Basin of Attraction of Hebb-rule Associative Memory as it Evolves

Akira Imada and Keijiro Araki

Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-01 Japan
phone: +81-7437-2-5083
email: akira-i@is.aist-nara.ac.jp and araki@is.aist-nara.ac.jp

Abstract. We are applying genetic algorithms to fully connected neural network model of associative memory. In an earlier paper, we reported that a Genetic Algorithm can enlarge storage capacity of Hebb-rule associative memory. The associative memory capacity obtained in that experiment was around 33% of the number of neurons, while capacity of the original Hebb-rule associative memory is 15% at most. However the size of basin of attraction obtained was rather small compared to the original one. In this paper, we present a new version of the previous work trying to control the basin size. As far as we know, this is the first attempt to address the size of basin of attraction of associative memory by evolutionary processes.

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. 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 preliminary step, we are now investigating basic behavior of the function of associative memory under simple evolutionary processes.

In 1982, Hopfield [1] proposed a neural network model as an associative memory system. He used Hebbian rule [2] to make the network store a set of patterns. He estimated the storage capacity to be at most 15% of the number of neurons with computer simulations. Since then many researchers have been trying to enlarge the storage capacity in various ways (see e.g., [3], [4]).

However, the enlargement of capacity spoils the error correcting capability of the network. The size of basin of attraction decreases as the number of storage patterns increases. In fact, Personnaz et al. [5] shows that the basin of attraction of a stored memory falls sharply as the number of patterns approaches to a half of the number of neurons for auto-associative memory. In this paper we explore the trade-off as storage capacity vs basin size with using a genetic algorithm.

Genetic algorithms are kinds of searching techniques ([6], [7]). Since the late 1980's, genetic algorithms have been extensively used to search the optimal

set of connection weights of neural networks and/or their optimal architectures (see [8], [9], and references quoted therein). However all of these researches were for layered neural networks, and applications of genetic algorithm 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 [3]. 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 paper, we showed that randomly generated connection matrices evolve to store some number of patterns using a genetic algorithm without any learning rules [10]. We also showed that the genetic algorithm enlarges the capacity of Hebb-rule associative memory by pruning some connections [11]. In these two experiments, we did not refer to the basin of attraction of the associative memory. In this paper, we redesign the genetic algorithm to try to control the size of basin of attraction.

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 mechanism of this *learning on a population level* of the genetic algorithm.

The following section provides an overview of the implementation of our genetic algorithm. Then results of the new implementation concerning the basin size are described in Section 3. Finally some conclusions are given.

2 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:

(1) At first, a weight matrix W_0 are determined in certain way. In this paper,

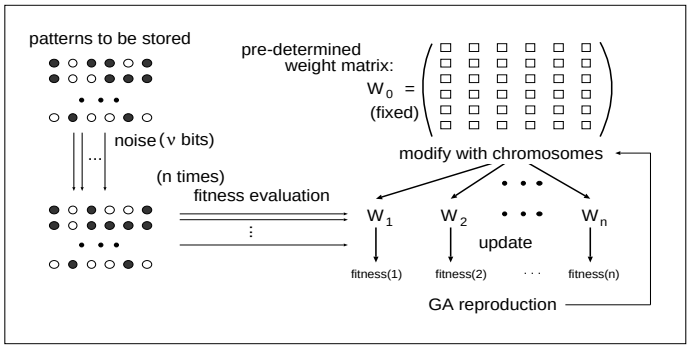

Fig. 1. Schematic representation of our genetic algorithm.

it is determined by Hebbian rule, i.e.,

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

where, ξ_i^μ is the i-th bit of the μ-th pattern which is to be given to the network. As we simulate with 49 neurons of Hopfield network, the size of W_0 is 49x49. This original weight matrix remains unchanged during evolution. **(2)** Then 256 chromosomes are produced 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 will be 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 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)** To evaluate fitness value f, ν random bits of each stored patterns are flipped at first. Then each inner product of one of stored patterns and updated neuron-states after giving its noisy input is averaged over every time step up to certain time t_0 and over all the patterns. This process is repeated for n different combinations of the ν-bit-noisy inputs and averaged over them. That is,

