©2010 International Journal of Computer Applications (0975 - 8887)
Volume 1 – No. 28

A Novel Approach to Classificatory Problem using Grammatical Evolution based Hybrid Algorithm

Rahul Kala
Department of Information Technology, Indian Institute of Information Technology and Management Gwalior, Gwalior, India

Anupam Shukla
Department of Information Technology, Indian Institute of Information Technology and Management Gwalior, Gwalior, India

Ritu Tiwari
Department of Information Technology, Indian Institute of Information Technology and Management Gwalior, Gwalior, India

ABSTRACT
Numerous problems that where intelligent systems find application are classificatory in nature. These include Face Recognition, Speaker Recognition, Word Recognition, etc. In this paper we propose a new model for these classificatory problems. This model is based on the Grammatical Evolution which is an Evolutionary Algorithm that uses chromosomes as a set of instructions over a predefined grammar. The model that we propose here is a type of fuzzy inference system. Rules are in form of a collection of points representing every class. The separation between the unknown input and these representative points determines the degree of belongingness of the unknown input to the specific class being considered. Multiple contributions from same classes are simply added together. The training data set is used for the purpose of generating the initial set of configurations of this fuzzy model. The fuzzy functions are parameterized by adding fuzzy parameters, like any neuro-fuzzy model. These parameters are trained by a validation data set using a training algorithm. The performance of the system over training and validation data set serve as the fitness function. Variable mutation rate is applied. We tested the effectiveness of the algorithm over the picture learning problem and received better results than numerous commonly used algorithms.

General Terms
Algorithms

Keywords
Classification, Hybrid Algorithms, Grammatical Evolution, Adaptive Neuro-Fuzzy Inference Systems, Fuzzy Inference Systems

1. INTRODUCTION
Classification is one of the major problems being extensively studied. In these problems, we have a set of classes and we are supposed to classify the input into one of these classes. If we take the example of speaker recognition, the problem is to classify the input features and map it to one of the many known speakers. Hence we are trying to map a set of features into a class, where the number of classes and their properties are well known in advance. The classificatory problems have a huge importance in the modern world due to their varied presence in different domains and fields. This attracts the interests of researchers to contribute effective problem solving models for these problems.Genetic Algorithms, Artificial Neural Networks and Fuzzy Inference Systems are extensively applied over these problems. A better approach to solve these problems is using the hybrid algorithms. The hybrid algorithms use a combination of the basic algorithms. They try to use the best features of all the algorithms and evolve a system that is able to perform much better than the individual algorithms. The hybrid algorithms are able to adapt much better to the inputs and model the given problem better than the traditional approaches.

One of such approach to hybrid algorithms is the Adaptive Neuro-Fuzzy Inference Systems. These systems build a fuzzy model. The model is parameterized by adding some fuzzy parameters. These parameters are then optimized by using a backpropagation algorithm over the modeled artificial neural network. Genetic Algorithms may be used by the system for the optimizations of the training. At the end of the training, we get the values of the various parameters. These parameters are then used to make the final fuzzy inference system. The system thus generated combines the best features of the genetic algorithms and artificial neural network to evolve a system that is able to perform better than the separate systems.

Evolutionary Algorithms are widely used for various kinds of searching and optimization problems. These algorithms use the analogy of the natural evolution system to solve the problem. Various genetic operators like mutation, crossover, etc. are applied to a population pool to generate higher generations. The performance of any individual or solution is judged by the fitness function. The fittest solution in the final population is regarded as the most optimal solution that the algorithm could find, within the constraints of time and computation.

Grammatical Evolution [21, 22] is another technique that is used to apply genetic algorithms over variable length chromosomes. This approach finds its analogy to the process of protein synthesis. The approach normally uses Backus Naur Form (BNF) to represent the syntax of the language. This ensures a syntactically solution is generated always. The various rules are used one after the other to get the final form of the solution depending upon the chromosome representation. The grammatical evolution technique makes it possible to use genetic algorithms over variable length chromosomes.

Fuzzy Inference Systems (FIS) is also an area of varied applications. Fuzzy models are used to make various fuzzy controllers and classifiers. The basic principle of fuzzy logic states that in real world inputs may not be crisp with a definite value. In reality they may be true only to a certain degree. This degree is called as the degree of membership. The functions that determine

the membership over the input ranges are called as the membership functions.

