九州大学学術情報リポジトリ
Kyushu University Institutional Repository

Genetic Algorithm Enlarges the Capacity of Associative Memory

Imada, Akira
Graduate School of Information Science, Nara Institute of Science and Technology

Araki, Keijiro
Graduate School of Information Science, Nara Institute of Science and Technology

https://hdl.handle.net/2324/6342

出版情報：1995
バージョン：
権利関係：

KYUSHU UNIVERSITY

Genetic Algorithm Enlarges the Capacity of Associative Memory

Akira Imada*
Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology
akira-i@is.aist-nara.ac.jp

Keijiro Araki
Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology
araki@is.aist-nara.ac.jp

Abstract

We propose a genetic algorithm for mutually connected neural networks to obtain a higher capacity of associative memory. In Hopfield network as an associative memory system, the memory capacity is at most 15% of the number of neurons. Here we applied our method to the Hopfield network, and obtained the capacity of 33%. We conjectured that this is due to both asymmetry and sparseness of the connection matrix introduced by the genetic algorithm.

1 Introduction

In 1982 Hopfield [1] proposed a neural network model as an associative memory system. He used the Hebbian rule [2] to make connection matrix memorize a set of patterns. He estimated experimentally the memory capacity to be 15% of the number of neurons. Later the capacity was verified more analytically by Amari et al. [3] with neurodynamics and by Amit et al. [4] with spin-glass theory. Since Kohonen [5] proposed several types of connection matrices for associative memory, many researchers have successfully improved the capacity in various ways. For example, Yanai et al. [6] tried that by introducing sparse connectivity, and Morita [7] by using a different dynamic system. Those mechanisms, however, are not well understood yet. Here we tried to improve the memory capacity by using a genetic algorithm.

Since the late 1980's there have been a lot of papers dealing with neural networks by genetic algorithms (see [8]). Most of them, however, were for layered neural networks (see [9, 10] for example), and only a few were for mutually connected neural networks [11]. In this paper, we applied a simple genetic algorithm to mutually connected neural networks, and we have found that it can improve the capacity of associative memory.

*8916-5 Takayama, Ikoma, Nara 630-01 Japan

We presented in another paper that randomly generated connection matrices can learn some patterns using a genetic algorithm without any learning rules [12]. In this paper, on the other hand, we applied a genetic algorithm to the connection matrices which learned some patterns in advance by Hebbian rule.

The overall goal for this research is twofold. One is to know if we can use the genetic algorithm as a more effective learning method than that proposed so far, and the other is to clarify the process of this learning mechanism of the genetic algorithm.

In the following sections we will present the detailed descriptions of our method, some results of our simulations, discussions about our conjecture of how we can enlarge the memory capacity, and the conclusions.

2 Method

We simulated associative memory with mutually connected neural networks with 49 neurons.

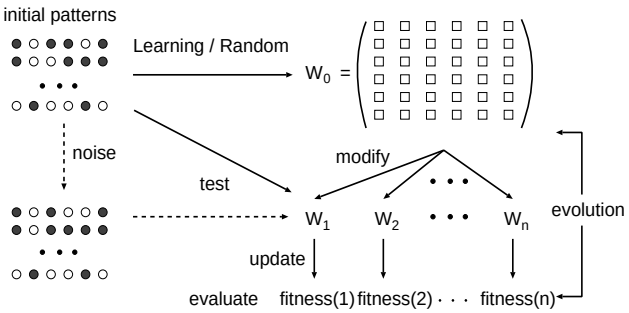

Figure 1: The overview of our genetic algorithm. The original connection matrix W_0 is remained unchanged during evolution. And 256 matrices are produced by slightly modifying this original W_0 in each generation. The modification is done by being multiplied by zero or -1 at a very small rate. Note that this time we gave no noise to the patterns.

2.1 Hebbian learning

The connection weights w_{ij} are determined in advance by Hebbian rule according to the initially given p-patterns, i.e.,

$$w_{ij} = \sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu,$$

where $\xi_i^\mu (i = 1, 2, \cdots, 49)$ is the i-th state of the μ-th pattern $\xi^\mu (\mu = 1, 2, \cdots, p)$, and is either 1 or -1 generated randomly. Note that some w_{ij} could be equal to zero by chance when p is even number.

