International Journal of Management & Information Technology
Volume 3, No 1, January, 2013

Performance Evaluation of Pattern Storage Network of Associative memory with Sub-optimal GA for Hand written Hindi 'SWARS'

Rajesh Lavania[1], Manu Pratap Singh[2]
Assistant Professor[1] , Reader[2]
Department of Computer Science Engineering, Institute of Engineering & Technology,
Dr. B. R. Ambedkar University, Khandari, Agra

ABSTRACT

In this paper we are performing the evaluation of Hopfield neural network as Associative memory for recalling of memorized patterns from the Sub-optimal genetic algorithm for Handwritten *'SWARS'* of Hindi language. In this process the genetic algorithm is employed from sub-optimal form for recalling of memorized patterns corresponding to the presented noisy prototype input patterns. The sub-optimal form of GA is considered as the non-random initial population or solution. So, rather than random start, the GA explores from the sum of correlated weight matrices for the input patterns of training set. The objective of this study is to determine the optimal weight matrix for correct recalling corresponds to approximate prototype input pattern of Hindi *'SWARS'*. In this study the performance of neural network is evaluated in terms of the rate of success for recalling of memorized Hindi *'SWARS'* for presented approximate prototype input pattern with GA in two aspects. The first aspect reflects the random nature of the GA and the second one exhibit the suboptimal nature of the GA for its exploration. The simulated results demonstrate the better performance of network for recalling of the memorized Hindi SWARS using genetic algorithm to evolve the population of weights from sub-optimal weight matrix.

Keywords: Hopfield neural networks, Associative memory, Pattern Storage, Genetic algorithm, Evolutionary Algorithm.

1. INTRODUCTION

Pattern storage & recalling i.e. pattern association is one of prominent method for the pattern recognition task that one would like to realize using an artificial neural network (ANN) as associative memory feature. Pattern storage is generally accomplished by a feedback network consisting of processing units with non-linear bipolar output functions. The Hopfield neural network is a simple feedback neural network (NN) which is able to store patterns locally in the form of connection strengths between the processing units. This network can also work for the pattern completion on the presentation of partial information or prototype input pattern. The stable states of the network represent the memorized or stored patterns. Since the Hopfield neural network with associative memory [1-2] was introduced, various modifications [3-10] are developed for the purpose of storing and retrieving memory patterns as fixed-point attractors. The dynamics of these networks have been studied extensively because of their potential applications [21-24]. The dynamics determines the retrieval quality of the associative memories corresponding to already stored patterns. The pattern information in an unsupervised manner is encoded as sum of correlation weight matrices in the connection strengths between the proceeding units of feedback neural network using the locally available information of the pre and post synaptic units which is considered as final or parent weight matrix.

Hopfield [1] proposed a fully connected neural network model of associative memory in which we can store information by distributing it among neurons, and recall it from the dynamically relaxed neuron states. If we map these states corresponding to certain desired memory vectors, then the time evolution of dynamics leads to a stable state. These stable states of the networks represent the stored patterns. Hopfield used the Hebbian learning rule [25] to prescribe the weight matrix for establishing these stable states. A major drawback of this type of neural networks is that the memory attractors are constantly accompanied with a huge number of spurious memory attractors so that the network dynamics is very likely to be trapped in these attractors [6], and thereby prevents the retrieval of the memory attractors. Hopfield type networks also 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 evolutionary algorithm to accomplish the search for global minima. There have been a lot of researchers who apply evolutionary techniques (simulated annealing and Genetic algorithm) to minimize the problem of false minima [10]. Imades & Akira [10-19] have applied evolutionary computation to Hopfield neural networks in various ways. A rigorous treatment of the capacity of the Hopfield associative memory can be found in [20]. The Genetic algorithm has been identified as one of prominent search technique for exploring the global minima in Hopfield neural network [24].

Developed by Holland [26], a 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. Each member of this population evaluates from a fitness function which is selected against some known criteria. The selected members of the population from the fitness function are used to generate the new population as the members of the population are then selected for reproduction based potential solutions from the operations of the genetic algorithm. The process of evaluation, selection, and recombination is iterated until the population converges to an acceptable optimal solution. Genetic algorithms (GAs) require only fitness information, not gradient information or other internal knowledge of a problem as in case of neural networks. Genetic algorithms have traditionally been used in optimization but, with a few enhancements, can perform classification, prediction and pattern association as well [27-29]. The GA has been used very effectively for function optimization and it can perform efficient searching for approximate global minima. It has been observed that the pattern recalling in the Hopfield type neural networks can be performed efficiently with GA [13]. The GA in this case is expected to yield alternative global optimal values of the weight matrix corresponding to all stored patterns.

The conventional Hopfield neural network suffers from the problem of non-convergence and local minima on increasing the complexity of the network. However, GA is particularly good to perform efficient searching in large and complex space to find out the global optima and for convergence. Considerable research into the Hopfield network has shown that the model may trap into four types of spurious attractors. Four well identified classes of these attractors are mixture states [4], spin glass states [58], compliment states and alien attractors [59]. As the complexity of the of the search space increases, GA presents an increasingly attractive alternative for pattern storage & recalling in Hopfield type neural networks of associative memory.

The neural network applications address problems in pattern classification, prediction, financial analysis, and control and optimization [30]. In most current applications, neural networks are best used as aids to human decision makers instead of substitutes for them. Genetic algorithms have helped market researchers performing market segmentation analysis [31]. Genetic algorithms and neural networks can be integrated into a single application to take advantage of the best features of these technologies [32].

Much work has been done on the evolution of neural networks with GA [33-37]. There have been a lot of researches which apply evolutionary techniques to layered neural networks. However, their applications to fully connected neural networks remain few so far. The first attempt to conjugate evolutionary algorithms with Hopfield neural networks dealt with training of connection weights [45] and design of the neural network architecture [46,47], or both [48-51]. Evolution has been introduced in neural networks at three levels: architectures, connection weights and learning rules [38]. 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. Training of neural networks using evolutionary algorithms started in the beginning of 90’s [16,52]. Reviews can be found in [24,27-29,35]. Cardenas et al. [53] presented the architecture optimization of neural networks using parallel genetic algorithms for pattern recognition based on person faces. They compared the results of the training stage for sequential and parallel implementations. The genetic evolution has been used as data structures processing for image classification [54].