In this paper we evolve a new algorithm that solves the classificatory problem. The algorithm is based on the approach of Grammatical Evolution. The basic model being followed is a Fuzzy model. The various parameters of the model are optimized by the use of Grammatical Evolution. We also take the use of training using the steepest descend approach to adjust the parameters of the model. These try to get a solution of the Grammatical Evolution into the nearest local minima. The Grammatical Evolution is responsible to find out the global minima.

This paper is organized as follows. Section 2 talks about the literature survey. In section 3 we discuss about grammatical evolution model. Section 4 would discuss the fuzzy model that used in solving the problem. Section 5 talks about the Genetic operators and the way they function in the algorithm. Section 6 talks about the training algorithm. In section 7 we present the results over the picture learning problem. Section 8 gives the conclusion.

2. LITERATURE REVIEW

The basic motivation behind the paper is hybrid systems. A lot of work is being done in order to develop hybrid systems for various problems. Rutkowski [25] presented the flexible neuro fuzzy systems. This is a widely used hybrid system that has largely reformed the manner of working of Fuzzy Systems. Kuo et al [14] proposed a hybrid model for learning fuzzy if-then rules. These have been applied to variety of problems [10, 15, 31]. A lot of work is going on in the various parts of these algorithms like clustering, genetic optimizations, rule forming, parameter updating etc [6, 19, 27]. The algorithms try to optimize the performance in clustering by designing various models based on the architecture of neuro-fuzzy systems [4, 16, 17, 23, 33].

Adams [1] proposed the use of Genetic Algorithms to decide the connectivity of associative memory that can be generalized to artificial neural networks. Maravall et al [18] also proposed a hybrid model of Genetic Algorithms with reinforcement learning for robotic controllers.

Classification is a major problem of study. People are trying to better adapt the present algorithms to make them more suitable for classificatory problems. Amin et al. [2] presented a model of single layered complex valued artificial neural network for classificatory problem. Here they had made use of new activation functions. Hu [12] used Choquet Integral with neural network and genetic algorithms for pattern classification. Estudillo et al [7] proposed multiplicative nodes instead of additive in neural networks to build neural network classifiers. Ang et al [3] used probability to genetically evolve a neural network from smallest size to larger sizes.

Genetic Algorithms are being constantly studied and modified for better performances at various situations. One of the major landmarks here are the Particle Swarm Optimizations. Nani used the Nearest Neighbor approach for prototype reduction using Particle Swarm Optimization and the found the Nearest Neighbor approach to be good [18].

Grammatical Evolution is another area of work. Tsoulus et al [32] proposed grammatical evolution for neural network construction and training. Neill et al [21, 22] has highlighted various developments in these fields in terms of representation and working of the algorithm. Ryann [26] proposed these algorithms for program generation in arbitrary language.

A lot of work is being done in the field of neural network for validating, generalizing and better training of the neural network [5, 8, 9, 29]. Classification also employs the use of Hidden Markov Models [11, 28, 30]. These are statistical models that can be used to predict the consequence of the unknown input. These models have found a variety of use in handwriting recognition, speech recognition etc.

Instantaneously trained neural networks [13, 24] are a good approach for faster training with a smaller generalization capability. These networks require very less training time as the weights are decided just by seeing the inputs. Self organizing maps have also been used extensively for the problem solving. Various other mathematical models have been proposed. These employ mathematical techniques like point to point matching for solving the problem.

3. GRAMMATICAL EVOLUTION

Grammatical Evolution is an effective means to apply genetic algorithm in a situation where the chromosome length is varying [21, 22]. Grammatical Evolution usually uses the notion of Backus Naur Form (BNF) to represent the language. It carries all information of the syntax of the language in the form of various rules. The grammar is a specification of the terminating characters, non-terminating characters, starting character, etc. Rules govern the generation of the grammar. The grammar may thus be represented by *{N, T, P, S}* where *N* is set of non-terminals, *T* is set of terminals, *P* is set of production rules and *S* is the start symbol.

The following is an example of a grammar:

N = { <expr>, <biop>, <uop>, <bool> }
T = { and, or, xor, nand, not, true, false, (,) }
S = { <expr> }
P can be represented as:
(A) <expr> ::= (<expr> <biop> <expr>)
| <uop> <expr>
| <bool>
(B) <biop> ::= and
| or
| xor
| nand
(C) <uop> ::= not
(D) <bool> ::= true
| false

Table 1 summarizes the production rules and the number of choices associated with each.

