International Journal of Computer Science and Applications,

Vol. 7 No. 4, pp. 34-49, 2010

EVOLUTIONARY RADIAL BASIS FUNCTION NETWORK FOR CLASSIFICATORY PROBLEMS

RAHUL KALA

Department of Information Technology, ABV-Indian Institute of Information Technology and Management Gwalior, Morena Link Road, Gwalior, Madhya Pradesh 474010, India
rahulkalaiiitm@yahoo.co.in
http://rkala.99k.org/

HARSH VAZIRANI

Department of Information Technology, ABV-Indian Institute of Information Technology and Management Gwalior, Morena Link Road, Gwalior, Madhya Pradesh 474010, India
harshiiitmg@gmail.com

NISHANT KHANWALKAR

Department of Information Technology, ABV-Indian Institute of Information Technology and Management Gwalior, Morena Link Road, Gwalior, Madhya Pradesh 474010, India
nis.iiitm@gmail.com

MAHUA BHATTACHARYA

Department of Information Technology, ABV-Indian Institute of Information Technology and Management Gwalior, Morena Link Road, Gwalior, Madhya Pradesh 474010, India
mahuab@hotmail.com
http://faculty.iiitm.ac.in/~mb/

Classification has been a major problem of study whose application includes speaker recognition, character recognition, etc. In this paper we first adapt the Radial Basis Function Network (RBFN) for classification problems and then use customized Evolutionary Algorithms to evolve the RBFN. The neurons of the RBFN correspond to some class out of the available output classes. Linear addition of only the same class neurons is taken and an additional layer is added that decides the final output on the basis of maximum activation of each class. Evolutionary algorithm has operators jump and add neuron that aid in optimization. Penalty has been used to restrict overgrowth of network. The algorithm was used to solve the problem of detection of PIMA Indian diabetes and gave a recognition rate of 82.37%, which was better than most of the commonly known algorithms in literature.

Keywords: Radial Basis Function Networks; Evolutionary Artificial neural Networks; Classification; Pattern Recognition; Medical Diagnosis; Diabetes.

1. Introduction

Artificial Neural Networks (ANN) are effective problem solving agents. They perform the task of learning from the historic data and generalizing the learning to new data. They are extensively used for the problems of functional predictions and classification. The problems of functional prediction deal with the calculation of the correct outputs to the

given input, where the outputs are usually continuous in nature. Classification deals with the mapping of the input to some output class. Major problems of study in literature are classificatory in nature. Speaker recognition, character recognition, signature recognition, sentence recognition, document classification, etc. are few examples of classificatory problems. The performance of an algorithm for the classificatory problems is measured by the number of inputs to which the algorithm could correctly determine the output class. The boundaries separating the various classes from each other in the feature space are called as the decision boundaries. Algorithms try to construct effective decision boundaries for classification purpose. The task is easy if the inter-class separation is high and intra-class separation is low. The problem is usually caused by data lying near the decision boundary which is usually difficult to classify.

ANNs store all information in the form of weights that enables them to solve problems. The performance of the ANN depends upon the choice of weight which is usually set by a training algorithm by analyses over the training data presented to the algorithm. The training algorithm tries to find an optimal point in the weight space or the error space of the problem such that the errors are minimized. Each model of ANN uses its own training algorithm. These algorithms are hence different for Multi Layer Perceptron, Radial Basis Function Networks, Learning Vector Quantization, recurrent Neural Networks, Self Organizing Maps, etc.

Most training algorithms are prone to be struck at some local minima. Hence there is a large application of Evolutionary Algorithms that help in finding the global minima in the complex error space. The Evolutionary Algorithms use their analogy with the natural evolution to evolve solutions along with generations. The higher generations are usually fitter than the preceding generations. As the generations proceed, the best and the average fitness increase. The best individual, when the stopping criterion is met, is regarded as the final solution to the problem. The Evolutionary Algorithms are extensively used as neural training agents in three separate heads of evolving a fixed architecture neural network, evolving a variable architecture neural network and evolving a learning rule. In this paper we use the evolutionary algorithms for evolving a variable architecture neural network.

