Searching Real-Valued Synaptic Weights of Hopfield's Associative Memory using Evolutionary Programming

Akira Imada
akira-i@is.aist-nara.ac.jp

Keijiro Araki
araki@is.aist-nara.ac.jp

Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-01 Japan
phone: +81-7437-2-5083

Abstract

We apply an evolutionary programming to fully connected associative memory model of Hopfield. There have been a lot of researches which apply evolutionary techniques to layered neural networks. However, their applications to Hopfield neural networks remain few so far. In earlier papers we reported that a genetic algorithm can evolve networks to function as associative memory by pruning some of its connections. In this paper we present that the associative memory networks can be produced more directly by searching synaptic weights in the space of real value.

1 Introduction

Associative memory is a dynamical system which has a large number of stable states with a domain of attraction around them[1]. If the system starts at any state in the domain of attraction, it will converge to the stable state.

In 1982, Hopfield proposed his fully connected neural network model of associative memory[2]. The model consists of N neurons and N^2 synapses, in which p bipolar patterns

$$\xi^\mu = (\xi^\mu_1, \cdots, \xi^\mu_N) \quad (\mu = 1, \cdots, p)$$

are to be memorized in the form of its stable equilibrium states. He employed a discrete-time, asynchronous update scheme, i.e., each neuron receives inputs from all other neurons and updates its state, according to,

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

where w_{ij} is a synaptic weight from neuron j to neuron i, $s_i(t)$ is the state of i-th neuron at time t, and $f(z) = 1$ if $z \geq 0$, -1 otherwise. Asynchronous update means that at most one neuron is updated its state at each time step. The synaptic weights w_{ij} are specified by the Hebbian rule[3], i.e.,

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

Then giving one of the memorized patterns, possibly including a few errors, to the network as a initial state will result in the stable state after certain time steps of updating. He suggested by using computer simulation that the maximum number of the patterns to be stored in a network with N neurons is $p = 0.15N$, if a small error in recalling is allowed.

Later, this was theoretically calculated by Amit, Gutfreund & Sompolinsky[4] by using *replica* method, and showed that $p = 0.14N$.

In 1987, McEliece, Posner, Rodemick & Venkatesh[5] proved that when

$$p < \frac{N}{4 \ln N}$$

holds, the Hopfield's model will recall the memory without error.

The storage capacity is strongly dependent on how synaptic weights are specified. This is conventionally referred to as *learning*. Other learning schemes instead of Hebbian rule have been proposed for their increased storage capacity. For example, *pseud-inverse matrix* method by Kohonen & Ruohonen[6], which has the capacity $p = 0.5N$, and *spectral algorithm* by Pancha & Venkatesh[7] which has the capacity $p = N$. Both are an extension of Hebbian learning.

The specification of synaptic weights also implies that pruning the connection of Hebbian weights. Derrida, Gardner & Zippelius[8] studied a symmetrically diluted Hebbian synaptic matrices and showed that the capacity, measured per surviving synapse, is significantly higher than that of the fully connected network.

In an earlier paper[9], we reported that fully-connected neural network with *random* synaptic weights evolved to store some number of patterns by optimally pruning some of its connections by means of a genetic algorithm[10]. In this experiment, a population of weight matrices were generated by multiplying each component of a pre-fixed random matrix by an allele of chromosomes which include a few of zeros. The resultant capacity was $p = 0.16N$.

We also showed that the genetic algorithm enlarges the capacity of Hebb-rule associative memory[11]. In this experiment, we used Hebbian rule to make a network learn a set of random bipolar patterns whose number exceeds the Hopfiedd's capacity. Hence this matrix does not function reliably as associative memories because of these over-loaded patterns ($p > 0.15N$). Then the genetic algorithm evolved this matrix in the same way as above, and eventually made it recover all the learned memory. The capacity was $p = 0.33N$.

What is the ultimate capacity by only exploring the learning scheme within the Hopfield's frame work ? The storage capacity, in general, is traded off against the basin size. Gardner[12] studied the optimal storage capacity as a function of the size of the basin of attraction. She showed that as basin size tends to zero the ultimate capacity will be

$$p = 2N,$$

as well as proposing an algorithm to obtain weight matrix to realize the capacity.

The storage capacity also depends on other aspects of neural networks. For example, we can increase the storage capacity exponentially instead of proportionally by modifying the architecture of the network. Chiueh & Goodman[13] proposed a new architecture in which storage capacity scales exponentially with the number of neurons N. To be more specific, for an input pattern ρN bits away from the nearest memory pattern, the storage capacity can be approached to c^N asymptotically, where c varies from 1 ($\rho = 0.5$) to 2 ($\rho = 0$) according to the specified basin size. Hence 2^N might be the ultimate upper bound for the capacity of the network.

