Open access Journal International Journal of Emerging Trends in Science and Technology

Performance Analysis of Hopfield Network Associative Memory using Evolutionary Algorithm for Superimposed Images of Numerals

Authors
Ruby Panwar, Amit Kumar, Meenakshi

ABSTRACT
This paper presents the implementation of a Hebbian learning rule and genetic algorithm to store and later, recall of superimposed images of numerals in Hopfield network associative memory. A set of ten objects (i.e. 0 to 9 numerals) has been considered as the pattern set. In the Hopfield network associative memory, the weighted code of input patterns provides an auto-associative function in the network. The storing of images is done by hebbian learning rule and recalling is done by using both hebbian rule and genetic algorithm. The simulated results shows that the genetic algorithm gives efficient results as compared to hebbian rule for superimposed images of numerals.

Keywords- *Hopfield Neural network, associative memory, hebbian learning rule, genetic algorithm, weight matrices, pattern recalling, population generation technique.*

INTRODUCTION

Building an *Intelligent Machine*, which can think and act completely as a human brain, is one of the upfront issues which is still to be realized by Computer Science fraternity. If we need to build such an *Intelligent Machine*, first we need to imitate the human brain. The human brain is difficult to study directly unlike animal brains. Indirect data such as brain waves pick up the electric potential of the brain surface, which gives us only crude information on the emotional state, and is not suitable for identifying information on the state of thinking, the essential part of the human brain function.

When a child is born, what does the child know? To our knowledge, the child knows only how to cry. The child probably does not know its parents. When the child grows, the step by step learning process begins. First, the child learns to drink milk. Then the child learns to identify its parents. Every time a child learns something, it is encoded into some portion of the brain. Yet there is a difference in the way the information is stored in brain. Some information or instances are "hard-coded" within the brain. As a result, we never forget certain things. For example, once we learn to swim, we never forget swimming. Though it appears normal to say that we know swimming, there is a mystery behind this. Why are we unable to forget the swimming? The reason might be that when we are fully trained to swim, it is hard-coded into our brain. There are many examples of unforgettable information. Another example is once we learn that 1 + 1 equals 2, we never forget that fact. Why? The reason is that it is completely learned. These examples demonstrate that we can learn, understand and remember certain things completely, partially and sometimes not at all. Depending on our capacity for learning, the information is stored in our brain.

Artificial neural networks (ANN) are a form of artificial intelligence modeled after how the brain processes information. In this biological paradigm there are neurons (nodes) which are small processing units. Neurons are connected to other neurons and these connections can have various integrities (weights). As signals travel through the network output is produced. It is the purpose of this experiment to see if the simplified artificial neural networks will function similarly to the complex human brain. Most importantly neural network models use some organizational principles such as learning, generalization, adaptivity and fault tolerance. ANN is useful only when the processing units are organized in a suitable manner to accomplish the given task.

For the Hopfield network, superimposed numerals are presented at the input and the network iterates until it reaches a steady state or a limit cycle. A pattern is recalled when stability is reached. In general, if the input pattern is some corrupted or noisy version of pattern stored in the network, then the recalled pattern should be ideal or uncorrupted version of the input pattern. Hopfield network will try to converge on the closest memory state whether the input state is valid or merely random noise. An associative memory is a primary application of the Hopfield network. The weights of the connections between the neurons have to be thus set that the states of the system corresponding with the patterns which are to be stored in the network are stable.

Developed by Holland (1975), an evolutionary searching (genetic algorithm) is a biologically inspired search technique. In simple terms, the technique involves generating a random initial population of individuals, each of which represents a potential solution to a problem Genetic algorithms have traditionally been used in optimization but, with a few enhancements, can perform classification, prediction and pattern association as well.

Much work has been done on the evolution of neural networks with GA. There have been a lot of researches which apply evolutionary techniques to layered neural networks. The first attempt to conjugate evolutionary algorithms with Hopfield neural networks dealt with training of connection weights. Evolution has been introduced in neural networks at three levels: architectures, connection weights and learning rules. The evolution of connection weights proceeds at the lowest level on the fastest time scale in an environment determined by architecture, a learning rule, and learning tasks. The evolution of connection weights introduces an adaptive and global approach to training, especially in the reinforcement learning and recurrent network learning paradigm. The evolution of learning rules can be regarded as a process of “learning to learn” in ANN’s where the adaptation of learning rules is achieved through evolution. The evolution of architectures enables ANN’s to adapt their topologies to different tasks without human intervention and thus provides an approach to automatic ANN design as both ANN connection weights and structures can be evolved.