Radial Basis Function Networks (RBFN) are type of ANN that have one input layer, one hidden layer and one output layer. The hidden layer calculates the norm of the input from the neuron. It passes the norm through a non-linear activation function. The linear layer does the linear weighted addition of the outputs of the hidden neurons. This is given as the final output of the system. Each of the neuron in the hidden layer corresponds to a point in the input space. Further each of these neurons has a spread that determines the extent of its influence. Bias may be added as additional inputs. The various parameters of these networks are trained by a training algorithm that normally uses gradient descend rule for the training. This sets the various system parameters and the system is able to give high performance.

Medical Diagnosis is the field that deals with the automatic detection of disease based on input attributes. Automated diagnosis systems have enabled early detection of diseases and hence greatly contribute towards timely preventive measures to be taken by the patient. These systems are used to assist the doctors as expert systems for decision

making. They rely heavily on the historical data. These systems are trained to extract knowledge out of the historical database. The historical database contains information on the presence or absence of disease for various attribute values. Based on the findings on this data, the system tries to generalize the findings on the new unknown data as well.

A lot of work is being done in order to develop hybrid systems for various problems. Rutkowski [Rutkowski (2004)] 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.* [Kuo *et al.* (2008)] proposed a hybrid model for learning fuzzy if-then rules. These have been applied to variety of problems [Grigore and Gavat (1998); Lee *et al.* (1999); Taur and Tao (2000)]. A lot of work is going on in the various parts of these algorithms like clustering, genetic optimizations, rule forming, parameter updating etc [Er and Zhou (2008); Mu *et al.* (2006); Sadhu *et al.* (2008)]. The algorithms try to optimize the performance in clustering by designing various models based on the architecture of neuro-fuzzy systems [Chaudhuri and Bhattacharya (2000); Lin and Hong (2007); Lin *et al.* (2007), Pinero *et al.* (2004), Vieira *et al.* (2003)].

Adams *et al.* [Adams *et al.* (2009)] proposed the use of Genetic Algorithms to decide the connectivity of associative memory that can be generalized to artificial neural networks. Maravall *et al.* [Maravall *et al.* (2009)] 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 and Murase [Amin and Murase (2009)] presented a model of single layered complex valued artificial neural network for classificatory problem. Here they had made use of new activation functions. Hu [Hu (2008)] used Choquet Integral with neural network and genetic algorithms for pattern classification. Estudillo *et al.* [Estudillo *et al.* (2008)] proposed multiplicative nodes instead of additive in neural networks to build neural network classifiers. Ang *et al.* [Ang *et al.* (2008)] 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 and Lumini [Nani and Lumini (2009)] used the Nearest Neighbor approach for prototype reduction using Particle Swarm Optimization and the found the Nearest Neighbor approach to be good [Maravall *et al.* (2009)]. Grammatical Evolution is another area of work. Tsoulus *et al.* [Tsoulus *et al.* (2008)] proposed grammatical evolution for neural network construction and training. O' Neill *et al.* [O'Neil and Ryan (2001); O'Neil and Brabazon (2005)] has highlighted various developments in these fields in terms of representation and working of the algorithm. Ryann [Ryann (2005)] 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 [Draghici (1997); Getnot *et al.* (2006); Graves *et al.* (2008); Som and Saha (2008)]. Classification also employs the use of Hidden Markov Models [Hewavitharana *et al.* (2002); Shi *et al.* (1998); Soryani and Rafat (2006)]. 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 [Kak (1998); Rajagopal (2003)] 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 [Kohonen (1997)]. Various other mathematical models have been proposed. These employ mathematical techniques like point to point matching for solving the problem.

In this paper we first propose a classificatory nature of the RBFN. Here we adapt the conventional RBFN to enable it give a high performance to the classificatory problems. In order to do so we make every neuron of the hidden layer associated with some output class. The corresponding calculations hence become class based. Each neuron calculates the class-specific activation which is added for the like classes. Then we make a winner-takes-all approach and select the highest activation among the classes. This class is declared as the final output class. The various system parameters that include the number of neurons per class, their corresponding centers and spreads are optimized by the use of customized Evolutionary Algorithm. This algorithm does an incremental evolution of the RBFN with customized operators.