At every iteration, the first non terminal symbol is replaced by a rule selected by the formula:

Rule = c mod r

Here c is the codon integer value, r is the number of rule choices

Table 1. The number of choices available from each production rule.

Rule	Number of Choices
A	3
B	4
C	1
D	2

Consider the following rule from the given grammar, i.e., given the non-terminal <*biop*>, which describes the set of binary operators that can be used, there are four production rules to select from. As can be seen, the choices are effectively labeled with integers counting from zero.

(B) <biop> ::= and (0)

| or (1)

| xor (2)

| nand (3)

If we assume the codon being read produces the integer 6, then

6 mod 4 = 2

would select rule (2) *xor*. That is, <*biop*> is replaced with *xor*. Each time a production rule has to be selected to transform a non-terminal, another codon is read. In this way the system traverses the genome.

The chromosome consists of a sequence of integers or codons. They are read one after the other and likewise the solution is modified by the application of the rules. It is possible that many codons may not be read and the solution may consist of only terminating characters. It may even be possible that the codons be read and still non-terminating symbols exist in the solution. In this case the solution may be neglected, given lowest fitness value or completed by default terminating symbols.

4. FUZZY MODEL

The algorithm has a fuzzy model over which the final system works. The fuzzy model that is built is used to classify the various inputs to the various classes. The various input features are used for this classification.

The whole working space is a multi-dimensional space where the number of dimensions are equal to the number of features used for classification. This in a 2 dimensional space is given in figure 1. A point here represents an input that we wish to classify. The problem is to map all these points into classes as given in figure 1. The boundaries in between the points are known as the decision boundaries.

The rules in this model consist of a set of points over this n-dimensional input space. The numbers of points are kept variable.

Suppose that the model has *p* points in this n-dimensional space. The collection of these *p* points may be represented as given in (1)

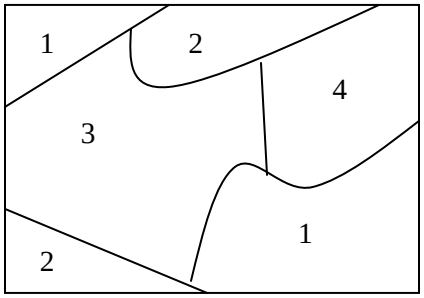

Fig 1: The multi dimensional view of classificatory problem

$$R = \{R_1, R_2, R_3, \ldots R_p\} \quad (1)$$

Each of these points is located on a n-dimensional input space. Hence these points may be represented as given in (2)

$$R = \{<r_{11}, r_{12}, r_{13}, \ldots r_{1n}>, <r_{21}, r_{22}, r_{23}, \ldots r_{2n}>, <r_{31}, r_{32}, r_{33}, \ldots r_{3n}> \ldots <r_{p1}, r_{p2}, r_{p3}, \ldots r_{pn}> \} \quad (2)$$

Whenever we need to find out the class of an unknown input, we basically try to see the effects of each of the p points. The effect of a point *T* varies according to the Gaussian function depending upon the distance of the unknown input and the point *T*. This means that a point *T* will have an effect *E* to an unknown input *I*, where *E* is given by the Gaussian function in (3)

$$E = e^{-\frac{d^2}{2c^2}} \quad (3)$$

Here *c* is a constant, *d* is the separation between the point *T* and the unknown input *I*. *d* is given by (4)

$$d = (T_1 - I_1)^2 + (T_2 - I_2)^2 + (T_3 - I_3)^2 \ldots + (T_4 - I_4)^2 \quad (4)$$

Each point *T* taken above will belong to a particular class. The effect of this point is actually the contribution of this point towards the deciding of the class to which the point belongs. If out of *p* points initially selected, *q* of them belong to particular class (say *j*), the net contribution of the class *j* is the additive contribution of each of these points. Let E_1, E_2, $E_3 \ldots E_q$ be the contributions of these *q* points that all belong to the same class *j*. The net effect of class *j* is given by (5)

$$C_j = \sum_{k=1}^{q} E_k \quad (5)$$

Once we have all the contributions of the participating classes, we simply take the maximum of them. The class with the maximum contribution is declared the winner and input is said to be belonging to this class. In other words we may say as given in (6)

$$\text{Class} = i \text{ such that } C_i > C_j \text{ for all } j \neq i \quad (6)$$

Here we see that each point that we selected, is associated with its location, the class to which it belongs and the Gaussian parameter *c*. This means that a point P_i may be clearly stated as (7)

$P_i = <R_i, L_i, c_i>$ (7)

Here *R* is the location given by (2), c_i is the Gaussian parameter given by (3) and L_i is the class to which the point belongs to.

