

# EVOLVING CONNECTION WEIGHTS FOR PATTERN STORAGE AND RECALL IN HOPFIELD MODEL OF FEEDBACK NEURAL NETWORKS USING A GENETIC ALGORITHM

T. P. Singh<sup>1</sup>, Suraiya Jabin<sup>2</sup>

<sup>1,2</sup> Department of Computer Science, Jamia Millia Islamia (Central University), New Delhi, India  
{thipen@yahoo.com}

## ABSTRACT

*In this paper, implementation of a genetic algorithm has been described to store and later, recall of some prototype patterns in Hopfield neural network associative memory. Various operators of genetic algorithm (mutation, cross-over, elitism etc) are used to evolve the population of optimal weight matrices for the purpose of storing the patterns and then recalling of the patterns with induced noise was made, again using a genetic algorithm. The optimal weight matrices obtained during the training are used as seed for starting the GA in recalling, instead starting with random weight matrix. A detailed study of the comparison of results thus obtained with the earlier results has been done. It has been observed that for Hopfield neural networks, recall of patterns is more successful if evolution of weight matrices is applied for training purpose also.*

## KEYWORDS

*Hopfield Neural Network, genetic algorithm, associative memory, weight matrices, population generation technique, fitness function*

## 1. INTRODUCTION

There are numerous examples which demonstrate that a human brain can learn, understand and remember certain things completely, partially and sometimes not at all. Our learning capacity decides what information is stored in our brain. Incomplete or partially learned information loses its strength with time and does not get retained in the brain. But through continuous practice or training, selected things can be hard coded in our brain [1].

The Artificial neural networks (ANN) attempt the modeling of the human brain in a serial machine and are nowadays well-established computational structures which use simple elementary units connected together with appropriate weighted connections [2-5]. Neural Networks are massive parallel computing systems consisting of an extremely large number of simple processors with many interconnections. The main characteristics of neural networks are that they have the ability to learn complex nonlinear input-output relationships, use sequential training procedures, and adapt themselves to the data [6].

Our memories function in an associative or content-addressable fashion. That means, a memory is not located in a particular net of neurons in isolation. The Hopfield neural network is a simple

recurrent artificial network which is capable of storing certain patterns in a manner similar to the brain - the full pattern can be recovered if the network is presented with only partial, occluded or noisy version of it. This is just because of the stability of the system, if few of the interconnections of the nodes are changed, the recalled pattern is not too badly corrupted, the network can respond in a way what is called the “best guess”.

## **2. BACKGROUND AND RELATED WORK**

Neural networks have been widely used for pattern recognition and classification problems [7-8]. Pattern recognition is a very broad area in neural networks and a very common task realized by NN is in handwriting or character recognition [9-10]. Hopfield [11] proposed a fully connected neural network model of associative memory in which information can be stored by distribution of it among the various nodes in the architecture, and later can be recalled when system is in dynamically relaxed state. Hopfield used the Hebbian learning rule [12], to prescribe the weight matrix. This type of network will most likely be trapped in non-optimal local minima close to the starting point, which is not desired. The presence of false minima will increase the probability of error in recall of the stored pattern. The problem of false minima can be reduced by adopting the any evolutionary algorithm to accomplish the search for global minima. Genetic algorithm is such an evolutionary search technique and has traditionally been used in optimization [7-8]. In this, a random initial population of individuals is generated, each of which represents a potential solution to the problem. Each member of that population’s fitness as a solution to the problem is evaluated against some known criteria. Members of the population are then selected for reproduction based upon that fitness, and a new generation of potential solutions is generated from the offspring of the fit individuals. The process of evaluation, selection, and recombination is iterated until the population converges to an acceptable solution.

A lot of work can be found in literature [13-15] regarding the evolution in neural networks. Evolution has been introduced in neural networks at three levels: architectures, connection weights and learning rules [16]. Our focus here is exploring the evolution at connection weights level. Earlier work of evolution of connection weights in Hopfield neural network with GA can be found in [17-19] and [20-23]. We do not find much references of training the network and then recalling of patterns - both by evolutionary technique simultaneously.