From the discussions above, it does not seem to be interesting to study Hopfield model of associative memory any more. However, there are almost infinite number of solutions in the interconnection space. A modest amount of them has been reported already, whilst all of these solutions has been found heuristically. One reason why we are interested in the Hopfield's model is to obtain the insight into the distribution of solutions in the search space. Furthermore, due to its simplicity, Hopfield model of associative memory could be used as a test function for evaluating evolutionary techniques. The model has lots of local minima, some of which we already know as well as the global minimum as the Gardener's limit mentioned above. As such, our interest is to search solutions which function as an associative memory by exploring in the huge connection space of $\boldsymbol{R}^{N^2}$ rather than to seek the method of realizing high storage capacity.

In this context *evolutionary programming*[14] or possibly *evolution strategy*[15] seems to be more direct than *genetic algorithm*. Evolutionary programming is a technique for searching real parameter values by selection after the mutation using normally distributed variances. Usually this variances also undergo adaptation. However, in this paper, we apply what Böck & Schwefel[16] call a *standard evolutionary programming*, where deviations are not self-adaptive.

In the next section, we describe the implementation of the evolutionary programming briefly.

2 Evolutionary Programming Implementation

Our evolutionary programming proceeds as follows:

(1) Some number of bipolar patterns ξ^μ ($\mu = 1, 2, \cdots, p$) are randomly produced. The goal of the evolutionary programming is to find a matrix which stores all these patterns as fixed points.

(2) All the individual weight matrices(NxN) in the population, W^ν ($\nu = 1, 2, \cdots, M$) are initialized. Population M is set to 128 in this paper. Here we experiment in the following two situations:

(a) All the components of matrices are set to zero, and (b) the matrix learn the patterns whose number exceeds the capacity. In both cases, networks do not function as associative memories. Note that all the individual matrices have the same form initially.

(3) A population of chromosomes which is constituted with N^2 alleles are produced. We denote $(i+N\cdot j)$-th allele of ν-th chromosome as η_{ij}^{ν}. All the initial value of these alleles are set to 0.5 here.

(4) Then each matrix is modified by the chromosomes as follows.

$$w_{ij}^{\nu}(\text{offspring}) = w_{ij}^{\nu}(\text{parent}) + \eta_{ij}^{\nu} \cdot N^{\nu}(0,1),$$

where w_{ij}^{ν} is ij-th component of the ν-th individual matrix W^{ν}, and $N^{\nu}(0,1)$ is a Gaussian random variable with mean zero and standard deviation one. This process produces a total of M offspring from M parents. Hence this is a $(M+M)$-strategy in evolution strategy terminology.
We note that recombination operator is not used to produce offspring.

(5) Fitness value f^{ν} for each individual weight matrix is evaluated with the pre-determined random patterns ξ^{μ}. First we calculate the average of all the inner products (overlap) of initial states and states at each time of update not more than certain time t_0. And then these are averaged over all sets of initial patterns, i.e.,

$$f^{\nu} = \frac{1}{p\cdot(t_0-1)\cdot N}\sum_{\mu=1}^{p}\sum_{t=2}^{t_0}\sum_{j=1}^{N}\xi_j^{\mu}s_j^{\mu}(t).$$

In this paper, t_0 is set to $2N$ i.e., twice the number of neurons, and we observed that was enough. Note that fitness 1 implies that all the initial patterns are stored as fixed points, 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 these oscillated solutions.

(6) The fitness value for *each* individual of M parents and M offspring is compared with those of q individuals randomly selected also from M parents and M offspring. The $2M$ individuals are ranked with number of wins out of these q tournaments, and M best individuals survive to the next generation. The number of tournaments q is set to 32 here.

(7) Unless highest fitness value reaches the value of 1 nor generation exceeds 12000, the allele value of each chromosome will be renewed according to,

$$\eta_{ij}^{\nu}(\text{new}) = \eta_{ij}^{\nu}(\text{old}) + \sqrt{0.01\cdot f^{\nu}}\cdot N^{\nu}(0,1)$$

then the processes from (4) to (7) are repeated.

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

3 Result 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 which function as associative memory, in the connection space of $\boldsymbol{R}^{2401}$.

The initialization of the 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 all zero. The other is to start with matrices which learn the memory patterns with Hebbian rule, but the number exceeds the capacity. In both cases, all individuals in the first generation are the same, and they do not function as associative memory.