The complete set of *p* points, or a valid solution may be represented by (8)

$P = \{<R_1, L_1, c_1>, <R_2, L_2, c_2>, <R_3, L_3, c_3>, \ldots <R_p, L_p, c_p>\}$ (8)

5. TRAINING ALGORITHM

A solution randomly generated from the Genetic approach may not be initially very appealing. It may be possible for a relatively optimal solution to lie somewhere very close to the configuration generated by the Genetic Algorithm. This encourages us to use a training algorithm to optimize the solutions.

It may however be noted that this training may result in making the algorithm computationally expensive. Hence we need to strike a right balance between the training algorithm to search for minima in the closer vicinity and the Genetic Algorithm to try to find global minima. The right balance between these two approaches may give us the computationally most optimal solution, within the constraints of time.

The training is carried out on a separate set of data elements. This is the validation data set. This data set is used to adjust the parameters of the model and fix it to the most optimal value. The parameters that can be adjusted are the position of all the points chosen R_i and the Gaussian parameters of these points c_i.

The training is carried out similar to the backpropagation training of an artificial neural network. The training uses the principle of steepest descend algorithm. The training in every epoch tries to adjust the value of the parameters to better match the requirements. Every input element of the validation data set is applied one after the other. We also introduce the concept of learning rate here which determines the rate at which the system trains itself. Unlike in any general neural network, in our system it is not assured that every epoch reduces the error, the errors may be increased on increasing epoch. We take the best configuration during the various stages of training to determine the best configuration of the system.

The algorithm for the modification of the parameters is given as follows:

ModifyParameters
Step1: For every cluster c in sample space
Step2: a1 ← contribution of desired output class in deciding final answer
Step3: a2 ← contribution of class c if different from desired output class else contribution of 2nd highest contributing class for the final output
Step4: if desired class = c
Step5: With a probability of 0.5 increase its Gaussian parameter c by K percent
Step6: With a probability of 0.5 move center closer to the given input by C percent
else
Step7: With a probability of 0.5 decrease its Gaussian parameter c by K percent
Step8: With a probability of 0.5 move center away from given input by C percent

The increase/decrease in percent in Gaussian parameter is given by the relation (9)

$K = \eta_1 / ((a_1-a_2+c_1)\ D)$ (If $a_1>a_2$) (9)

$K= \eta_2 (a_1-a_2+c_2)/ MD$ (else wise)

Where K is the percent change in firing power,
D is the distance between input and cluster being considered (c)
M is the maximum distance between any two points in the n dimensional space
η_1, η_2 is the learning rate
c_1, c_2 are constants

The percent increase/decrease in center is given by the following relation (10)

$C = \eta_3 / ((a_1-a_2+c_3)\ D)$ (If $a_1>a_2$) (10)

$C = \eta_4 (a_2-a_1)+c_4 / MD$ (else wise)

Where K is the percent change in firing power,
D is the distance between input and cluster being considered (c)
M is the maximum distance between any two points in the n dimensional space
η_3, η_4 is the learning rate
c3,c4 are constants

6. GENETIC ALGORITHM

The whole algorithm is based on a Genetic approach, where we try to optimize the whole model. Here we have used the Grammatical Evolution technique that serves as the base for the application of Genetic Algorithm over variable length of the chromosome. We study the Grammatical Evolution along with the applied Genetic operators in the next sub sections

6.1 Grammatical Evolution

The Grammatical evolution uses a set of rules that are used to model the solution. These rules are in form of the Backus Norm Form (BNF). The rules we use here are

