Genetic Algorithm Enlarges Capacity of 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 propose a technique to obtain higher capacity of associative memory with mutually-connected neural networks by a simple genetic algorithm. In Hopfield network as an associative memory system, capacity is at most 15% of the number of neurons. We successfully obtained the capacity of about twice as much as that of Hopfield network by applying our method to connection matrices which memorize binary patterns by Hebbian rule.

1 Introduction

In 1982, Hopfield[1] proposed a neural network model as an associative memory system. He used Hebbian rule [2] to make the 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 derived more analytically by Amari et al. with neurodynamics [3], and by Amit et al. with spin-glass theory [4]. Since Kohonen[5] proposed several types of connection matrices for associative memory, many researchers have tried to improve the capacity in various way. For example, Yanai et al. tried that by introducing sparse connectivity [6], and Morita by using different dynamic system [7]. So far, however, mechanisms of obtaining higher capacity are not well-understood yet. Here we tried to improve the memory capacity by using

genetic algorithm.

Since 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 for example [9, 10]), 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.

We presented in another report that randomly-generated connection matrices can learn some patterns without any learning rules through genetic algorithms[12]. But in this paper, we applied genetic algorithms to connection matrices which learned some patterns in advance by Hebbian rule. One of our goals is to improve capacity obtained by Hebbian learning through evolutions.

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

2 Method

We simulated with mutually-connected neural networks with 49 neurons. At the beginning of genetic algorithm, connection matrix is determined by Hebbian learning, i.e.

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

for given P patterns ξ^μ $(\mu = 1, 2, \cdots, P)$, where

$$\xi^\mu = (\xi_1^\mu, \xi_2^\mu, \xi_3^\mu, \cdots, \xi_N^\mu),$$

and each ξ_i^μ $(i = 1, 2, \cdots, N)$ is either 1 or -1 generated randomly. As described above, the w_{ij} is determined by Hebbian rule according to a set of patterns given before evolution, and this remained fixed during the simulation.

We multiplied each w_{ij} by $e_{ij} \in \{-1, 0, 1\}$ which is generated randomly where generation rates of -1 and 0 were both 0.02. By repeating this process 256 times, we produce 256 connection matrices which are slightly different from each other. In our simulation, we can obtain the higher capacity for the larger

population. However we chose 256 as a population because of our resources.

Chromosomes of our genetic algorithm are made up of these e_{ij}. Each individual matrix in the population is produced from this chromosome and original connection matrix. And the chromosomes change their allele value while keeping its size constant during the evolution .