The above experiment starting from Hebbian matrices is reminiscent of the Kirkpatrick-Sherrington model of spin-glass[17]. In fact if $p = 1$ this is exactly the same as their model. Our situation however, is rather extreme, i.e., over-loaded memorized patterns.

The effect on evolution of varying the number of patterns to be stored is studied. Starting with a certain number p, we repeat the simulations 30 times with different random number seeds for each p. If we find the perfect solution(s) we increment the number p. Thus, we found a matrix evolved to store 6 patterns in the former case, and 13 patterns in the latter case. In Figure 1, we show the results. The convergence speed is at generation 7848 and 10858, respectively.

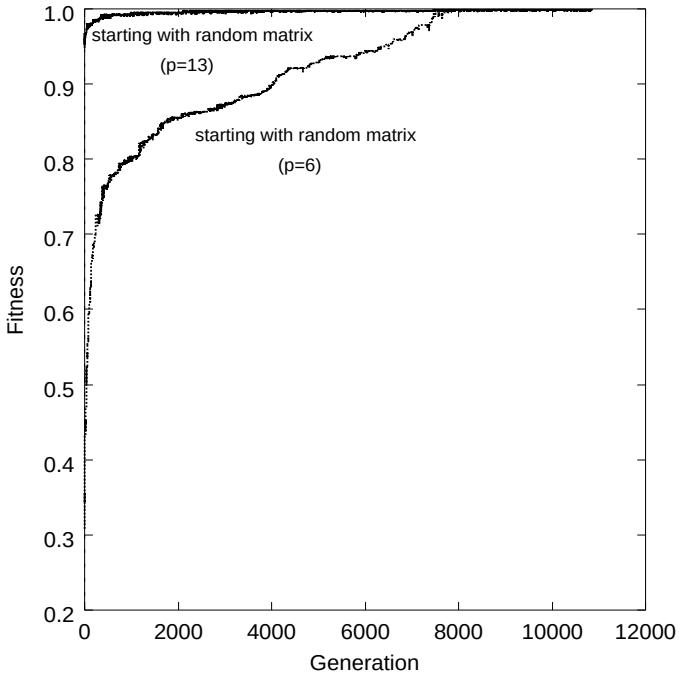

Figure 1: **Fitness of the Best of Generation**

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

$$\sigma = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}w_{ji} / \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^2.$$

In the simulation starting with over-loaded Hebbian matrix, the degree of symmetry decreases and asymptotically approaches to the value around 0.6 as the corrupted memory is recovered.

We conjecture that 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 the asymmetry of the synaptic weights introduced by the evolutionary programming destabilizes that spurious states, as Herz, Grinstein & Solla[18] and Parisi[19] suggested in their analysis using Ising model.

On the contrary, in the simulation starting with random matrix, we can see the ratio fluctuate around 0.1, 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.

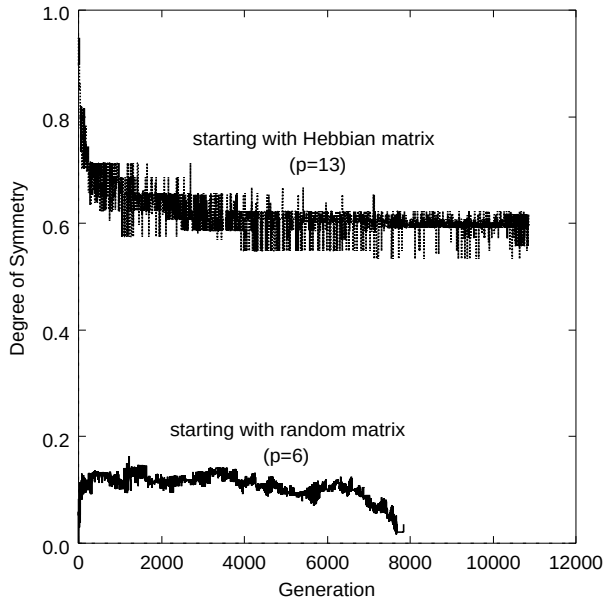

Figure 2: **Degree of Symmetry**

To say 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, we used somewhat similar definition to the one by Verleysen, Legat, & Jespers[20], That is, 1000 initial state patterns are generated by randomly picking up a memory

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 original memory 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 show the results of Hebbian synaptic weights without evolution for $p = 8$ for convenience. We can see that the network resulted from the above two experiments still maintain the tolerance of noisy input, although the size of basin of attraction is much smaller than the Hebb rule associative memory.

