Basin of Attraction of Associative Memory as it Evolves from Random Weights

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

Araki, Keijiro
Department of Computer Science and Computer Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University

Basin of Attraction of Associative Memory as it Evolves from Random Weights

Akira Imada[1] and Keijiro Araki[2]

[1] Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-01 Japan
akira-i@is.aist-nara.ac.jp

[2] Department of Computer Science and Computer Engineering
Graduate School of Information Science and Electrical Engineering
Kyusyu University
6-1 Kasuga-koen, Kasuga, Fukuoka, 816 Japan
araki@c.csce.kyusyu-u.ac.jp

Abstract. We apply genetic algorithms to fully connected Hopfield associative memory networks. Previously, we reported that a genetic algorithm can evolve networks with random synaptic weights to store some number of patterns by pruning some of its synapses. The associative memory capacity obtained in that experiment was around 16% of the number of neurons. However the size of basin of attraction was rather small compared to the original Hebb-rule associative memory. In this paper, we present a new version of the previous method 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 recall is a process in which an incomplete or a noisy input of a stored pattern results in the retrieval of its complete pattern. The error-correcting capability is due to their distributed storage of patterns among neurons, though they have a limited capacity. We are investigating basic behaviors of the associative memory under simple evolutionary processes.

In 1982, Hopfield[1] proposed a neural network model of an associative memory system. He specified synaptic weights by Hebbian rule[2] to store a set of random bipolar(± 1) 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]).

Within the capacity, *all* the incorrect inputs at a distance ρ ($\rho \in (0, 1/2)$) from each of the N-bit stored patterns are corrected (McElice et al.[5], Personaz[6]), which implies that each stored pattern has a spherical domain of attraction of

same radius ρN (Pancha et al.[7]). However, as p increases, the size of the basin of attraction guaranteed by the Hebbian rule becomes smaller (see, e.g., [8]). It could be optimized so that the radius of an attraction domain will be extended to a half of the distance from its nearest neighbor (in N-dimensional space). In this paper we explore the trade-off between storage capacity and basin size with using a genetic algorithm.

Genetic algorithms are a class of global search algorithm([10], [11]). Since the late 1980's, genetic algorithms have been used extensively to search the optimal set of synaptic weights of neural networks and/or their optimal architectures (see [12], [13], 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 networks have a fixed architecture. However, sparse connectivity has a possibility to enhance the original associative memory capacity of Hopfield network[3]. Therefore, genetic algorithms can be used for Hopfield neural networks to explore their architectures as well as to determine their synaptic weights.

Previously, we showed that randomly generated weight matrices evolve to store some number of patterns using a genetic algorithm without any learning rules[14]. We also showed that the genetic algorithm enlarges the capacity of Hebb-rule associative memory[15]. The genetic algorithm was based on pruning some synaptic weights of the networks. In these two reports, 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 to understand the mechanism of the *learning* under evolutionary processes, rather than to use the genetic algorithm as a more effective learning method than those proposed so far.

The following section provides a brief review of associative memory. Section 3 provides an overview of our genetic algorithm implementation. Then results of the new implementation on the basin size are described in Section 4.

2 Associative memory

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

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

where ξ_i^{μ} is the i-th bit of the μ-th pattern. To store these patterns, each connection weight has to be appropriately specified. Under Hebbian learning, 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 an 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 $i\text{-}th$ 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. 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 more frequent and finally they become 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 $i\text{-}th$ neuron at time t when the $\mu\text{-}th$ 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. Each neuron is chosen according to pre-assigned order (once per cycle), instead of chosen randomly, at each step of updating.

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 stored as *fixed points*. We evaluate the storage capacity by the upper bound of the number of patterns stored as fixed points. However the goal in this paper is to obtain larger size of basins rather than the higher capacity.

A comprehensive review of associative memory can be found in [9].