This paper is organized as follows. In section 2 we discuss the classificatory model of the RBFN. The training algorithm for the modified RBFN is discussed in section 3. In section 4 we present the evolutionary classificatory RBFN. The results over the PIMA Indian Diabetes database are given in section 5. The conclusion remarks are given in section 6.

2. Classificatory RBFN

In this section we present a modified model of the RBFN that is more suited for the classificatory problems. The conventional RBFN has output nodes that give output values based on the system inputs. These outputs may lie on a continuous scale bound between the highest and lowest values. The model is well-suited for the functional prediction problems but does not give an optimal performance to the classificatory problems where there is a single discrete output corresponding to the name of the class to which the input belongs. The role of the system is to determine one class as the final output class.

Modified RBFN consists of four layers as shown in figure 1. The layer-wise working of the various neurons is given in figure 2. The first layer is the input layer. Here the various inputs are given to the system. Let the input applied to this layer be $I <I_1, I_2, I_3, ...I_n>$. Here n is the number of inputs or attributes in the problem. The second layer is the non-linear distance layer. Like any conventional RBFN, any neuron of this layer corresponds to some point in the input space. This neuron calculates its distance from the applied input. The computed distance is passed through a non-linear activation function. This activation is given to the next layer for further processing. Let there be a total of a neurons in this layer given by $(H_1)^{c1} <h_1^1, h_2^1, h_3^1, ... h_n^1>$, $(H_2)^{c2} <h_1^2, h_2^2, h_3^2, ... h_n^2>, ...(H_a)^{ca} <h_1^a, h_2^a, h_3^a, ... h_n^a>$. Here c_i for any neuron i denotes the class that the

neuron corresponds to and is tagged with all the hidden neurons. The output o_i^{ci} for any neuron H_i of this layer that corresponds to class c_i is given by equation (1).