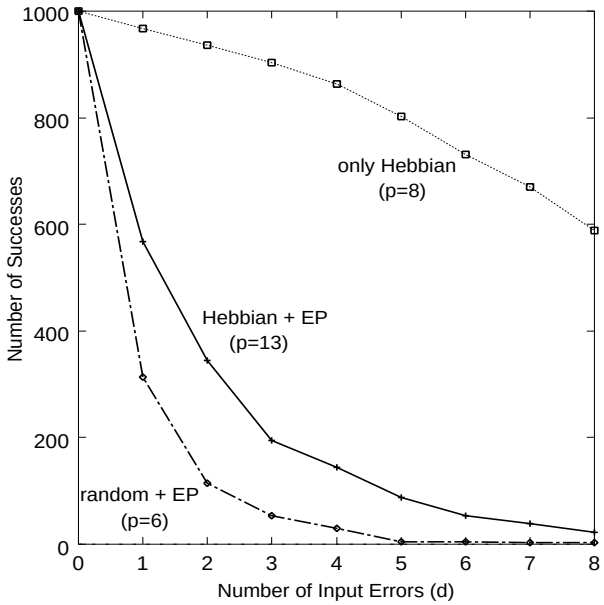

Figure 3: **Error Correcting Capability**

4 Conclusions

We have described an application of an evolutionary programming to fully connected neural network 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. Hence, solution is a point in the space of $\boldsymbol{R}^{2401}$. This size of Hebb-rule associative memory would store at most 7 random bipolar patterns as fixed points. The evolutionary programming finds solution which stores 6 ($p = 0.12N$) and 13 ($p = 0.27N$) patterns starting with random and Hebbian matrices, respectively, and their degrees of symmetry are different from each other. We conjecture that one of the reasons of these phenomena are more or less due to asymmetry of weight matrix introduced by the evolutionary programming.

The results and analysis reported here are only the beginning of our continuing research project, and many other tasks, such as multi-modality, are still waiting for exploration.

Acknowledgments

We thank Peter Davis at Advanced Telecommunication Research Institute (ATR) for providing us great insight into the dynamics of Hopfield neural networks.

References

[1] J. Komlós, and R. Paturi (1988) *"Convergence Results in an Associative Memory Model."* Neural Networks 1, p239.

[2] J. J. Hopfield (1982) *"Neural Networks and Physical Systems with Emergent Collective Computational Abilities."*, Proceedings of the National Academy of Sciences, USA, 79, p2554.

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

[4] 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, p1530.

[5] R. J. McEliced, E. C. Posner, E. R. Rodemick, and S. S. Venkatesh (1987) *"The Capacity of the Hofpield Associative Memory."* IEEE Trans. Information Theory IT-33, p461.

[6] T. Kohonen, and M. Ruohonen (1973) *" Representation of Associated Data by Matrix Operators."* IEEE Trans. Computers C-22(7), p701.

[7] 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, p225.

[8] B. Derrida, E. Gardner, and A. Zippelius (1987) *"An Exactly Soluble Asymmetric Neural Network Model."* Europhys. Lett. 4, p167.

[9] 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, p803.

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

[11] A. Imada, and K. Araki (1995) *"Genetic Algorithm Enlarges the Capacity of Associative Memory."*, Proceedings of 6th International Conference on Genetic Algorithms, p413.

[12] E. Gardner (1988) *"The Phase Space of Interactions in Neural Network Models."* J. Phys 21A, p257.

[13] T. D. Chiueh, and R. M. Goodman (1991) *"Recurrent Correlation Associative Memories."* IEEE Trans. Neural Networks 2(2), p275.

[14] D. B. Fogel (1992) *"An Analysis of Evolutionary Programming."* Proceedings of 1st Annual Conference on Evolutionary Programming, p43.

[15] I. Rechenberg (1965) *"Cybernetic Solution path of an Experimental Problem."* Royal Aircraft Establishment, Library Translation, p1122 (Hants UK).

[16] T. Böck, and H.-P. Schwefel (1993) *"An Overview of Evolutionary Algorithms for Parameter Optimization."* Evolutionary Computation 1(1), p1.

[17] S. Kirkpatrick, and D. Sherrington (1978) *"Infinite-ranged models of Spin-glasses."* Phys. Rev. B17, p4384.

[18] 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, p538.

[19] G. Parisi (1986) *"Asymmetric Neural Networks and the Process of Learning."* J. Phys. A19, L675.

[20] 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, p265.