In this paper we are exploring the GA for efficient recalling of memorized patterns as auto associative memory from the Hopfield neural network corresponding to the presented input pattern vector of handwritten Hindi *‘SWARS’* characters. The recalling in this associative memory network is performed under the consideration of reducing the effect of false minima by using evolutionary searching method like genetic algorithm. In this approach the GA starts from the suboptimal weight matrix as the initial population of solution. The suboptimal weight matrix reflects the encoded patterns information of the training set by using unsupervised Hebbian learning rule i.e. sum of correlation weight matrices. Each correlation term is corresponding to individual pattern information. Hence, the GA starts from the sum of correlation matrices for training set which we call as parent weight matrix, and it determines the optimal weight matrix for the presented noisy prototype input patterns of the handwritten *‘SWARS’* of Hindi language. The performance of pattern storage network is evaluated as rate of success in recalling of correct memorized pattern correspond to the presented prototype input pattern of handwritten *‘SWARS’* with GA which starts from sub-optimal solution i.e. sub-optimal GA. The simulated results indicate the better performance of the suboptimal genetic algorithm (SGA) as compared with Hebbian rule in success rate for recalling of correct memorized *‘SWARS’* characters.

In the following sections we will present the description of patterns used for training, the Hopfield neural network used for storing the patterns, the GA used for recalling the already stored patterns, experiments detail, discussion of our results obtained through simulation, and the conclusion of our investigations.

2. SAMPLE PATTERN REPRESENTATION

The patterns used for the simulations are shown in Figure 1. Each pattern consists of a 5 X 5 pixel matrix representing a handwritten character of Hindi *‘SWARS’*. White and black pixels are respectively assigned corresponding values of -1 and +1.

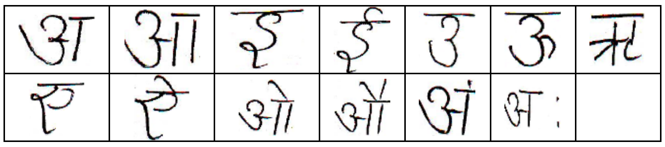

Figure 1: The set of patterns used for training

Now, the input pattern vector for the storage corresponding to hand written character of Hindi *‘SWARS’* is constituted with the series of bipolar values +1 and -1. For example, the pattern vector for character **v** can be written as:

[1 -1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 -1 1 1 1 1]

In the general form we can represent the l^{th} pattern vector as:

$$x^l = [a_1^l, a_2^l, - - - - - - - - - - - - a_{25}^l] \tag{1}$$

where l= 1 to 13 and i, j= 1 to 25

3. THE HOPFIELD NEURAL NETWORK

The proposed Hopfield model consists of N (25 = 5 X 5) neurons and $N \times N$ connection strengths. Each neuron can be in one of the two states i.e. ±1, and $L(13)$ bipolar patterns have to be memorized in the Hopfield neural network of associative memory.

Hence, to store $L(13)$ number of patterns in this pattern storage network, the weight matrix **w** is usually determined by the Hebbian rule as follows:

$$w = \sum_{l=1}^{L} x_l^T x_l \tag{2}$$

or, $$w_{ij}^l = \tfrac{1}{N}\left(a_i^l a_j^l\right) \tag{3}$$

and, $$w^l = \frac{1}{N}\sum_{i,j}^{N} a_i^l a_j^l \tag{4}$$

or, $$w_{ij} = \frac{1}{N}\sum_{l=1}^{L} a_i^l a_j^l \ (i \neq j) \text{ and } w_{ii} = 0 \tag{5}$$

where { $a_i^l, i = 1,2,3----N; l = 1,2,3----L$ and $i \neq j$; with set of L patterns to be memorized and N is the number of processing units}. The network is initialized as:

$$s_i^l(0) = a_i^l(0) \quad \text{for all} \quad i = 1 \quad \text{to} \quad N \tag{6}$$

The activation value and output of every unit in Hopfield model can represent as:

$$y_i = \sum_{j=1}^{N} w_{ij} s_i(t) \ ; \ i, j = 1,2,3----N; i \neq j \tag{7}$$

and $$s_i(t+1) = \text{sgn}(y_i) \tag{8}$$

where $\text{sgn}(y_i) = \pm 1$ for $y_i \geq 0$ and $y_i < 0$ respectively

Associative memory involves the retrieval of a memorized pattern in response to the presentation of some prototype input patterns as the arbitrary initial states of the network. These initial states have a certain degree of similarity with the memorized patterns and will be attracted towards them with the evaluation of the neural network.

Hence, in order to memorize 13 handwritten Hindi *'SWARS'* of in a 25-unit bipolar Hopfield neural network, there should be one stable state corresponding to each stored pattern. Thus at the end, the memory pattern should be fixed-point attractors of the network and must satisfy the fixed-point condition as:

$$y_i^l s_i^l = \sum_{\substack{j=1 \\ i \neq j}}^{N} w_{ij} s_j^l \tag{9}$$

or, $$y_i^l = s_i^l \sum_{j=1}^{N} w_{ij} s_j^l \quad \text{where } y_i^l \geq 0 \tag{10}$$

Therefore, the following activation dynamics equation must satisfy to accomplish the pattern storage:

$$f\left(\sum_{j \neq i}^{N} w_{ij} s_j^l\right) = s_i^l; \tag{11}$$

where $i, j = 1,2,---, N; i \neq j$

Let the pattern set be $P = \{x^1, x^2, ---, x^L\}$

where $[x^1 = (a_1^1, a_2^1, ---, a_N^1),$

$x^2 = (a_1^2, a_2^2, ---, a_N^2),$

-

-

-

$x^L = (a_1^L, a_2^L, ---, a_N^L)$

with $N = 1,2,---,25$

and $L = 1,2,---,13]$. (12)