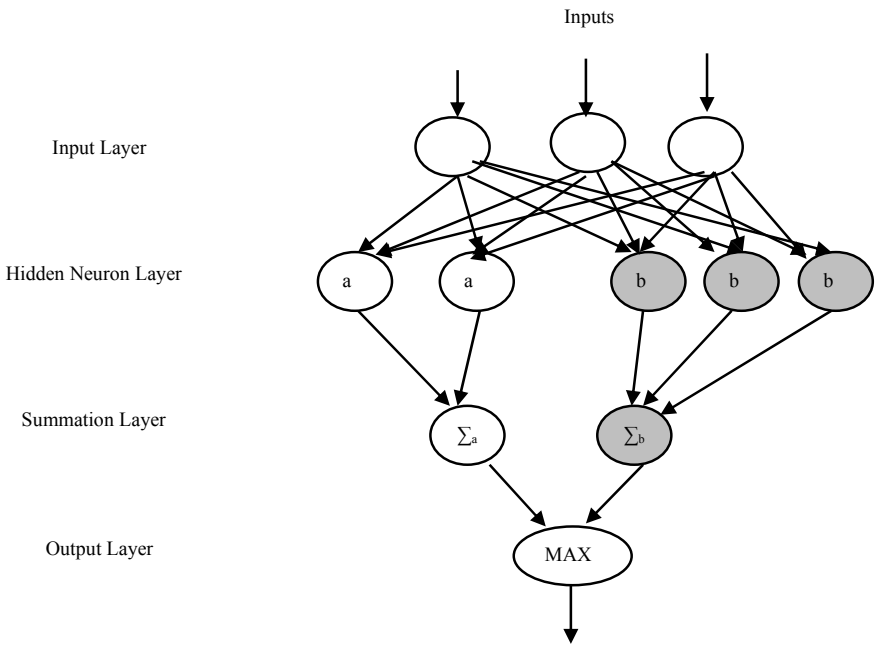

Fig. 1. The architecture of the classificatory RBFN

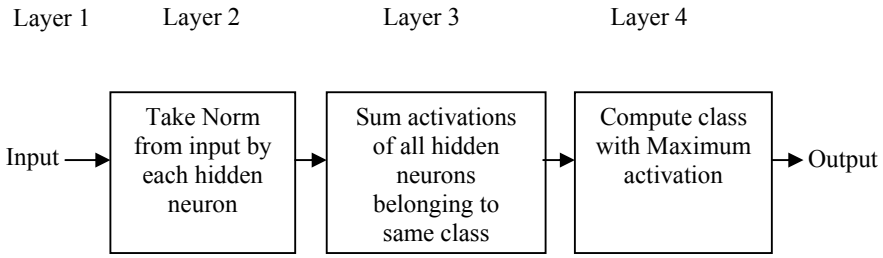

Fig. 2. Working of the classificatory RBFN

$$o_i^{c_i} = \exp(-\frac{\|H_i - I\|}{2\sigma_i^2}) \tag{1}$$

Here σ_i is the spread of the i^{th} hidden neuron. ||x|| is the Euclidian norm given by (2).

$$\|H_i - I\| = \sqrt{\sum_{j=1}^{n}\left(h_j^i - I_j\right)} \tag{2}$$

The third layer is the weighted addition layer. This layer performs a weighted addition of the outputs of neurons that are of same class. The number of neurons of this layer is equal to the number of output classes possible in the system. For simplicity we take all the corresponding weights as unity which is fixed in the classificatory RBFN structure. Each neuron of this class also corresponds to a class whose addition it is performing. The output C_k for any class k is given by equation (3). Here the summation adds activations o_i^{ci} of all hidden neurons whose classes c_i corresponds to the class k.

$$C_k = \sum_{i,c_i=k} o_i^{c_i} \tag{3}$$

The next layer selects the maximum activation and designates it as the final class using the winner-takes-all criterion. This is the final output of the system given by equation (4).

$$O = c : C_c < C_k \vee k \neq c \tag{4}$$

Each neuron of the hidden layer has its own spread that signifies the spread of its effect in the input space. Higher spread means an influence even in the distant areas and vice versa. The higher spread denotes a larger generalizing capability of the RBFN. The smaller spread makes the RBFN more localized in nature. The generalizing capability of the RBFN is its capability to correctly classify unknown inputs. This however requires a larger training time. Also it may not be possible many times to train the RBFN below some significant spread. The localized RBFNs with smaller spread perform better in training data, but may not perform well in the testing data. They require less training time due to the localized nature.

The proposed model of the RBFN may be visualized as multiple neurons deployed at different places, each trying to affect its locality. Each neuron competes to influence the output in favor of its own class. The influence decreases in a Gaussian manner as we move away from this neuron in the input space, just like the conventional RBFN. The various neurons corresponding to the same class are deputed at distant places in the input space and hence contribute to the class score. Since they are at distant location, they may be visualized as a sub-class of the main class. The output layer selects the class with the best score and designates it as the final output.

3. Training Algorithm

The RBFN discussed above has numerous parameters that need to be set in order the system to perform well. These parameters or weights are set by a training algorithm used in this system. This training algorithm is inspired from the Kohonen's Self Organizing Map Training Rule. Consider that we have a classificatory RBFN with hidden nodes as $(H_1)^{c1} <h_1^1, h_2^1, h_3^1, \ldots h_n^1>$, $(H_2)^{c2} <h_1^2, h_2^2, h_3^2, \ldots h_n^2>, \ldots (H_a)^{ca} <h_1^a, h_2^a, h_3^a, \ldots h_n^a>$. The training is supposed to find the exact location of each of the hidden node and its corresponding spread. We are given the training input as a set of inputs and outputs. The input contains a set of attributes and the output contains the corresponding classes which the inputs belong. Each combination of input and output is given to the training algorithm a number of times. The processing of all the inputs and outputs by the training algorithm and according adjustment of the system parameters is termed as one epoch of the training.

Consider one input $I <I_1, I_2, I_3, \ldots I_n>$ be given to the system for training. Let the associated class of this output be o. The center h_j^i of any hidden neuron i and associated class c_i of the system, corresponding to the j^{th} attribute at iteration t is modified according to equation (5).

$$h_j^i(t+1) = \begin{cases} h_j^i(t) + \eta_1(t)(I_j - H_i(t)) & if\ c_i = o \\ h_j^i(t) - \eta_1(t)(I_j - H_i(t)) & if\ c_i \neq o \end{cases} \tag{5}$$

