeros in them respectively, which are of the elitest individual in each generation. The definition of degree of symmetry η here is

$$\eta = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}w_{ji} / \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^2.$$

As we can see in those Figures, the behaviors of these parameters are close to those of fitnesses.

As described earlier, -1 in the chromosome reverses the roll of enforcement or suppression of the weight, and 0 prunes the connection. To ascertain the effect of -1 and 0 in the chromosome, we made the following two experiments. One is with the chromosomes chosen from $\{-1, 1\}$ and another is from $\{0, 1\}$ under the same condition of a simulation with the chromosomes chosen from $\{-1, 0, 1\}$ above. We showed the results in Figure 5. Although the chromosome chosen from (a):$\{-1, 0, 1\}$ worked most effectively, as we can see in Figure 5, the chromosome made up of either (b):$\{-1, 1\}$ or (c):$\{0, 1\}$ also worked. The number of fixed points resulted from the experiment(b) was four, whereas one from the experiment(c). The roll of -1 seems to be

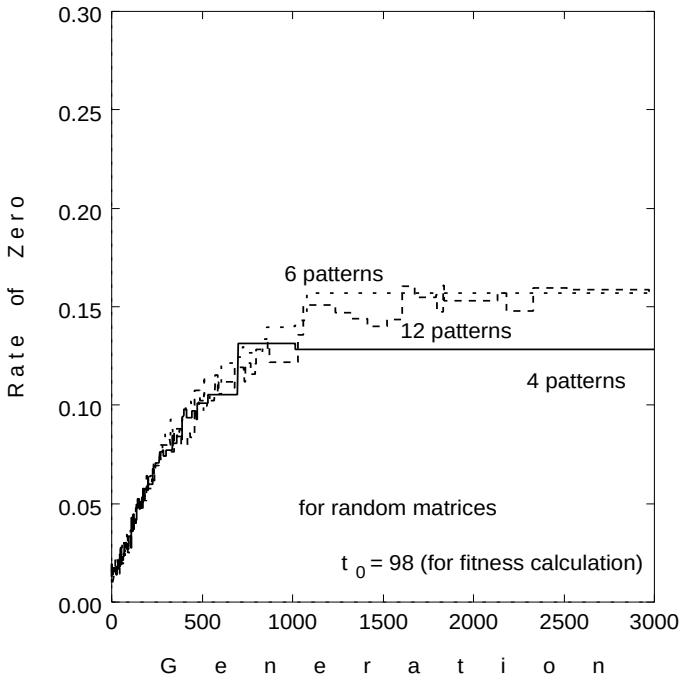

Figure 3: **Rate of Zero**

more important than 0, but we need 0 to obtain six fixed points in the experiment(a).

Consequently we may conjecture that both of the rate of zero and the degree of symmetry have something to do with the learning, though we need further research.

4 Conclusions

We successfully made the random matrices memorize some patterns only by means of genetic algorithm. This time we simulated with the neural networks with 49 nodes, and we obtained the maximum of six fixed points of associative memory. Although the memory capacity of the connection matrix achieved are less than that of Hopfield network, it may lead to useful applications to obtain the higher capacity of associative memory, as well as helping to clarify the process of learning. The matrices we obtained here are quite different from those of Hopfield network. Those are asymmetric and include about 20% of zeros. And the behaviors of both rate of zeros and degree of symmetry are very close to those of fitnesses. Hence, we may conjecture

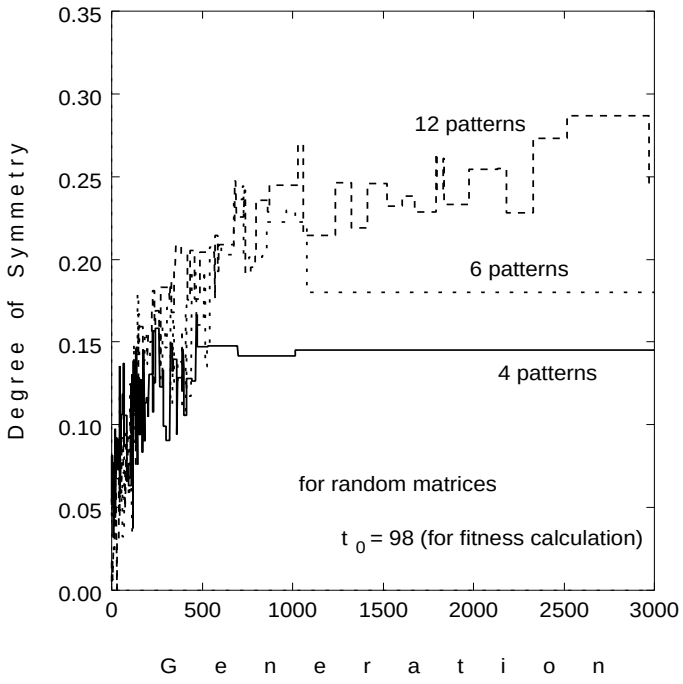

Figure 4: **Degree of Symmetry**

that these parameters play an important roll to make them memorize some patterns.