What is new about the application of this algorithm to mutually connected neural network is that the connection matrix W_0 calculated in this way remains fixed during the evolution.

2.2 population

We multiply each w_{ij} by $e_{ij} \in \{-1, 0, 1\}$ which is generated randomly where the generation rates of -1 and 0 are both at $1/50$. Here we must notice that -1 reverses the role of excitation/inhibition of connection weight, and 0 prunes the connection. By repeating this 256 times, we produce 256 connection matrices which are slightly different from the original W_0.

In our simulation, the larger the population, the higher the capacity we obtained. In this paper, however, the population size are set to 256, considering the limited resources of our computers.

2.3 chromosome

The chromosome of each individual matrix is made up of these e_{ij}. The chromosome has the length of N^2, and keep its size constant during evolution.

2.4 asynchronous update and fitness

We evaluate fitness of each chromosome during the states of neurons are updated. In the beginning, the i-th neuron is assigned a value of ξ_i^μ. The state of each neuron is asynchronously updated by

$$s_i^\mu(t+1) = sgn(\sum_{j=1}^{N} w_{ij} s_j^\mu(t)),$$

where $s_i^\mu(t)$ is the state of i-th neuron at time t, and N is the total number of neurons. At each step of updating, a neuron is chosen according to pre-assigned order (once in a cycle), instead of chosen randomly. For the remainder of this paper, the term "update" will refer to this asynchronous update.

As shown in Figure 2, the fitness f of each modified matrix is calculated by summing up the inner products of initial states and the states after being updated 98 steps, over all sets of given initial patterns. That is,

$$f = \frac{1}{49 \cdot p} \sum_{\mu=1}^{p} \sum_{j=1}^{49} s_j^\mu(0) s_j^\mu(98)$$

evaluate the fitness value of each individual in the population.

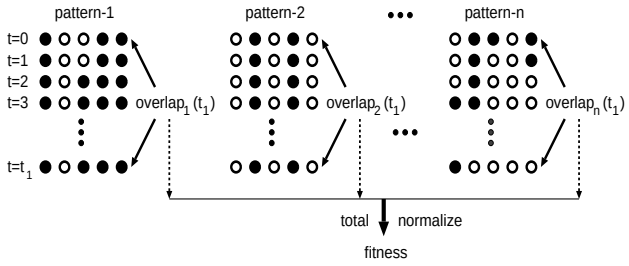

Figure 2: The evaluation of fitness. The fitness is calculated by summing up the inner products of initial states(at $t = 0$) and the states after being updated t_1 steps, over all sets of initially given patterns, and normalizing it.

The time of updates was twice as much as the number of neurons here. And this was enough for the present purpose of selecting individuals of higher fitness. The modification of the original Hebbian learned matrix is made only slightly, and we eliminate the solution of limit cycle by keeping diagonal elements of connection matrix zero. As described above, when states of neurons are updated, a neuron is selected once in a N steps according to the pre-assigned order. So if all the neurons do not change their states in this cycle, they will not change afterwards. Indeed, if the number of neuron which change its state at N-th step is zero, or, if the number at $2N$-th step is one, we can evaluate that the solution is quite stable. And this worked well to find solutions of high fitness in our genetic algorithm. Therefore we set it to $2N$, i.e. 98 in this paper. We must notice, however, that we need another way of fitness evaluation in the case of randomly generated connection matrix [12].

2.5 mutation and crossover

The individuals which have higher fitness produce other individuals to make the next generation. In making offsprings we used the following system of mutation and crossover.

As the mutation, we add one at the rate of $1/100$ to the value of allele, which is either -1, 0 or 1, and if the allele value becomes 2 we change it into -1.

