Evolved Asymmetry and Dilution of Random Synaptic Weights in Hopfield Network Turn a Spin-glass Phase into Associative Memory

Akira Imada
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

Keijiro Araki
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 evolutionary computations to Hopfield's neural network model of associative memory. Previously, we reported that a genetic algorithm can enlarge the Hebb rule associative memory by pruning some of over-loaded Hebbian synaptic weights. In this paper, we present the genetic algorithm also evolves *random* synaptic weights to store some number of patterns.

1 Introduction

There have been a lot of researches which apply evolutionary techniques to layered neural networks (see e.g., [1, 2] and references quoted therein). However, their applications to fully-connected recurrent neural networks remain few so far. In this paper, we present some basic behaviors of the Hopfield model of associative memory under simple evolutionary processes.

Associative memory is a dynamical system which has a number of stable states with a domain of attraction around them [3]. Applying a vector $\boldsymbol{x}_0$ as the initial state of the dynamical system results in the state trajectory $\{\boldsymbol{x}_t\}_{t\geq 0}$. If we start the system with $\boldsymbol{x}_0 = \boldsymbol{u} + \boldsymbol{\delta u}$ in the vicinity of some fixed point $\boldsymbol{u}$, and the state trajectory converges to $\boldsymbol{u}$, i.e., $\lim_{t\to\infty} \boldsymbol{x}_t = \boldsymbol{u}$, then we can regard the system as an associative memory.

In 1982, Hopfield proposed his fully connected neural network model of associative memory [4]. The model consists of N neurons and N^2 synapses. Each neuron can be in one of two states ± 1, and p bipolar patterns $\xi^\nu = (\xi_1^\nu, \cdots, \xi_N^\nu)$ $(\nu = 1, \cdots, p)$ are to be memorized as equilibrium states. He employed a discrete-time, asynchronous update scheme. That is, each neuron updates its state, one at a time, according to

$$s_i(t+1) = f\left(\sum_{j\neq i}^{N} w_{ij} s_j(t)\right),$$

where $s_i(t)$ is a state of i-th neuron at time t, w_{ij} is a synaptic weight from neuron j to neuron i, and $f(z) = 1$ if $z \geq 0$ and -1 otherwise.

The behaviors of the collective states of the individual neurons are characterized by the synaptic weights. When these synaptic weights are determined appropriately, networks store some number of patterns as fixed points. Hopfield specified w_{ij}'s by the Hebbian rule [5], i.e.,

$$w_{ij} = \sum_{\nu=1}^{p} \xi_i^\nu \xi_j^\nu \ (i \neq j),\ w_{ii} = 0.$$

Then giving one of the memorized patterns, possibly including a few errors, to the network as an initial state will result in the stable state after certain time steps. He suggested by using computer simulation that the maximum number p of the patterns to be stored in a network with N neurons is $0.15N$, if a small error in recalling is allowed. Later, this was theoretically calculated by Amit et al. [6] by using *replica* method. They showed that the storage capacity is $p = 0.14N$. In 1987, McEliece et al. [7] proved that when $p < N/4\ln N$ holds, the Hopfield's model will recall the memory without error.

The storage capacity depends strongly on how synaptic weights are specified. The specification of the synaptic weights is conventionally referred to as *learning*. Other learning schemes instead of Hebbian rule have been proposed for their increased storage capacity. The *pseud-inverse matrix* method by Kohonen et al. [8] and *spectral algorithm* by Pancha et al. [9] are the examples, and the capacity is enlarged to $p = 0.5N$ and $p = N$, respectively. Both are extensions of the Hebbian learning.

Then, what is the ultimate capacity when only the learning scheme is explored within the Hopfield's framework? Generally the storage capacity is traded off against basin size. Gardner[10] studied the optimal storage capacity as a function of the size of basin of attraction. She showed that as basin size tends to zero the ultimate capacity will be $p = 2N$, proposing an algorithm to obtain the weight matrix.