Here $\eta_1(t)$ is the learning rate

The spread σ_i for any hidden node i is given by equation (6)

$$\sigma_i(t+1) = \begin{cases} \sigma_i(t) + \eta_2(t)\|I - H_i\| & if\ c_i = o \\ \sigma_i(t) - \eta_2(t)\|I - H_i\| & if\ c_i \neq o \end{cases} \tag{6}$$

The learning rate $\eta_1(t)$ and $\eta_2(t)$ are kept variable. The learning rate decreases linearly as we proceed with the iterations. The linear decrease in the learning rate, along with iterations is given by equation (7)

$$\eta(t+1) = \eta(t) - \frac{\alpha}{Iter} \tag{7}$$

Here *Iter* is the total number of iterations and α is a constant that denotes the change in the learning rate with time.

4. Incremental Evolution of Classificatory RBFN

The classificatory RBFN discussed in section 2, along with the training algorithm in section 3 marks a complete ANN system for the task of classification. However the system suffers from sub-optimal training as a result of the limitations of the training algorithm. In this section we carry forward the task of extending the system onto an

evolutionary base. The task is two folded to optimize the parameters (neuron locations and spread) as well as to evolve the network architecture.

In this paper we have used an incremental evolution technique. The maximum number of neurons in the hidden neurons per class (L) is set by the system. This number is initially kept to a very low value. This restricts the entire evolutionary process to generate larger networks. As a result the evolutionary process tries to solve the problem given using smaller number of neurons. As the generations increase, this number is increased. This later allows the system to develop larger networks with high number of neurons in the hidden layer. The increase in the maximum number of neurons per class L at any generation g is given by equation (8).

$$L(g) = L(0) + L_{\max} \frac{g}{L} \tag{8}$$

Here $L(0)$ is the least number of neurons per class, which may normally be kept as 1 and G is the maximum number of generations. L_{max} is the maximum change in L, g is the current generation.

It is natural that as L increases, the number of neurons in the system would increase. The larger networks may have a better performance in the training data, but not perform well in the testing data. The larger networks would lead to a loss of generality of the system. We hence need to restrict the network from selection of very large networks. This is done by adding penalty into the fitness of the network. The fitness function of the EA is hence given by equation (9). The aim of the EA is to find parameters and architecture such that equation (9) is maximized.

$$Fit = \frac{\text{Number of correct classifications}}{\text{Total Number of Training Inputs}} + \beta \text{ x Toal Number of Neurons} \tag{9}$$

Here β is the penalty constant.

The other task of the EA is the fixing of the system parameters. This includes the centers and spread of all the hidden neurons. The EA does this optimization by including these parameters in the individual representation. While calculating the fitness function, a few iterations of the training algorithm are also executed. This is used as a local search strategy by the EA. Hence it may be summarized that the EA does the task of intelligent placement of multiple agents in the fitness space. It tries to search the entire space to find global minima. The training algorithm on the other hand tries to converge each of these agents to a nearby local minima. This local search by the training algorithm has a great impact on the EA that enables it to converge early and better explore the fitness space.

We discuss the individual representation as well as the various genetic operators in the subsequent sections. The general algorithm is given in figure 3.

4.1. *Individual Representation*

The Classificatory ANN discussed in section 2 is represented as an individual of the EA. Here the number of neurons in the input layer, addition layer and the output layer are fixed. Further the weight of all connections from the hidden layer to the addition layer is

taken as unity and is hence fixed. The EA is supposed to optimize the number of neurons in the hidden layer, their corresponding output classes, centers and spreads. These are taken into account while deciding the individual representation.

The individual in this problem consists of c numbers. Here c is the number of output classes. Let these numbers be $<k_1, k_2, k_3, \ldots k_c>$. Each number k_i denotes the number of neurons corresponding to class i. Hence it is necessary that these numbers obey the relation (10) as discussed above.

$$k_i \leq L(g) \tag{10}$$

These are then followed by c independent sequences. Each of these sequence i contains k_i number of centers are spreads. Each center is made up of n attributes.