3 Genetic Algorithm Implementation

The genetic algorithm used in this study is as follows. **(1)** A weight matrix W_0 is generated in certain way. In this paper, we use a weight matrix obtained from the previous version of our genetic algorithm (no noisy-input) in which random weights eventually evolve to store all the given patterns perfectly as fixed points. As we simulate with 49 neurons of Hopfield network, the size of W_0 is 49×49. This original weight matrix remains unchanged during the evolution. **(2)** Then 256 chromosomes are generated randomly. The chromosome has a fixed length of $2401(=49\times49)$ 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. As will be mentioned in procedure(3) below, each component of the original weight matrix w_{ij} is multiplied by one of these alleles. Allelic value of 0 corresponds to pruning the connection to which it 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 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. In each generation, this produces 256 weight matrices slightly different from W_0. We note that the larger the population number, the higher the performances. Due to computational resource's limits, however, we set it to 256. **(4)** To evaluate fitness value f, randomly chosen ν bits of each stored pattern are flipped. Then they are given to the network and updated. Each inner product of the updated state and its corresponding stored pattern is averaged over all time steps 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 implies that all the noisy patterns are retrieved correctly, while fitness less than 1 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 the oscillatory 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 are mutated in time with probability 0.01 (mutation rate), where mutation rotates the value of randomly chosen allele in chromosome $c^{(n)}(i)$ cyclically, i.e.,

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

(7) Unless highest fitness value reaches the value of 1 nor generation exceeds 12000, 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 empirically.

4 Results and Discussions

In this section, we compare the results of the fitness evaluation using noisy inputs with the one obtained from our previous no-noisy version. All these simulations were carried out on networks consisting of 49 neurons.

In Figure 1, we show the best-fitness vs. generation resulted from fitness evaluation with noisy/no-noisy input. The dotted line in the figure is the representative sample from no-noisy version of the genetic algorithm. A network with random weight is evolved and can store eventually a maximum of 8 patterns as fixed points at the 6449-*th* generation. Then this matrix is used as W_0 in noisy counterpart of our genetic algorithm. The results are shown with solid line in the figure. The perfect solution emerged at the 6369-*th* generation. This matrix also stores the above 8 patterns as fixed points. In this latter experiment, we evaluate fitness by averaging over 20 repetitions giving 5 random noises each time. Note that in this case, the best-individual will not be necessarily the best in the next generation even under the elitist strategy. We also run this noisy version starting with random weight matrix instead, however we have not obtained 100% correct individuals to date (not shown in the figure).

This fitness evaluation with noisy input is designed to enlarge the size of basin of attraction. In Figure 2, we show the degree of tolerance to noisy input of the two networks appeared in Figure 1, together with the original Hebb-rule associative network. In the figure, we plot the similarity of updated neuron-states for the noisy input to its complete pattern as a function of number of noisy bits given to the input. Network starts out with an initial configuration of neuron-states, 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 are to obtain a configuration of the neuron-states after updating 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

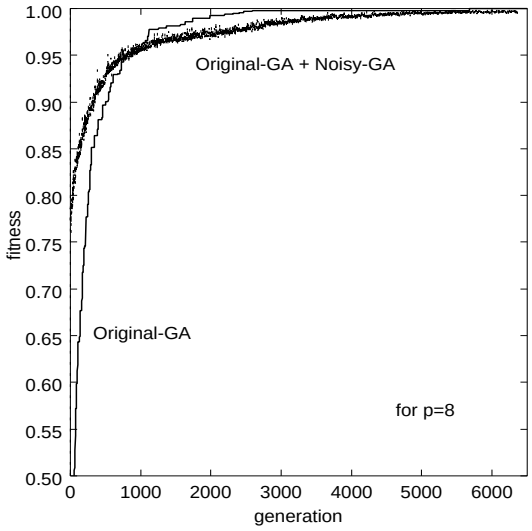

Fig. 1. Fitness vs. generation for fitness evaluation with noisy/no-noisy input.

