Evolution of Associative Memory with Environmental Change

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

Evolution of Associative Memory with Environmental Change

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

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

Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-01 Japan

Abstract

There have been a lot of researches which apply evolutionary techniques to layered neural networks. However, their applications to Hopfield neural networks remain few so far. We have been applying a genetic algorithm to fully connected associative memory model of Hopfield, and reported elsewhere that the network can store some number of patterns only by evolving weight matrices with the genetic algorithm. In this paper, we present that evolving both weight matrices and patterns to be stored results in the higher capacity of associative memory than in the case of fixed patterns above. From an Artificial Life perspective, the patterns to be stored are considered to be an environment.

1 Introduction

Associative memory is a process in which we can store information by distributing it among neurons. One of the most appealing features is its capability to tolerate noisy and/or partial input in recalling the information, though it has a limited capacity.

These networks might be used as a memory organ of digital organisms of our Artificial Life research in the future. However, as a preliminary step, we are now investigating basic behavior of the function of associative memory under simple evolutionary processes. We are now using a genetic algorithm ([1], [2]) to evolve the networks.

Genetic algorithms have been extensively exploited for layered neural networks (see e.g, [3]), but curiously have not been applied to Hopfield model. Probably this is because the Hopfield network has a fixed architecture. However, sparse connectivity of the network has a possibility to enhance the associative memory capacity [12]. Hence the genetic algorithm may be used also as a method to determine the architecture of Hopfield networks. In fact, we showed that the genetic algorithm enlarges the capacity of Hebb-rule associative memory by pruning some of its connections [4]. We also reported that a random weight matrix was evolved to store some number of patterns only by means of the genetic algorithm [5].

In the context of Artificial Life, it has been shown that environmental change enhances some performances [6], [7]. Here we give the patterns to be stored an opportunity to be modified in every certain generations so that the fitness of individuals are improved. When we take the patterns as a language of individuals in a population, it is interesting to see if slight occasional modifications of their language enhance the individual's adaptability.

The goal of this research is twofold. One is to improve the techniques to enlarge the associative memory capacity which are proposed so far, and the other is to know if the ability of associative memory can be adaptive to the changing environment.

In the following section, we provide an outline of our experiment, a brief review of associative memory and the implementation of our genetic algorithm. The difference from our previous implementation is described in detail.

2 Experiment

We apply a simple genetic algorithm to fully connected associative memory model of Hopfield. To be more specific, an auto-associative, discrete-time asynchronous-update, bipolar-state network. The connection weights are chosen randomly either from -1 or 1 at the start of the genetic algorithm, therefore the network can not store any patterns initially. The objective of the experiment is to make this network store some number of patterns without any learning algorithm besides the genetic algorithm.

We set the size of the network to 49 neurons in this paper, considering computational expense. This size of *Hebb-rule associative memory* would have a capacity of 7 patterns at most. In our previous simulation with the networks of the same size, we succeeded in evolving a random weight matrix eventually to store 8 patterns [5]. In this new version of the experiment, we occasionally change the patterns to be stored to see if it enhances the capacity.

From Artificial Life point of view, each individual in the population tries to memorize a given set of patterns, increasing its accuracy, generation by generation. And our concern in this paper is how these occasional changes in language affect the accuracy and the capacity.

Before describing the implementation of our genetic algo-

rithm, let us review some fundamental concepts relating to associative memory.

Associative Memory

Hopfield network which has N neurons can store some number of N-bit bipolar patterns, if all the connection weights are determined adequately. The patterns are n-dimensional random vectors whose elements take the values ± 1 randomly with equal probability. The *μ-th* pattern is denoted as

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

where ξ_i^μ is the *i-th* bit of the *μ-th* pattern which takes the value of either -1 or 1. To store these patterns, each connection weight has to be adequately adjusted. If we use Hebbian rule, for example, the weight w_{ij} is determined as follows:

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

where p is the number of patterns to be stored. This process of determining w_{ij} is referred to as *learning*.

These w_{ij}'s define a so called energy function on an N-dimensional bipolar state space, i.e.,

$$E(S) = \frac{1}{2} \sum_{i,j(i \neq j)} w_{ij} S_i S_j,$$

where S moves over all possible combinations of N-dimensional vector $\{-1, 1\}^N$ and S_i is the *i-th* component of the state S. If the number of stored patterns is small enough, this function takes a local minimum where S corresponds to one of stored patterns. In this sense the stored patterns are said to be *attractors* and the size of this local minimum are referred to as *basin of attraction*. As the number of stored patterns p increases, spurious minima become non-negligible and finally these local minima are randomized.