Now, the initial weights have been considered as $w_{ij} \approx 0$ (near to zero) for all $i's$ and $j's$. From the synaptic dynamics as vectors we have the following equation for encoding the patterns information as:

$$W^{new} = W^{old} + X^T.X \tag{13}$$

and $$W^{old} = W^{new} \tag{14}$$

similarly for the L th patterns, we have:

$$W^L = W^{L-1} + \left(X^L\right)^T X^L \tag{15}$$

Thus, after the learning for all the patterns, the final parent weight matrix can be represented as:

$$W^L = \begin{bmatrix} 0 & \frac{1}{N}\sum_{l=1}^{L} a_1^l a_2^l & \frac{1}{N}\sum_{l=1}^{L} a_1^l a_3^l & --- & \frac{1}{N}\sum_{l=1}^{L} a_1^l a_N^l \\ \frac{1}{N}\sum_{l=1}^{L} a_2^l a_1^l & 0 & \frac{1}{N}\sum_{l=1}^{L} a_2^l a_3^l & --- & \frac{1}{N}\sum_{l=1}^{L} a_2^l a_N^l \\ | & | & | & | & | \\ | & | & | & | & | \\ \frac{1}{N}\sum_{l=1}^{L} a_N^l a_1^l & \frac{1}{N}\sum_{l=1}^{L} a_N^l a_2^l & \frac{1}{N}\sum_{l=1}^{L} a_N^l a_3^l & --- & 0 \end{bmatrix} \tag{16}$$

Now, to represent W^L in the convenient representation form, let us assume following notations:

$S_1 S_2 = \sum_{l=1}^{L} a_1^l a_2^l$, $S_1 S_3 = \sum_{l=1}^{L} a_1^l a_3^l$, $S_1 S_N = \sum_{l=1}^{L} a_1^l a_N^l$,
$S_2 S_1 = \sum_{l=1}^{L} a_2^l a_1^l$, $S_2 S_3 = \sum_{l=1}^{L} a_2^l a_3^l$, $S_2 S_N = \sum_{l=1}^{L} a_2^l a_N^l$,

$$S_N S_1 = \sum_{l=1}^{L} a_N^l a_1^l \text{, } S_N S_2 = \sum_{l=1}^{L} a_N^l a_2^l \text{--, } S_N S_3 = \sum_{l=1}^{L} a_N^l a_3^l \quad (17)$$

So that, from equation (16) & (17) , we get:

$$W^L = \frac{1}{N}\begin{bmatrix} 0 & s_1 s_2 & s_1 s_3 & --- & s_1 s_N \\ s_2 s_1 & 0 & s_2 s_3 & --- & s_2 s_N \\ | & | & | & | & | \\ | & | & | & | & | \\ s_N s_1 & s_N s_2 & s_N s_3 & --- & 0 \end{bmatrix} \quad (18)$$

This square matrix is considered as the parent eight matrix because it represents the partial solution or sub-optimal solution for the pattern recalling corresponding to the presented prototype input pattern vector. The next generation population of solutions will be evolved from this sub-optimal weight matrix. Thus, we consider that the GA will now start from this sub-optimal initial solution rather than random one. So, we consider this as sub-optimal GA. Hopfield suggested that the maximum limit for the storage is $0.15N$ in a network with N neurons, if a small error in recalling is allowed. Later, this was theoretically calculated as $p = 0.14N$ by using replica method [3]. Wasserman [39] showed that the maximum number of memories 'm' that can be stored in a network of 'n'neurons and recalled exactly is less that cn^2 where 'c'is a positive constant greater than one. It has been identified that the storage capacity strongly depends on learning scheme. Researchers have proposed different learning schemes, instead of the Hebbian rule to increase the storage capacity of the Hopfield neural network [55,56] Gardner showed that the ultimate capacity will be $p = 2N$ as a function of the size of the basin of attraction [57]. Imada and Akira [10] applied the genetic algorithm to the Hopfield model as an associative memory, and obtained the capacity of 33% of the number of neurons. It has also been observed that the possibility of false minima may occur during the recalling of memorized patterns. However the GA has been identified as being particularly good at performing efficient searching in large and complex spaces to determine the global optima or minimize the possibility of false minima. Kumar and Singh investigated [24] that the GA of the evolutionary algorithms is much suitable choice to reduce the affect of false minima from the Hopfield neural network during the recalling of memorized patterns.

4. PATTERN RECALLING WITH GENETIC ALGORITHM

In GA implementation we consider the cycle of generating the new population with better individuals and restart the search until an optimum solution is found. In this process the two fitness evaluation functions have been used. The first fitness function is determining the best matrices of the weight populations those settle the network in a stable state corresponds to correct memorized pattern for presented input pattern. This input pattern is one of the memorized patterns. The second fitness evaluation function is selecting the weight matrices from the populations of the network in a stable state correspond to the correct memorized pattern for presented prototype noisy or approximate input pattern. It indicates that the stable states of the network will use for the evaluation of weight populations. Thus in the recalling process, stable state of the network correspond to the stored pattern should retain for the selected weight matrix on the presentation of prototype input pattern.

In this implementation process the two fitness evaluation functions are used. The first fitness evaluation function determines the suitable weight matrices which are responsible to generate the correct recalling of the memorized pattern for the error-free or exact input pattern that has been used in the training set. It means that, at the first level of filtering only those weight matrices will select which provide the correct pattern auto-association for the samples of training pattern set. Thus, at this level no approximate prototype input pattern is presented. It represents only weight matrices those exhibit the pattern association during the training of the network and should carry in the next generation of population, whereas the second fitness evaluation function is used after the crossover operator. The crossover operator is applied only to chromosomes those have been passed from the first fitness evaluation function. The second fitness evaluation function applies to determine the population of weight matrices those are responsible for recalling of correct memorized pattern for presented approximate or noisy prototype input pattern. Thus, the second fitness evaluation function is actually selecting the final population of chromosomes which are required for obtaining the optimal solution. The parameters used for genetic algorithm are summarized in Table 1.

Table 1: Parameter used for genetic algorithm.

