A Novel Approach to Classificatory problem using Neuro-Fuzzy Architecture

Rahul Kala[1], Anupam Shukla[2], Ritu Tiwari[3]

[1,2,3] Department of Information Technology, Indian Institute of Information Technology and Management, Gwalior, MP, INDIA
[1] rahulkalaiiitm@yahoo.co.in, [2]dranupamshukla@gmail.com, [3] rt_twr@yahoo.co.in

Citation: R. Kala, A. Shukla, R. Tiwari (2011) A Novel Approach to Classificatory problem using Neuro-Fuzzy Architecture, *International Journal of Systems, Control and Communications* 3(3) 259-269.

Final Version Available At:
http://inderscience.metapress.com/content/l01856467755n46x/

Abstract. Classification is one of the major problems solved by the artificial neural networks. The problem deals with the mapping of the input data into classes. Voice recognition, pattern matching, face recognition, character recognition etc. are these types of problems. In the past few years we have seen a great increase in the methods to solve these problems. In this paper we have proposed a new method for solving these problems. The method is inspired from the neuro-fuzzy logic approach to problem solving. Here we have proposed the various changes that may be made in various layers for the system to better handle classificatory problems. We first cluster the training data so that the rest of the algorithm works efficiently with limited data to handle. The clustering is based on class identification of various inputs given as training data. A sort of fuzzy logic approach serves as a means to classify the unknown input to a class based on the rules formed from inputs given in training data. Rules are in form of representative of every cluster and their matching class. The centre and power of the representative are the parameters that are optimized using a training algorithm similar to the backpropagation algorithm of the artificial neural networks. This algorithm is optimized by Genetic Algorithms. We tested the algorithm on the famous classificatory problem of picture learning. We got a good efficiency which proves the effectiveness of the algorithm.

Keywords: Neuro-Fuzzy Networks, Machine Learning, Clustering, Fuzzy Logic, Artificial neural Network, Classification, Pattern Matching

1 Introduction

Classification problems have developed rapidly over the past few years. The problem is to classify the inputs into known number of classes as outputs. A good amount of work in these kinds of problems is by the help of artificial neural networks [4, 7 and 12]. The neural networks have a great power to approximate any unknown function. This power is used to approximate and hence classify the inputs to classes. Another good amount of work is using the Neuro-Fuzzy Networks [8, 14 and 21]. These involve the combined concept of fuzzy logic and neural networks to map the given input data into the output data. These systems may are optimized by the Genetic Algorithms [13, 6 and 11] which further improves the efficiency of the system. When applied to practical data, these systems give a good efficiency and are able to solve the various problems.

In Neuro-Fuzzy systems, we first cluster the input data by using a clustering algorithm like C-means clustering. The use of clustering is to reduce the training data into manageable number that can be solved efficiently in further steps. The fuzzy logic is used to form fuzzy rules that determine the relation between the input and output. Each cluster contributes to the fuzzy rules. The fuzzy rules involve many unknown parameters. These parameters may be varied to optimize the efficiency of the system. The various parameters of the fuzzy logic are optimized by the neural network training. We use steepest descend algorithm to vary each of the parameter to get the final output. We also use Genetic Algorithms for optimizations of the neural networks.

The backpropagation algorithm in the artificial neural networks is used to optimize the parameters of the fuzzy logic. The various parameters are determined by the steepest descend approach of the neural network. The neural network hence tries to find out the most efficient combination of the various parameters that gives the maximum efficiency. At every step the output is calculated and matched with the desired output. The error is used to modify the parameters. The genetic algorithms are used for optimization of the neural network.

2 Related Works

As discussed above, there are so many problems that are classificatory in nature. A lot of work has been done in each of the individual problems. Many of these problems use artificial neural networks to approximate the output. The neural networks with back propagation have been extensively used in almost all the problems. Genetic Algorithms have also been applied to the neural networks for optimized training. A lot of work is being done in the field of neural network for validating, generalizing and better training of the neural network [4, 7 and 12].

The problems also employ the concept of Hidden Markov Models [9, 11 and 13]. 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 [10, 19] 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.

Neuro Fuzzy networks are relatively new. These systems emerged as a motivation for the combination of the powers of neural networks and fuzzy logic. These have been applied to variety of problems [8, 14 and 21]. A lot of work is going on for their application in various domains. These networks very optimally solve the problem. A lot of work is going on in the various parts of these algorithms like clustering, genetic optimizations, rule forming, parameter updating etc [5, 17 and 20].