S ::= <model>
<model> ::= <point> + <model>
| <point>
<point> ::= <R> + <L> +<c>
<R> ::= $<R_1>$ + $<R_2>$ + $<R_3>$ + $<R_4>$ + … +$<R_n>$
<L> ::= <1>
| <2>
| <3>
…..
<q>
<c> ::= 0.<numberlist>
$<R_1>$, $<R_2>$, $<R_3>$… $<R_n>$ ::= 0.<numberlist>
<numberlist> ::= <digit> + <numberlist>
| <digit>
<digit> ::= 0
| 1
| 2
…..
| 9

This may be reduced to
S ::= <model>
<model> ::= $<R_1>$ + $<R_2>$ + $<R_3>$ + $<R_4>$ + … +$<R_n>$ + <L> +<c> + <model>
| $<R_1>$ + $<R_2>$ + $<R_3>$ + $<R_4>$ + … +$<R_n>$ + <L> +<c>

<L> ::= <1>
| <2>
| <3>
…..
<q>
<c> ::= 0.<numberlist>
<R_1>, <R_2>, <R_3>… <R_n> ::= 0.<numberlist>
<numberlist> ::= <digit> + <numberlist>
| <digit>
<digit> ::= 0
| 1
| 2
…..
| 9

Here *q* is the number of classes into which the mapping needs to be done and *n* in the number of features used in the problem.

Here we assume that inputs in all dimensions have been normalized to lie between 0 and 1.

6.2 Initial Solution

The first major step is the generation of initial solutions for the genetic algorithm to run. At first one solution is generated using a special analysis of the input data.

Clustering is done over the training data by the simple rule of finding groups of input data that all belong to the same class. We make sure that the closed region formed by these points should be occupied by members of this class only. This means that no member of any other class can lie in the region occupied by these clusters. For 2 input systems, this region would be a circle. This is shown in figure 2.

This is done because the basic philosophy is to make representatives of each cluster and position them onto the graph. Using this algorithm, we make sure that we may be able to comfortably replace a cluster by a representative. No other node of a different class would be found nearby this cluster. Hence the node representing this cluster would be free to dominate inside the cluster and in neighboring areas.

We also select a representative for every node. This is the centre of the entire region where a cluster is located in the n-dimensional input space. Let us assume that the training data in a cluster are $<I_{11},I_{12},I_{13},I_{14}\ldots I_{1n}>$, $<I_{21},I_{22},I_{23},I_{24}\ldots I_{2n}>$, $<I_{31},I_{32},I_{33},I_{34}\ldots I_{3n}>\ldots..<I_{p1},I_{p2},I_{p3},I_{p4}\ldots I_{pn}>$. The center for this data is given by (11)

$$\text{Centre} = <(\max(z_1)+\min(z_1))/2, (\max(z_2)+\min(z_2))/2 \ldots (\max(z_n)+\min(z_n))/2> \quad (11)$$

Here $z_1=\{I_{11},I_{21},I_{31}\ldots I_{p1}\}$
$z_2=\{I_{12},I_{22},I_{32}\ldots I_{p2}\}$
$z_n=\{I_{1n},I_{2n},I_{3n}\ldots I_{pn}\}$

We define the radius of the cluster as the square of distance between the center of the cluster and the most distant point in the cluster. Hence the radius of the cluster can be written as given in (12)

$$R=\text{Max}\{(C_1-I_{i1})^2+(C_2-I_{i2})^2+(C_3-Ii_3)^2+\ldots\ldots\ldots..+(C_n-I_{in})^2\} \quad (12)$$

For all i in cluster

Here $<C_1, C_2, C_3, C_4\ldots C_n>$ is the center coordinates

In order that this represents a valid cluster, we further have the condition that point of any other class must not lay in the region of this cluster. Hence for any point $<I_{x1},I_{x2},I_{x3},I_{x4}\ldots I_{xn}>$ in the system, if this point is of a different class than the class of our input, then the equation (13) is always false.

$$(C_1-I_{x1})^2+(C_2-I_{x2})^2+(C_3-I_{x3})^2+\ldots\ldots\ldots..+(C_n-I_{xn})^2 <= R \quad (13)$$

Here $<C_1,C_2,C_3,C_4\ldots C_n>$ is the center coordinates

and R is the radius of the cluster

This concept in a 2 input system is given in figure 2.