The retrieval of one of stored memories is as follows. 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 the *i-th* neuron at time t when the *μ-th* pattern is given to the network. For each neuron, the weighted sum of its inputs is computed, and if this is greater than or equal to zero, the state of the neuron takes the value of 1, and -1 otherwise. In this paper, a neuron is chosen according to pre-assigned order (once in a cycle) at each step of updating, instead of chosen randomly.

When input is one of the stored patterns which is given small noise within the size of basin of attraction, this input pattern relaxes to the original stored pattern after several steps of the update. If we do not give any noise to the input, on the other hand, i.e., one of stored patterns is given to the network, all the states remain unchanged from the start, and the patterns are said to be memorized as *fixed points*. We evaluate the ability of weight matrix by the upper bound of the number of patterns stored as fixed points.

When Hopfield [8] proposed a neural network model as an associative memory system, in which he used Hebbian learning rule, he estimated the memory capacity to be at most 15% of the number of neurons with using computer simulations. Later the capacity was verified more analytically by Amit et al. [10] with spin-glass theory, and by Amari et al. [11] with statistical neuro-dynamical method. Since then, many researchers have been successfully trying to enlarge the capacity. For example, Yanai et al. [12] made it by introducing sparse connectivity, Morita [13] by replacing the sigmoid function with a non-monotonic transfer function, and Willshow et al. [14], Palm [15] and Meunier et al. [16] by using sparsely coded patterns instead of random patterns. Here we are exploring this by means of genetic algorithms.

To know more about associative memory, see for example [17].

Genetic Algorithm

In this subsection we describe the implementation of our genetic algorithm briefly. As in Figure 1, our genetic algorithm proceeds as follows:

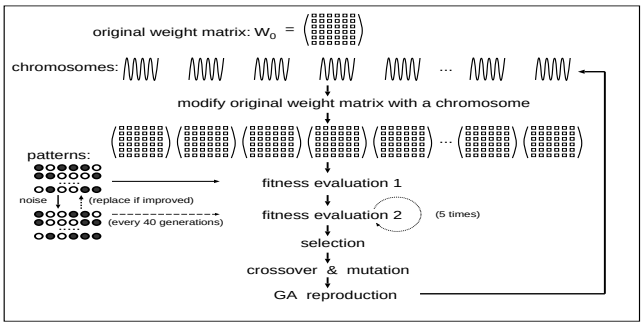

Figure 1: **Schematic representation of our GA**

(1) At first a weight matrix $W_0(49\text{x}49)$ is produced randomly so that each component of the matrix w_{ij} are chosen from $\{-1, 1\}$. Hence the matrix does not store any patterns at this moment. This original weight matrix remains unchanged during evolution. **(2)** Then 256 chromosomes are produced also randomly. The chromosome has a fixed length of 2401(=49x49) alleles, and its alleles are chosen randomly from $\{-1, 0, 1\}$, where the probability of choosing either -1 or 0 is set to 0.02 here. Each component of the original weight matrix w_{ij} is multiplied by one of these alleles. Allele 0 implies to prune the connection which is multiplied, and -1 to reverse excitatory/inhibitory connection. We denote the i-th allele of the n-th chromosome as $c^{(n)}(i)$. **(3)** Each chromosome modifies the original weight matrix W_0, which produces 256 weight matrices slightly different from W_0. The modification was made as follows:

$$w_{ij}^{(n)} = w_{ij} \cdot c^{(n)}(49i + j),$$
$$(i, j = 0, 1, 2, \cdots, 48; \quad n = 1, 2, 3, \cdots 256),$$

where $w_{ij}^{(n)}$ denotes i-j component of the n-th weight matrix in the population. We must note that the larger the population number, the higher the performances. In these experiments, however, we set it to 256 just because of our computer resources. **(4)** Evaluation of fitness value f for each individual weight matrix is made with a set of randomly predetermined patterns ξ^μ (see preceding subsection). First we calculate the total of all the inner products (overlap) of initial states and states at each time of update not more than certain time t_0. And then these are summed up over all sets of initial patterns, i.e.,

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

In this paper, t_0 is set to 98 i.e., two times the number of neurons, and we observed that was enough. We must note that fitness 1.000 implies that all the initial patterns are stored as fixed points, while fitness less than 1.000 includes many possible cases. If we evaluated the fitness at one point of updating time instead, some solutions might be limit cycle. We adopted the above time-consuming fitness evaluation in order to avoid these oscillated solutions. **(5)** Every 40 generations, we give a perturbation to one randomly chosen bit of each pattern, and for the new set of modified patterns we evaluate fitness of each individual of the generation in the same way as above. If the highest fitness value exceeds the previous value, we replace the new patterns with old ones. This process is repeated five times for the same population. **(6)** Two parent matrices are chosen uniformly at random from the upper 40% of the population which is ranked by fitness. Then those are recombined to make one child chromosome, and an individual in the lower 60% of the population is replaced with it. This process is repeated until all the individuals in the lower subpopulation are replaced. **(7)** Recombinations are made with uniform crossover. We tested several types of crossover including one- and two-point-crossover, and observed that uniform crossover outperformed the others. Furthermore, the offspring from each crossover are mutated in times, where mutation rotates the value of randomly chosen allele in chromosome $c^{(n)}(i)$ cyclically i.e.,