Considering this with the result that genetic algorithm can enlarge the memory capacity that we presented in another report [14], genetic algorithm could have its own unique learning mechanism which we have not found in the already known learning algorithms.

Also, the fact that we do not use any artificial learning rules in this simulation is very interesting from the biological point of view [15].

5 Future works

We simulated with mutually connected neural networks of 49 neurons. But to make our simulation more reliable, we are exploring these simulations with those of 100 neurons.

In this report, weight values are discrete and transfer function is signus function. But we are introducing continuous values with the sigmoid function to

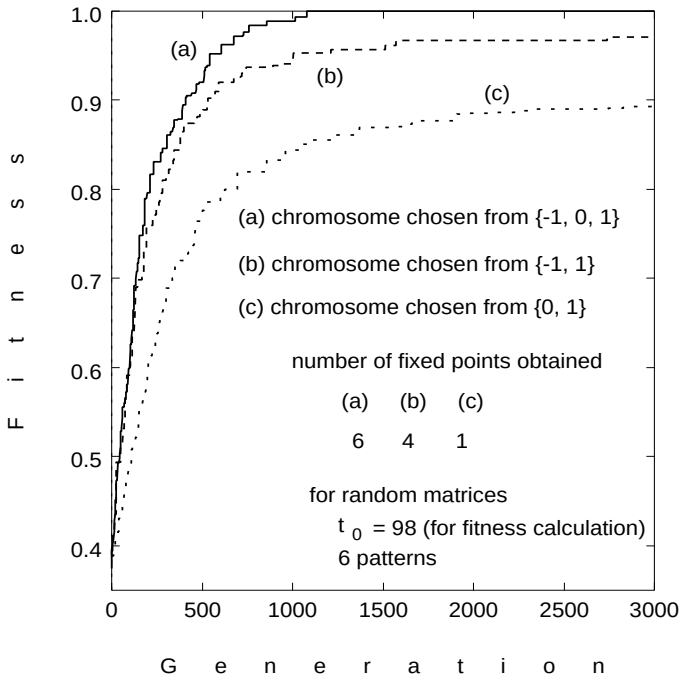

Figure 5: **The Roll of -1 and 0 in the Chromosome**

make our simulation more effective.

In addition, we used either -1, 0 or 1 to modify weight values and they do not change the absolute value of connection weights except for the case of making them zero. So we are thinking of other methods to change the weight values more flexibly through evolution.

Acknowledgments

We wish to thank Peter Davis at ATR (Advanced Telecommunication Research Institute) for giving us his insightful ideas on the evolution of mutually connected neural networks. Thanks also go to Shin Ishii at ATR for his useful comments on associative memory, and other people at NAIST (Nara Institute of Science and Technology) who gave us helpful discussions.

References

[1] Kohonen,T. Associative Memory: A System-Theoretical Approach,

Springer-Verlag, Berlin Heidelberg. (1977)

[2] Hopfield,J.J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proceedings of the National Academy of Sciences,USA,79,2554-2558.(1982)

[3] Hebb,D.O. The Organization of Behavior. Wiley, New York.(1949)

[4] Yanai,H., Sawada,Y., and Yoshizawa,S. Dynamics of an Auto-Associative Neural Network Model with Arbitrary Connectivity and Noise in the Threshold. Network 2 No.3 295-314 (1991)

[5] Morita,M. Associative Memory with Nonmonotone Dynamics. Neural Networks, 6, 115-126 (1993)

[6] Whitley,D., Schaffer,J.D.(Ed.) Combinations of Genetic Algorithms And Neural Networks. IEEE Computer Society Press. (1992)

[7] Whitley,D., and Hanson,T. Optimizing Neural Network using Faster More Accurate Genetic Search. ICGA-89, 391-396 (1989)

[8] Montana,D., and Davis,L. Training Feedforward Neural Networks using Genetic Algorithms. ICGA-89, 762-767 (1989)

[9] Harp,S., Samad,T., and Guha,A. Towards the Genetic Synthesis of Neural Networks. Proc.of ICGA-89, (1989)

[10] Miller,G., Todd,P.,and Hegde,S. Designing Neuran Networks using Genetic Algorithm. Proc.of ICGA-89, (1989)

[11] Kitano,H. Designing Neural Networks using Genetic Algorithm with Graph Generation System. Complex Systems 4. 461-476 (1990)

[12] Gruau,F., and Whitley,W. Adding Learning to the Cellular Development of Neural Networks Evolution and Baldwin Effect. Evolutionary Computation. 213-233 (1993)

[13] Fujita,S., and Nishimura,H. An Evolutionary Approach to Associative Memory in Recurrent Neural Networks. Technical Report at Hyogo University of Education.HUIS-94-3.(1994)

[14] Imada,A., and Araki,K. Genetic Algorithm Enlarges the Capacity of Associative Memory. ICGA-95(submitted)

[15] Habel,D.H. ,and Weisel,T.N. Brain Mechanisms of Vision. Scientific America, Vol.241 No.3, 150-162 (1979)