And then crossover is as follows. Two chromosomes were chosen randomly from the upper 40% of

the population, which is rearranged in every generation in descending order of their fitness value, and a new chromosome is produced by taking either allele value of the two parent chromosomes, allele by allele. Then this is replaced with the one in the lower 60% of the population. This process is repeated until all the chromosomes in the lower 60% of the population are replaced with new chromosomes, in other words, a uniform crossover. We show the schematic diagram of the uniform crossover in Figure 3.

We are trying other types of crossover, but up to now we obtain the best results when we use this uniform crossover.

We determined those parameters above through some trial and error tests and set them fixed in the simulation of this paper.

We show the overview of our genetic algorithm in Figure 1. As we do not give any noise to initial patterns in this paper, each basin volume has not been evaluated yet.

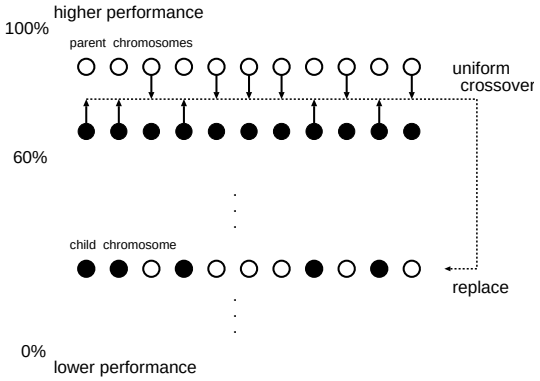

Figure 3: Schematic diagram of uniform crossover. Two chromosomes were chosen randomly from the upper 40% of the population, and a chromosome is produced by taking either allele value of the two parent chromosomes, allele by allele. This is repeated until all chromosomes in the lower population are replaced.

3 Results and Discussions

We applied the genetic algorithm above to the Hopfield net work, and obtained twice as much associative memory capacity. In this section we are describing the results of our simulations with some discussions.

3.1 General feature of the evolution

To grasp the overview of our evolution, let us start from the general feature of the evolution of Hebbian learned matrices.

3.1.1 mild modification

As we described above, our method modify original connection matrix W_0 by multiplying zero and -1 at a very small rate. In this situation, only a few elements of W_0 are modified even after long generations. The following is an example of the modification of connection matrix.

- total number of elements: $49^2 = 2401$
- number of modified elements: 316 (13.2%)
- number of pruned connections: 62 (2.6%):

This is an example for the most successful case where 16 patterns are given to the network, and all of them are successfully memorized after evolving 10795 generations. 13.2% of the whole connections are modified, and 2.6% are pruned through evolution. Though we will describe this more precisely later, here we have to notice that we observe the same tendencies, be it successful or not so, throughout our simulations in this paper. We have to also notice that all the diagonal elements of connection matrix remain zero during evolution, because those are set to zero by Hebbian rule before evolution, and can not be changed by being multiplied by any e_{ij}. Therefore we do not allow a neuron to have feedback to itself. In this sense, the modification might be said to be very mild.

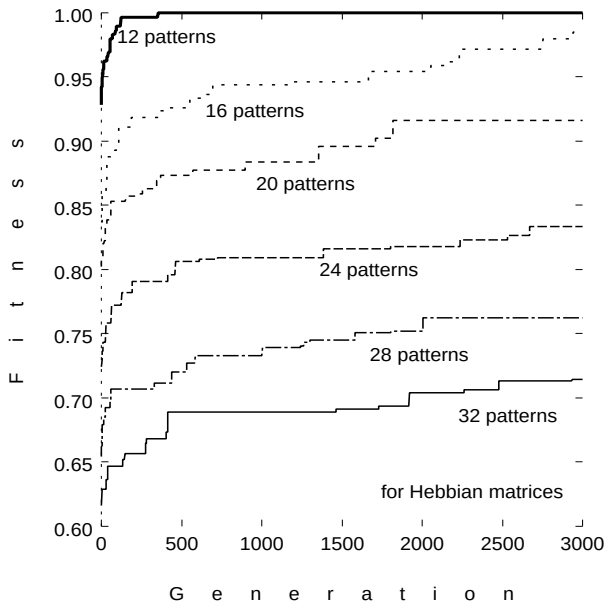

Figure 4: Examples of fitness versus generation, for 12, 16, 20, 24, 28 and 32 patterns.

