Application of an Evolution Strategy to the Hopfield Model of 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. In the Hopfield model, almost infinite number of combinations of synaptic weights give the network a function of associative memory. Furthermore, there is a trade-off between the storage capacity and size of basin of attraction. Therefore, the model could be thought of as a test suite of multi-modal and/or multi-objective optimization. As preliminary stages, we investigate the basic behaviors of associative memory under simple evolutionary processes. In this paper, we present some experiments using an evolution strategy.

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 an evolution strategy.

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. 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. Networks store some number of patterns as fixed points when these synaptic weights are determined appropriately. Hopfield specified w_{ij}'s by the Hebbian rule[5]:

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

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 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. 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 as-

pects 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, we are interested in exploring synaptic weight space with Hopfield's scheme unchanged.

Previously, with using a genetic algorithm [12, 13], we succeeded in evolving fully-connected neural network with random synaptic weights to store some number of patterns [14]. It was made by optimally pruning some of its connections as well as balancing the number of excitatory/inhibitory synapses. Chromosomes consisting of mostly 1 but a few of 0 and -1 were used to modify a pre-determined random matrix R_{ij}. That is, individuals in a population are

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

The resultant storage capacity was $p = 0.16N$.

We also showed that the genetic algorithm enlarges the capacity of Hebb-rule associative memory[15], where we used an over-loaded Hebbian weight matrix J_{ij}, which means that the matrix learned a set of patterns more than the storage capacity ($p > 0.15N$). This matrix

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

was evolved in the same way as above, and eventually stored all the learned patterns as fixed points. The capacity $p = 0.33N$ was obtained.

In the genetic algorithm above, we used ternary chromosomes $\{1, 0, -1\}^{N^2}$. In this paper, on the other hand, we encode synaptic weight matrix directly to real-valued chromosomes $\mathbf{R}^{N^2}$, and apply Evolution Strategy [16] to them as a continuous parameter optimization problem.

2 Evolution of Hopfield Model

The conceptual algorithm is as follows:

begin
  initialize 256 individuals
  evaluate fitness values of the individuals
  while termination criterion not fulfilled **do**
    until the rest of the best 40% replaced **do**
      select two parents from the best 40% individuals
      recombine them to produce an offspring
      mutate it
      evaluate its fitness value
    od
  od
end

The goal of the algorithm is to find a set of real-valued synaptic weights which stores all the pre-determined bipolar patterns, ξ^ν ($\nu = 1, 2, \cdots, p$), as fixed points.

We describe each procedure a bit more in detail.

(1) (initialization) Individuals are $2N^2$-dimensional vectors whose components are N^2 synaptic weights (object variables) and the same number of standard deviations (strategy parameters). The synaptic weights are initialized in two ways: (a) random synaptic weights , and (b) over-loaded Hebbian weights. In both cases, networks in the first generation do not function as associative memories. Population is set to 128 in this paper.

(2) (fitness evaluation) When a pattern ξ^ν is given to the network, state of neurons varies at each time of update (unless ξ_i^ν is a fixed point). The difference between the initial state and updated state at time t is measured 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 traditionally referred to as *overlap*. The averaged value of $m^\nu(t)$ over all the stored patterns at certain time t_0 might be used as a fitness value. However, this would sometimes result in the solution which would give high fitness only at certain times including t_0 (periodic behaviors). Hence we calculated the fitness value by averaging it over times, i.e.,

$$f = \frac{1}{(t_0 - 1) \cdot p} \sum_{t=2}^{t_0} \sum_{\nu=1}^{p} m^\nu(t).$$

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

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

(4) (recombination) *Discrete recombination*, which is similar to *uniform crossover* [17] in binary chromosomes, are operated both on object variables and strategy parameters of the selected parents, 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. This process is repeated until all the individuals in the worst 60% of the population ($= \lambda$) are replaced.