Classification is a major problem of study which finds immense interest because of its applicability. A lot of work is being done to adapt the neuro-fuzzy systems for the classification problems. The algorithms try to optimize the performance in clustering by designing various models based on the architecture of neuro-fuzzy systems [1, 2, 3, 16, 18 and 22].

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. Algorithm

In this section we study the various aspects of the algorithm. The algorithm works entirely on the principles of the neuro-fuzzy inference system. In this algorithm we first cluster the training inputs in to data clusters. This is done by the clustering algorithm. We use the fuzzy logic type approach to map the inputs to the output with the help of training data provided. We assume that each cluster can be approximated by just one representative. This representative affects the output of the unknown inputs. The output of any input is the cumulative effect firing of all the cluster representatives.

The fuzzy approach has parameters. These are the centre and firing power of each cluster representative. These parameters are adjusted using a training algorithm similar to the backpropagation algorithm of neural network approach. We also use Genetic Algorithms to optimize the data. The training algorithm and genetic algorithms are applied over the validating data which should be set aside for testing the algorithm. The whole process of the algorithm is shown in figure 1.

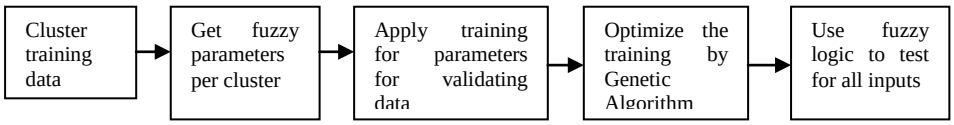


Fig. 1. The general algorithm

3.1 Clustering

The first step of the algorithm is to cluster the given input training data. The use of clustering is to reduce the number of points that participate in the neuro-fuzzy steps of

the algorithm. The limited number of training data reduces the computation for the later steps. Also they filter out the irrelevant data.

The clustering of the training data is done keeping in mind that this is the classification problem. Here the output is not found in continuous patterns. Rather the output is the class to which the input belongs. Hence we keep in mind that for any cluster we make of the training data, all cluster members belong to the same class.

Clustering is done 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 Fig 2.

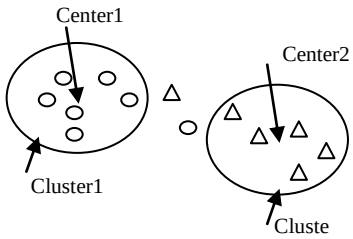


Fig. 2. The clusters of the given input. Center is the mean of extreme x and y coordinates.

This is done because the basic philosophy of the algorithm is to make representatives of each cluster and position them onto the graph. These are made to fire by a certain power to give the output of the unknown input. Using this algorithm, we make sure that we may be able to comfortably replace a cluster by a representative. No other node would be found firing 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 (1)

$$C = <(\max(z_1)+\min(z_1))/2,\ (\max(z_2)+\min(z_2))/2...(\max(z_n)+\min(z_n))/2> \quad (1)$$

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 (2)

$$R^2=\text{Max}\{(C_1-I_{i1})^2 + (C_2-I_{i2})^2 + (C_3-I_{i3})^2+\ldots\ldots\ldots..+ (C_n-I_{in})^2\} \quad (2)$$

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 (3) is always false.

$$(C_1-I_{x1})^2+(C_2-I_{x2})^{2+}(C_3-I_{x3})^2+\ldots\ldots\ldots+(C_n-I_{xn})^2 <= R^2 \quad (3)$$

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.

We also define firing power of every cluster. We assume that every such cluster would participate in the fuzzy system and would contribute an amount proportional to the power. The firing power of any cluster is directly proportional to the radius of the cluster. It is defined by (4)

$$P=\alpha R^2 \quad (4)$$

Here α is const
R is radius of the graph
P is the power

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

3.2 Fuzzy Logic

The fuzzy logic of the system is a concept that is used to map the outputs to the inputs. We have the cluster of inputs available from the previous step. Here we use a special type of fuzzy relation that deals with each and every output class separately. When the computation from all the class is over, the results are compared to get the final answer. This is the class to which the given input belonged. The fuzzy logic approach is used to drive the system by the various fuzzy rules made.