3.1.2 evolvabilty

The size of the neural network used in this simulation was 49 neurons. And we simulated with the num-

ber of initial patterns to be memorized from 12 to 32. Since the connection matrices in this paper had learned those patterns by Hebbian rule beforehand, even at the first generation the fitness values are quite high when the number of patterns are small enough. But still we can observe the fitness increase. We show some results in Figure 4.

3.1.3 convergence

Naturally, the smaller the number of patterns, the faster the fitness attains to the highest value. We show the generation at which the elitest individual attained the fitness of 1.000 in Table 1. For more than 16 patterns, we have not obtained the fitness of 1.000 yet, though we could obtain higher capacity with more population and/or more generation.

Table 1: Generation number at which the elitest individual in the generation attains fitness of 1.000, for 12, 13, 14, 15 and 16 patterns.

patterns	12	13	14	15	16
generation	354	1268	6003	6688	10795

3.2 Twice as much memory capacity

We succeeded in doubling the associative memory capacity of Hopfield network with the genetic algorithm mentioned above. This section explore the improvement of the capacity in detail.

3.2.1 overlap

In our simulations, we first generate some patterns ξ^μ randomly, and then determine connection matrix W_0 according to the patterns by Hebbian rule. To see the extent to which our method improve memory capacity, we evaluated the memory capacity of this original connection matrix W_0. Network with connection matrix W_0 are given the patterns ξ^μ and updated 98 steps in the same way as in the genetic algorithm. And the overlapping rate defined by

$$\frac{1}{49 \cdot p} \sum_{\mu=1}^{p} \sum_{j=1}^{49} s_j^\mu(0) s_j^\mu(98)$$

are calculated, which is also the same as the fitness defined above. When an updated state $s_j^\mu(98)$ overlaps with its initial state $s_j^\mu(0)$, the product of them is 1, whilst -1 otherwise. So, if all the updated states overlap with their corresponding initial states, the overlapping rate will equal to unity. And if the half of them overlap with their initial states, this will be zero. The results obtained are shown in Table 2, for 15, 16, 17 and 18 patterns, with the fitness values of the elitest individual evolved from the same original connection matrix W_0.

We can always obtain the higher memory capacity after modifying the connection matrix by the genetic algorithm.

Here we must note that 0.801 in Table 2, for example, simply means $(49 \cdot 15 - 73 \cdot 2)/(49 \cdot 15)$. This implies that the total of 73 updated states are different from the corresponding initial states. These figures in Table 2 seems to indicate that the capacity of this system could be 16 patterns.

Table 2: Overlapping rate of the states updated 98 steps with their corresponding initial states before and after genetic algorithm, for 15, 16, 17 and 18 patterns.

patterns	15	16	17	18
before GA	0.801	0.842	0.772	0.752
after GA	1.000	1.000	0.990	0.982

3.2.2 fixed point

In Hopfield network, when the number of patterns to be memorized is small enough, all the patterns are guaranteed to be memorized as a fixed point, i.e.,

$$s^\mu(t) = s^\mu(0), \quad \text{for } \forall t.$$

The same does not hold for the patterns greater than the capacity, around eight in this case. Take an example of our case for 16 patterns, which exceeds the capacity. Although a few patterns are still memorized as a fixed point, most of them do not converge. An example of a pattern which does not converge is shown in Table 3 (a). This is a time series of Hamming distance between the initial states and each updated states, i.e. the number of neurons different from its corresponding initial states. When this pattern is given to the network, overlapping states increases with time of update. After applying the genetic algorithm, however, this pattern was memorized as a fixed point, hence the Hamming distances remain zero from the beginning as shown in Table 3 (b).

In this simulation, the matrix which has the fitness of 1.000 implies that all the patterns initially given to the network are memorized as a fixed point.