The storage capacity also depends on the other aspects of neural networks. For example, we can increase the capacity *exponentially* instead of *proportionally* by modifying the architecture of the network. Chiueh et

al. [11] proposed a new architecture in which storage capacity scales *exponentially* with N, the number of neurons. They showed that 2^N might be the ultimate upper bound for the capacity.

However, for some reason, we are interested in exploring synaptic weight space with Hopfield's scheme unchanged.

In this framework, we previously used a genetic algorithm [13, 14] to modify the Hopfield's Hebbian synaptic weights [12]. If the patterns which exceed the capacity are learned by Hebbian rule, the memories are more or less collapsed. We succeeded in evolving this overloaded Hebbian weights to re-store all the patterns as fixed points, and obtained the capacity $p = 0.33N$. In this paper, we apply the genetic algorithm to more challenging random synaptic weights.

2 GA Implementation

In this simulation, a weight matrix R_{ij} is produced randomly before the start of the GA run, and remains unchanged. Chromosomes in each generation modify this initial matrix to produce a population of weight matrices as follows.

$$w_{ij} = c_{ij} \times R_{ij}, \quad c_{ij} \in \{1, 0, -1\}.$$

These are evaluated their fitness values. According to the fitness values, two parent chromosomes are selected to be recombined to produce one offspring. The offspring is mutated occasionally and reconstructs the next generation.

The cycle of reconstructing the new population with better individuals and restarting the search is repeated until a perfect solution is found or a maximum number of generation has been reached. The specific details are as follows.

(1) (initialization) Chromosomes are N^2-dimensional vectors, and they are initialized so that their components are randomly selected from $\{1, 0, -1\}$ with the probability of .98, .01 and .01, respectively.

(2) (fitness evaluation) When one of stored patterns ξ^ν is given to the network as an initial state, the state of neurons varies from time to time afterwards (unless ξ^ν is a fixed point). In order for the network to function as an associative memory, these two states must be similar. The similarity as a function of time is defined by,

$$m^\nu(t) = \frac{1}{N} \sum_{i=1}^{N} \xi_i^\nu s_i^\nu(t),$$

where $s_i^\nu(t)$ is the state of the i-th neuron at time t. This is referred to as *overlap* by convention. In evaluating the fitness value, the temporal average of the overlap $\langle m^\nu \rangle$ is calculated for each stored pattern. Then these are also averaged over all patterns. That is, the fitness value f is

$$f = \frac{1}{t_0 \cdot p} \sum_{t=1}^{t_0} \sum_{\nu=1}^{p} m^\nu(t).$$

In this paper, t_0 is set to $2N$, twice the number of neurons. Note that the fitness 1 implies all the p patterns are stored as fixed points, while fitness less than 1 includes many possible other cases.

(3) (selection) We use $(\mu + \lambda)$-selection in evolution strategy terminology. Two parent chromosomes are chosen *randomly* from the best 40% of the population $(= \mu)$.

(4) (recombination) Recombinations are made with *uniform crossover* [16], operating on alleles of the selected parents' chromosomes, i.e., two parents $(u_1, \cdots, u_n)$ and $(v_1, \cdots, v_n)$ produce an offspring $(w_1, \cdots, w_n)$ such that w_i is either u_i or v_i with equal probability.

(5) (mutation) Mutation is made by rotating the allele as follows.

$$\{1\} \to \{-1\}, \quad \{-1\} \to \{0\}, \quad \{0\} \to \{1\}$$

The procedures (3)–(5) are repeated until all the individuals in the worst 60% of the population $(= \lambda)$ are replaced with the offspring.

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

3 Results and Discussion

Since the initial matrix to start with is completely random, all the networks in the first generation do not function as an associative memories, but are full of spin-glass attractors. The goal of this paper is to find the optimal combinations of $c_{ij} \in \{1, 0, -1\}$ which modify the initial matrix by multiplying them to the corresponding components of the matrix.

All simulations were carried out on networks with 49 neurons.