In the present approach, we have made an effort to implement the evolution of weight matrices for both training and recall purposes simultaneously; using a genetic algorithm. The advantage of using this approach is that it is minimizing the randomness from the genetic algorithms because, instead of starting from the random solution, it starts from approximate optimum solution. Therefore, the process of recall is more efficient and relatively faster compared to earlier implementations.

## **3. METHODOLOGY AND IMPLEMENTATION DETAILS**

For testing the above discussed approach, an architecture of symmetric Hopfield neural network of size 36 (i.e. 6X6) nodes was created. The patterns used for the experiment were randomly generated using bipolar values +1 and -1. The total numbers of patterns taken for the experiment were 10.

### **3.1 EVOLUTION OF WEIGHT MATRICES USING HYBRID APPROACH (GENETIC ALGORITHM AND HEBB’S LAW) FOR STORING PATTERNS**

For storing a pattern, there must be a stable state which should satisfy the following activation dynamics equation given by Hebb:

$$f\left(\sum_j w_{ij} \cdot s_j\right) = s_i$$

Here initial weights have been considered as  $w_{ij} \approx 0$  (near to zero) for all i's and j's.

Now the updation of weight matrices takes place as follows:

$$w_{ij}^{new} = w_{ij}^{old} + \sum_{i,j} s_i s_j$$

After storing one pattern a weight matrix of the order N is obtained. Now, we apply population generation technique to evolve a population of weight matrices from the above original parent weight matrix. The original weight matrix remains unchanged during the evolution. Total N + 1 numbers of chromosomes are produced after using the mutation and elitism. Each chromosome is having a fixed length of  $N \times N$  alleles and can be represented as follows:

0	$S_1 S_2$	$S_1 S_3$	-----	$S_1 S_N$	$S_2 S_1$	-----	$S_N S_1$	$S_N S_2$	$S_N S_3$	-----	0
---	-----------	-----------	-------	-----------	-----------	-------	-----------	-----------	-----------	-------	---

Each component of the original weight matrix  $w_{ij}$  i.e.  $S_i S_j$  is multiplied by one of these alleles.

We denote the ith allele of the nth chromosome as  $A_i^n$ . Each chromosome modifies the original weight matrix and produces  $N$  weight matrices slightly different from the original one.

For each of these N weight matrices, we try to store the remaining patterns in the taken Hopfield architecture using Hebb's equation and filter in those weight matrices with which storing of patterns remains successful. This process is repeated until one (or more) optimal weight matrix(ces) are obtained at which all patterns are stored successfully.

### 3.2 GENETIC ALGORITHM IMPLEMENTATION FOR RECALLING PROTOTYPE PATTERNS

In recalling process, a population of weight matrices is produced randomly from the parent weight matrix(ces) those obtained after the storing process. Same population generation technique is used which is discussed above.

#### *Fitness Evaluation*

Selection of better or efficient next generation of weight matrices, fitness evaluation function has been used. When one of the stored patterns  $X^L$  is given to the network as an initial state, the state of neurons varies from time to time until  $X^L$  becomes a fixed point. In order to store the pattern in the network, these two states must be similar. The similarity as a function of time is defined by,

$$z^L = \frac{1}{N} \sum_{i=1}^N x_i^L S_i^L(t)$$

Here  $S_i^L(t)$  is the state of the  $i$ th neuron at time  $t$ . The fitness function is evaluated as follows:

$$f = \frac{1}{NL} \sum_{i=1}^N \sum_{l=1}^L z^L(t)$$

If the value of  $f$  is 1, it implies that all the initial patterns have been stored as fixed points. Thus, we consider only the generated weight matrices those consist with fitness evaluation value 1. Thus filtered new population will be used for generating the next better population of weight matrices with the crossover operator.

### **Crossover Operator**

Crossover is an operation which is responsible for the recombination of the selected population of weight matrices. This operator forms a new solution by taking some parameters from one parent and exchanging it with another at the very same point. Hence, on applying this crossover operator with the constraint that the number of chromosomes or components selected for exchange should be equal from both the weight matrices. Newly generated weight matrices are again evaluated against the fitness function defined above.