more specific, each of the stored pattern ξ^μ is added ν bits of noise randomly, and given to the network to test the tolerance to the noise. This is repeated 800 times. Then the similarity $\zeta(\nu)$ is defined by

$$\zeta(\nu) = \frac{1}{800 \cdot p \cdot 49} \sum_{k=1}^{800} \sum_{\mu=1}^{p} \sum_{j=1}^{49} \xi_j^\mu \cdot s_{j\nu k}^\mu(98),$$

where $s_{j\nu k}^\mu(t)$ is 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. The slower the decay of the curve, the broader the size of basin of attraction. As we can see in the figure, the decay of the curve for the network obtained from no-noisy version is extremely steep. It implies that although the genetic algorithm can evolve a random matrix to store all the 8 patterns as fixed points, the size of basin of attraction is much smaller than that of the original Hebb-rule associative memory. However, genetic algorithm with noisy input in evaluating fitness improves the size significantly, though it is still smaller than the original Hebb-rule associative memory.

5 Conclusions

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

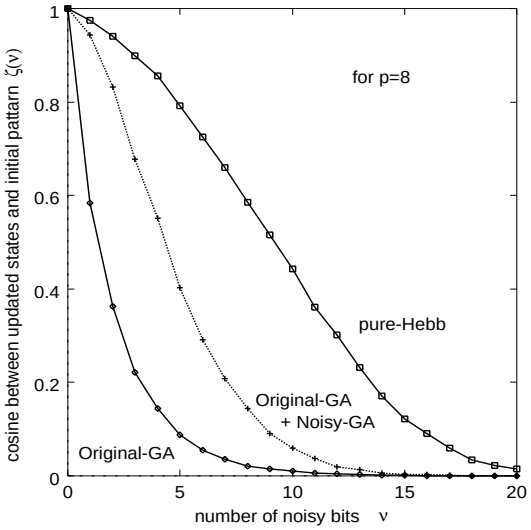

Fig. 2. Similarity of updated states of a noisy input to its corresponding memory.

fitness values. The goal is to enlarge the size of basin of attraction of the network. Unfortunately, we found that 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. However, we are partly successful in enlarging the small basin size obtained in the previous implementation by approximately a half of the size of the original Hebb-rule associative memory while maintaining its storage capacity.

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 exploring the evolution starting with Hebbian learned weight matrix instead of random weight matrix.

Finally, we are also thinking of using this Hopfield model of associative memory as a test function of multi-modality as well as an example of a multi-objective function.

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, and 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. McEliece R. J., Posner E. C., Rodemick E. R., and Venkatesh S. S., The Capacity of the Hofpield Associative Memory. *IEEE Trans. Information Theory* **IT-33**, (1987) 461–482.
6. Personnaz, L., Guyon, I., and Dreyfus, G., Collective Computational Properties of Neural Networks: New Learning Mechanisms. *Physical Review* A**34**(5) (1986) 4217–4227.
7. Pancha G., Venkatesh S. S., Feature and Memory-Selective Error Correction in Neural Associative Memory. in M. H. Hassoun (eds.) *Associative Neural Memories: Theory and Implementation.*, Oxford University Press, (1993) 225–238.
8. Gardner, E., The Phase Space of Interactions in Neural Network Models. *Journal of Physics* A: Math. Gen., **21A** (1988) 257–270.
9. Hassoun, M. H. (eds.), *Associative Neural Memories: Theory and Implementation.* Oxford University Press (1993)
10. Holland, J., *Adaptation in Natural and Artificial Systems.* The University of Michigan Press (1975)
11. Goldberg, D., *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley (1989)
12. Schaffer, J. D., Whitley, D., and 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.
13. Yao, X., A Review of Evolutionary Artificial Neural Networks. *International Journal of Intelligent Systems* **8** (1993) 539–567.
14. Imada, A., and 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.
15. Imada, A., and Araki, K., Genetic Algorithm Enlarges the Capacity of Associative Memory. *Proceedings of 6th International Conference on Genetic Algorithms* (1995) 413–420.

This article was processed using the LaTeX macro package with LLNCS style