In the present work, our objective is to analyze the performance of neural networks of Hopfield type for recalling of the already stored patterns with evolutionary algorithm. In this process, first the patterns of training set have been encoded in the neural network using conventional hebbian learning rule. It is expected that all the patterns of training set has been successfully stored as the associative memory feature of Hopfield type neural network. As a result of this learning process, we obtain the expected optimized weight matrix. Now, we employ the genetic algorithm to evolve the population of this approximate optimal weight matrix obtained by hebbian learning rule. The fitness of every evolved population of weight matrices is evaluated by using fitness evaluation function. This process of evaluation of weight matrices continues till the last matrix of the generated population is not examined. This selected population of weight matrices reflects the optimal solution for recalling process in a way that on presentation of any noisy input pattern, it produces the correct corresponding stored pattern. The advantage of 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 recalling by proposed hybrid evolutionary system will be efficient and relatively faster compared to simple genetic algorithm and conventional hebbian approach. The detailed comparative analysis of the results obtained during the simulation is presented and discussed with the help of graphs and tables.

SIMULATION DESIGN AND IMPLEMEN-TATION DETAILS

This section describes the experiments designed to evaluate the performance of Hopfield neural network with the genetic algorithm for the taken set of objects recalling.

Set of Patterns Used For Training

The patterns used for the simulations are shown in Figure 1. Each pattern consisted of a 6 X 6 pixel matrix representing an object of the set. White and black pixels are respectively assigned corresponding values of -1 and +1.

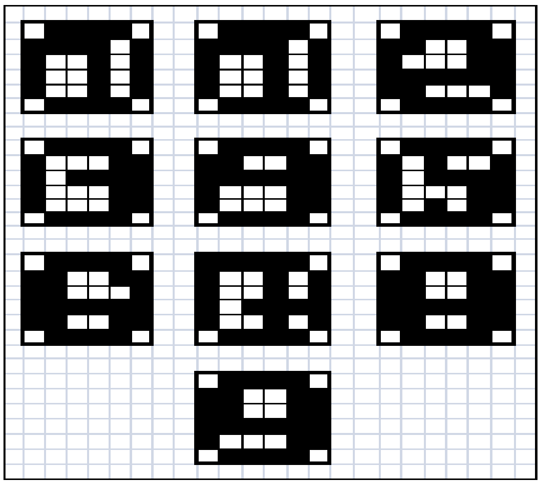


Figure 11: Set of patterns after superimposing two numerals (0 with other numerals 1 to 9)

The above figure shows the set of patterns formed after mixing numerals 1 to 9 with 0, same patterns can also be generated for other numerals 1 to 9. For example, mixed 0 with 1 can be written as:- [-11111-11111-111-1-11-111-1-11-111-1-11-11-11111-1]

Experiments

Experiments were taken on same Hopfield network architecture (i.e. 36 neurons network) by mixing two numerals with each other (shown in figure 2). Each experiment is based on two aspects which are *recalling the objects with Hebbian rule and recalling the same objects with genetic algorithm*. The inputs for different runs are carried out by superimposing numerals that are already stored in the network.

The genetic operators used in finding out results are summarized in Table 1.

Table 1: Genetic operator used in experiments

Training Algorithms	Genetic Operator Used
Hebbian rule	None
Genetic algorithm	Population generation technique (mutation + elitism), Crossover and Fitness evaluation technique

The parameters used in experiments are described in Table 2.

Table 2: Parameters used for Hebbian learning rule

Parameter	Value
Initial state of neurons	Randomly Generated Values Either −1 and 1
Threshold values of neurons	0.00

The task associated to the Hopfield neural networks in performing experiments is to store the taken set of objects as patterns with the appropriate recalling of the patterns by mixing numerals with one another.

Hopfield Neural Network

The proposed Hopfield model consists of N (36 = 6 X 6) neurons and N*N connection strengths. Each neuron can be in one of two states i.e. ±1, and L bipolar patterns have to be memorized in associative memory. For storing L patterns, we could choose a Hebbian rule given by the summation of the Hebbian terms for each pattern. i.e.

$w_{ij} = \frac{1}{L}\sum_{l=1}^{L} x_i^L x_j^L (i \neq j);$ and $w_{ii} = 0$………………….. (1)

To store 10 objects in a 36-unit bipolar Hopfield neural network, at each corresponding one stable state and that must satisfy the following activation dynamics equation:

$f(\sum_j w_{ij}.s_j) = s_i;$ where i=1,2,3…………………….N. (2)

Let the pattern set is X = {x^1, x^2 ,...................................x^L}……….. (3)

where $x^1 = (a_1^1, a_2^1, \ldots\ldots a_N^1)$

$x^2 = (a_1^2, a_2^2, \ldots\ldots a_N^2)$

.

.

.

