Baldwin Effect on the Evolution of Associative Memory

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
phone: +81-7437-2-5083

Abstract

We are applying genetic algorithms to fully connected associative memory model of Hopfield. In an earlier paper, we reported that the network can store some number of patterns only by evolving weight matrices with a simple genetic algorithm. The associative memory capacity obtained in that experiment was around 16% of the number of neurons. We have tried various ways to enlarge it. In this paper, we present that Baldwin effect on the evolution can enhance the capacity to around 39%. 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.

Key Words

Associative memory, Storage capacity, Hebbian learning, Genetic algorithm, Baldwin effect

1 Introduction

Associative memory is a process in which a complete or a noisy input of the pre-determined stored patterns results in the retrieval of it. They have an error-correcting capability because of their distributed storage of patterns among neurons, though they have a limited capacity. We are thinking of the possibility to use it as a memory organ of digital organisms of our Artificial Life research in the future. However, as a first step, we are now investigating basic behavior of the function of associative memory under simple evolutionary processes.

In 1982, Hopfield (1982) proposed a neural network model as an associative memory system. He used Hebbian rule (Hebb, 1949) to make connection matrix store a set of patterns. He estimated the storage capacity to be at most 15% of the number of neurons with computer simulations. Later the capacity was verified more analytically by Amit, Gutfreund & Sompolinsky (1987) with spin-glass theory, and by Amari & Maginu (1987) with statistical neuro-dynamical method. Since then many researchers have been trying to enlarge the storage capacity in various ways (see e.g., (Yanai, Sawada & Yoshizawa, 1991; Meunier, Yanai, & Amari, 1991; Morita, 1993)). Here we are trying that using genetic algorithms.

Genetic algorithms are a kind of searching techniques (Holland, 1975; Goldberg, 1993). Since the late 1980's, the genetic algorithms have been extensively used to search the optimal set of connection weights of neural networks and/or their optimal architectures (see (Schaffer, Whitley & Eshelman, 1992; Yao, 1993) and references quoted therein).

However, all of these researches were for layered neural networks, and applications of genetic algorithms to Hopfield networks remain few so far, probably because of the fact that Hopfield network has a fixed architecture. However, sparse connectivity has a possibility to enhance the original

associative memory capacity of Hopfield network (Yanai, Sawada & Yoshizawa, 1991). Therefore, genetic algorithms are also valid for Hopfield neural networks to explore their architectures as well as to determine their connection weights.

In an earlier papers, we showed that randomly generated connection matrices can store some number of patterns using a genetic algorithm without any learning rules (Imada & Araki, 1995a). We also showed that a genetic algorithm enlarges the capacity of Hebb-rule associative memory by pruning some connections (Imada & Araki, 1995b). In these two experiments, individuals in the population do not learn the patterns in their life time. The effect of life time learning on evolutions was first studied as a biological process by Baldwin (1986). Now this is known as Baldwin effect. As an analog of this effect, many researches concerning the combination of genetic algorithms and neural networks have addressed the relationship between learning on a population level through evolution and on an individual level during its life time. Gruau & Whitley (1993) show enhancement of performances of their neural networks by introducing Baldwin effect on the evolution (see also (Hinton & Nowlan, 1987; and Parisi & Nolfi, 1995), for example).

In this paper, we redesigned our previous genetic algorithm so that each individual in the population can learn a set of pre-determined random patterns before its fitness evaluation, i.e., Baldwin effect on the evolution. And we found that it can enlarge the storage capacity, as well as make convergence speed much faster.

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 those proposed so far, and the other is to clarify the process of this learning mechanism of the genetic algorithm.

The following section provides a brief review of the associative memory. Section 3 provides an overview of the implementation of our genetic algorithm. The results of Baldwin effect on the evolution are compared with our previous results of no Baldwin version in Section 4. Finally some conclusions are given.

2 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 by

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

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 p patterns, each connection weight has to be adequately adjusted. If we use Hebbian rule, for example, the weight w_{ij} is determined as follows:

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

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 attractor*. 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 stored patterns which is given small noise within the size of basin of attractor, this input pattern relaxes to the original stored pattern after several steps of the update. In this simulation, however, we do not give any noise to input pattern. In this case, when one of a 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.

To know more about associative memory, see for example (Hassoun, 1993).