First, the effect on evolution of varying p is studied. We repeat each simulation 30 times with different random number seed for the same p. If we find the perfect solution(s) then we increment p. Thus, we found a matrix evolved to store a maximum of 7 patterns. We shows representative sample of the best fitness vs. generation in Figure 1. In this experiment, optimization are made by pruning synaptic connections as well as balancing the number of excitatory/inhibitory

synapses, with using $c_{ij} \in \{1, 0, -1\}$. For comparison purposes, we also experimented with $c_{ij} \in \{1, -1\}$ and with $c_{ij} \in \{1, 0\}$. The results for 7 patterns are also shown in Figure 1. We can see that balancing the number of excitatory/inhibitory synapses plays more important roll than pruning synapses. The number of patterns each resultant matrix can store as fixed points is 3 and 0, respectively. And the maximum p in each experiment is 6 and 3, respectively.

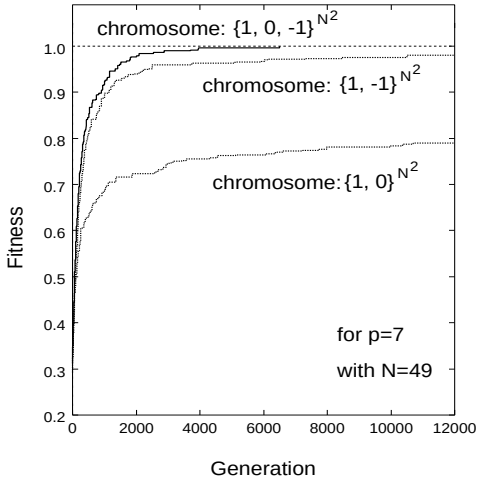

Figure 1: **Fitness of the Best of Generation**

Asymmetry and Dilution of Synaptic Weights

We conjectured that the emergence of retrieval states is due to the destabilization of the spin-glass attractors by the asymmetry and dilution of synaptic weights, as Herz et al. [17] suggested.

To see this, we investigated the time evolution of degree-of-symmetry and dilution rate of weight matrices. The degree-of-symmetry is defined by

$$\sigma = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}w_{ji} \cdot \left(\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^2\right)^{-1},$$

following after Krauth et al. [18]. As in Figure 2, both ratios increase from 0 and asymptotically approach to the value between 0.15 and 0.2 as retrieval states emerge.

Basin of Attraction

To regard a network an associative memory system, it should show the tolerance for certain amount of noises. That is, each memory pattern must have a certain domain of basin of attraction.

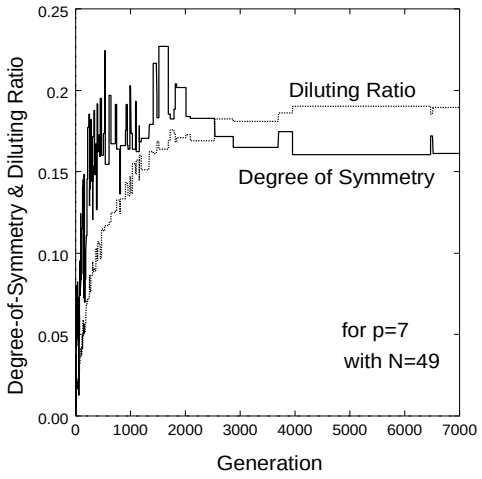

Figure 2: **Degree of Symmetry and Dilution Rate**

As the criterion of basin size, the definition by Verleysen et al. [19] are used. We generate 1000 initial state patterns by randomly picking up a stored pattern and flipping d bits. Then they are given to the network and updated the state $2N$ times (possibly it reaches a stable state). The resulting states is then compared with the corresponding stored pattern, and the run is called a success if they match *exactly*. In Figure 3, the number of successes out of 1000 runs are plotted against the number of input errors d. We also plot the result of the original Hebbian synaptic weights for convenience. Although the networks resulted from the above experiments have extremely small size of basin of attraction, we can see that they still have an error-correcting capability.