(5) (mutation) Previously, we also applied *evolutionary programming* [18] to this model of associative memory [19]. For comparison purposes, we use the similar mutation operation to that implementation. That is, instead of usual mutation in evolution strategy

$$\begin{cases} x_i' = x_i + \sigma_i' \\ \sigma_i' = \sigma_i + \exp N_i(0,1) \end{cases}$$

we mutate object variable x_i and strategy parameter σ_i as follows.

$$\begin{cases} x_i' = x_i + \sqrt{f_i} \cdot N_i(0,1) \\ \sigma_i' = \sigma_i + \sqrt{0.01 \cdot f_i} \cdot N_i(0,1) \end{cases}$$

where $N_i(0,1)$ is normally distributed random variable of mean 0 and standard deviation 1 sampled for each individual i.

The above parameter values were determined mainly on the basis of trial and error, while they have not necessarily been optimized.

3 Results and Discussion

All simulations were carried out on networks with 49 neurons. Hence, the first goal of the experiment is to find a solution out of many possible solutions in the connection space $\boldsymbol{R}^{2401}$.

The initialization of individual matrices in the first generation is made in the following two ways, as mentioned above. One is to start with individual matrices whose components are randomly chosen small values. The other is to start with matrices which learn patterns exceeding the capacity with Hebbian rule. In both cases, individuals in the first generation do not function as associative memory.

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 11 patterns in the former case, and 17 patterns in the latter case.

Figure 1 shows representative samples of the best fitness vs. generation for evolutions started with both random and over-loaded Hebbian matrices. They converge at generation 1062 and 4374, respectively.

Intuitively, these two matrices seem to belong to different classes of solutions. To see this, we investigated the time evolution of degree-of-symmetry of these matrices, defined by

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

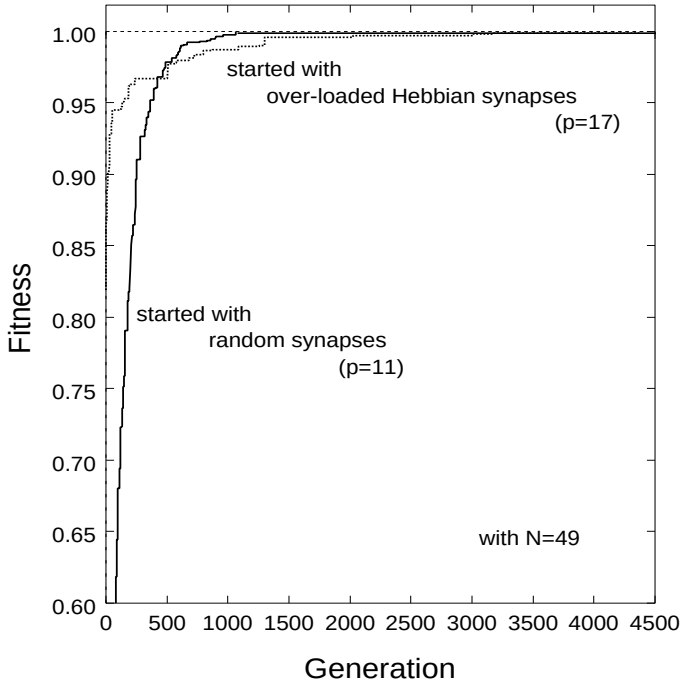

Figure 1: **Fitness of the Best of Generation**

following after Krauth et al. [20]. In the simulation starting with over-loaded Hebbian matrices, the degree of symmetry decreases from the perfect symmetry and asymptotically approaches to the value around 0.5 as its corrupted memory is recovered. In this level of over-capacity, the memory states are not randomized, but some resultant spurious equilibrium states prevent the input state from converging to its memory state. Then we conjecture that the asymmetry of the synaptic weights introduced by the evolution strategy destabilizes that spurious states, as Herz et al. [21] suggested in their analysis using Ising model.

In the simulation starting with random matrix, on the other hand, we can see the ratio increases from zero and approaches to the value around 0.4, showing that we need some extent of symmetry to obtain the function of associative memory.

However the roll of asymmetry of the synaptic weights is still an open question.