3 Implementation of our genetic algorithm

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

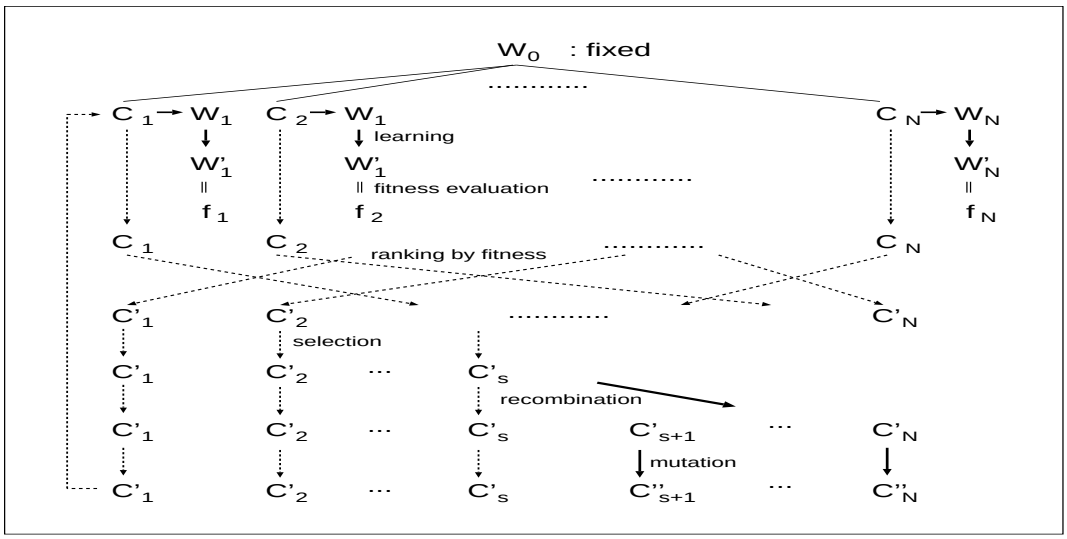

Figure 1: Schematic representation of our genetic algorithm.

(1) At first a weight matrix W_0(49x49) [1] is produced randomly so that each component of the matrix w_{ij} is 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 these alleles are chosen randomly from $\{-1, 0, 1\}$, where the probability of choosing either -1 or 0 is set to 0.02 here. As is mentioned in procedure(3) below, 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

[1] As we will describe later, we simulate with 49 neurons of Hopfield network.

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:

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

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)** Each weight matrix learns a set of pre-determined random patterns ξ^μ by Hebbian learning rule as follows.

$$w_{ij} = w_{ij} + \lambda \sum_{\mu=1}^{p} \xi_i^\mu \xi_{j\neq i}^\mu.$$

We set the value of λ to 0.4 here, which was found to work well with some experiments. These Hebbian learned weight matrices are used only for fitness evaluation in procedure(5) below, and do not affect their chromosomes. Hence This is a Baldwin effect on the evolution. **(5)** Evaluation of fitness value f for each individual weight matrix is made with a set of randomly pre-determined 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 time-consuming fitness evaluation above in order to avoid these oscillated solutions. **(6)** Two parent chromosomes are chosen to be recombined uniformly at random from the upper 40% of the population which is ranked by fitness. **(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 each mutated in times, where mutation rotates the value of randomly chosen bit 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)** Unless highest fitness value reaches the value of 1.000 nor generation exceeds 12000, the individuals in upper subpopulation (40%) survive to constitute the next generation with their offspring(60%), and the processes from (2) to (8) are repeated.

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

4 Results and Discussions

To see the Baldwin effect on the evolution of associative memory capacity, we show the results of two preliminary experiments, at first. One is evolution from randomly determined weight matrix without using any learning algorithms, and the other is evolution from Hebbian learned weight matrix. And then we will describe the Baldwin effect on the evolution of random weight matrix using Hebbian rule as a life time learning of each individual. All these simulations were carried out on networks with 49 neurons.

Evolution from random weight matrix

Genetic algorithm per se has an ability to make a random weight matrix store certain number of patterns. In fact, we succeeded in obtaining the weight matrix which can store 8 patterns (16%

of the number of neurons) as fixed points by evolving randomly determined weight matrix (Imada & Araki, 1995b). The genetic algorithm used in this experiment was similar to the one described in previous section except for skipping the Hebbian learning process before fitness evaluation. In Figure 2(left), we show the results of best-so-far fitness vs generation for 4, 6, 8, and 9 patterns. These are samples of fastest convergence among several runs with different random number seed, or samples of maximum number of stored patterns as fixed points when it does not converge. [2] For 9 patterns, we did not obtain the weight matrix which stores all these patterns perfectly as fixed points, even after repeating 30 runs with different random number seed. (Among these runs, the maximum number of patterns stored as fixed points was 8.)