Table 4: The comparison of the number of patterns memorized as a fixed point before and after genetic algorithm, for 15, 16, 17 and 18 patterns.

patterns	15	16	17	18
before GA	2	4	3	3
after GA	15	16	15	14

Table 3: An example of time series of Hamming distances between initial states and each updated states, i.e. the number of neurons different from its initial state at each time, up to 220 time steps. Time is from top left towards right. (a): Worst case out of 16 patterns before genetic algorithm, and (b): after genetic algorithm for the same pattern.

(a)																								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	2	2	3	3	3	3	3	3	3	3	3	3	4	4	5	6	6	6	6	7	7
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	9	9	9	10	10	10	10	10	10	10	10	10	10	11	11	11	11	11	11
11	11	11	11	11	11	11	11	11	11	11	12	12	12	12	12	12	12	12	12	12	12	13	13	13
13	13	14	14	15	15	15	15	15	15	15	15	15	15	15	15	15	15	16	17	17	17	17	17	18
18	18	18	18	18	18	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19
19	19	19	19	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	...				
(b)																								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...				

The number of patterns which are memorized as a fixed points p_0 could also be the criterion for the memory capacity. In Table 4, we show the number of patterns memorized as a fixed point before and after the genetic algorithm, for 15, 16, 17 and 18 patterns, which also seems to indicate 16 patterns are the capacity.

From the discussions above, we conjectured the capacity of our method in this paper to be 16 patterns, that is 33% of the number of neurons. In the Hopfield network, on the other hand, the capacity of associative memory is about 15% of the number of neurons as described earlier. As we experimented with 49 neurons here, the eight patterns are around this limit. Therefore we may conclude that we can double the associative memory of Hopfield network.

3.3 Features of the connection matrices

To investigate what is essential for the improvement in memory capacity, we checked two parameters in the elitest connection matrix in each generation, i.e. the degree of symmetry and the number of connections pruned.

3.3.1 degree of symmetry

We calculated the degree of symmetry η of the matrices by

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

At the beginning of the evolution, the connection matrices are symmetric because they are determined by

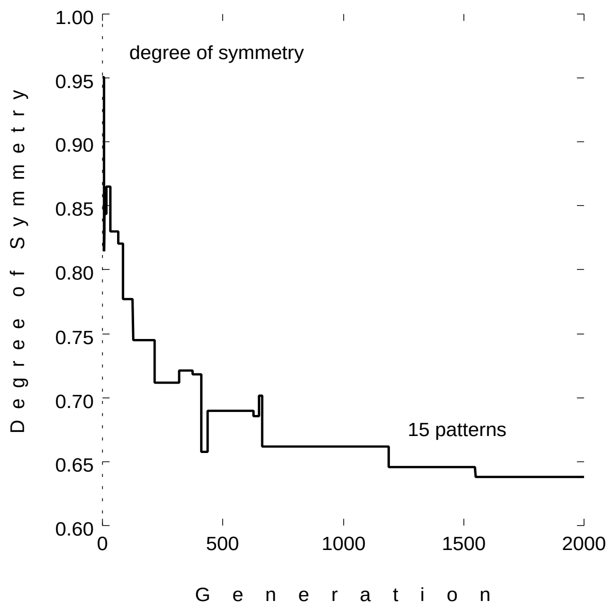

Figure 5: A typical example of transition of degree of symmetry at an early stages of evolution, for 15 patterns. (This is the different sample from the one in Figure 6). We can see rapid decrease in earlier generations.

Hebbian rule. And as shown in Figure 5 the connection matrix changes quite rapidly from symmetry to asymmetry as higher fitnesses are attained. We also show examples of transition of the degree of symmetry, for 15, 16, 17 and 18 patterns in Figure 6. For these patterns the values converge to around 70%

Consequently, we may conjecture that asymmetry introduced by 0 and -1 in the chromosome have something to do with the effectiveness of the learning.