This process will continue for all the weight matrices from the population. It is possible to obtain more than one optimal weight matrices for the prototype pattern recalling.

The different parameters used in experiments are summarized below in Table 1 and Table 2:

Table 1: Parameters used in training

Parameter	Value
initial states of neurons	randomly generated values either -1 or +1
threshold value	0.00

Table 2: Parameters used in GA implementation

Parameter	Value
initial states of neurons	randomly generated values either -1 or +1
threshold value	0.00
mutation probability	0.5
mutation population size	N+1
crossover population size	N*N

## **4. RESULTS AND DISCUSSION**

The results being discussed in this section show that there is a significant improvement in recalling success rate of the given patterns with induced error while 'evolution of weight matrices' has been used for storing the patterns.

Each experiment constitute of 1000 runs of recalling each pattern with different level of errors, after storing them by both methods. In recalling, the success is considered only if the recalling of a pattern is made within 20 iterations (i.e. mutation, elitism, crossover operation and fitness evaluation function). The errors have been induced in the patterns randomly by flipping bits at different positions.

Table 3: Recalling Success results while there was no error, 1-bit error and 2-bits error induced in prototype input patterns

	Pattern No.	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
	→										
<b>Recall Success (in %) using GA</b>	While storing patterns without Evolution of Weight Matrices	100	100	100	100	100	100	100	100	100	100
	While storing patterns by Evolution of Weight Matrices	100	100	100	100	100	100	100	100	100	100

Table 4: Recalling Success results while there was 3-bits error induced in prototype input patterns

	Pattern No.	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
	→										
<b>Recall Success (in %) using GA</b>	While storing patterns without Evolution of Weight Matrices	78.3	89.8	78.9	75.9	83.0	79.4	77.6	76.7	78.9	69.0
	While storing patterns by Evolution of Weight Matrices	98.4	97.7	100	99.9	98.9	99.3	99.8	99.1	95.6	100

Table 5: Recalling Success results while there was 4-bits error induced in prototype input patterns

	Pattern No.	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
	→										
<b>Recall Success (in %) using GA</b>	While storing patterns without Evolution of Weight Matrices	54.3	32.4	38.2	45.2	43.2	44.4	40.6	39.5	38.0	47.5
	While storing patterns by Evolution of Weight Matrices	85.4	76.6	72.1	75.3	72.1	69.6	87.8	71.2	70.1	68.3

Table 3 shows that the results of recalling are perfectly same for both methods – either we store the patterns by simple Hebb’s law or by evolution of weight matrices followed by Hebb’s law. In both cases, patterns were correctly recalled 100% time. But significant changes have been observed (table 4 and 5) while the induced error in the presented prototype input patterns has been 3-bits and 4-bits. This shows that within the given framework of the simulation, if the training of the network has been done using evolution it produces better results in recalling the erroneous patterns.

