Hopfield Model of Associative Memory as a Test Function of Evolutionary Computations

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 genetic algorithms to Hopfield's neural network model of associative memory. Previously, using ternary chromosomes, we successfully evolved both random weight matrix and over-loaded Hebbian weight matrix to function as an associative memory. In this paper, we present a real-encoded genetic algorithm to evolve random synaptic weights to store some number of patterns as associative memory. The goal is to study how can we use the Hopfield model as a test suite for evolutionary computations.

1 Introduction

In the field of Evolutionary computations, various kinds of test functions has been proposed (see, e.g., [1, 2, 3]). In this paper, we discuss the Hopfield model of associative memory as a more challenging test suite for the evolutionary computations. Hopfield model of associative memory has almost infinite number of combinations of synaptic weights which give the network a function of associative memory [4]. At the same time, there is a trade-off between its storage capacity and the error correcting capability. Hence we can think of the model as an example of multi-modal functions as well as multi-objective functions. Previously, as a real-valued parameter optimization problem, we applied Evolutionary Programming (EP) [5], Evolution Strategy(ES) [6], and Breeder Genetic Algorithm (BGA) [3] to the model. And we observed that they can easily search the solutions for the storage of small number of patterns [7, 8]. However, they have not been able to locate the global optimal solution suggested by Gardner [9], so far. As a preliminary stage for using the Hopfield model to test evolutionary computations, we apply, in this paper, a more simple form of the genetic algorithm (GA) [10] in order for the analysis to be made clear. That is, real-coded genetic algorithm using uniformly radom mutation with discrete recombination under truncation selection. The ultimate goal of this research is to locate the solution of Gardner's limit.

2 Hopfield Model

Associative memory is a dynamical system which has a number of stable states with a domain of attraction around them [11]. 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 [12]. 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. Namely, 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 [13], 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. [14] by using *replica* method. They showed that the storage capacity is $p = 0.14N$. In 1987, McEliece et al. [15] 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. [16] and *spectral algorithm* by Pancha et al. [17] 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[9] 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.

3 GA Implementation

In this simulation, a population of weight matrices are produced randomly when the GA starts. In each generation, they are modified through discrete crossovers and uniformly random mutations, and their fitness values are evaluated. According to the fitness values, individuals of the next generation are selected, using a $(\mu + \lambda)$-strategy in ES terminology. 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. Specific details of the operations are as follows.

Representaion: Chromosomes are N^2-dimensional real-valued vectors. Each allelic value of the chromosome is mapped from the corresponding component of the individual's synaptic weight matrix.

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 other cases.

Selection: We use the truncation-selection [3], i.e., two parent chromosomes are chosen *randomly* from the best T% of the population.

Recombination: Discrete Recombination [3] operates 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.

Mutation: Mutation is made by randomly picking up an allele with the probability p_m, and replace it with uniformly random number selected from $[-1, 1]$.

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

4 Results and Discussion

Since all the networks in the first generation have synaptic weights chosen randomly from $[-1, 1]$, they do not function as associative memories because of a full of spin-glass attractors (low m stable attractors). The goal is to search the combinations of the real-valued synaptic weights which make the network store all the given patterns as fixed point attractors.

All simulations were carried out on networks with 49 neurons. Note that this size of the network would store 8 patterns under Hebbian learning algorithm. The parameters of the genetic algorithm: population number, truncation rate T, and mutation probability p_m are set to 256, 40, and 0.01, respectively.

First, the effect on the genetic algorithm of varying the number of patterns 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 9 patterns. We shows representative sample of the best fitness vs. generation of the run in Figure 1.

We also run the same genetic algorithm without recombination operator for the same patterns as above experiment. This is a continuous version of *Random Mutation Hill Climbing* [18] or *Random Bit Climber* [19].

The result is shown also in Fig.1. In some problems, simple hill climbing or mutation without crossover outperform the genetic algorithm, however, we can see that the crossover plays an important roll in this experiment.

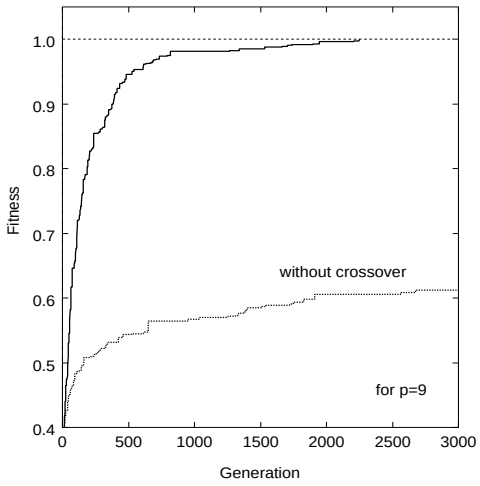

Figure 1: **Fitness of the Best of Generation**

We investigated the time evolution of degree-of-symmetry of the weight matrix, which 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. [20]. We show the result in Fig.2. As the spin-glass attractors are destabilized and retrieval states are emerged, the ratio increases from 0 (totally asymmetric) and asymptotically approaches to the value around 0.25.