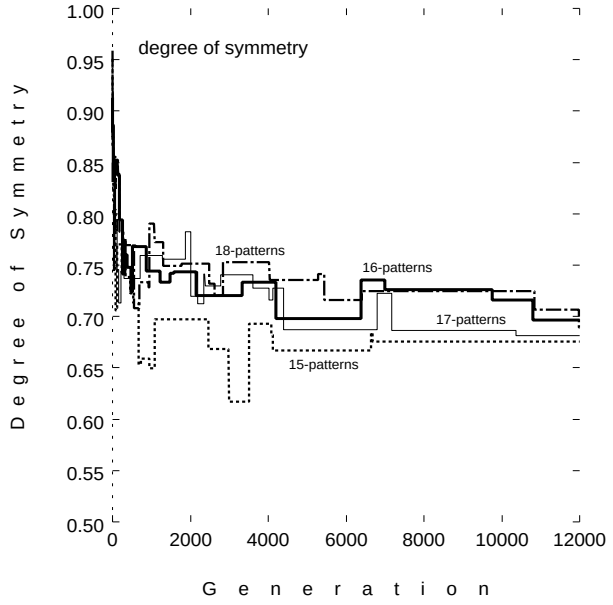

Figure 6: Transition of the degree of symmetry of the connection matrices up to 12000th generation. Examples of 15, 16, 17 and 18 patterns. The values converge to around 70%

3.3.2 rate of pruned connection

When we start the genetic algorithm, the diagonal elements of the original weight matrix W_0 are all zero. In addition, when the patterns given to the network are even number, some of other elements could also be zero, whilst in the case of odd number of patterns this dose not happen. And these elements of zero do not alter during evolution. In producing individuals in each generation, some of the non-zero elements of the original Hebbian matrix become zero by being multiplied by zero in chromosome. This implies that the connections are pruned, as we described before. Here we define the rate of pruned connection ζ as {the number of non zero elements multiplied by zero in producing individuals} divided by {total number of connections}. As shown in Figure 7, the rate of pruned connection increases quite rapidly until higher fitness is obtained, and does not alter drastically afterwards. We also show an example for 15, 16, 17 and 18 patterns in Figure 8. We can observe the rates converge approximately to the value of 3%.

Hence, we may conjecture that pruned connections also play an important role to improve the associative memory capacity of Hopfield network.

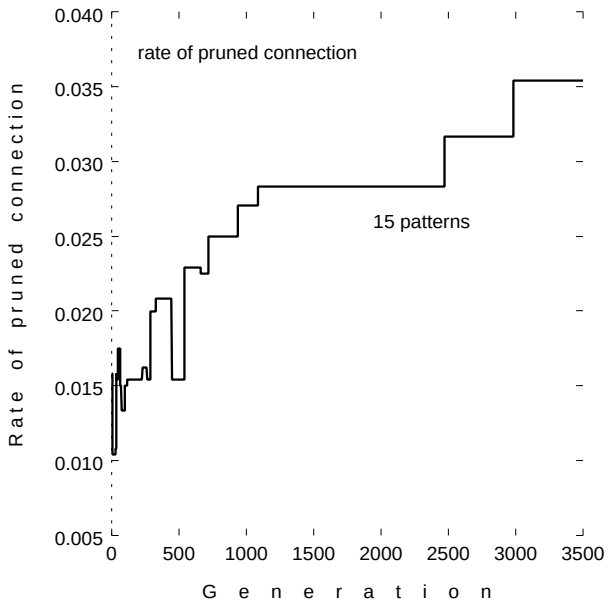

Figure 7: An Early stage of the rate of pruned connection. An example of 15 patterns. We can see quite a rapid increase.

3.3.3 roles of 0 and -1 in the chromosome

As described earlier, -1 in the chromosome reverses the role of excitation/inhibition 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 chromosomes chosen from $\{-1, 1\}$ and the other is from $\{0, 1\}$, with other parameters remain exactly the same as those chosen from $\{-1, 0, 1\}$ above. We showed the results in Figure 9. As we can see in Figure 9, although the chromosome chosen from (a):$\{-1, 0, 1\}$ worked most effectively, the chromosome chosen from (b):$\{-1, 1\}$ also worked. In the case of the chromosome chosen from (c):$\{0, 1\}$, however, it was trapped in a local minimum at an very early stage of evolution. The number of fixed points obtained in the experiment(b) was 14, whereas 9 in the experiment(c). The role of -1 seems to be more important than 0, but we also need 0 to obtain 16 fixed points in the experiment(a). And in addition, -1 helps to avoid being trapped in a local minimum.