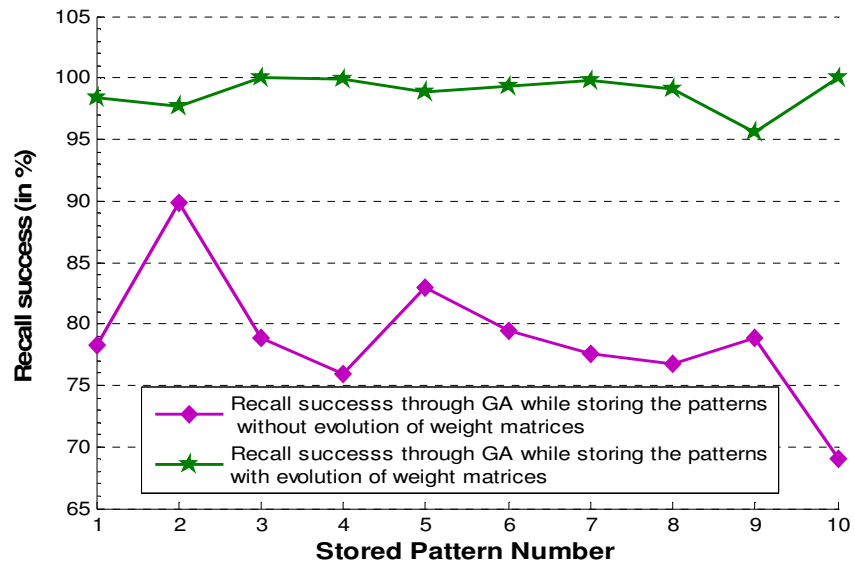


Figure 1: Comparison Graph of Recalling Success with 3-bits error induced in prototype input patterns using both approaches – storing with evolution and without evolution of weight matrices

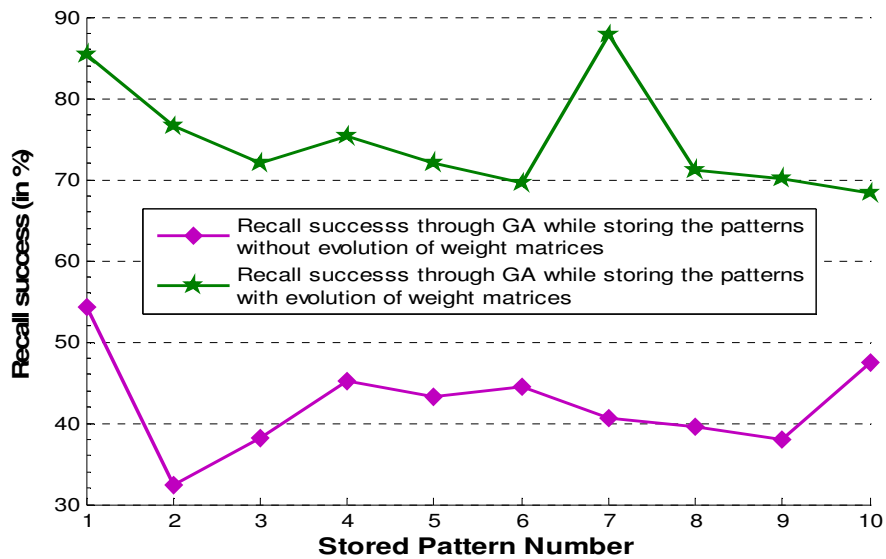


Figure 2: Comparison Graph of Recalling Success with 4-bits error induced in prototype input patterns using both approaches – storing with evolution and without evolution of weight matrices

Figure 1 and 2 are presenting the comparison chart of performance of two approaches (i.e. recall by GA after storing without and with evolution of weight matrices methods) graphically based on results provided in Tables 4 and 5.

The simulation program was developed in MATLAB 7.0. Due to limitation of resources, we could execute only 5-runs (i.e. for no error, 1-bit error, 2-bit error, 3-bit error and 4-bit error) and could not go beyond that.

## 5. CONCLUSION

In the present paper, we have implemented a genetic algorithm for storing the patterns in Hopfield Neural Network Associative Memory and later recalling these patterns with induced noise again using GA. The implementation yielded encouraging results but the work is still in early stage and the model is to be tested on a larger data set with a larger induced errors. Also, due to the limited computational power, the network size tested has been limited to 36 nodes which may be increased to the higher side.