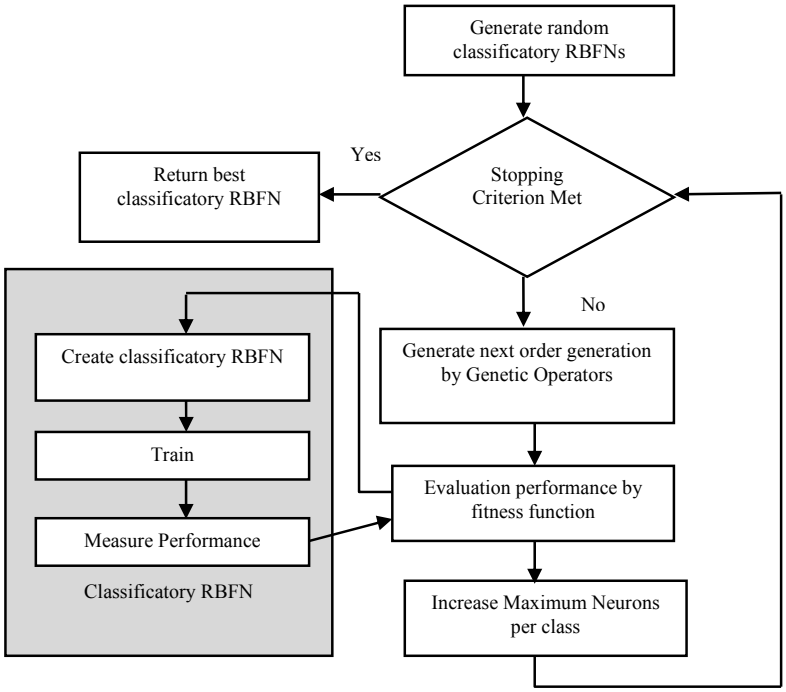

Fig. 3. The Incremental Evolution of Classificatory RBFN

4.2. *Crossover*

Crossover deals with the mixing of properties of two individuals to make new individuals. This operator plays a major role in convergence of the algorithm at the minima as the generations increase. Two parent individuals result in the generation of two children. In this algorithm, crossover is carried out separately for each sequence one by one. Here each sequence corresponds to an output class. The number of neurons of the

first child is the same as the first parent chromosome and the number of neurons of the second child is the same as the second parent chromosome. The individual centers and spread are taken randomly from either of the two parents. This is similar to the application of scattered crossover per neuron in a sequence. In this way we get the two new individuals of the next generation ready.

4.3. *Mutation*

Mutation is responsible for the exploratory nature of the EA. This operator tries to shift the individual by small amounts in the search space. This prohibits early convergence and contributes towards the search for the global minima. In this algorithm we apply a Gaussian mutation for both the center and the spread of every neuron of every class. The architecture of the network is not changed by this operator. The mutation rate needs to be kept high enough since small variations are also made by the RBFN training algorithm. Very large mutation values may however result in randomness.

4.4. *Grow*

This operator results in minor growth of the classificatory RBFN. The growth happens by the addition of neurons. These are located at random points in the input space and have random spread values. If the resulting network (after training algorithm optimizations) gives a good fitness performance, these networks are carried forward in the evolutionary process and influence the other solutions. If however the fitness is low, they die along with generations. The growth of a classificatory RBFN takes place independently for all the sequences. For every sequence we add a new random hidden neuron to it with a probability of 0.5. Hence if there were C number of output classes in the problem, we may expect addition of $C/2$ number of neurons in the total RBFN.

4.5. *Add*

Using this operator we add new random individuals to the population pool. The number of neurons of each class is set to the maximum permissible value which varies with generations. This operator is largely responsible for the growth or the incremental evolution of the classificatory RBFNs. If the networks are less than the threshold size, this operator contributes greatly by producing larger networks. It is natural that the performance of the larger networks would be high with negligible over-growth penalty. Hence the fitness value would be high and they would dominate the evolutionary process. However towards the end this operator would produce over-sized networks. Now the performance of these networks may be high, but the penalty would be large. The high penalty may not overcome the minute performance boost that these network witness. Hence the overall fitness would be low and the individuals generated by this operator might die in the subsequent generations.