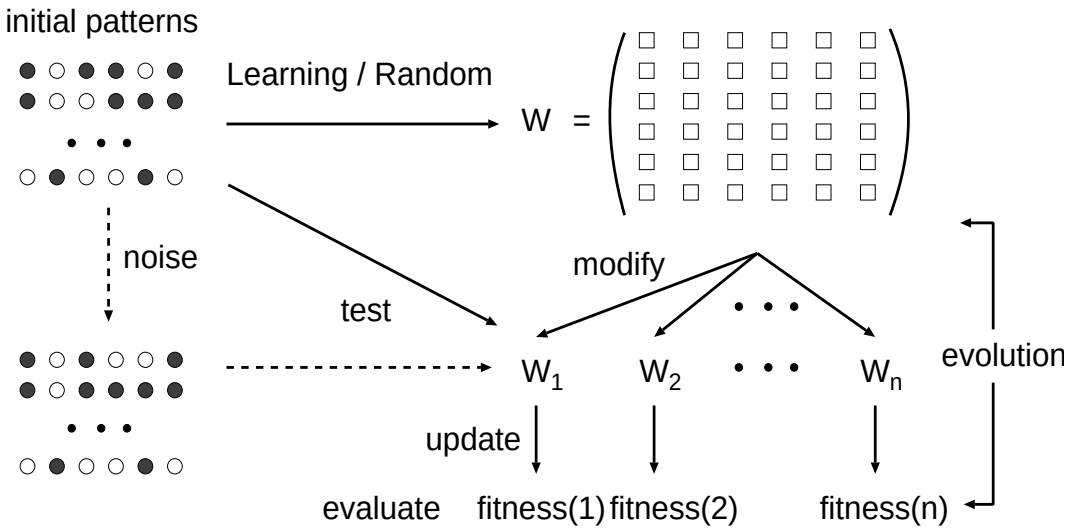

Figure 1: **Overview of the Genetic Algorithm**

We used the following system of mutation and crossover. We add 1 at the rate of 0.01 to the value of allele i.e. -1, 0 or 1 (if the value becomes 2 we change it into -1). And then two chromosomes were chosen from upper 60% of the population which is ordered according to their fitness value (we defined it later), and exchange each allele by the probability of 0.56, which is reffered to as uniform crossover. We are trying other types of crossovers, but so far, we have not obtained the better results than this uniform crossover.

We evaluate fitness of each chromosome during the state 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. For the remainder of this paper, the term "update" will refer to

this asynchronous update.

Fitness value f of each modified matrix is evaluated by summing up inner products of initial states and final states after updates for all sets of given initial patterns. The time of updates was twice as much as the number of neurons. So,

$$f = \frac{1}{N \cdot P} \sum_{\mu=1}^{P} \sum_{j=1}^{N} s_j^{\mu}(0) s_j^{\mu}(T),$$

where P is the total number of patterns, and T is the time of updates, i.e. $T = 2N$ here.

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

3 Results and Analysis

First of all, we simulated with the initial patterns of number from 4 to 98. Since we started with connection matrix which learned those patterns by Hebbian rule as described earlier, fitness values of the first generation are quite high when the number of patterns are small enough. However the increase of fitness value can be still observed. We showed some results in Figure 2. Take the case of 12 patterns as an example, we obtain the fitness of 1.000 just after 354th generation.

We generated each set of initial patterns randomly, and the fitness values vary according to covariance of the set of patterns. However we were able to observe the general tendencies described here.

We obtained the fitness of 1.000 within 3000 generations if the number of patterns are less than 14. The fitnesses converge faster in smaller number of patterns. In the case of 12, 13 and 14 patterns, for example, the elitest individual attained the fitness of 1.000 at the generation of 354, 1263 and 6003 respectively. But in the case of 15 patterns, we did not obtain the fitness of 1.000 even after the 12000th generation. In this sense, our system has its own limit in memory capacity.

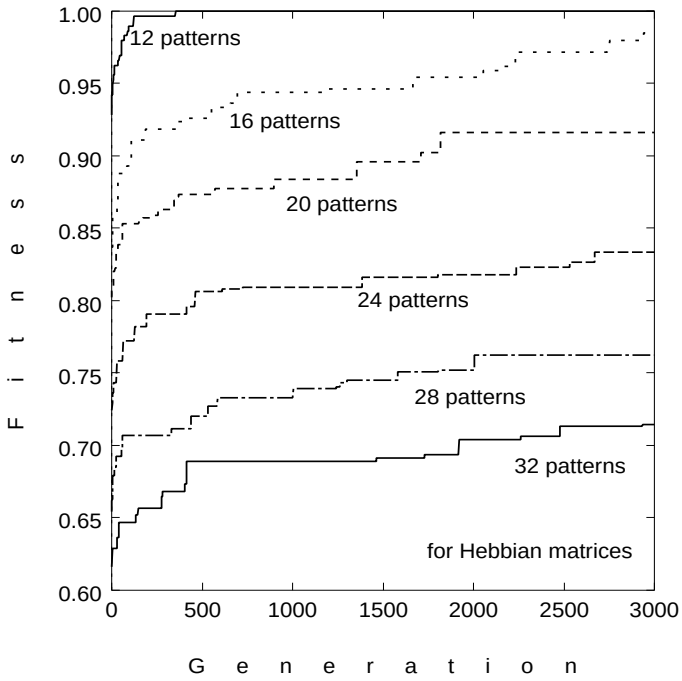

Figure 2: **Generation vs. fitness in GA after Hebbian learning**

3.1 Improvement of Memory Capacity

We compared memory capacity of connection matrices before and after genetic algorithm for the same set of patterns. As shown in Table 1, we can obtain the higher memory capacity after modifying the connection matrices by genetic algorithm.