## REFERENCES

1. Ubiquity, Vol. 4, Issue 37, Nov. 12 – 18 (2003) <http://www.acm.org/ubiquity/>
2. Freeman J.A., Skapura D.M.: *Neural Networks: Algorithms, Applications and Programming Techniques*, Reading, MA: Addison Wesley (1991)
3. Antognetti P., Milutinovic V.: *Neural Networks: Concepts, Applications, and Implementations* (Eds.), Vol I-IV, Prentice Hall, Englewood Cliffs, NJ (1991)
4. Anderson J.A., Rosenfeld E.: *Neurocomputing: Foundations of Research* MIT Press, Boston, MA (1988)
5. Hinton G. E., Sejnowski T.J.: *Neural Network Architectures for AI*, Tutorial Number MP2, National Conference on Artificial Intelligence (AAAI-87) (July 1987)
6. Jain A. K., Robert P. W. D., Mao J.: Statistical Pattern Recognition: A Review, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1(2000) 4-33
7. Mangal M., Singh M.P.: Analysis of Multidimensional XOR Classification Problem with Evolutionary Feed forward Neural Networks, *International Journal on Artificial Intelligence Tools*, Vol. 16, No.1 (2007) 111-120
8. Mangal M., Singh M.P.: Analysis of Classification for the Multidimensional Parity-Bit-Checking Problem with Hybrid Evolutionary Feed-forward Neural Network, *Neurocomputing*, Vol. 70 (2007) 1511-1524
9. Burges C.J.C.: Handwritten digit string recognition, In M.A. Arbib, (eds): *The Handbook of Brain Theory and Neural Networks*, Cambridge, MA: MIT Press (1995) 447-450
10. Mori S.: *Historical review of theory and practice of handwritten character recognition, Fundamentals in Handwriting Recognition*, S. Impedovo (ed.), NATO ASI Series F Computer and Systems Sciences, Vol. 124, Springer-Verlag-(1994) 43-69
11. Hopfield J. J.: Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy Sciences, USA*, Vol. 79 (1982) 2554 - 2558
12. Hebb D.: *The Organization of Behavior, A Neuropsychological Theory*, Wiley, New York (1949)
13. Salcedo-Sanz, S., Yao X.: A Hybrid Hopfield Network-Genetic Algorithm Approach for the Terminal Assignment Problem, *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, Vol. 34, No. 6 (2004) 2343-2353
14. Yao, X., Xu, Y.: Recent Advances in Evolutionary Computation, *J. Comput. Sci. & Technol.*, Vol. 21 No. 1(2006) 1-18
15. Islam, M. M., Yao, X.: A Constructive Algorithm for Training Cooperative Neural Network Ensembles, *IEEE Transactions on Neural Networks*, Vol. 14, No. 4 (2003)
16. Yao, X.: Evolving artificial neural networks, *Proceeding of the IEEE*, Vol. 87, No. 9 (1999) 1423-1447
17. Imada, A., Araki K.: Genetic Algorithm Enlarges the Capacity of Associative Memory, *Proceedings of the 6th International Conference on Genetic Algorithms*. (1995) 413-420

18. Imada, A., Araki K.: Evolved Asymmetry and Dilution of Random Synaptic Weights in Hopfield Network Turn a Spin-glass Phase into Associative Memory. *The 2nd International Conference on Computational Intelligence and Neuroscience proceedings of Joint Conference of Computer Science*, Vol. 2 (1997) 223-226
19. Imada, A., Araki K.: Hopfield Model of Associative Memory as a Test Function of Evolutionary Computations, *The 1st International Workshop on Frontiers in Evolutionary Algorithms, Proceedings of Joint Conference of Computer Science*, Vol. 1 (1997) 180-183.
20. Shrivastava, S., Singh, M.P.: Performance evaluation of feed-forward neural network with soft computing techniques for hand written English alphabets, *Applied soft Computing Journal*, Vol. 11(1) (2011) 1156-1182
21. Kumar, S., Singh, M.P.: Pattern recalling analysis of English alphabets using Hopfield model of feedback neural network with evolutionary searching, *International Journal of Business Information Systems*, Vol. 6(2) (2010) 200-218
22. Che, Z.-G, Chiang, T.-A, Che, Z.-H.: Feed-forward neural networks training: A comparison between genetic algorithm and back-propagation learning algorithm, *International Journal of Innovative Computing, Information and Control*, Vol. 7(10) (2011) 5839-5850
23. Singh, T.P., Jabin, S., Singh, M. : Evolving Weight Matrices to increase the Capacity of Hopfield Neural Network Associative Memory using Hybrid Evolutionary Algorithm, *Proceedings of 2010 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC 2010)*, art. no. 5705809, pp. 434-438, doi 10.1109/ICCIC.2010.5705809