Rules are one of the most essential parts of the system. In this system we make one rule per cluster of the inputs. As every cluster belongs to a separate class of output, every rule maps to a particular class in output. At any given input, this rule fires to get the output. The output of the rule is given by the relation (5).

$O_i = P_i / D_i$ (5)

Here O_i is the individual rule output for cluster i

P_i is the firing power of the cluster i

D_i is the square of distance between the center of cluster i and given input

Here P may be assumed as the weight of the rule which is an adjustable parameter.

Aggregation in this system involves simple addition of all individual outputs. The final output for any specific class is hence given by relation (6)

$C_j = \sum O_i$ (6)

Where C_j is the Output for any class j.

O_i is the output for individual rule i

We solve separately for every class that is possible. The class getting the maximum score of C_i is declared the winner and the output is mapped to that particular class. Hence the class that the input maps to is given by the relation (7)

Class = i such that $C_i > C_j$ for all $j \neq I$ (7)

Hence using and making a fuzzy system, we can know the class to which every known or unknown input belongs.

3.3 Training Algorithm

The training algorithm is used to further optimize the system. We had introduced two parameters in the system. The first was the firing power of the neuron. The second was the center of the cluster. We assumed that the clusters fire from the center with specific power. The most effective class of clusters win and corresponding output is believed to be the final answer.

In this system we use the validation data for the training through the neural network. This is not the same data that was used for the clustering purposes. The system may be trained by any data especially set aside for training. All the training data are applied one after the other repeatedly. Like in artificial neural network, we use the steepest descend approach. Training also does not guarantee that always the error would be reduced. It may be possible that the training increases the error.

The parameter modification modifies the center and the firing power of every cluster. The modification algorithm iterates through every cluster. It sees whether the class to which the cluster being considered belongs is same as the class of the training data being considered. If the two classes are same, the algorithm tries to increase the contribution of this cluster. On the other hand, if the classes do not match, the contribution is tried to be minimized.

The rate of increase or decrease of the contribution depends on the distance between the training data and the cluster being considered. The more the distance, the lesser would be the change made. Hence an input only affects the regions near it in the n-dimensional space.

The modification also depends upon the difference in contribution of the class being considered and the contribution of the desired class. If the class being considered is the same as that of the desired class, the difference with the next highest class is taken. If the difference is large, the change is small and vice versa. However if the condition is that the difference is negative the converse is true. This means that whenever the contribution of the class being considered is more than the contribution of the desired class, we try to make a large change, so that this difference can be reduced and eventually reversed.

We also have introduced the concept of learning rate. The learning rate determines the rate at which the changes are made. In the implementation, we assumed a constant learning rate. But the same may be changed with time.

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

ModifyParameters

Step1: For every cluster c in sample space
Step2: $a_1 \leftarrow$ contribution of desired output class in deciding final answer
Step3: $a_2 \leftarrow$ contribution of class c if different from desired output class else contribution of 2^{nd} highest contributing class for the final output
Step4: if desired class = c
Step5: With a probability of 0.5 increase its power 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 power 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 power is given by the relation (8)

$K = \dot{\eta}_1 / ((a_1-a_2+c_1) D)$ (If $a_1>a_2$) (8)
$K= \dot{\eta}_2 (a_1-a_2+c_2)/ MD$ (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
$\dot{\eta}_1$, $\dot{\eta}_2$ is the learning rate
c_1,c_2 are constants

The percent increase/decrease in center is given by the relation (9)
$C = \dot{\eta}_3 / ((a_1-a_2+c_3) D)$ (If a1>a2) (9)
$C = \dot{\eta}_4 (a_2-a_1)+c_4 / MD$ (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
$\dot{\eta}_3$, $\dot{\eta}_4$ is the learning rate
c_3,c_4 are constants

We use especially reserved points called the validating data for the testing of the system. The performance of the system is good if it gives good results on the validating data set. We retain the system that gives us the maximum efficiency in the validating data set.

3.4 Genetic Algorithm

The genetic algorithm is used for further optimizations. In this system also we use genetic algorithm to mix two possible set of values of parameters and obtain another set of values. The network may perform better with this set of values.