Parameter/ Operation/ procedure	**Suboptimal Genetic Algorithm**		**Random Genetic Algorithm**
	$p_m = 0.001$	$p_m = 0.500$	
Chromosome length	$N \times N$	$N \times N$	$N \times N$
Mutation probability	0.001	0.500	1.000
Mutation population size	N	N	N
Crossover population size	$N \times N$	$N \times N$	$N \times N$
Crossover type	Uniform	Uniform	Uniform
Number of fitness functions	2	2	2
Initial population	Suboptimal weight matrix	Suboptimal weight matrix	Random weight matrix

4.1. The Mutation Operator

The mutation operator plays a secondary role in the genetic algorithm. Mutation performs the modification of the value of each gene of a solution with some probability p_m. Nevertheless the choice of p_m is critical to GA performance and has been studied by DeJong [40]. The typical value of mutation probability is in the range 0.005-0.05. The idea of adapting mutation and crossover to improve the performance of GAs has been used by researchers using the different criterion

[41-42]. Whitely et al. [43] idea for adaptation is based on the Hamming distance between solutions; while in Srinivas et al. [44] approach p_m is based on fitness values of the solutions.

The mutation operator produces the population of N weight matrices or chromosomes of same order as the original parent matrix on applying it N times. Thus, each chromosome is having a fixed length of $N \times N$ genes or alleles. In this process of mutation we select randomly any genes i.e. $s_i^r s_j^r$ from parent chromosome. In which r is the position of the gene in the parent chromosome or weight matrix. We consider another randomly generated chromosome of the same order as $N \times N$ in which on the same randomly selected position i.e. r, the allele values are non zero in the interval -1 and +1 and for other positions the values are zero i.e. $A_{ij}^r \neq 0 \; and \; A_{ij}^o = 0$, where A_{ij}^r is the value of the gene at the position r in the generated chromosome of order $N \times N$ and A_{ij}^o is the value of the gene at the position $'o'$ in the generated chromosome of order $N \times N$. Now, we add this randomly generated chromosome with the parent chromosome and generate the new chromosome or weight matrix as:

$$w_{ij}^{new} = s_i^r s_j^r + A_{ij}^r$$

and, $$w_{ij}^{new} = s_i^o s_j^o + A_{ij}^o \tag{19}$$

where $A_{ij}^r \neq 0 (o \neq r) \; and \; A_{ii}^o = 0$.

The steps for mutation operator:

Step 1: Generate the mutation positions in the chromosome randomly.

Step 2: Modify the parent chromosome at the positions generated in step 1, using equation (19).

Step 3: Repeat step 1 and 2 until a number N of mutated chromosome populations have been created.

4.2. Elitism

Elitism is used when creating each generation so that the genetic operators do not loose good solutions. This involves copying the Hebbian-encoded weight matrix i.e., the suboptimal solution unchanged in the new population, which includes W^L for creating the total number M (i.e. $N+1$) of chromosomes.

4.3. The First Fitness Evaluation

The first fitness evaluation function (f) is used for selecting a good or efficient next generation of weight matrices. Evaluation of f for each individual weight matrix is carried out with a set of randomly pre – determined patterns x^L . When one of the stored patterns x^l is given to the network as an initial state, the state of neurons varies over 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 [10]

$$z^l = \tfrac{1}{N} \sum_{i=1}^{N} x_i^l s_i^l (t) \tag{20}$$

Here $s_i^l(t)$ is the state of the i^{th} neuron at time t . In evaluating the fitness value, the temporal average overlap $\langle z^{\mu} \rangle$ is calculated for each stored pattern, as follows. First the total of the inner products of the initial states and states is calculated at each time of update not greater than a certain time t_0 . After that, these values are summed up over whole set of initial patterns [10] i.e.,

$$f = \tfrac{1}{t_0 L} \sum_{t=1}^{t_0} \sum_{l=1}^{L} z^l (t) \tag{21}$$

Here t_0 has been set to N (the number of processing units). We must note that fitness 1 implies that all the initial patterns have been stored as fixed points. Thus, we consider only those generated weight matrices that have the fitness evaluation value 1. Hence, all the selected weight matrices will be considered as the new generation of the population. This new population will be used for generating the next better population of weight matrices with the crossover operator.

4.4. The Crossover Operator

The power of GAs arises from crossover. Crossover is a structured and randomized exchange of genetic material between solutions, with the probability that ‘good’ solutions can generate ‘better’ ones. Thus, crossover is an operation which may be used to combine multiple parents and make off spring. This operator 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 them with ones from another at the very same point. Here, we are applying the recombination with the uniform crossover. In this process, the network selects randomly (with uniform distribution) a string of non – zero chromosomes from a selected weight matrix and exchanges it with string of non – zero chromosomes from another selected weight matrix. Thus, a large population of the weight matrices will be generated. Hence, on applying this crossover operator with the constraint that the numbers of genes or alleles selected for exchange should be equal for the two weight matrices, the modification has been made in the selected weight matrices as follows:

$$w_{1,N\times N}^{new} = w_{1,old} \bigcup [w_{1,t\times u}^{sel} \leftarrow w_{2,t\times u}^{sel}]$$

And $$w_{2,N\times N}^{new} = w_{2,old} \bigcup [w_{2,t\times u}^{sel} \leftarrow w_{1,t\times u}^{sel}] \tag{22}$$

where $t,u \leq N$

Here $w_{1,N\times N}^{new}$ & $w_{2,N\times N}^{new}$ are newly generated weight matrices after crossover operator, $w_{1,old}$ & $w_{2,old}$ are two selected weight matrices randomly from mutated population to perform the crossover operator, $w_{1,t\times u}^{sel}$ & $w_{2,t\times u}^{sel}$ are the selected sub matrices randomly from $w_{1,old}$ & $w_{2,old}$ respectively for exchange to perform crossover operator, and $\bigcup$ is the operator for construction of new weight matrix through crossover operator.

In this way, we can generate the population of K weight matrices i.e. $N \times N$ by applying the crossover operator on the population of M chromosomes.

Thus, we have the new large population of K weight matrices from the crossover operator as follows:

$$\left\{w_{1,N\times N}^{new}, w_{2,N\times N}^{new}, ---, w_{K,N\times N}^{new}\right\} \tag{23}$$

Steps for Crossover Operation

Step 1: Initialize the crossover population size limit with value $N \times N$.