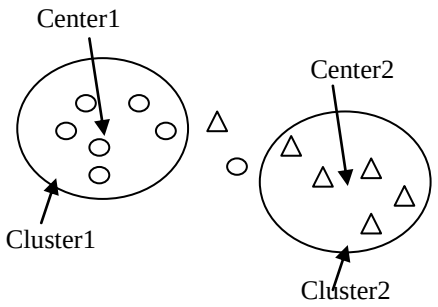

Fig 2: The clusters of the given input. Center is the mean of extreme x and y coordinates.

The center serves as the location of the point. We also define gaussian parameter of every cluster. It is defined by (14)

$$c=\alpha R \quad (14)$$

Here α is const
R is radius of the graph
c is the Gaussian parameter

The algorithm for the clustering is given by the following algorithm.

Cluster()
Step1: While all points are not clustered
Step2: Add any random point to a new cluster
Step3: While it is possible to add new points in this new cluster
Step4: Search for the point p that is closest to any point in the cluster
Step5: If cluster formed by adding p to this cluster is possible then add p to new cluster
Step6: Find center and initial power of this cluster

This gives one of the solution. By modifying the locations of the various points and their corresponding Gaussian parameters by a random percentage, we may be able to obtain some more initial solutions. These solutions are approximately 30% of all the initial solutions.

The rest of the solutions are generated purely randomly in nature. The number of points are although restricted in a way that the solutions with less number of points are preferred to the solutions with more number of points. This saves use of excessive memory as well as makes the algorithm computationally faster.

The probabilities of the number of points are measured relative to the number of points present in the initial solution generated by clustering. The various probabilities are given in table 2.

Table 2. The probabilities of occurrences of various number of points

Number of points in Initial Solution ± percent	Probability
10%	0.5
50%	0.35
100%	0.15

For computational reasons, we do not generate initial solutions that have more than double the number of points as available in the initial solution that was generated by clustering.

All the points in any initial solution are sorted in the order of L_i. This means that they are sorted in the order of the class to which the points belong. Also all solutions need to have all possible classes present in at least one point within it. If a solution is found in which a particular class is not represented by any of the points, the solution is omitted.

6.3 Crossover

The crossover operator we use here is a conventional two-point crossover. The two points are chosen at random places on the chromosome. This splits both the participating chromosomes into three parts. The middle part is taken from one of the parent and the other from the other two parent to generate two distinct solutions.

The first crossover point is chosen among the first 10% of the length of the entire chromosome. The 2nd crossover point is chosen within the first 50% length of the chromosome.

The crossover points may be taken as the mean value from the two participating chromosomes.

6.4 Mutation

The mutation is the Genetic operator which tries to add various new features in the participating chromosomes. The mutation is very necessary in this algorithm to add features that might not have been generated at the time of generation of the initial set.

We have a variable mutation rate here. Initially we keep the mutation rate high and then we reduce the mutation rate as the algorithm proceeded to newer generations. The mutation rate decreases according to the Gaussian function with respect to the generations. This is given in (15)

$$E = e^{-\frac{d^2}{2c^2}} \qquad (15)$$

In the mutation operation, we pickup two elements at random. The first is increased by α percent. The other is reduced by α percent. The resultant is the new chromosome that is passed into the new generation. In mutation also, the first element is chosen from the first 10% length of the chromosome.

6.5 Fitness Function

The fitness function we use here is the performance of the system over the training as well as the validation data set. The more the performance, the better is the solution. We measure the number of outputs the system is able to classify correctly.

7. RESULTS

We applied this algorithm over the picture learning problem. The algorithm was coded using JAVA. A picture was made using two colors '#' and '.'. The picture was marked with some data as the training data and some as the validation data set. The whole training data set and validation data set was given to the algorithm for training.

On the completion the algorithm was made to predict the complete picture. We saw that our system was able to predict the picture to quite a good extent. The performance of the system was reasonably high, as much as 87.10%. We also compared this with the standard neural network training and the standard neuro fuzzy training. The training was found to be much better using our algorithm. The results are given in table 3.

8. CONCLUSION

In this paper we have proposed a method of solving the classificatory problem using a hybrid model. The whole model was built over a Genetic Algorithm using Grammatical Evolution techniques. The Genetic Algorithms were optimizing the parameters of a Fuzzy Model that we had built to solve the classificatory problem. Training similar to the neural network training was adopted to bring the system from any location to the nearest local minima. The training was restricted to a few epochs to strike a balance between the search for local minima and the global minima which the Genetic Algorithm was trying to find out. A variable mutation rate was used, that was kept high initially and low as the newer generations were generated.