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

The mutation rate is set to 0.01 in this paper. **(8)** The processes from (2) to (8) are repeated until best-so-far-fitness converges to 1.000 or generation reaches 12000. In repeating the processes, individuals in upper subpopulation (40%) survive to constitute the next generation with their offspring (60%).

These operations and parameter values were determined mainly on the basis of trial and error.

3 Results and Discussions

In our previous simulation [5], where the patterns to be stored remained fixed during evolution, the network were able to store up to 8 patterns as fixed points. In those simulations, we repeated 30 runs for 9 patterns with different random number seed, nevertheless we did not obtain the weight matrix which stored all these 9 patterns. So we concluded the associative memory capacity obtained with that method was 8 patterns for 49 neurons. On the other hand, new version of the simulation in this paper, where the patterns to be stored were also evolved occasionally, we obtained the weight matrix which stored all the 9 patterns as fixed points. Again we repeated the simulation for 10 patterns with different random number seed, however none of the best-fitted matrix did not store them as fixed points. We show the fitness vs generation for 9 and 10 patterns in Figure 2.

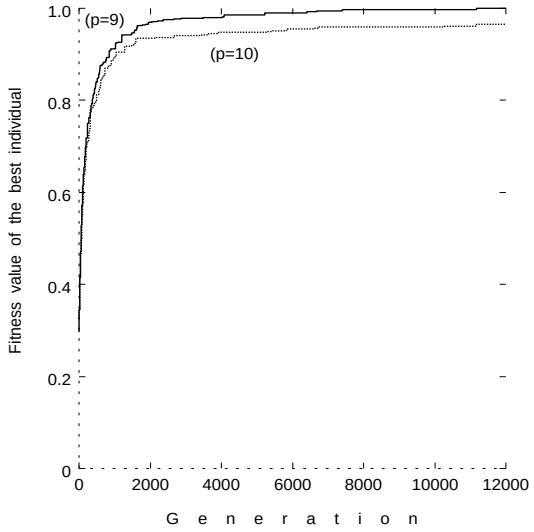

Figure 2: **Fitness vs Generation**

In this new version of the simulation, one bit of each pattern is flipped at every 40 generations, and these modified patterns are checked if they give a higher fitness value to the best-so-far individual. When this occurs, the old patterns are replaced with the new ones.

At early stages of evolution, the modified patterns tend to give individuals higher fitness value. However, as evolution proceeds the number of individual's accepting new patterns decreases asymptotically. In Figure 3 we show the number as evolution proceeds.

It shows that the effect of learning by the genetic algorithm itself in each generation is very weak. Therefore even the individual which learned those patterns most efficiently have a room to accept the slight changes in patterns at early stages. As evolution proceeds, however, accumulated effects of the learning make them difficult to accept the changes.

To ascertain this, we experimented with Hebbian matrix instead of random one as the original weight matrix

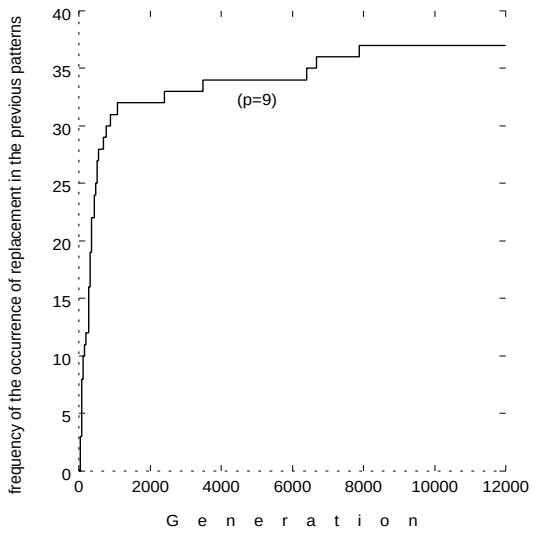

Figure 3: **Frequency of Replacement of Patterns**

W_0. However, the modified patterns were never accepted from the first generation at all (not shown in the figures in this paper), probably because Hebbian learning effect is much stronger than the one by genetic algorithm.