Though this capability of storing 8 patterns with 49 neurons is almost the same as in the original Hebb-rule associative memory of Hopfield [3], the structure of weight matrix in both cases is very different from each other. Hebbian learned weight matrix in the Hopfield model is symmetric and all the neurons are fully connected except for itself. [4] In the case of this experiment, on the other hand, evolutions are started from a totally asymmetric weight matrix with no zero components, and during the evolution, symmetry is recovered to some extent and some components become zero. We must note that the zero components introduced by the genetic algorithm imply to prune the connections between neurons. In Figure 2(right), we show zero-density and degree-of-symmetry [5] of the weight matrix which eventually stores 9 patterns. As shown in the figure, both the zero-density and the degree-of-symmetry are increasing from zero and asymptotically approach to the value of approximately 0.2. We conjectured that both these recovery of symmetry and pruning some of the connections play an important roll to store the patterns effectively.

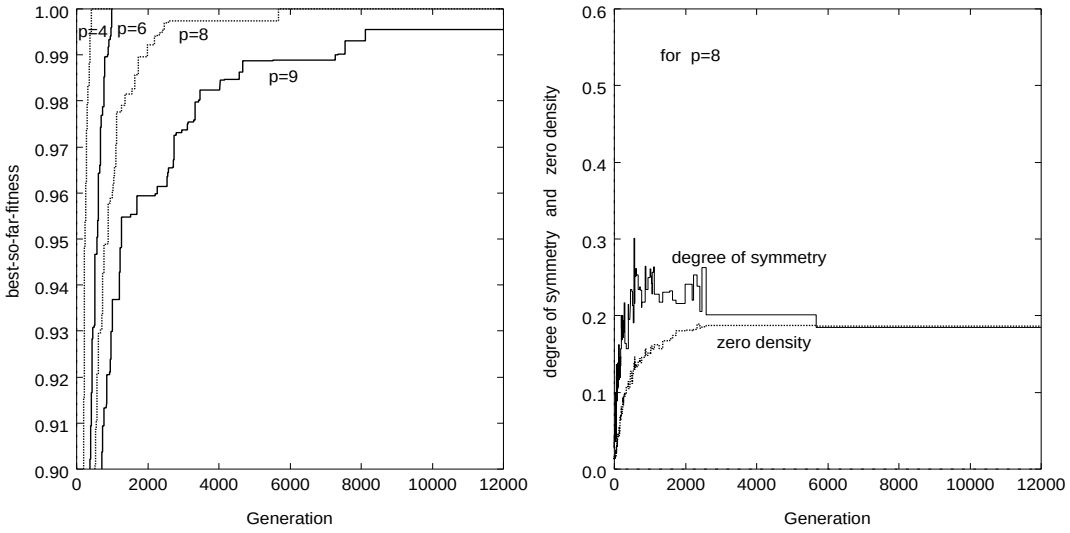

Figure 2: Evolution departing from *random matrix*. Fitness vs Generation(left), zero-density and degree of symmetry(right).

Evolution from Hebbian learned weight matrix

Since we use Hebbian learning rule as a life time learning in the simulation of Baldwin effect, it is interesting to see whether Hebbian learned matrix itself can increase its capacity through evolution.

[2] Hereafter the same way is used to represent the fitness vs generation curves.

[3] According to our experiment, it stored the maximum of 9 patterns as fixed points. This is somewhat larger than the original estimate by Hopfield and later study of its capacity though, we use this as a base of comparison since our objective here is to obtain the maximum performance among many runs.

[4] We must note that when the number of patterns is even, some components could be zero as a result of Hebbian learning.

[5] It is defined as $\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}w_{ji} / \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^2$.

As we mentioned above, Hebbian learned weight matrix originally can store 9 patterns at most. We applied a genetic algorithm to the Hebbian learned weight matrix (Imada & Araki, 1995b). The genetic algorithm was the same as Figure 1 except that W_0 is the Hebbian learned matrix, and again the matrices in the population skip the learning process before fitness evaluation. As a result, we were able to enlarge the capacity to 16 patterns (33% of the number of neurons) through the evolution. In Figure 3(left), we show the best-so-far-fitness-curve for 14, 16 and 17 patterns. Here also we run the simulations 30 times for 17 patterns, however we did not obtain matrix which has perfect fitness. (The maximum number of patterns stored as fixed points was 14.)