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.

As the criterion of basin size, the definition by Verleysen et al. [21] 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 for the same matrix shown in Fig.1. We also plot the result of the original Hebbian synaptic weights for convenience. We can see that the networks resulted from the above experiments have extremely small size of basin of attraction, though they

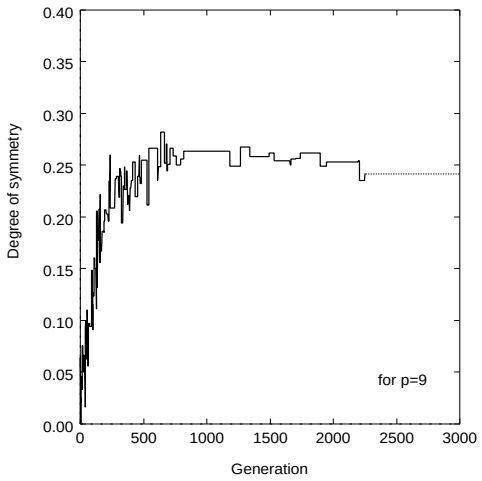

Figure 2: **Degree of Symmetry**

still have a little error-correcting capability. The genetic algorithm seems to find solutions by sacrificing the size of basin of attraction.

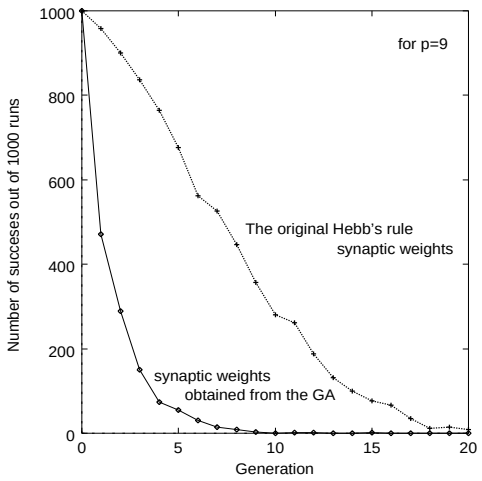

Figure 3: **Error Correcting Capability**

5 Conclusions

We have described an application of a real-encoded genetic algorithm to Hopfield model of associative memory. The task is to find a combination of $49^2 = 2401$ synaptic weights which gives the fully-connected 49 neurons of network a function of associative memory. The genetic algorithm located the solution for a maximum of 9 patterns. All these patterns are eventually stored as fixed

points. For more than 9 patterns, however, the searches are all stagnated in the middle of evolution, in spite of the fact that there are solutions for up to 98 patterns as Gardner suggested.

The analysis and results reported are only the beginning of our research, and we are exploring the way to find powerful algorithms to be able to locate the global optimum in the sense of Gardner.

References

[1] K. DeJong (1975) *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Ph.D thesis, University of Michigan.

[2] H.-P. Schwefel (1981) *Numerical optimization of Computer Models.* Wiley.

[3] H. Mühlenbein, D. Schlierkamp-Voosen (1996) *Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization* Evolutionary Computation 1(1), 25.

[4] P. Baldi, and S. Venkatesh (1987) *Number of Stable Points for Spin-glasses and Neural Networks of Higher Orders.* Physical Review Letter 58, 913.

[5] D. B. Fogel (1992) *An Analysis of Evolutionary Programming.* Proceedings of 1st Annual Conference on Evolutionary Programming, 43.

[6] I. Rechenberg (1965) *Cybernetic Solution path of an Experimental Problem.* Royal Aircraft Establishment, Library Translation No.1122 (Hants UK).

[7] A. Imada, and K. Araki (1996) *Searching Real-Valued Synaptic Weights of Hopfield's Associative Memory using Evolutionary Programming.* (submitted)

[8] A. Imada, and K. Araki (1996) *Application of Evolution Strategy to Hopfield model of Associative Memory.* (submitted)

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

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

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

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

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

[14] 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.

[15] 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.

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

[17] 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.

[18] S. Forrest, and M. Mitchell (1993) *Relative Building Block Fitness and the Building Block Hypothesis.* Foundation of Genetic Algorithms 2, Morgan Kauffmann, 109.

[19] L. Davis (1991) *Bit-Climbing, Representational Bias, and Test Suite Design.* Proceedings of the 4th International Conference on Genetic Algorithms, 18.

[20] 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.

[21] 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.