$x^L = (a_1^L, a_2^L, \ldots\ldots a_N^L)$

where N = 1,2,3…………………….35 (4)

Now, the initial weights has been considered as $w_{ii} \approx 0$ (near to zero) for all i's and j's. Now,

$$w_{ij}^{new} = w_{ij}^{old} + X_i^1.X_j^1 \quad (5)$$

or $$w_{ij}^{new} = w_{ij}^{old} + \sum_{i,j} s_i s_j \quad [\because s_i = x_i \text{ for all i}] \quad (6)$$

and $$w_{ij}^{old} = w_{ij}^{new} \quad (7)$$

Similarly for the L^th pattern

$$w_{ij}^{L} = w_{ij}^{L-1} + \sum_{i,j} s_i^L s_j^L \quad (8)$$

We can generalize this as

$$W_{ij}^{L} = \frac{1}{L}\sum_{l=1}^{L}\sum_{i,j} s_i^l.s_j^l \quad (9)$$

or $$W^{L} = \frac{1}{L}\sum_{l=1}^{L} s^l (s^l)^T \quad (10)$$

After storing all the patterns, the final weight matrix looks like as follows

$$W^L = \begin{bmatrix} 0 & S_1S_2 & S_1S_3 & --------- & S_1S_N \\ S_2S_1 & 0 & S_2S_3 & --------- & S_2S_N \\ | & | & | & | & | \\ S_NS_1 & S_NS_2 & S_NS_3 & --------- & 0 \end{bmatrix}$$

The Genetic Algorithm Implementation

In this simulation, a population of weight matrices is produced randomly from the parent weight matrix when GA starts. In each generation, this population is modified through uniform random mutations and discrete crossovers and their fitness values are evaluated. According to the fitness values, individuals of the next generation are selected, using a (μ+λ) -strategy in ES terminology.

The cycle of generating the new population with better individuals and restarting the search is repeated until an optimum solution was found. The first fitness function is evaluating the best matrices of the weights population on the basics of the settlement of the network in the stable state corresponding to the stored pattern on the presentation of the already stored pattern as the input pattern. The fitness evaluation function is selection of the weight matrices on the basis of settlement of the network in the stable state corresponding to the correct or exact stored pattern on the presentation of prototype input pattern as the already stored pattern. It indicates that the stable states of the network will be used for the evaluation of the weight's population. Thus in the recalling process, stable state of the network corresponding to the stored pattern should be retained for the selected weight vector on the presentation of prototype input pattern. All those weight matrices which can successfully recall the respective stored patterns by providing the same as input pattern (with no error) at a time will be considered as fitted weight matrix.

RESULTS AND DISCUSSIONS

The results presented in this section demonstrate that large significant difference exists between the performance of genetic algorithm and conventional Hebbian rule for recalling objects those have been stored in Hopfield neural network using Hebbian learning rule. These results recommend that, in all cases, recalling of any approximate pattern through genetic algorithm outperformed the recalling of the same patterns through conventional Hebbian rule. While there are mixed patterns presented in the input pattern (tables given below) shows the results for recalling the stored objects using both Hebbian rule and genetic algorithm. In total 1000 times the recalling was made through both the algorithms separately for each object.

In table 3, the numeral 0 is superimposed with other numerals (1to 9) and then the results are carried out by recalling mixed pattern as 0 or 1 or not distinguished with Hebbian learning rule and genetic algorithm. When the superimposed patterns are recalled by using genetic algorithm then the recalling rate is higher than the hebbian learning rule. As patterns are superimposed with each other, obviously noise will arise. GA has shown that out of the mixed patterns it works better for identifying the original patterns, numerals in our case. There has been some cases when the recalling was made 100 percent like in the combination of numeral 6 and numeral 8, this is because of the superimposed pattern was at the least hamming distance from either of the original numeral. There has been cases when the recalling was significantly low, which again can be explained by the maximum hamming distance between superimposed pattern and either of the original numeral.

Table 3: Results when 0 superimposed with other numerals (1 to 9)

	After 1000 iterations success (%)							
	By Hebb's rule				By Genetic algorithm			
Numerals mixed together	*Numerals identified as '0'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*	*Numerals identified as '0'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*
(0,1)	0.6	1.4	98	2	45.6	12.6	41.8	58.2
(0,2)	0.9	0.7	98.4	1.6	69.6	30.2	0.2	99.8
(0,3)	0.6	0.8	98.6	1.4	41.0	32.1	26.9	73.1
(0,4)	0.9	1.3	97.9	2.1	55.6	44.3	0.1	99.9
(0,5)	0.7	1	98.3	1.7	31.6	25.6	42.8	57.2
(0,6)	1	0.8	98.2	1.8	35.6	54.6	9.8	90.2
(0,7)	0.7	1.3	98	2	29.6	36.8	33.6	66.4
(0,8)	1	0.8	98.2	1.8	35.6	32.6	31.8	68.2
(0,9)	0.7	1.2	98.1	1.9	26.8	35.9	37.3	62.7

Graph 1: Results when 0 superimposed with other numerals (1 to 9)