The resultant weight matrix are totally different from the one obtained in the evolution

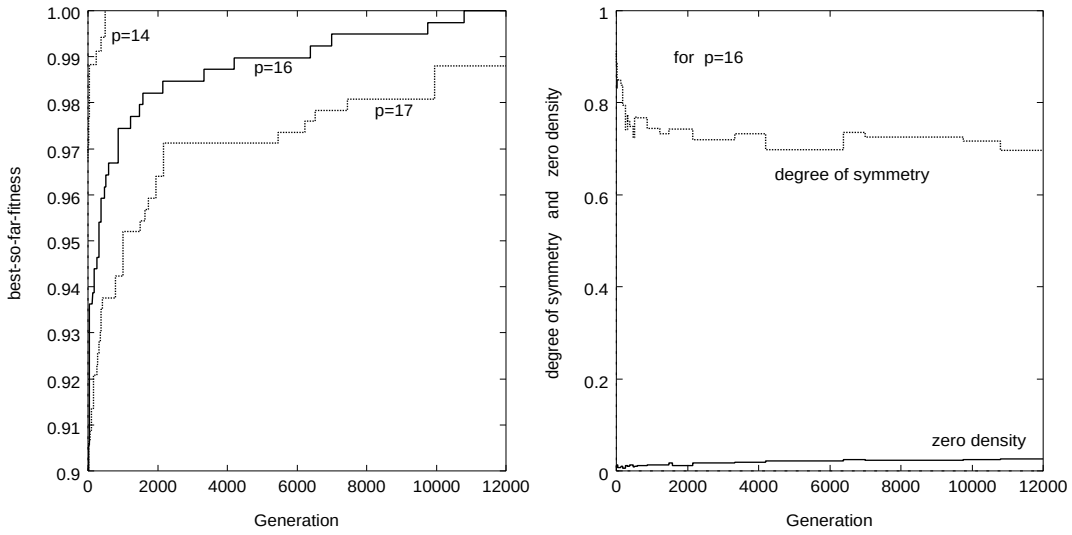

Figure 3: Evolution departing from *Hebbian matrix*. Fitness vs Generation(left), zero-density and degree of symmetry(right).

starting from random matrix above. In Figure 3(right), we show the zero-density [6] and the degree-of-symmetry of the weight matrix which eventually stores 16 patterns. As shown in the figure, the matrix obtained in this experiment is more symmetric, and the genetic algorithm prune much fewer connections than in the experiment described above. The degree of symmetry and the zero density approaches to the value of approximately 0.7 and 0.03, respectively.

Baldwin effect on the evolution

In Figure 4(left), we show the best-so-far-fitness-curve of the Baldwin evolution from random matrix for 17, 19 and 20 patterns. For 20 patterns, we run the simulations for 30 times as before, but number of patterns stored as fixed points were 17 at most. Therefore we might conclude that the Baldwin effect on the evolution enhances the capacity to 19 patterns (39% of the number of neurons). Although the increase in capacity is not so drastic if we compare it to the evolution starting from Hebbian learned matrix above, the speed of convergence is tremendously improved in Baldwin version of the simulation. For example, in the evolution from Hebbian learned matrix for 16 patterns, fitness of 1.000 was obtained at the 10795-*th* generation (see Figure 3(left)), while in the Baldwin version for 17 patterns, it was obtained just at the 996-*th* generation (see Figure 4(left)).

Since learning of the patterns affects each individual only in its life time in this experiment, the property of Hebbian learning does not change as generation proceeds. As in Figure 4(right), the degree of symmetry remains unchanged from the beginning (around 0.8). This asymmetry is due

[6] As this matrix learns even number of patterns, some of its components are zero as a result of Hebbian learning. Here we show only the density of zero introduced by the genetic operators.

to -1, 0, and 1 randomly assigned in each matrix before the life time learning. We must note that the component of each weight matrix in the population never be zero after individual's learning in this third experiment.