Some experiments in Artificial Life literatures show that changing environment in every generation enhances the evolvability of individuals (see for example [6] and [7]). In this paper a set of patterns to be given to the individual networks could be taken as an environment. We tried to change the environment in every generation. Although it works at the beginning (until around the 100-th generation), the effect is saturated afterwards without attaining fitness of 1.000 (not shown here neither).

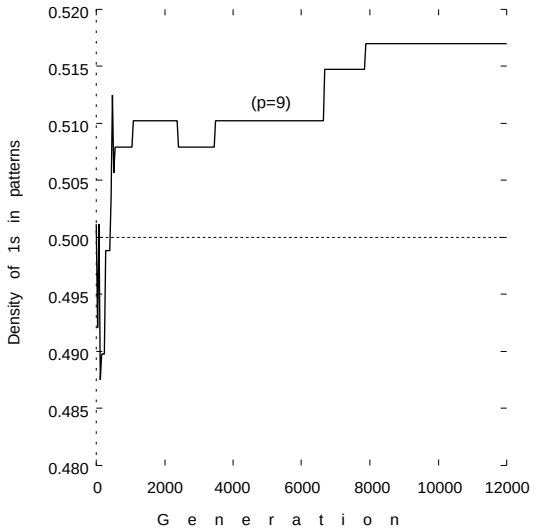

Figure 4: **Density of "1" in Patterns**

To see what happens in these replaced patterns, we counted the number of 1's in patterns which were successfully modified and adopted as new patterns. In Figure 4, we show an example of the density of 1's in the replaced patterns as a function of generation numbers. We can see in the figure that these accepted patterns tend to have more 1's than in the previous patterns. So we conjectured that the enhancement in memory capacity is due to the decrease of -1's in the patterns.

As we mentioned in the previous section, Willshow et al. [14], Palm [15] and Meunier et al. [16] showed that sparsely coded patterns enhanced the capacity of *Hebb-rule associative memory.* Hence we expected that the density of 1's would be decreased through evolution, though we do not use Hebbian learning. However, we obtained the opposite results here. Yet we do not know the reason for that.

4 Conclusions

By slightly modifying patterns occasionally in the evolution of associative memory, we obtain a little higher memory capacity than in the case of fixed patterns. Acceptance of the modified patterns occurs more frequently at early stages of evolution, and asymptotically decreases its chance as generation proceeds. The improvement in capacity is due to the increase of the number of 1's in patterns.

The enhancement in capacity in this method is not so drastic than we expected, comparing to other experiments of changing environment [6], [7]. However, from an Artificial Life point of view, these phenomena observed here might be interesting when we take the patterns as a language of the individuals. Changes in language could be more easily allowed when the culture is young and a little higher rate of "1" is more acceptable in this society. Furthermore, to consider evolvability or elaboration of language, we have to incorporate communication between individuals or species probably.

The result in this paper is just a beginning of our research, we can not draw further conjectures yet. So we will keep researching to obtain further interesting phenomena in Artificial Life context.

Acknowledgments

We thank Peter Davis at Advanced Telecommunication Research Institute (ATR). The ideas in this paper were originally derived from discussions with him.

References

[1] J. Holland (1975). *Adaptation in Natural and Artificial Systems.* The University of Michigan Press.

[2] D. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

[3] J. D. Schaffer, D. Whitley, 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* 1-37.

[4] A. Imada, and K. Araki (1995). Genetic Algorithm Enlarges the Capacity of Associative Memory. *Proceedings of 6th International Conference on Genetic Algorithms* 413-420.

[5] A. Imada, and K. Araki (1995). Mutually Connected Neural Network Can Learn Some Patterns by Means of GA. *Proceedings of the World Congress on Neural Networks* **1** 803-806.

[6] L. Zhou, and S. Franklin (1994). Character Recognition Agents. *Artificial Life IV: Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems* 301-306.

[7] T. Unemi, M. Nagayosi, N. Hirayama, T. Nade, K. Yano, and Y. Masujima (1994). Evolutionary Differentiation of Learning abilities: A Case Study on Optimizing Parameter Values in Q-Learning by a Genetic Algorithm. *Artificial Life IV: Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems* 331-336.

[8] 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.

[9] D. O. Hebb (1949). *The Organization of Behavior.* Wiley.

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

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

[12] 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.

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

[14] D. J. Willshow, O. P. Buneman, and H. C. Longuett-Higgins (1969). Nonholographic Associative Memory. *Nature* **222** 960-962.

[15] G. Palm (1980). On Associative Memory. *Biological Cybernetics* **36** 19-31.

[16] C. Meunier, H. Yanai, and S. Amari (1991). Sparsely Coded Associative Memories: Capacity and Dynamical Properties. *Network* **2** 469-487.

[17] M. H. Hassoun (Ed.) (1993). *Associative Neural Memories: Theory and Implementation.* Oxford University Press.