4.6. *Other Operators*

The other operators used are rank based fitness scaling, stochastic uniform selection and elite. Rank based scaling rates the individuals on the basis of their rank. This prohibits very fit individuals from domination. Stochastic uniform selection selects the fittest individuals for participation in the evolution process for the next generation. Elite operator transfers fittest individuals of every generation directly to the next generation. This algorithm plays a major role in this algorithm to preserve small sized network from being dying out with generations, in case they are most fit as per the performance and size tradeoff.

5. Results

The proposed algorithm was implemented in JAVA for the experimental purposes. The database used for the experimentation was of PIMA Indian Diabetes. The data was taken from the UCI Machine Learning Repository [Sigillito (1990)]. The aim is the detection of Diabetes in a patient given the various attributes. The PIMA Indian Diabetes data set consists of a total of 8 attributes. These decide the presence of diabetes in a person. This database places several constraints on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. The first attribute is the number of times the women was pregnant. The next attribute is Plasma glucose concentration a 2 hours in an oral glucose tolerance test. We further have the attributes Diastolic blood pressure (mm Hg), Triceps skin fold thickness (mm), 2-Hour serum insulin (mu U/ml), Body mass index (weight in kg/(height in m)^2), Diabetes pedigree function and Age (years).

The general methodology involves the division of database into training and testing data sets. The training data set is used for training the system and the testing data set is used to measure the performance. This is shown in figure 4. The entire database of the disease has a total of 768 instances of data. This was divided into two parts. The first part consisted of the training database containing 535 (~70%) instances of data. The second part was the testing database consisting of 233 (~30%) instances of data. In the training phase the proposed algorithm was given the training instances to evolve the classificatory RBFN. The next stage was testing. Here the evolved classificatory RBFN was given unknown data from the testing data set. The output was collected and compared with the standard output. This determined the performance or the classifying ability of the system.

The maximum number of neurons per class (L_{max}) was fixed to be 30. The number of individuals was 100. The execution was carried over 1000 generations. At any generation crossover contributed 40% of the population and Mutation contributed 20% of the population. The contributions of Add, Grow and Elite operators were 24%, 15% and 1% respectively. The training algorithm was executed for 20 epochs. The initial learning rates η_1 and η_2 were fixed as 0.005 and 0.001 respectively. The algorithm gave an accuracy of 84.07% in the training data and 82.37% in the testing data using the mentioned constants values. Hence we may easily conclude that the system was able to effectively solve the problem of detection of diabetes in the patients.

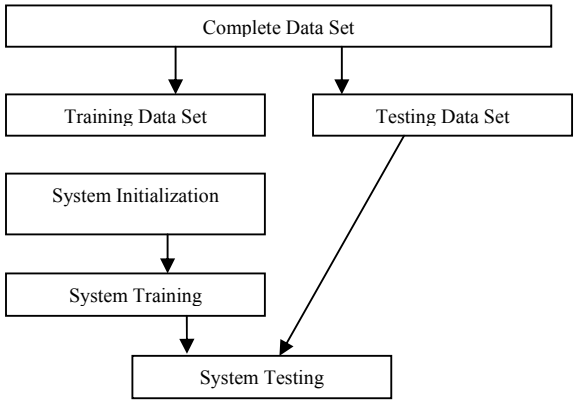

Fig. 4. The testing methodology

We further applied a few commonly used methods to the same data set to compare the efficiency of the proposed algorithm against these methods. The first method applied was of Artificial Neural Network (ANN) trained with Back Propagation Algorithm (BPA). Here we used a single hidden layer which consisted of 12 neurons. The activation functions for the hidden layer was `tansig` and `purelin`. The training function used was `traingd`. The other parameters were a learning rate of 0.05 and a goal of 10-1. Training was done till 2000 epochs. After the network was trained and tested, the performance of the system was found out to be 77.336% for the training data set and 77.7358% for the testing data set.

The second method applied was on ensembles. Here we had used 4 modules or ANNs. Each one of them was trained separately using the same training data set. The 4 ANNs were more or less similar to each other with small changes. These had 12, 14, 10 and 12 neurons respectively. The numbers of epochs were 2500, 2000 and 4000. The four ANNs were trained separately. Here we had used a probabilistic polling in place of the normal poling. The resulting system had a total performance of 78.7276% for the training data and 76.9811% for the testing data.