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

where $s_{j\nu k}^\mu(t)$ is the state of the *j-th* neuron at time t when the *k-th* combination of ν-bit-noisy μ-*th* pattern is given to the network. In this paper, t_0 is set to 98, i.e., twice 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)** Two parent chromosomes are chosen uniformly at random from 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. **(6)** Recombinations are made with uniform-crossover. We tested several types of crossover including one-

and two-point-crossover, and observed that the 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.,

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

The mutation rate is set to 0.01 in this paper. **(7)** 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 (7) are repeated.

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

3 Results and Discussions

In this section, we present the effect of fitness evaluation with noisy inputs. All these simulations were carried out on networks with 49 neurons, and the number of patterns to be stored remains fixed at eleven in this paper.

In Figure 2, we show the best-so-far-fitness vs generation resulted from fitness evaluation with noisy input. This is a sample from the evolutions starting with the weight matrix which learns a set of 11 patterns by Hebbian rule in advance. Though this number of patterns to be stored is more than the storage capacity of original Hebb-rule associative memory, the resultant best-fit individual in the 12000-*th* generation stores all these 11 patterns as fixed points. It is not surprising that we obtain 100% correct individual in spite of the fact that the fitness never attains the value of 1.000 in the evolution, since these are averaged value over all repetition of noisy inputs. Note that the best-so-far-individual is not necessarily the best-individual in the next generation even under the elitist strategy as in this paper. In this experiment, fitness values are evaluated with 8 (= n) different combinations of inputs added 6 (= ν) bits random noises. The number of noisy bits ν are set to 6 according to our preliminary experiments (not shown in this paper).

These experiments raise our next interest as to whether fitness evaluation with noisy input enlarges the size of basin of attraction. In Figure 3, we show the degree of tolerance to noisy input with different number of repetition n for 6 (= ν) bits of noisy input, by plotting the similarity of updated neuron-states after giving noisy input to its complete pattern as a function of number of noisy bits given to the input pattern. The network starts out with an initial configuration of neuron-states (-1's and 1's) and this configuration changes in discrete time steps according to asynchronous update. When one of noisy input of stored patterns is given to the network, we obtain a configuration of the neuron-states after updating long enough time steps (= 98 here). And this is compared to the initial configuration of complete version of its input. The comparison is made by using cosine of the angle between two vectors which represent the two configurations above. These are averaged both over all inputs of stored patterns and over several runs (= 800 here). To be more specific, all the stored pattern ξ^{μ}

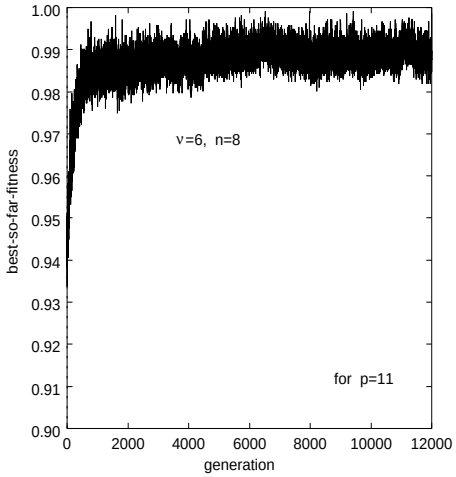

Fig. 2. Fitness vs generation for fitness evaluation with noisy input.