Step 2: Extract two chromosomes from among the $M(i.e.N+1)$ chromosomes randomly.

Step 3: Obtain a sub matrix of order $t \times u$ randomly in each extracted chromosome for exchanging the values.

Step 4: Exchange the sub matrices between the chromosomes.

Step 5: Include both chromosomes in the crossover population.

Step 6: Check whether the population size is equal to $N \times N$. If not, go to step 2 again.

4.5. The Second Fitness Evaluation

In the process of recalling the stored pattern, corresponding to a approximate prototype input pattern of handwritten Hindi *'SWARS'*, the best suitable weight matrix is selected from the generated population of *K* weight matrices.
Let the state of the network corresponding to the already stored l^{th} pattern be

$$N(s^l) = \{s_1^l, s_2^l, ---s_N^l\} \tag{24}$$

This represents one of the stable states of the network for the memorized l^{th} pattern.

Let the prototype of presented input pattern be $x^{l+\varepsilon}$. This pattern represents the noisy or distorted form of the already stored pattern x^l in the network. We have the population K of weight matrices after the crossover operation. Now, we select the weight matrices from this population to evaluate its fitness. Let $w^{POP,k}$ be the F^{th} weight matrix from the generated population K of weight matrices. Now, we assign this selected weight matrix to the network and use the activation dynamics to determine the output state of the network as

$$s_i^{l+\varepsilon} = \sum_{j=1}^{N} w_{ij}^{POP,k} s_j^{l+\varepsilon}(t+1) \tag{25}$$

If, $s_i^{l+\varepsilon}(t+1) = s_i^l(t)$; $\quad \forall i = 1:N$ (26)

It implies that the network settles in the same stable state which corresponds to the already stored pattern, so that $w^{POP,k}$, is selected from the fitness function if it is able to settle the network in the stable state corresponding to the already stored l^{th} pattern, i.e.

$$N(s^{l+\varepsilon}) = N(s^l) \tag{27}$$

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

5. EXPERIMENTS DETAIL

To do the simulation we are considering the Hopfield neural network of 25 neurons. This neural network is trained for pattern storage with the Hebbian learning rule for given training set of handwritten *'SWARS'* of Hindi language. In the training set every pattern consists with 25 bipolar features. The pattern information of the training set is encoded in the form of connection strengths of interconnections between the neurons of the network. In this way the parent weight matrix is constructed which represents the encoded pattern information of memorized patterns.

Now we start with the process of recalling for any presented noisy prototype input pattern of already memorized pattern. The process of recalling is accomplished with two different methods namely the Hebbian rule and the sub optimal GA. The implementation of above stated methods is performed with experiments to study the performance behavior of these methods.

The recalling through Hebbian rule starts by applying the presented prototype input pattern to the Hopfield network and then we continue to iterate the network till it reaches to stability. The stable state of the network reflects any one of the memorized pattern. In this method the possibility of false recalling or false minima is likely to occur in most of the cases if the presented input pattern is a noisy pattern.
The recalling process through the suboptimal GA can be described in the algorithmic form as:

```
Sub Optimal Genetic Algorithm ()

    {

    read suboptimal parent weight matrix;

    initialize input pattern to the network;
do

{

perform mutataion and elitism;

 select solutions for next population by first fitness
function ;

perform crossover;

evaluate population by second fitness function;

}

while (convergence not achieved OR 20 times)

}
```

6. RESULTS AND DISCUSSION

The results presented in this section are demonstrating that, within the simulation framework presented above, large significant difference exists between the performance of genetic algorithm and the Hebbian rule for recalling of memorized handwritten *'SWARS'* of Hindi language in Hopfield neural network using the Hebbian learning rule.

The results are indicating about the difference between sub-optimal GA and Hebbian rule for rate of success to evaluate the performance of these two techniques for associative memory feature. In the experiments, the prototype input patterns used for recalling purpose consists the error which has been generated randomly with respect to memorized patterns. Tables 2-7 show the results of recalling the patterns which containing zero-,one-,two-,three-,four-,and five-bit errors from stored patterns in the Hopfield neural network. The performance of sub optimal GA corresponding to zero-bit, and one-bit error in prototype input patterns is found much better with respect to conventional technique. The results clearly indicate that the Hebbian rule works well for a noiseless pattern, for most of the cases, but its performance degrades substantially and recalling success goes down to a maximum of 1.30% in the case of one-bit error, 0.20% in the case of two-bit error, 0.01% in the case of three-bit error, and 0.000% in the cases of four- and five bit errors. On the other hand, GA recall the pattern successfully even when high noise is presented in the input test pattern i.e. four-bit and five-bit errors. It is observed that in most of the cases the performance of the suboptimal GA outperform the conventional Hebbian method. It is also observed that variance in the mutation probability in sub optimal GA does not have any substantial impact in the performance of this algorithm.

Table .2: Results for recalling Hindi *'SWARS'* which involve zero-bit error from the memorized *'SWARS'*.

'SWARS'	**Recalling Success (in %)**		
	Hebbian Rule	**Sub-optimal GA**	
		p_m **=0.001**	p_m **=0.40**
अ	94.20	100	100
आ	87.10	100	100
इ	89.20	100	100
ई	83.60	100	100
उ	92.60	100	100
ऊ	88.60	100	100
ऋ	82.00	100	100
ए	100.80	100	100
ऐ	86.8	100	100
ओ	85.60	100	100
औ	92.20	100	100
अं	85.20	100	100
अः	81.60	100	100

Table 3: Results for recalling Hindi *'SWARS'* which involve one-bit error from the memorized *'SWARS'*.