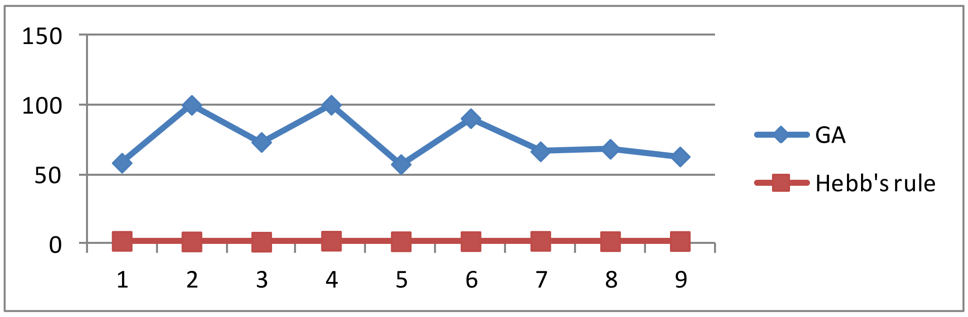


Table 4: Results when 1 superimposed with other numerals (0 to 9)

	After 1000 iterations success (%)							
	By Hebb's rule				*By Genetic algorithm*			
Numerals mixed together	*Numerals identified as '1'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*	*Numerals identified as '1'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*
(1,0)	*0.6*	*1.4*	*98*	*2*	*45.6*	*12.6*	*41.8*	*58.2*
(1,2)	*1.1*	*0.4*	*98.5*	*1.5*	*49.4*	*13.3*	*37.3*	*62.7*
(1,3)	*1*	*0.4*	*98.6*	*1.4*	*41.0*	*25.4*	*33.6*	*66.4*
(1,4)	*1*	*0.6*	*98.4*	*1.6*	*50.7*	*35.3*	*14.0*	*86.0*
(1,5)	*0.8*	*0.4*	*98.8*	*1.2*	*25.6*	*11.6*	*62.8*	*37.2*
(1,6)	*1.2*	*0.4*	*98.4*	*1.6*	*22.6*	*34.9*	*42.5*	*57.5*
(1,7)	*0.4*	*0.2*	*99.4*	*0.6*	*39.0*	*26.7*	*34.3*	*65.7*
(1,8)	*1.3*	*0.4*	*98.3*	*1.7*	*15.3*	*38.6*	*46.1*	*53.9*
(1,9)	*1*	*0.5*	*98.5*	*1.5*	*25.4*	*33.5*	*41.1*	*58.9*

Graph 2: Results when 1 superimposed with other numerals (0 to 9)

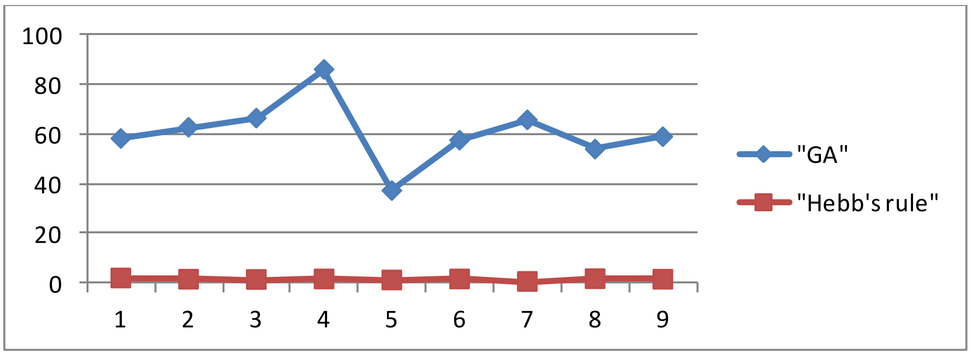


Table 5: Results when 2 superimposed with other numerals (0 to 9)

	After 1000 iterations success (%)							
	By Hebb's rule				By Genetic algorithm			
Numerals mixed together	*Numerals identified as '2'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*	*Numerals identified as '2'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*
(2,0)	0.9	0.7	98.4	1.6	69.6	30.2	0.2	99.8
(2,1)	1.1	0.4	98.5	1.5	49.4	13.3	37.3	62.7
(2,3)	0.3	0.5	99.2	0.8	36.0	46.4	17.6	82.4
(2,4)	0.4	0.9	98.7	1.3	33.2	63.1	3.7	96.3
(2,5)	0.4	0.8	98.8	1.2	39.0	58.6	2.4	97.6
(2,6)	0.2	0.1	99.7	0.3	26.5	15.2	58.3	41.7
(2,7)	0.4	0.9	98.7	1.3	32.5	63.3	4.2	95.8
(2,8)	0.2	0	99.8	0.2	19.2	9.3	71.5	28.5
(2,9)	0.2	0.4	99.4	0.6	19.6	41.3	39.1	60.9