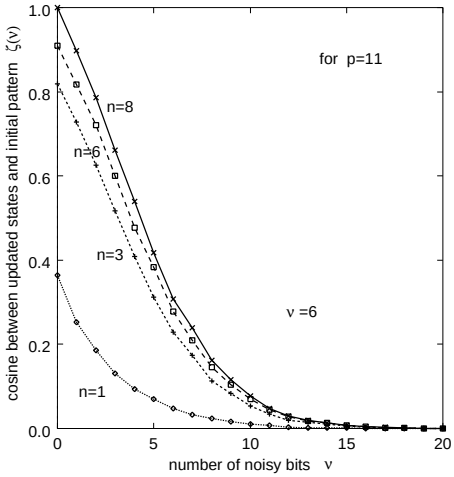

Fig. 3. Similarity of updated states for noisy input to its corresponding correct pattern.

are added ν bits of noise randomly and given to the network to test the tolerance to them. These are repeated 800 times here, and the state of the j-th neuron at time t when the k-th set out of these 800 combinations of ν-bit-noisy μ-th pattern is given to the network is denoted as $s^{\mu}_{j\nu k}(t)$. Then the similarity $\zeta(\nu)$ is defined by

$$\zeta(\nu) = \frac{1}{800 \cdot p \cdot 49} \sum_{k=1}^{800} \sum_{\mu=1}^{p} \sum_{j=1}^{49} \xi^{\mu}_{j} \cdot s^{\mu}_{j\nu k}(98).$$

In general, if the curve moves toward right side it indicates we obtain broader size of basin of attraction. As shown in the figure, we can enlarge the size of basin by increasing the number of repetition n. For more than 8 repetitions, however, we can not see the performance enhancement any more. And in addition, the enhancement of the size of basin are only slight compared to our previous results of no-noisy counterpart of the genetic algorithm for p=11.

We also run this noisy version of the genetic algorithm starting with Hebbian learned weight matrix for the number of patterns from 6 to 16, however we have obtained only a few 100% correct individuals to date (not shown in this paper).

4 Conclusions

In this paper, we have described an application of a genetic algorithm to fully connected neural network model of associative memory. The feature of the genetic algorithm is to use noisy input of the storage patterns in evaluating fitness

values. In this implementation, it is very hard for the best-fit individual in the population to converge to 100% correct solution, compared to our previous implementation where fitness is evaluated with no-noisy input. One of the goals of this paper, however, is to enlarge the size of basin of attraction of the network, and this was partly successful.

Though the results reported are not so satisfactory, these are only the beginning of our continuing research project. There might be other implementations to control the size of basin of attraction more effectively, which we are now exploring. And in addition, we are also investigating the evolution starting with random weight matrix instead of Hebbian learned weight matrix.

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. Hopfield, J. J.: Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences, USA* **79** (1982) 2554–2558
2. Hebb, D. O.: *The Organization of Behavior* Wiley (1949)
3. Yanai, H., Sawada, Y, Yoshizawa, S.: Dynamics of an Auto-Associative Neural Network Model with Arbitrary Connectivity and Noise in the Threshold. *Network* **2**(3) (1991) 295–314
4. Morita, M.: Associative Memory with Nonmonotone Dynamics. *Neural Networks* **6** (1993) 115–126
5. Personnaz, L., Guyon, I., Dreyfus, G.: Collective Computational Properties of Neural Networks: New Learning Mechanisms. *Physical Review* A**34**(5) (1986) 4217–4227
6. Holland, J.: *Adaptation in Natural and Artificial Systems* The University of Michigan Press (1975)
7. Goldberg, D.: *Genetic Algorithms in Search, Optimization, and Machine Learning* Addison-Wesley (1989)
8. Schaffer, J. D., Whitley, D., Eshelman, L. J.: 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* (1992) 1–37
9. Yao, X.: A Review of Evolutionary Artificial Neural Networks. *International Journal of Intelligent Systems* **8** (1993) 539–567
10. Imada, A., Araki, K.: Mutually Connected Neural Network Can Learn Some Patterns by Means of GA. *Proceedings of the World Congress on Neural Networks* **1** (1995) 803–806
11. Imada, I., Araki, K.: Genetic Algorithm Enlarges the Capacity of Associative Memory. *Proceedings of 6th International Conference on Genetic Algorithms* (1995) 413–420