'SWARS'	**Recalling Success (in %)**		
	Hebbian Rule	**Sub-optimal GA**	
		p_m **=0.001**	p_m **=0.40**
अ	0.50	100	100
आ	1.02	100	100
इ	2.10	100	100
ई	2.30	100	100
उ	2.01	100	100
ऊ	1.08	100	100
ऋ	1.03	100	100
ए	1.11	100	100
ऐ	1.09	100	100
ओ	2.10	100	100
औ	0.90	100	100
अं	0.81	100	100
अः	1.25	100	100

Table 4: Results for recalling Hindi *'SWARS'* which involve two-bit error from the memorized *'SWARS'*

'SWARS'	**Recalling Success (in %)**		
	Hebbian Rule	**Sub-optimal GA**	
		p_m **=0.001**	p_m **=0.40**
अ	0.00	81.25	97.50
आ	0.20	87.19	90.10
इ	0.03	100	99.20
ई	0.10	95.38	98.10
उ	0.16	75.73	85.20
ऊ	0.10	98.20	89.20
ऋ	0.19	89.44	98.10
ए	0.09	87.30	99.50
ऐ	0.19	99.50	100.00
ओ	0.02	86.50	95.50
औ	0.00	70.00	80.58
अं	0.10	70.53	83.76
अः	0.11	82.60	84.40

Table 5: Results for recalling Hindi *'SWARS'* which involve three-bit error from the memorized *'SWARS'*.

'SWARS'	**Recalling Success (in %)**		
	Hebbian Rule	**Sub-optimal GA**	
		p_m **=0.001**	p_m **=0.40**
अ	0.00	30.95	38.40
आ	0.00	76.50	73.73
इ	0.00	87.95	84.94
ई	0.00	90.09	90.21

उ	0.00	69.40	71.04
ऊ	0.00	66.67	66.40
ऋ	0.00	77.53	80.05
ए	0.00	73.50	74.20
ऐ	0.00	95.30	98.10
ओ	0.00	70.39	64.23
औ	0.00	50.02	54.88
अं	0.00	44.04	51.57
अः	0.00	60.75	62.19

Table 6: Results for recalling Hindi *'SWARS'* which involve four-bit error from the memorized *'SWARS'*.

'SWARS'	**Recalling Success (in %)**		
	Hebbian Rule	**Sub-optimal GA**	
		p_m **=0.001**	p_m **=0.40**
अ	0.00	08.70	10.68
आ	0.00	21.28	23.59
इ	0.00	22.62	20.54
ई	0.00	29.94	21.90
उ	0.00	16.04	13.15
ऊ	0.00	16.71	12.08
ऋ	0.00	19.68	19.74
ए	0.00	19.77	17.41
ऐ	0.00	29.23	25.98
ओ	0.00	20.41	19.49
औ	0.00	08.89	07.75
अं	0.00	10.61	08.88
अः	0.00	16.18	14.03

Table 7: Results for recalling Hindi *'SWARS'* which involve five-bit error from the memorized *'SWARS'*.

'SWARS'	**Recalling Success (in %)**		
	Hebbian Rule	**Sub-optimal GA**	
		p_m **=0.001**	p_m **=0.40**
अ	0.00	0.08	0.60
आ	0.00	1.90	2.00
इ	0.00	1.70	1.76
ई	0.00	2.01	3.08
उ	0.00	0.56	1.06
ऊ	0.00	2.50	1.59
ऋ	0.00	3.50	2.58
ए	0.00	3.08	1.98
ऐ	0.00	3.80	2.80
ओ	0.00	2.30	0.80
औ	0.00	1.00	0.78
अं	0.00	1.80	2.60
अः	0.00	2.61	1.98

Amit [3] claimed that the capacity of deterministic Hopfield model with the Hebbian rule is about 0.15N for the noisy prototype input patterns, where N the number of nodes in the network is. If such a network is overloaded with a number of patterns exceeding its capacity, its performance rapidly deteriorates toward zero. Here, we have stored the 13 handwritten *'SWARS'* of Hindi language in a network of 25 nodes and the performance of the GA suggests that on inducing 5-bit error in presented prototype input pattern the network is able to recall the stored patterns. It implies that the network capacity has increased up to 0.45N. Thus, the numbers of attractors are existing here and successfully explored during the recalling process. It is quit obvious to understand that the GA has searched the suitable optimal weight matrices which are responsible to generate sufficiently large number of attractions. Hence, the Hebbian rule which has been used to encode the pattern information is not the optimal weight matrix for finding the global minima of the problem due to the limited capacity of the Hopfield model. Thus capacity has been increased with GA by exploring the optimal weight matrices for the encoded patterns.

The simulation program, which is developed in MATLAB-7, to test the Hebbian rule, the suboptimal GA, and the random GA for the recalling of handwritten *'SWARS'* of Hindi language, stores the patterns in the Hopfield neural network of 25 neurons. It is to note that during suboptimal GA, the success is considered only if the recalling is done within 20-iterations.

7. CONCLUSION

The simulation results i.e. Tables 2-7 indicates that the sub optimal genetic algorithm has more success rate than the Hebbian rule for the stated problem. It has been found that suboptimal GA can give more than one convergent weight matrices for any prototype input pattern in comparison to the Hebbian rule, if the prototype input pattern is correctly recognized. This shows the more chances for GAs to explore better solution than the Hebbian rule. Further the sub optimal GA is starting from sub optimal weight matrix so it has more chances to explore more number of convergent weight matrices with respect to random GA. In the purposed method it can be seen that the two fitness evaluation functions are used. There are two basic advantages of the two fitness evaluation functions:

1. The randomness of the GA has minimized, because the population is filtered twice. Hence, the less number of populations will be generated and the generated population will be more fitted for the solution.
2. As the number of population has minimized, the searching time will also be reduced. Thus, the GA has improved in its implementation because it is less random and consuming less time for searching the optimal solution.

The direct application of GA to the pattern association has been explored in this research. The aim is to introduce an alternative approach to solve the pattern association problem. The results from the experiments conducted on the algorithm are quite encouraging. Nevertheless more work needs to be perform especially on the tests for noisy input patterns. We can also use this concept for pattern recognition for hand written '*VYANJANS'* of Hindi language, overlapped alphabet and curve scripts. It is also proposed to undertake the following problems in future research program.

1. We would like to train the Hopfield neural network with the genetic algorithm and then the pattern recalling with genetic algorithm.
2. We would like to apply the same approach on the quantum Hopfield neural network.
3. We would like to introduce other types of sigmoid function, chromosome, fitness function, crossover to change the weight values through evolution.