Graph 3: Results when 2 superimposed with other numerals (0 to 9)

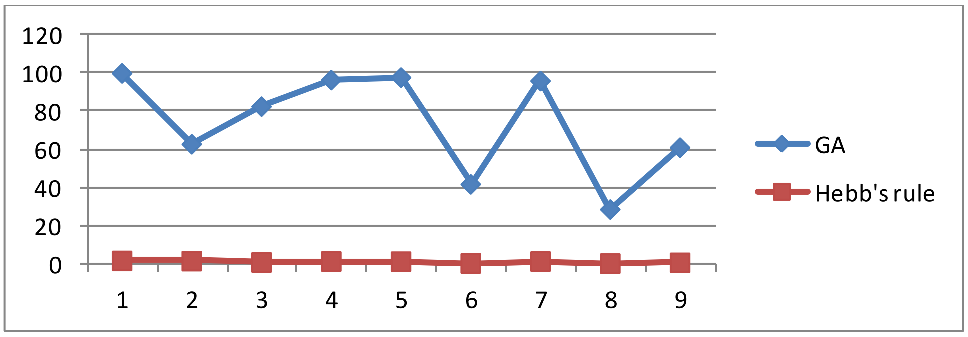


Table 6: Results when 3 superimposed with other numerals (0 to 9)

	After 1000 iterations success (%)							
	By Hebb's rule				By Genetic algorithm			
Numerals mixed together	*Numerals identified as '3'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*	*Numerals identified as '3'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*
(3,0)	0.6	0.8	98.6	1.4	41.0	32.1	26.9	73.1
(3,1)	1	0.4	98.6	1.4	41.0	25.4	33.6	66.4
(3,2)	0.3	0.5	99.2	0.8	36.0	46.4	17.6	82.4
(3,4)	0.2	0.5	99.3	0.7	11.3	31.6	57.1	42.9
(3,5)	0.4	0	99.6	0.4	25.3	2.6	72.1	27.9
(3,6)	0.6	0.4	99	1	28.2	24.3	47.5	52.5
(3,7)	0.6	0.1	99.3	0.7	29.3	10.6	60.1	39.9
(3,8)	0.6	0.3	99.1	0.9	31.8	25.6	42.6	57.4
(3,9)	0.2	0.5	99.3	0.7	13.6	38.9	47.5	52.5

Graph 4: Results when 3 superimposed with other numerals (0 to 9)

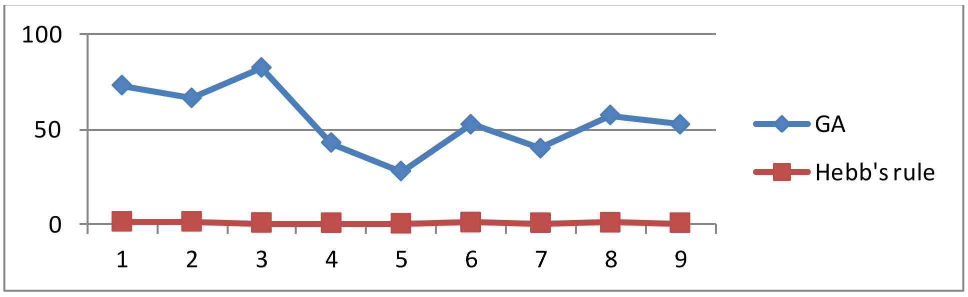


Table 7: Results when 4 superimposed with other numerals (0 to 9)

	After 1000 iterations success (%)							
	By Hebb's rule				By Genetic algorithm			
Numerals mixed together	*Numerals identified as '4'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*	*Numerals identified as '4'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*
(4,0)	0.9	1.3	97.8	2.2	55.6	44.3	0.1	99.9
(4,1)	1	0.6	98.4	1.6	50.7	35.3	14.0	86.0
(4,2)	0.4	0.9	98.7	1.3	33.2	63.1	3.7	96.3
(4,3)	0.2	0.5	99.3	0.7	11.3	31.6	57.1	42.9
(4,5)	0.6	0.4	99	1	51.6	32.1	16.3	83.7
(4,6)	0.9	0.3	98.8	1.2	62.3	25.3	12.4	87.6
(4,7)	0.8	0.8	98.4	1.6	42.3	41.3	16.4	83.6
(4,8)	0	1	99	1	12.0	45.9	42.1	57.9
(4,9)	0.4	0.5	99.1	0.9	28.6	39.6	31.8	68.2

Graph 5: Results when 4 superimposed with other numerals (0 to 9)