With those results described here we may conclude that both full connectivity and symmetry of the connection matrices prevented the networks from attaining high memory capacity in Hopfield network, though

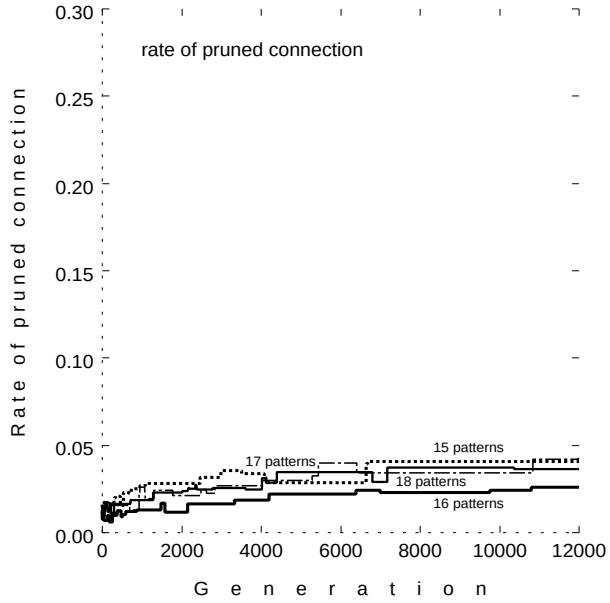

Figure 8: Transition of rate of pruned connection up to 12000th generation. Examples of 15, 16, 17 and 18 patterns. The values converge to around 3 through 4%.

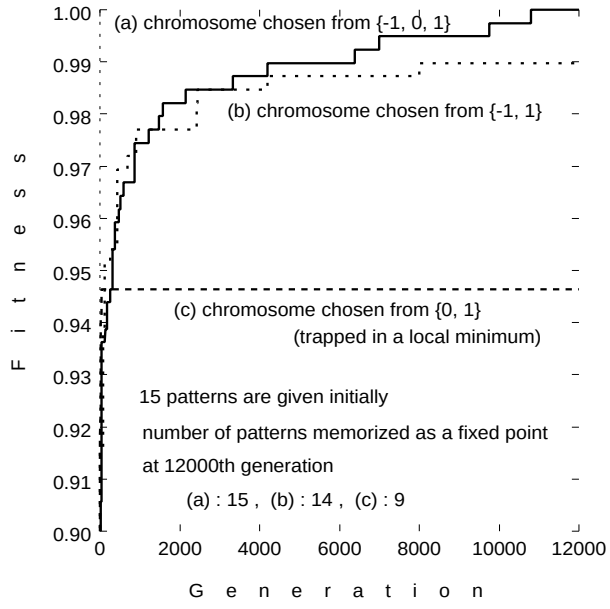

Figure 9: The evolution of the fitness with the chromosome chosen from (a): $\{1, 0, -1\}$, (b): $\{1, -1\}$ and (c): $\{1, 0\}$. In the case of (c), the evolution was trapped in a local minimum.

we need further analysis to explain these phenomena.

3.4 Generality of those simulations

In each number of patterns above, we repeated simulations with 20 (or more) different seeds of random numbers with other parameters remain fixed, and we were able to observe the general tendencies described here. Some of the example of the data from 15 patterns are shown in Table 5.

The figures in this paper are mainly those of best results out of these simulations.