REFERENCES

[1] Hopfield, J. J., "Neural Networks and Physical Systems with Emergent Collective Computational

Abilities", Proceedings of the National Academy Sciences, USA, 79, pp.2554 – 2558, (1982).

[2] Hopfield, J. J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proceedings of the National Academy Sciences, USA, 81, pp.3088 – 3092, (1984).

[3] Amit, D. J., Gutfreund, H., and Somopolinsky, H., "Storing Infinite Number of Patterns in a Spin-glass Model of Neural Networks", Physical Review Letters, vol. 55(14), pp. 461-482, (1985).

[4] Amit, D. J., "Modeling Brain Function: The World of Attractor Neural Networks", Cambridge University Press New York, NY, USA, (1989).

[5] Haykin, S., "Neural Networks: A Comprehensive Foundation", Upper Saddle River: Prentice Hall, Chap 14, pp. 64, (1998).

[6] Zhou, Z., and Zhao, H., "Improvement of the Hopfield Neural Network by MC-Adaptation Rule", Chin. Phys. Letters, vol. 23(6), pp. 1402-1405.

[7] Zhao, H., "Designing Asymmetric Neural Networks with Associative Memory", Physical Review, vol. 70(6) 066137-4.

[8] Kawamura, M., and Okada, M., "Transient Dynamics for Sequence Processing Neural Networks", J. Phys. A: Math. Gen., vol. 35 (2), pp. 253, (2002).

[9] Amit, D. J., "Mean-field Ising Model and Low Rates in Neural Networks", Proceedings of the International Conference on Statistical Physics, 5-7 June Seoul Korea, pp. 1-10, (1997).

[10] Imada, A., and Araki, K., "Genetic Algorithm Enlarges the Capacity of Associative Memory", Proceedings of the sixth International Conf. on Genetic Algorithms, pp. 413 – 420, (1995).

[11] Imada, A., and Araki, K., "Mutually connected neural networks can learn some patterns by means of GA", Proceedings of World Congress on Neural Networks, vol. 1, pp. 803-806, (1995).

[12] Imada, A., and Araki, K., "Basin of attraction of associative memory as it is evolves from random weights", Proceedings of First Asia-Pacific Conference on Simulated Evolution and Learning", pp. 271-278, (1996).

[13] Imada, A. and Araki, K., "Basin of attraction of associative memory as it is evolves from random weights", Proceedings of First Asia-Pacific Conference on Simulated Evolution and Learning, pp. 41-44, (1996).

[14] Imada, A. and Araki, K., "Lamarckian evolution of associative memory", Proceedings of IEEE International Conference on Evolutionary Computation, pp. 678-680, (1996).

[15] Imada, A. and Araki, K., "Evolution of Hopfield model of associative memory by the breeder genetic algorithm", Proceedings of the Seventh International Conference on Genetic Algorithms, pp. 784-791, (1997).

[16] Imada, A. and Araki, K., "Applications of an evolutionary strategy to the Hopfield model of associative memory", Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 679-683, (1997).

[17] Imada, A. and Araki, K., "Hopfield model of associative memory as a test function of evolutionary computations", The first international workshop on frontiers in evolutionary algorithms, Proceedings of Joint Conference of Computer Science, vol. 1, pp. 180-183, (1997).

[18] Imada, A. and Araki, K., "Searching real valued synaptic weights of Hopfield's associative memory using evolutionary programming", The Sixth Annual Conference on Evolutionary Programming, Springer-Verlag, Lecture Notes in Computer Science, vol. 1213,pp. 13-22, (1997).

[19] Imada, A. and Araki, K., "Random perturbations to Hebbian Synapses of associative memory using genetic algorithms", Proceeding of International Work Conference on Artificial and Natural Neural Network, Springer-Verlag, Lecturer notes in computer science, vol. 1213, pp. 398-407, (1997).

[20] McEliece, R. J., Posner, E. C., Rodemich, E. R. and Venkatesh, S. S., "The capacity of the Hopfield associative memory", IEEE Trans Information Theory IT-33 4, pp. 461-482, (1987).

[21] Hopfield, J. J. and Tank, D. W., "Neural Computation of Decisions in Optimization Problems", Biological Cybernetics, vol. 52 (3), pp. 141-152, (1985).

[22] Tank, D. W. and Hopfield, J. J., "Simple Neural Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", IEEE Trans. Circuits and Syst., vol. 33(5), pp. 533-541, (1986).

[23] Jin, T. and Zhao, H., "Pattern Recognition using Asymmetric Attractor Neural Networks", Phys. Rev., vol. E 72(6), pp. 066111-7, (2005).

[24] Kumar, S. and Singh, M. P., "Pattern Recall Analysis of the Hopfield Neural Network with a Genetic Algorithm", Computers and Mathematics with Applications, vol. 60(4), pp. 1049-1057, (2010).

[25] Hebb, D., "The Organization of Behavior: A Neuropsychological Theory", Wiley, New York, (1949).

[26] Holland, J. H., "Adaptation in Natural and Artificial Systems", The University of Michigan Press, Ann Arbor, Michigan, (1975).

[27] Mangal, M. and Singh, M. P., "Analysis of Multidimensional XOR Classification Problem with Evolutionary Feed-forward Neural Networks", International Journal on Artificial Intelligence Tools, vol. 16(1), pp. 111-120, (2007).

[28] Mangal, M. and Singh, M. P., "Analysis of Classification for the Multidimensional Parity-Bit-Checking Problem with Hybrid Evolutionary Feed-forward Neural Network", Neurocomputing, vol. 70 (7-9), pp. 1511-1524, (2007).

[29] Mangal, M. and Singh, M. P., "Handwritten English Vowels using Hybrid Evolutionary Feed-forward Neural Network", Malaysian Journal of Computer Science, vol. 19(2), pp. 169-187, (2006).

[30] Paliwal, M. and Kumar, U. A., "Review: Neural networks and statistical techniques", A review of applications, Expert Systems with Applications, vol. 36(1), pp. 2-17, (2009).