To regard a network as 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, we used somewhat similar definition by Verleysen et al. [22]. We generate 1000 initial state patterns by randomly picking up a stored pattern and flipping d bits. Then they are given to the

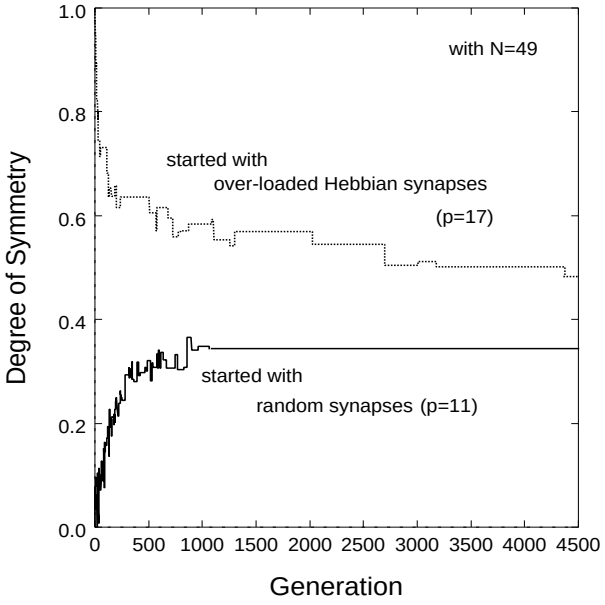

Figure 2: **Degree of Symmetry**

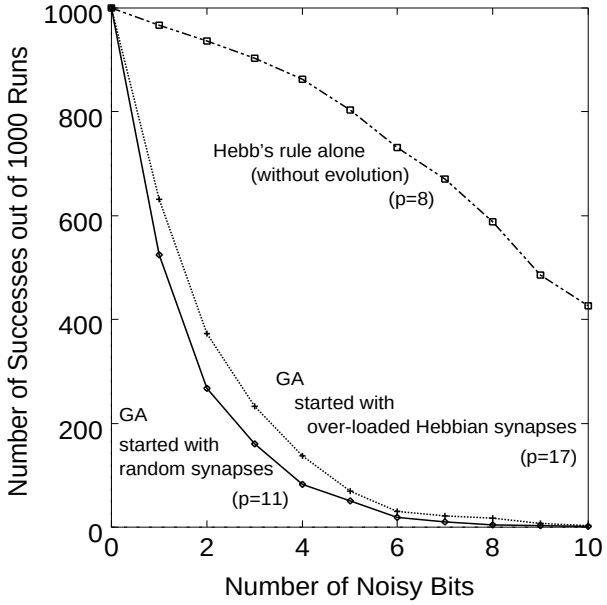

Figure 3: **Error Correcting Capability**

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 of the above two experiments 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 two experiments have extremely small size of basin of attraction, they still have an error-correcting capability.

4 Conclusions

We have described an application of an evolution strategy to Hopfield model of associative memory. The objective of these experiments is to find some solutions of real-valued synaptic weight matrix which gives networks a function of associative memory.

The simulations in this paper were made with 49 neurons. This size of Hebb-rule associative memory would store around 8 random bipolar patterns as fixed points. The evolution strategy finds solution which stores 11 ($0.22N$) and 17 ($0.35N$) patterns starting with random and Hebbian matrices, respectively. We conjecture these phenomena are more or less due to asymmetry of weight matrix introduced by the evolution strategy.

The results and analysis reported here are only the beginning of our continuing research project, and many other tasks including more elaborate tuning of the parameters are still waiting for exploration.

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, *"A Review of Evolutionary Artificial Neural Networks."* (1993) 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] J. Holland (1975) *Adaptation in Natural and Artificial Systems.* The University of Michigan Press

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

[14] 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.

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

[16] I. Rechenberg (1965) *"Cybernetic Solution path of an Experimental Problem."* Royal Aircraft Establishment, Library Translation, 1122 (Hants UK).

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

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

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

[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] 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 275 Springer-Verlag, 538

[22] 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.