4 Results

The algorithm presented was coded to test the impact of the algorithm. We took a much generalized problem of classification as proposed in [10, 19]. This is the picture learning problem. In this problem, we are given an image consisting of two colors. We measure the ability of the neural network to predict this image. We train the network by giving it some of the points of the image and telling it the color of the image. Then we try to reconstruct the image by feeding all the coordinates as the input and getting the output.

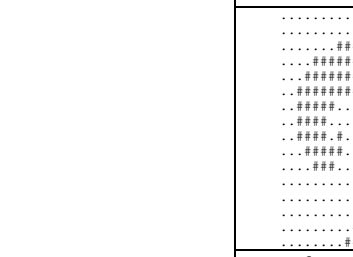
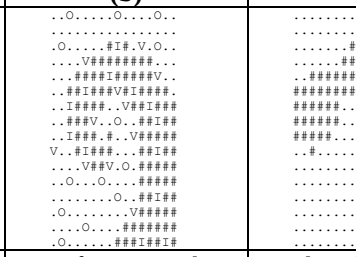
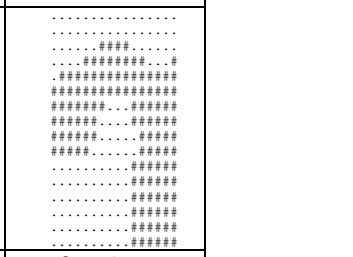
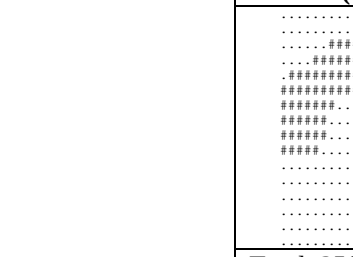
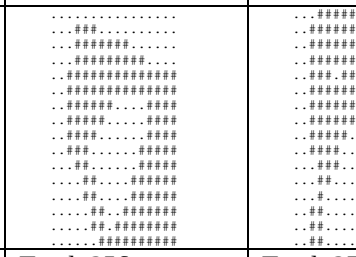
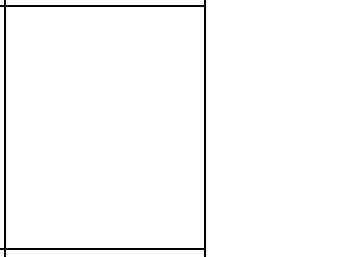
We took the whole image. It was a collection of pixels of two colors. We then took training data as points on the pixels. The training data was a collection of arbitrary points from both the classes. We also took data separately for the validation of the system. This also was a collection of points from both the classes. We applied the algorithm and noted the result.

The efficiency of the algorithm in various phases is given in table 1(a)-(e). It can be easily seen that the efficiency improves as a result of each and every step that was discussed in the algorithm. It may again be noted that the efficiency of the algorithm would increase with the increase of the number of training data sets. Higher training data would give better results.

The same problem was also tried to be solved by neural network and conventional neuro-fuzzy system. The same training data was given to the two systems, as was given to our neuro-fuzzy system. The results are shown in table 1(f) and (g). The results clearly show that the neural network failed to give results to the problem with much computation.

Hence it can be easily inferred from the results that the neuro-fuzzy system that we have proposed here gives the best results as compared to the other systems. The system gave better results than both the conventional neuro-fuzzy system as well as the artificial neural networks. The problem that we had considered was a generalized problem. This algorithm hence very efficiently solves the classification problems and gives high results.

Table1: Results of the algorithm

Original Image (a)	**Training Data** (O, I: training data of fuzzy clustering V: training data for neural network) **(b)**	**Image after only application of fuzzy system (c)**	**Image after application of neural and fuzzy (d)**
			
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: 218 Incorrect: 38 Training size: 26 No of Clusters: 11 Efficiency: 85.15%	Total: 256 Correct: 223 Incorrect: 33 Training size: 11 Efficiency: 87.10%
Image after application of neural and fuzzy with genetic (e)	**Neural Network (f)**	**Conventional Neuro Fuzzy System (g)**	
			
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%	

5 Conclusion

In this paper we considered a general classificatory problem. The whole architecture of the solution of the problem was that is used in the conventional neuro-fuzzy

systems. The algorithms of the neuro-fuzzy systems were modified in order to better adjust with the classification problems. When all this was done we got a good efficiency of 87% which shows the impact of the algorithm.