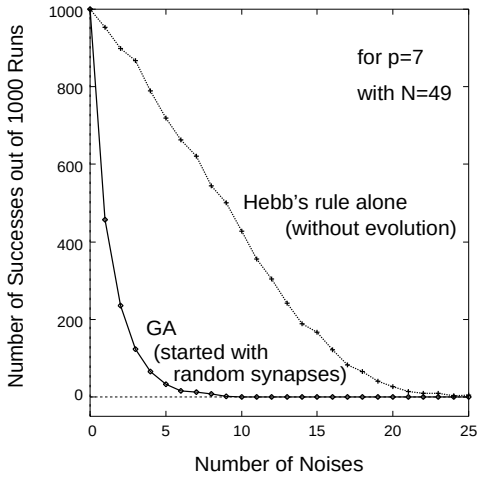

Figure 3: **Error Correcting Capability**

4 Conclusions

We have described an application of a genetic algorithm to Hopfield model of associative memory. The genetic algorithm modify a pre-specified weight matrix by multiplying the alleles of the ternary chromosome consists of 1, 0, and -1 to the corresponding components of the starting matrix.

The simulations were made with 49 neurons. This size of Hebb-rule associative memory would store at most 8 random bipolar patterns as fixed points. The patterns stored by the matrix found by the genetic algorithm is a little fewer than that. However this is amazing in that the evolution was started with completely random synaptic weights only by pruning some of its connection and by balancing the number of excitatory/inhibitory synapses.

We conjecture that the success is more or less due to the asymmetry and dilution of synaptic weights introduced by the genetic algorithm.

References

[1] J. D. Schaffer, D. Whitley, and 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.

[2] X. Yao (1993) *"A Review of Evolutionary Artificial Neural Networks."* International Journal of Intelligent Systems, vol.8, 539.

[3] J. Komlós, and R. Paturi (1988) *"Convergence Results in an Associative Memory Model."* Neural Networks 1, 239.

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

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

[6] D. J. Amit, H. Gutfreund, and H. Sompolinsky (1985) *"Storing Infinite Number of Patterns in a Spin-glass Model of Neural Networks."* Phys. Rev Lett. 55, 1530.

[7] R. J. McEliced, E. C. Posner, E. R. Rodemick, and S. S. Venkatesh (1987) *"The Capacity of the Hopfield Associative Memory."* IEEE Trans. Information Theory IT-33, 461.

[8] T. Kohonen, and M. Ruohonen (1973) *" Representation of Associated Data by Matrix Operators."* IEEE Trans. Computers C-22(7), 701.

[9] G. Pancha, S. S. VenKatesh (1993) *"Feature and Memory-Selective Error correction in Neural Associative Memory."* in M. H. Hassoun eds. Associative Neural Memories: Theory and Implementation, Oxford University Press, 225.

[10] E. Gardner (1988) *"The Phase Space of Interactions in Neural Network Models."* J. Phys 21A, 257.

[11] T. D. Chiueh, and R. M. Goodman (1991) *"Recurrent Correlation Associative Memories."* IEEE Trans. Neural Networks 2(2), 275.

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

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

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

[15] 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, vol.1, 803.

[16] G. Syswerda (1989) *"Uniform Crossover in Genetic Algorithms."* Proceedings of the 3rd International Conference on Genetic Algorithms, 2.

[17] J. A. Hertz, G. Grinstein, and S. A. Solla (1987) *"Irreversible Spin Glasses an Neural Networks."* in L. N. van Hemmen and I. Morgenstern eds. Heidelberg Colloquium on Glassy Dynamics. Lecture Notes in Physics No.275 Springer-Verlag, 538.

[18] W. Krauth, J.-P. Nadal, and M.Mezard (1988) *"The Roles of Stability and Symmetry in the Dynamics of Neural Networks."* J. Phys. A: Math. Gen. 21, 2995.

[19] M. Verleysen, J.-D. Legat, and P. Jespers (1993) *"Analog implementation of an Associative Memory: Learning Algorithm and VLSI Constraints."* in M. H. Hassoun eds Associative Neural Memories: Theory and Implementation, Oxford University Press, 265.