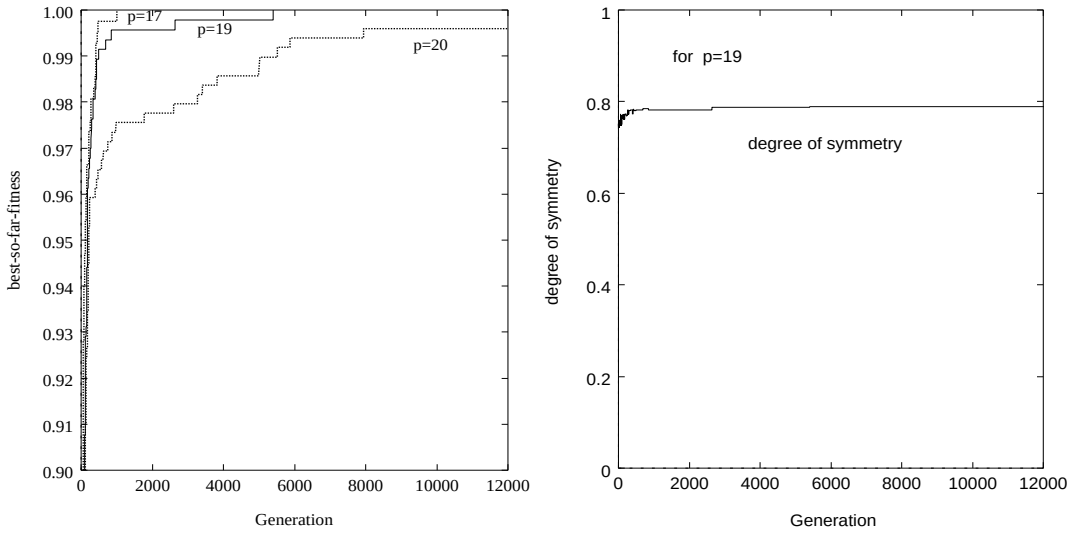

Figure 4: *Baldwin effect.* Fitness vs Generation(left), degree of symmetry(right).

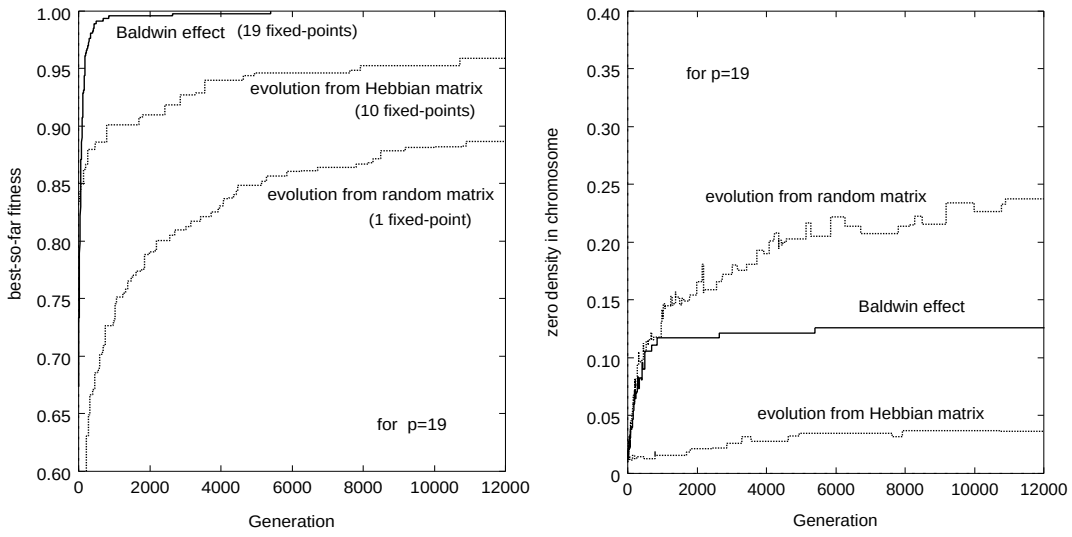

Figure 5: Comparative plot of *Baldwin effect* and departures from *random/Hebbian matrix.* Fitness vs Generation(left), zero-density in the chromosome(right).

For further comparison of the evolution under Baldwin effect with simple evolution from both Hebbian matrix and random matrix, we show each case of best-so-far-fitness vs generation for the same set of 19 patterns in Figure 5(left). We can see from the figure that evolution under Baldwin effect outperforms the other two. In the evolution under Baldwin effect, the weight matrix which store all these 19 patterns as fixed points emerged at the 5408-*th* generation. On the other hand, in the case of evolution from Hebbian matrix and random matrix never converge, and final number

of patterns stored as fixed points are 10 and 1, respectively.

We conjecture that the evolutionary pressures in these genetic algorithms are different from each other. In the Baldwin version, the pressure works to enhance learnability rather than to increase capacity directly. That is, individuals which have higher ability to learn tend to survive. We can see it, for example, in the assignment of zeros in their chromosomes during an evolution. For these three cases, we show the zero density in chromosome of best-so-far individual against generation in Figure 5(left). We can see significant difference among them.