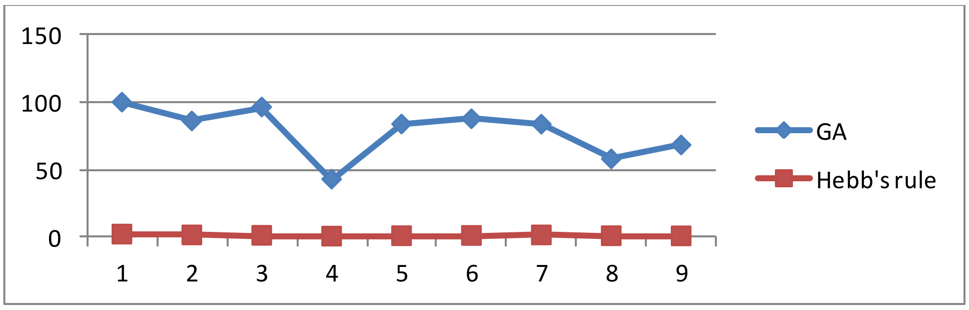


Table 8: Results when 5 superimposed with other numerals (0 to 9)

	After 1000 iterations success (%)							
	By Hebb's rule				By Genetic algorithm			
Numerals mixed together	*Numerals identified as '5'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*	*Numerals identified as '5'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*
(5,0)	0.7	1	98.3	1.7	31.6	25.6	42.8	57.2
(5,1)	0.8	0.4	98.8	1.2	25.6	11.6	62.8	37.2
(5,2)	0.4	0.8	98.8	1.2	39.0	58.6	2.4	97.6
(5,3)	0	0.4	99.6	0.4	25.3	2.6	72.1	27.9
(5,4)	0.6	0.4	99	1	51.6	32.1	16.3	83.7
(5,6)	0.9	0.5	98.6	1.4	35.9	21.9	42.2	57.8
(5,7)	0.4	0.8	98.8	1.2	22.7	35.6	41.7	58.3
(5,8)	0.9	0.4	98.7	1.3	54.6	21.9	23.5	76.5
(5,9)	0.8	0.6	98.6	1.4	39.8	28.6	31.6	68.4

Graph 6: Results when 5 superimposed with other numerals (0 to 9)

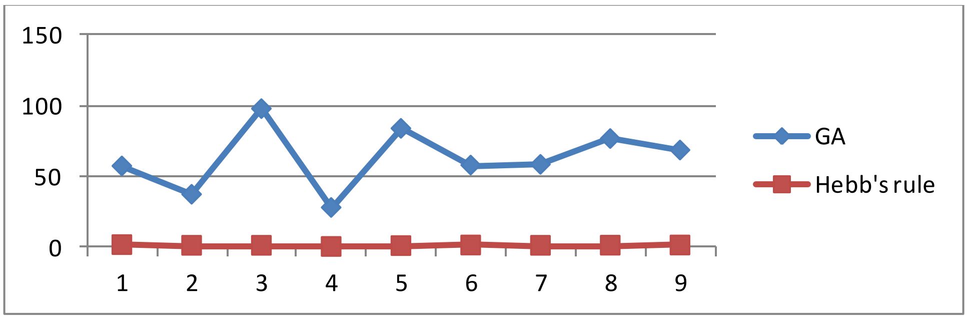


Table 9: Results when 6 superimposed with other numerals (0 to 9)

	After 1000 iterations success (%)							
	By Hebb's rule				By Genetic algorithm			
Numerals mixed together	*Numerals identified as '6'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*	*Numerals identified as '6'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*
(6,0)	1	0.8	98.2	1.8	35.6	54.6	9.8	90.2
(6,1)	1.2	0.4	98.4	1.6	22.6	34.9	42.5	57.5
(6,2)	0.2	0.1	99.7	0.3	26.5	15.2	58.3	41.7
(6,3)	0.4	0.6	99	1	28.2	24.3	47.5	52.5
(6,4)	0.9	0.3	98.8	1.2	62.3	25.3	12.4	87.6
(6,5)	0.9	0.5	98.6	1.4	35.9	21.9	42.2	57.8
(6,7)	0.1	1	98.9	1.1	19.6	56.9	23.5	76.5
(6,8)	0	100	0	100	6.9	93.1	0	100
(6,9)	0.1	0.4	99.5	0.5	21.8	45.9	32.3	67.7

Graph 7: Results when 6 superimposed with other numerals (0 to 9)

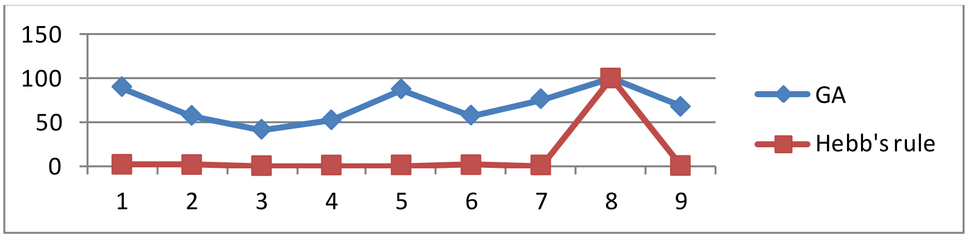


Table 10: Results when 7 superimposed with other numerals (0 to 9)