Table 5: Example of data from ten simulations with different seeds of random numbers with other parameters remain fixed, for 15 patterns. f, ζ, η, δ and p_0 in the first row represent fitness, rate of pruned connection, degree of symmetry, rate of modified element of the connection matrix and rate of patterns memorized as a fixed point, respectively.

	f	ζ	η	δ	p_0
pattern - a	0.986	0.035	0.673	0.199	12
pattern - b	1.000	0.041	0.676	0.201	15
pattern - c	0.997	0.042	0.654	0.209	14
pattern - d	0.997	0.039	0.659	0.190	14
pattern - e	0.992	0.024	0.696	0.166	12
pattern - f	0.984	0.037	0.623	0.196	13
pattern - g	0.995	0.030	0.708	0.162	13
pattern - h	1.000	0.052	0.634	0.250	15
pattern - i	0.995	0.027	0.696	0.164	13
pattern - j	0.989	0.046	0.635	0.214	13

4 Conclusions

First, we proposed a new simple technique of genetic algorithm to modify the connection matrix of a mutually connected neural network to enlarge the associative memory capacity of a Hopfield network. The feature of our technique is to keep the original connection matrix unchanged during the evolution. The individuals of each generation are slightly different from the original weight matrix, keeping the diagonal elements zero. This is done by multiplying -1 or 0 at a very small rate to the original matrix, which introduces both asymmetry of the weight matrix and sparseness of the connections. The genetic algorithm searches the best allocation of zero and -1 in the chromosome to obtain a higher capacity.

Next, we described that this technique is quite successful in improving the associative memory capacity of a Hopfield network. The capacity was always higher in the modified connection matrix than in the original matrix learned by Hebbian rule. We obtained the capacity of about 33% of the number of neurons

after applying the genetic algorithm whereas the original Hebbian matrix has the capacity of about 15%.

We suppose this effect is due to the sparseness of connection and the asymmetry of weight matrix introduced by the genetic algorithm.

Our investigations, however, have not been conclusive on this point, due to the difficulty of specifying exactly what is meant by these asymmetry and sparseness. In the present studies we have shown that there is already some possibility to use our method to obtain high capacity of associative memory.

5 Future works

The results reported in this paper should be thought of as a first attempt to establish a method to obtain higher associative memory capacity and construct a model to explain the mechanism. And we are thinking of the following procedures as a next step.

First, to make the results more reliable, we want to simulate neural networks with at least 100 neurons instead of 49 neurons. The problem is the time of simulation. So we are beginning to explore them by parallel processing on a distributed system.

Second, to make our simulation more effective, we are introducing continuous values with the sigmoid function instead of discreate values with the signus function. In addition, we use either -1 or 0 in the chromosome to modify the weight values, and they do not change the absolute value of the connection weights except for the case of making them zero. So, we are thinking of other types of chromosome and crossover to change the weight values more flexibly through evolution.

Third, we are trying to combine some weight matrices obtained from independent simulations in order to attain much higher capacity, though we have not succeeded yet.

Finally, we should construct a model to account for the phenomena described in this paper.

Acknowledgements

We thank Peter Davis and Shin Ishii at ATR (Advanced Telecommunication Research Institute) for giving us helpful comments and suggestions about the evolution of associative memory of recurrent neural networks. Thanks also go to other people at ATR as well as at NAIST (Nara Institute of Science and Technology) who gave us useful discussions at various stages of this research.

References

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

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

[3] S. Amari, and K. Maginu (1987). Statistical Neurodynamics of Associative Memory. *Neural Networks* **1**:63-73.

[4] D. J. Amit, H. Gutfreund, and H. Somplinsky(1987). Statistical Mechanics of Neural Networks near Saturation. *Annals. of Physics* **173**:30-67.

[5] T. Kohonen (1977). *Associative Memory: A System-Theoretical Approach.* Berlin Heidelberg: Springer-Verlag.

[6] H. Yanai, Y. Sawada, and S.Yoshizawa (1991). Dynamics of an Auto-Associative Neural Network Model with Arbitrary Connectivity and Noise in the Threshold. *Network* **2**(3):295-314.

[7] M. Morita (1993). Associative Memory with Nonmonotone Dynamics. *Neural Networks* **6**:115-126.

[8] J. D. Schaffer, and D. Whitley (1992) (ed.), *Proceedings of the Workshop on Combinations of Genetic Algorithms And Neural Networks.* The IEEE Computer Society Press.

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

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

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

[12] A. Imada, and K. Araki (1995). Genetic Algorithm as a Learning algorithm in Associative Memory. WCNN'95 (to appear).