In this paper we present a solution to the classificatory problem using neuro-fuzzy architecture. A better mathematical modeling might give ever better results. Also we had kept the learning rate constant. This rate can be modified according to time and input to give even better results. The more precise mathematical treatment of the approach may be done in future.

References

1. Amin, F., Murase, K..: Single-layered complex-valued neural network for real-valued classification problems. In: Neurocomputing (2008), doi:10.1016/j.neucom.2008.04.006
2. Ang, J.H. et al.: Training neural networks for classification using growth probability-based evolution. In: Neurocomputing (2008), doi:10.1016/j.neucom.2007.10.011
3. Chaudhuri, B.B., Bhattacharya U.: Efficient training and improved performance of multilayer perceptron in pattern classification. In: Neurocomputing 34 (2000) 11-27
4. Draghici, S.: A neural network based artificial vision system for license plate recognition. In: international Journal of Network Security, International Journal of Neural Systems, Vol. 8, No. 1, 1997
5. Er, M.J., Zhou, Y: A novel framework for automatic generation of fuzzy neural networks. In: Elsevier Journal of Neurocomputing 71 (2008) 584–591
6. Gernot, A., Fink, P., Thomas: Unsupervised Estimation of Writing Style Models for Improved Unconstrained Off-line Handwriting Recognition
7. Graves, A., Fernandez, S., Liwicki, M., Bunke, H., Schmidhuber, J.: Unconstrained Online Handwriting Recognition with Recurrent Neural Networks. In: Advances in Neural Information Processing Systems 20, 2008
8. Grigore, O., Gavat, I.: Neuro-fuzzy Models for Speech Pattern Recognition in Romanian Language. In: CONTI'98 Proceedings. Timisoara, Romania, pp. 165–172
9. Hewavitharana, S., Fernando, H. C., Kodikara, N.D.: Off-line Sinhala Handwriting Recognition using Hidden Markov Models.
10. Kak, S. C.: On generalization by neural networks. In: Elsevier Journal of Information Sciences 111 (1998) 293-302
11. Shi, D., Shu, W., Liu, H.: Feature Selection for Handwritten Chinese Character Recognition Based on Genetic Algorithms.
12. Som, T., Saha, S.: Handwritten character recognition by using Neural-network and Euclidean distance metric. In: Social Science Research Network
13. Soryani, M., Rafat, N.: Application of Genetic Algorithms to Feature Subset Selection in a Farsi OCR. In: Proceedings of World Academy of Science, Engineering and Technology, Volume 18, December 2006, ISSN 1307-6884
14. Lee, S. G. et al: A Neuro-Fuzzy Classifier for Land Cover Classification. In: 1999 IEEE International Fuzzy Systems Conference Proceedings, August 22-25, 1999, Seoul, Korea
15. Lin, C. J., Hong, S. J.: The design of neuro-fuzzy networks using particle swarm optimization and recursive singular value decomposition. In: Elsevier Journal of Neurocomputing 71 (2007) 297–310
16. Lin, C. J., Chung, I.F., Chen, C. H.: An entropy-based quantum neuro-fuzzy inference system for classification applications. In: Neurocomputing 70 (2007) 2502–2516

17. Mu, C. S., Chien, H. C., Eugene, L., Jonathan, L.: A new approach to fuzzy classifier systems and its application in self-generating neuro-fuzzy systems. In: Neurocomputing 69 (2006) 586–614
18. Pinero, P., et al: Sleep stage classification using fuzzy sets and machine learning techniques. In: Neurocomputing 58–60 (2004) 1137 – 1143
19. Rajagopal, P.: The Basic Kak Neural Network with Complex Inputs. In: Master of Science in Electrical Engineering Thesis, University of Madras, 2003
20. Sandhu, P. S., Salaria, D. S., Singh, H.: A Comparative Analysis of Fuzzy, Neuro-Fuzzy and Fuzzy-GA Based Approaches for Software Reusability Evaluation. In: Proceedings of Worlds Academy of Science, Engineering and Technology Volume 29 May, 2008, ISSN 1307-6884
21. Taur, J.S., Tao, C.W.: A New Neuro-Fuzzy Classifier with Application to On-Line Face Detection and Recognition. In: Journal of VLSI Signal Processing 26, 397–409, 2000
22. Vieira, A, Barradas, N.: A training algorithm for classification of high-dimensional data. In: Neurocomputing 50 (2003) 461 – 472Appendix: Springer-Author Discount