9. ACKNOWLEDGMENTS

This work has been supported and sponsored by Indian Institute of Information Technology and Management Gwalior.

10. REFERENCES

[1] Adams, R., Calcraft, L. & Davey, N. (2009). Using a genetic algorithm to investigate efficient connectivity in associative memories, Neurocomputing, 72(4-6) 732-742

[2] Amin, M. F. & Murase, K. (2009). Single-layered complex-valued neural network for real-valued classification problems, Neurocomputing, 72(4-6) 945-955

[3] Ang, J. H., Tan, K.C. & Al-Mamun A (2008). Training neural networks for classification using growth probability-based evolution, Neurocomputing, 71(16-18) 3493-3508

[4] Chaudhuri, B.B. & Bhattacharya U. (2000). Efficient training and improved performance of multilayer perceptron in pattern classification, Neurocomputing 34, 11-27

[5] Draghici, S. (1997) A neural network based artificial vision system for license plate recognition, International Journal of Network Security, International Journal of Neural Systems, 8 (1)

Table 3: The results of the algorithm and other algorithm

Original Image **(a)**	**Training Data** (O, I: training data V: Validation data) **(b)**	**Image after application of our algorithm** **(c)**	**Neural Network** **(d)**	**Neuro Fuzzy System** **(e)**
....###...... ....#########... ...###########.. ..#############. ..#####...###### ..####.....##### ..####.#...##### ...#####...##### ....###....##### ....##### ....##### ....###### ....####### ....########	..O.....O....O.. .... .O.....#I#.V.O.. ....V########... ...####I#####V.. ..##I###V#I####. ..I####..V##I### ..###V..O..##I## ..I###.#..V##### V..#I###...##I## ....V##V.O.##### ..O...O....##### ....O..##I## .O........V##### ....O....####### .O......###I##I#	....####...... ....########...# .############### ################ #######...###### ######....###### ######.....##### #####......##### ....###### ....###### ....###### ....###### ....###### ....######	....###.......... ...#######...... ...#########.... ..############## ..############## ..######....#### ..#####.....#### ..####......#### ..###......##### ...##......##### ....##....###### ....##....###### .....##..####### .....##.######## ......##########	...#####......## ..######.......# ..########...... ..#############. ..###.########.. ..#############. ..######...####. ..######....###. ..#####.....##.. ..####......###. ...###......#### ...##........### ...#......#..### ..##.....##...## ..##....###...## ..##....####.###
No of points: 256 No of #: 115 No of .: 141	No of training data: 37 No for clustering: 26 No for neural: 11	Total: 256 Correct: 223 Incorrect: 33 Efficiency: 87.10%	Total: 256 Correct: 215 Incorrect: 41 Training size: 37 Efficiency: 83.98%	Total: 256 Correct: 188 Incorrect: 68 Training size: 37 Efficiency: 73.43%

[6] Er, M.J., Zhou, Y. (2008) A novel framework for automatic generation of fuzzy neural networks, Neurocomputing, 21(4-6) 584-591

[7] Estudillo, F.J. Martinez, Martinez, C. H., Gutierrez, P.A. & Estudillo, A.C. Martinez (2008) Evolutionary product-unit neural networks classifiers, Neurocomputing, 72(1-3) 548-561

[8] Gernot, A., Fink, P. & Thomas (2006). Unsupervised Estimation of Writing Style Models for Improved Unconstrained Off-line Handwriting Recognition, Proc. Tenth International Workshop on Frontiers in Handwriting Recognition

[9] Graves, A., Fernandez, S., Liwicki, M., Bunke, H. & Schmidhuber, J. (2008). Unconstrained Online Handwriting Recognition with Recurrent Neural Networks, Advances in Neural Information Processing Systems

[10] Grigore, O. & Gavat, I. (1998) Neuro-fuzzy Models for Speech Pattern Recognition in Romanian Language, Proc. European Symposium on Intelligent Techniques CONTI'98. Timisoara, Romania, pp. 165–172

[11] Hewavitharana, S., Fernando, H. C. & Kodikara, N.D. (2002). Off-line Sinhala Handwriting Recognition using Hidden Markov Models