	After 1000 iterations success (%)							
	By Hebb's rule				By Genetic algorithm			
Numerals mixed together	*Numerals identified as '7'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*	*Numerals identified as '7'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*
(7,0)	0.7	1.3	98	2	29.6	36.8	33.6	66.4
(7,1)	0.4	0.2	99.4	0.6	39.0	26.7	34.3	65.7
(7,2)	0.4	0.9	98.7	1.3	32.5	63.3	4.2	95.8
(7,3)	0.1	0.6	99.3	0.7	29.3	10.6	60.1	39.9
(7,4)	0.8	0.8	98.4	1.6	42.3	41.3	16.4	83.6
(7,5)	0.4	0.8	98.8	1.2	22.7	35.6	41.7	58.3
(7,6)	0.1	1	98.9	1.1	19.6	56.9	23.5	76.5
(7,8)	0	0.4	99.6	0.4	5.9	35.6	58.5	41.5
(7,9)	0.8	0.5	98.7	1.3	45.9	22.9	31.2	68.8

Graph 8: Results when 7 superimposed with other numerals (0 to 9)

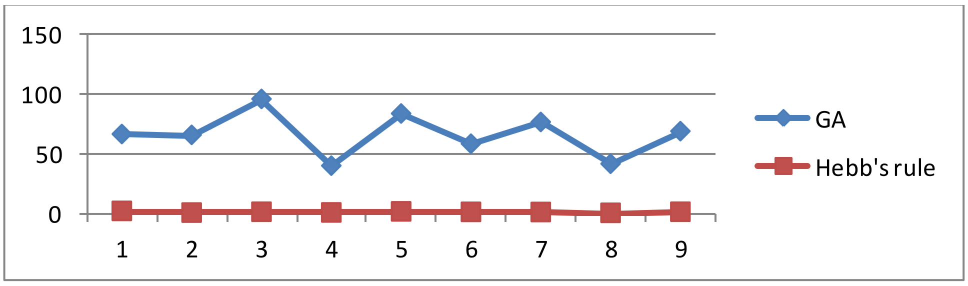


Table 11: Results when 8 superimposed with other numerals (0 to 9)

	After 1000 iterations success (%)							
	By Hebb's rule				By Genetic algorithm			
Numerals mixed together	*Numerals identified as '8'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*	*Numerals identified as '8'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*
(8,0)	1	0.8	98.2	1.8	35.6	32.6	31.8	68.2
(8,1)	1.3	0.4	98.3	1.7	15.3	38.6	46.1	53.9
(8,2)	0.2	0	99.8	0.2	19.2	9.3	71.5	28.5
(8,3)	0.3	0.6	99.1	0.9	31.8	25.6	42.6	57.4
(8,4)	0	1	99	1	12.0	45.9	42.1	57.9
(8,5)	0.9	0.4	98.7	1.3	54.6	21.9	23.5	76.5
(8,6)	0	100	0	100	6.9	93.1	0	100
(8,7)	0	0.4	99.6	0.4	5.9	35.6	58.5	41.5
(8,9)	100	0	0	100	100	0	0	100

Graph 9: Results when 8 superimposed with other numerals (0 to 9)

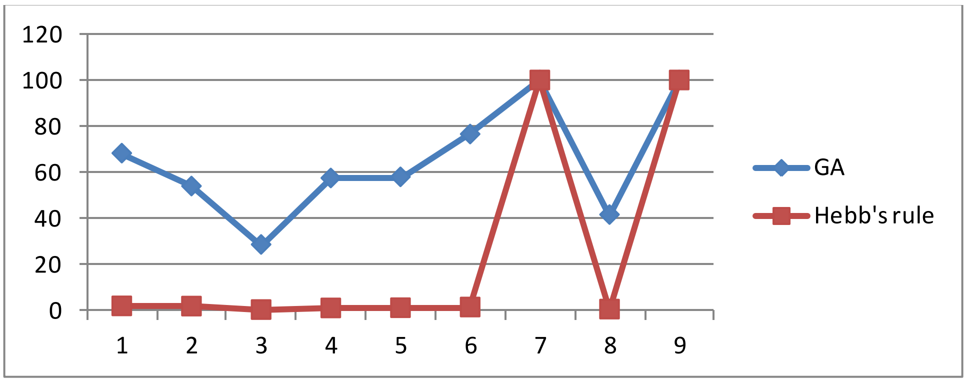
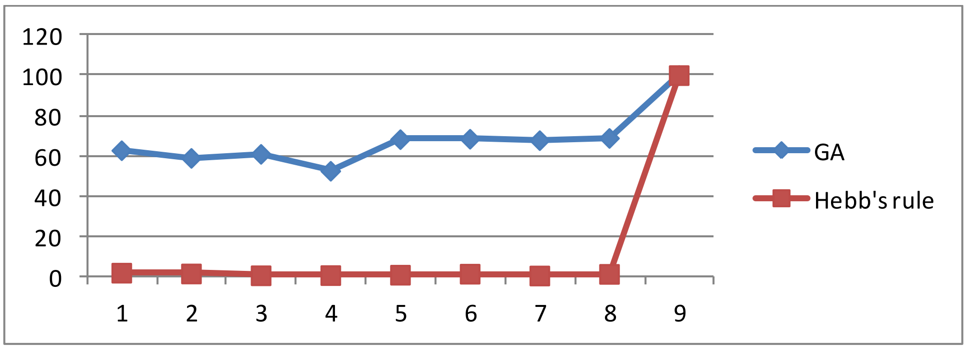