In Hopfield network, 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. In the case of Hopfield network full connection of the neurons and symmetric connection matrices might prevent the network from having a large capacity. The genetic algorithms above, however, prune some of the connections during the evolution by multiplying zero to the weight. And the final connection matrices are not symmetrical. And we suppose that the improvement of the capacity of our simulations is partly due to those facts.

The performance of the GA-modified-matrix is always higher than the original Hebbian-learning-matrix, but their performances decrease as the number of patterns increases. Although we did not obtain the fitness of 1.000

Table 1: **Hebbian learning vs. GA-after-Hebbian-learning. The 2nd row indicates the sum of overlapping between initial patterns and updated pattern. And the 3rd row indicates fitness values at the 3000th generation.**

number of patterns	4	8	12	16	20
Hebb only	1.000	0.990	0.837	0.758	0.633
Hebb + GA	1.000	1.000	1.000	0.985	0.916

for 12 patterns as in Table 1, we regard the network memorized them, because network can retrieve a pattern that is quite close to the initial pattern. Therefore, we conclude that we can obtain the "effective" memory capacity of about twice as much as Hopfield network.

3.2 Features of GA-Modified-Connection-Matrices

We traced dilutedness (zero-rates) and symmetricity in the elitest connection matrix of each generation.

3.2.1 Symmetricity

We calculated the symmetricity η of a matrix 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 Hebbian rule. And then the connection matrix changed quite rapidly from symmetry to asymmetry until high fitness values are obtained, where the value is around 70% for any number of initial patterns.

3.2.2 Two Stable States of Dilutedness

The behavior of dilutedness, on the other hand, is somewhat different. When we start the algorithm, the zero rates are $49/49^2 \sim 0.02$, because the orthogonal factors of Hebbian matrix are all zero. And from the first generation, the elitest connection matrix has a certain level of asymmetricity and does not change largely afterwards. Here we noticed the fact that there were two stable levels of dilutedness depending on simulations. One is around 4

through 5% and another is 20 through 30%.

With those results described here we may conclude that both full-connectivity and symmetricity of the connection matrices prevent the networks from obtaining higher capacity, though we need further analysis.

4 Conclusions

First, we proposed a new simple technique to modify the connection matrix of a mutually-connected neural network. The feature of our technique is to keep the original connection matrix unchanged during evolution. The individuals of each generation are slightly different from the original weight matrix. This is done by multiplying -1 or 0 at small rates to the original matrix, which introduces asymmetricity and dilutedness to the connection matrix of each individual. The genetic algorithm searches the best manner of difference toward a higher capacity.

Next, we described that this technique is quite successful in improving the capacity of associative memory of the mutually-connected neural networks. 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 30% 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 dilutedness and the asymmetricity of the connection matrix introduced by the genetic algorithm.

5 Future works

We chose 49 as the number of neurons of mutually-connected neural networks. But we want to simulate neural networks with at least 100 neurons to make our simulation more reliable. Problem is the time of simulation. So we are beginning to explore them with parallel machines.

In the method here, weight values are discrete and transfer function is signus function. But we are introducing continuous values with sigmoid function to make our simulation more effective. In addition, we used either -1, 0 or 1 to modify the 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 types of crossover to change the weight values more flexibly

through evolution.

Acknowledgments

We thank Peter Davis and Shin Ishii at ATR (Advanced Telecommunication Research Institute) for giving us helpful 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.

References

[1] 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)

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

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

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

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

[6] 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)

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

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

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

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

[11] 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)

[12] Imada,A., and Araki,K. Mutually Connected Neural Network Can Learn Some Patterns by Means of GA. WCNN-95 (submitted)