[12] Hu, Yi-Chung (2008). Nonadditive grey single-layer perceptron with Choquet integral for pattern classification problems using genetic algorithms, Neurocomputing, 72(1-3) 331-340

[13] Kak, S. C. (1998). On generalization by neural networks, Information Sciences, 111, 293-302

[14] Kuo, R.J., Hong, S.M., Lin, Y. & Huang, Y.C. (2008). Continuous genetic algorithm-based fuzzy neural network for learning fuzzy IF–THEN rules, Neurocomputing, 71(13-15) 2893-2907

[15] Lee, S. G. et al (1999) A Neuro-Fuzzy Classifier for Land Cover Classification, Proc. 1999 IEEE International Fuzzy Systems Conference Proceedings, Seoul, Korea

[16] Lin, C. J. & Hong, S. J. (2007) The design of neuro-fuzzy networks using particle swarm optimization and recursive singular value decomposition, Neurocomputing 71(1-3) 297–310

[17] Lin, C. J., Chung, I.F. & Chen, C. H. (2007). An entropy-based quantum neuro-fuzzy inference system for classification applications, Neurocomputing, 70(13-15) 2502–2516

[18] Maravall, Dario, Lope, Javier de, Martin & Jose Antonio H (2009). Hybridizing evolutionary computation and reinforcement learning for the design of almost universal controllers for autonomous robots, Neurocomputing, 72 (4-6) 887-894

[19] Mu, C. S., Chien, H. C., Eugene, L. & Jonathan, L. (2006) A new approach to fuzzy classifier systems and its application in self-generating neuro-fuzzy systems, Neurocomputing 69 (4-6) 586–614

[20] Nani, L. & Lumini, A. (2009) Particle swarm optimization for prototype reduction, Neurocomputing, 72(4-6) 1092-1097

[21] O' Neill, M. & Ryan, C. (2001) Grammatical Evolution, IEEE Transactions on Evolutionary Computing 5(4)

[22] O' Neill, M., Brabazon, A. (2005). Recent Adventures in Grammatical Evolution, In Proc. of CMS 2005 Computer Methods and Systems. Vol. 1, pp. 245-253

[23] Pinero, P. et al (2004) Sleep stage classification using fuzzy sets and machine learning techniques, Neurocomputing, 58–60, pp 1137 – 1143

[24] Rajagopal, P. (2003) The Basic Kak Neural Network with Complex Inputs, Master of Science in Electrical Engineering Thesis, University of Madras

[25] Rutkowski, L. (2004). Flexible Neuro-Fuzzy Systems, Structures, Learning and Performance Evaluation, Kluwer Academic Publishers

[26] Ryan, Conor, Collins, JJ & Neill, Michael O, Grammatical Evolution - Evolving Programs for an Arbitrary Language

[27] Sandhu, P. S., Salaria, D. S. & Singh, H. (2008) A Comparative Analysis of Fuzzy, Neuro-Fuzzy and Fuzzy-GA Based Approaches for Software Reusability Evaluation, Proc of Worlds Academy of Science, Engineering and Technology Vol. 29

[28] Shi, D., Shu, W. & Liu, H. (1998). Feature Selection for Handwritten Chinese Character Recognition Based on Genetic Algorithms, In Proc. Intl' Conf. on Systems, man and Cybernetics, Vol. 5, No 5, pp 4201-4206

[29] Som, T. & Saha, S. (2008) Handwritten character recognition by using Neural-network and Euclidean distance metric, Social Science Research Network, Available at SSRN: http://ssrn.com/abstract=1118755

[30] Soryani, M. & Rafat, N. (2006). Application of Genetic Algorithms to Feature Subset Selection in a Farsi OCR, Proceedings of World Academy of Science, Engineering and Technology, Vol. 18, ISSN 1307-6884

[31] Taur, J.S. & Tao, C.W (2000) A New Neuro-Fuzzy Classifier with Application to On-Line Face Detection and Recognition, Journal of VLSI Signal Processing 26, pp 397–409

[32] Tsoulos, I., Dimitris, G.& Glavas, E. (2008) Neural network construction and training using grammatical evolution, Neurocomputing, 72(1-3) 269-277

[33] Vieira, A & Barradas, N. (2003) A training algorithm for classification of high-dimensional data, Neurocomputing, 50, 461 – 472