[31] Kuo, R. J., Chang, K., and Chien, S. Y., " Integration and Self-Organizing Feature Maps and Genetic-Algorithm-Based Clustering Method for Market Segmentation", Journal of Organizational Computing and Electronic Commerce, vol. 14(1), pp. 43-60, (2004).

[32] Cao, Q. and Parry, M. E., "Neural network earnings per share forecasting models: A comparison of backward propagation and the genetic algorithm, Decision Support Systems, vol. 47(1), pp. 32-41, (2009).

[33] Pal, S. K., De, S. and Ghosh, A., "Designing Hopfield type networks using genetic algorithms and its comparison with simulated annealing", Int. Journal of Pattern Recognition and Artificial Intelligence, vol. 11(3), pp. 447-461, (1997).

[34] Xu, J., He, J. and Yao, X., "Solving Equations by Hybrid Evolutionary Computation Techniques", IEEE Transactions on Evolutionary Computation, vol. 4(3), pp. 295-304, (2000).

[35] Salcedo-Sanz, S. and 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 (6), pp. 2343-2353, (2004).

[36] Amin, A. H. M., Mahmood, R. A. R. and Khan, A. I., "Analysis of Pattern Recognition Algorithms using Associative Memory Approach: A Comparative Study between the Hopfield Network and Distributed Hierarchal Graph Neuron (DHGN)", IEEE 8th International Conference on Computerand Information Technology Workshops (CIT Workshoap-2008), pp. 153–158, (2008).

[37] Blumenstien, M., Liu, X. Y. and Verma, B., "An investigation of the modified direction feature for cursive character recognition", Pattern Recognition, vol. 40 (2), pp. 376-388, (2007).

[38] Yao, X., "Evolving artificial neural networks", Proceeding of the IEEE, vol. 87 (9), pp. 1423-1447, (1999).

[39] Wasserman, P. D., "Neural Computing: theory and practice", Van Nostrand Reinhold Co., New Yark, NY, USA, (1989).

[40] DeJong, K. A., "An analysis of the behavior of a class of genetic adaptive systems", Ph. D. dissertation, University of Michigan, (1975).

[41] Davis, L., "Adapting operator probabilities in genetic algorithms", Proceedings of Third Int. Con. Genetic Algorithms, pp. 61-69, (1989).

[42] Fogarty, T. C., "Varying the probably of mutation in genetic algorithms", Proceedings Third Int. Con. Genetic Algorithms, pp. 104-109, (1989).

[43] Whitley, D. and Starkweather, D., "Genitor-II: A distributed genetic algorithm, J. Expt. Theor. Artif. Int., vol. 2 (3), pp. 189-214, (1990).

[44] Srinivas, M. and Patnaik, L. M., "Adaptive probabilities of crossover and mutation in genetic algorithms", IEEE Trans. on Systems, Man and Cybernetics, vol. 24(4), pp. 656-667, (1994).

[45] Montana, D. and Davis, L., "Training feed forward neural networks using genetic algorithms", Proceedings of Eleventh International Joint Conference on Artificial Intelligence, SanMateo, CA, USA, pp. 762-767, (1989).

[46] Liu, Y. and Yao, X., "Evolutionary Design of Artificial Neural Networks with different nodes", Proceedings of IEEE International Conference on Evolutionary Computation, ICEC 96, Nagoya, Japan, pp. 670-675, (1996).

[47] Liu, Y. and Yao, X., "Evolving neural networks for Hang Seng stock index forecast", Proceedings of the 2001 Congress on Evolutionary Computation, CEC 01, Seoul, Korea, vol. 1,pp. 256-260, (2001).

[48] Kocalka, P. and Vojtek, V., "Problem Solving based on Evolutionary Neural Network Algorithms", Proceedings of 23rd Int. Conf. On Information Technology Interfaces, ITI 2001, Pula, Croatia, pp. 145-150, (2002).

[49] Wicker, D., Rizki, M. M. and Tamburino, L.A., "E-Net: Evolutionary neural network synthesis", Neuro-computing, vol. 42, pp. 171-196, (2002).

[50] Plagianakos, V. P. and Vrahatis, M. N., "Training neural networks with threshold activation function and constrained integer weights", Proceedings of the IEEE Int. Joint Conference on Neural Networks, IJCNN 2000, Como, Italy, vol.5, pp.161-165, (2000).

[51] Arifovic, J. and Gencay, R., "Using Genetic Algorithms to select architecture of a Feed forward Artificial Neural Network", Physica A: Statistical Mechanics and Its Applications, vol. 289(3-4), pp. 574-594, (2001).

[52] Yao, X., "Evolutionary artificial neural networks", International Journal Neural System, vol. 4(3), pp. 203-222, (1993).

[53] Cardenas, M., Melin, P. and Cruz, L., "Parallel Genetic Algorithms for Architecture Optimization of Neural Networks for Pattern Recognition, Soft Computing for Recognition Based on Biometrics", Studies in Computational Intelligence, vol. 312, pp. 303-315, (2010).

[54] Cho, S. Y. and Chi, Z., "Genetic evolution processing of data structures for image classification", IEEE Transactions on Knowledge and Data Engineering, vol. 17(2), pp. 216-231, (2005).

[55] Kohonen, T. and Ruohonen, M. "Representation of Associated Data by Matrix Operators", IEEE Tran Computers, vol. C-22(7), pp. 701-702, (1973).

[56] Pancha, G. and Venkatesh, S. S., "Feature and Memory-Selective Error Correction in Neural Associative Memory", Associative Neural Memories: Theory and Implementation, M. H. Hassoun, ed., Oxford University Press, pp. 225-248, (1993).

[57] Gardner, E., "The Phase Space of Interactions in Neural Networks Models", J Physics, vol. 21A, pp. 257-270, (1988).

[58] Kasko, B., "Neural Networks and Fuzzy System: A Dynamical systems approach to Machine Intelligence", Prantice-Hall, Englewood Cliff, NJ, (1992).

[59] Kumar, S., Saini, S. and Prakash, P., "Alien attractors and memory annibilation of Structured Sets in Hopfield networks", IEEE Trans Neural Networks vol. 7(5), pp. 1305-1309, (1996).