As was mentioned in earlier section, we evaluate the ability of weight matrix by the upper bound of the number of patterns stored as fixed points. We summarize the capacity of those cases described so far in Table 1. The figures in this table denote the ratio of the number of patterns stored as fixed points to the total number of patterns given to the network. We can see the capacity of the matrices of pure Hebbian learning, those evolved from random/Hebbian matrix, and those resulted from Baldwin effect are 9, 8, 16, 19, respectively.

Table 1: Ratios of patterns stored as fixed points.

p	8	9	10	$\cdots$	16	17	$\cdots$	19	20
random + GA	1.000	0.889	—	—	—	—	—	—	—
without GA	—	1.000	0.999	—	—	—	—	—	—
Hebbian + GA	—	—	—	—	1.000	0.824	—	—	—
Baldwin effect	—	—	—	—	—	—	—	1.000	0.850

In this experiment of Baldwin effect, the results of learning of individuals do not change their chromosomes, but only affect the selection after fitness evaluation. However, it is reported that incorporation of the results of learning into chromosomes also enhances the performance of genetic algorithms (see (Gruau & Whitley, 1993; Whitley, Gordon & Mathias, 1994; Bartscht, Engel & Müler-Schloer, 1995), for example). This is known to be Lamarckian inheritance. We are also investigating the effect of Lamarckian inheritance on the evolution of associative memory. Since in the implementation of genetic algorithm in this paper, it is hard to incorporate the results of learning to chromosomes, we use a different implementation where the chromosomes are made up of components of the weight matrix (Imada & Araki, 1996a). And we obtained almost the similar capacity to the result in this paper (Imada & Araki, 1996b).

5 Conclusions

The simulations in this paper were carried out with 49 neurons of fully connected neural networks. This size of Hebbian-learning associative memory network stores at most 9 patterns as fixed points. The no-learning version of our genetic algorithm evolved the network which eventually stores 8 patterns (16% of the number of neurons) starting from random matrix. On the other hand, each individual's Hebbian learning of the patterns in every generation before its fitness evaluation (Baldwin effect) results in the matrix which store 19 patterns (39%) eventually. We conjecture that this is due to the evolution which enhances the individual's learning ability rather than the Hebbian learning itself. Part of the reason for this is that the genetic algorithm without Baldwin effect has only an ability to evolve the Hebbian matrix to store 16 patterns (33%). In addition, the structure of chromosome of the Baldwin evolution is very different from other two types of evolution which directly try to search the networks of higher storage capacity. However the more precise reasons for this phenomena are still an open question. We leave this question to future research.

Acknowledgments

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

Reference

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

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

Baldwin, B. J. (1986). A New Factor in Evolution. *The American Naturalist* **30**, 441-451 and 536-553.

Bartscht, E., Engel, J., and Müler-Schloer, C. (1995). The Investigation of Lamarckian Inheritance with Classifier Systems in a Massively Parallel Simulation Environment. *Proceedings of 3rd European Conference on Artificial Life.* 920-933.

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

Gruau, F., Whitley, D. (1993). Adding learning to the cellular Development of Neural Networks: Evolution and the Baldwin Effect. *Evolutionary Computation*, **1**(3), 213-233.

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

Hebb, D. O. (1949). *The Organization of Behavior.* Wiley.

Hinton, G. E., and Nowlan, S. J. (1987). How Learning Guides Evolution. *Complex System*, **1**, 495-502.

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

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

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

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

Imada, A., and Araki, K. (1996a). Simple Genetic Algorithm as a Learning Method for Associative Memory. (to submit)

Imada, A., and Araki, K. (1996b). Lamarckian Evolution of Associative Memory. (to submit)

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

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

Parisi, D., and Nolfi, S. (1995). How Learning can Influence Evolution within a Non-Lamarckian Framework. *Plastic Individuals in Evolving Populations.* Santa Fe Institute Series, Addison-Wesley.

Schaffer, J. D., Whitley, D., and Eshelman, L. J. (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.

Whitley, D., Gordon, V. S., and Mathias, K. (1994). Lamarckian Evolution, the Baldwin Effect and Function Optimization. *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature.* 6-15.

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

Yao, X. (1993). A Review of Evolutionary Artificial Neural Networks. *International Journal of Intelligent Systems* **8**, 539-567.