Table 12: Results when 9 superimposed with other numerals (0 to 9)

	After 1000 iterations success (%)							
	By Hebb's rule				By Genetic algorithm			
Numerals mixed together	*Numerals identified as '9'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*	*Numerals identified as '9'*	*Numerals identified as the other numerals from super-imposed images*	*Numerals not identified*	*Total success*
(9,0)	0.7	1.2	98.1	1.9	26.8	35.9	37.3	62.7
(9,1)	1	0.5	98.5	1.5	25.4	33.5	41.1	58.9
(9,2)	0.2	0.4	99.4	0.6	19.6	41.3	39.1	60.9
(9,3)	0.5	0.2	99.3	0.7	13.6	38.9	47.5	52.5
(9,4)	0.4	0.5	99.1	0.9	28.6	39.6	31.8	68.2
(9,5)	0.8	0.6	98.6	1.4	39.8	28.6	31.6	68.4
(9,6)	0.1	0.4	99.5	0.5	21.8	45.9	32.3	67.7
(9,7)	0.8	0.5	98.7	1.3	45.9	22.9	31.2	68.8
(9,8)	100	0	0	100	100	0	0	100

Graph 10: Results when 9 superimposed with other numerals (0 to 9)



CONCLUSION AND FUTURE SCOPE

In this research, two numerals are mixed together by which input pattern is generated and that input pattern is recalled by using two approaches Hebbian learning rule and Genetic algorithm. Hebbian learning rule fails while distinguishing mixed pattern as 0 or 1 whereas Genetic algorithm provides an efficient results. In future, the similar approach of genetic algorithm may be extended to the superimposed images preprocessed by using image processing techniques. The simulation was carried using Matlab and on a single machine. If parallel machines are to be used, we may get faster results and more maneuvering in the algorithm (like refinement in fitness functions etc) can be done and results may be analyzed.

REFERENCES

1. Tanaka-Ynamawaki,M., (1992) "Human Generated Random Numbers and a Model of the Human Brain Functions", Department of Computer Science and Systems Engineering Faculty of Engineering, Miyazaki University, Miyazaki Japan, pp 889-2192.
2. Jesan,J.P. and Lauro,M.D., (2003) "Human Brain and Neural Network Behaviour". ACM publication.
3. Findley,D.M., (2009) "Thinking Like Human: A study of Human Brain and artificial Neural Network." Westmoore High School 12613 S. Western Ave.Oklahoma City, pp. 73170.
4. Mangal, M. & Singh, M. P., (2007) "Analysis of Multidimensional XOR Classification Problem with Evolutionary Feed-forward Neural Networks", International Journal on Artificial Intellig-ence Tools, Vol. 16, No.1, pp.111-120.
5. Yao, X., (1999) "Evolving artificial neural networks", Proceeding of the IEEE, vol.87, no. 9,pp.1423-1447.
6. Pal,S.K., De,S. & Ghosh, A., (1997) "Designing Hopfield type networks using genetic algorithms and its comparison with simulated annealing", Intl Journal of Pattern Recognition and Artificial Intelligence, Vol. 11, No. 3 pp-447-461.
7. Salcedo-Sanz,S. & Yao,X., (2004) "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, pp. 2343-2353.
8. Jiyou Xu,J.H., & Yao,X., (2000) "Solving Equations by Hybrid Evolutionary Computation Techniques", IEEE Transactions on Evolutionary Computation, Vol. 4, Issue 3, pp. 295-304.
9. Mangal, M. & Singh, M. P., (2006) "Handwritten English Vowels using Hybrid Evolutionary Feed-forward Neural Network", Malaysian Journal of Computer Science, Vol. 19, No. 2, pp. 169-187.
10. Imada, A., Araki K.., "Evolved Asymmetry and Dilution of Random Synaptic Weights in HopfieldNetwork 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.
11. 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.
12. 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.
13. 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.
14. 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.