The third method applied was of Adaptive Neuro Fuzzy Inference System (ANFIS). Here we used the same training as well as testing data sets. The initial Fuzzy Inference System was generated using a grid partitioning method. Each of the attributes had 2 Membership Functions with it. The system was allowed to be trained for a total of 100 epochs. The final system so obtained had a performance of 88.9720% for the training data and 66.5236% for the testing data.

The fourth system was evolutionary ANN. Here we tried to evolve the ANN weights as well as the correct connectionist architecture. Here the chromosome stored the presence or absence of connections in between neurons (along with weights) to eliminate a fully connected architecture. The parameters of the GA were a maximum number of 25 neurons, 25 as the population size with an elite count of 2. The initial population was

uniformly created within the search space. Double vector representation was chosen. Rank based fitness selection was used. Stochastic Uniform selection method was used. Crossover ratio was 0.8. The algorithm was run for 75 generations. The final system had a performance of 77.38% for the training data set and 73.819% in the testing data set.

The last system applied was the conventional Radial Basis Function Network (RBFN). Here the neurons had a spread of 55. The system so generated had a performance of 79.25% on the training data and 78.41% on the testing data.

The performance of the various models is analyzed in table 1. We can easily see that the proposed system gave the best performance than all the commonly known methods. We may hence generalize the results to other similar classificatory problems as well. The algorithm could effectively solve the problem of diagnosis in reasonably small times.

Table 1. The Accuracies with various methods.

S. No.	Algorithm	Training Accuracy	Testing Accuracy
1.	**Evolutionary Classificatory RBFN**	**84.07%**	**82.37%**
2.	ANN with BPA	77.33%	77.73%
3.	Ensemble (with BPA)	78.72%	76.98%
4.	ANFIS	88.97%	66.52%
5.	Evolutionary ANN	77.38%	73.81%
6.	RBFN	79.25%	78.41%

6. Conclusions

In this paper we solved the problem of classification using a novel concept of evolutionary RBFNs. We first adapted the conventional RBFN to solve the classification problems. This was done by making all the operations specific to a class. Every hidden neuron hence belonged to a class. Finally the class with a maximum activation was regarded as the output of the system. The training algorithm of the system was developed using the Kohonen's Self Organizing Map learning rule. Here an input attracted the neuron if it belonged to the same class, else repelled. Similarly the spread increased in case of like classes and vice versa. EA was used for the incremental evolution of the classificatory RBFNs so developed. The maximum permissible neurons in the RBFN per class increased along with generations. This resulted in the growth of the RBFN. The various system parameters were also tuned by the EA. The training algorithm of the RBFN was used as a local search strategy by the EA.

The algorithm was applied over the problem of detection of PIMA Indian Diabetes. The data for this problem was taken from UCI Machine Learning Repository. The data was divided into training and testing data sets. The training data set consisted of about 70% of the data and the testing data set consisted of 30% data. Experimental results show that the algorithm could effectively solve the problem with a high recognition rate of 82.37%. This was higher than all other commonly used methods including ANN with BPA, ensemble, evolutionary ANNs and RBFNs. This proves the greater classifying

power of the algorithm over other commonly used algorithms. These results may further be generalized to similar data sets. This shows the greater classifying potential of the proposed algorithm over other algorithms.

In this paper we only used a single data set for the experimental purposes. The use of other data sets may be done in future. Also comparison with other neural models and hybrid methods may be carried out for different data sets. The performance of the system largely depends upon the data set being used. This architecture may even be made modular to have added advantages of computational speedup and better learning capabilities for complex input data. Also the algorithm may be made adaptive in nature by dynamic behavior of the various system parameters. All this may be done in future.

Acknowledgements

The authors wish to thank the Director, ABV-Indian Institute of Information Technology and Management Gwalior, India for providing all facilities and support for conducting this research.

References

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

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

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

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

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)

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

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

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*

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*

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

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

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

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

Kohonen, T. (1997). *Self-Organizing Maps*, Springer series in information science Vol 30, Second Edition, Springer-Verlag, Heidelberg

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Sigillito, Vincent, 1990, UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html], The Johns Hopkins University. Available At: http://archive.ics.uci.edu/datasets/Pima+Indians+Diabetes

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

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

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

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

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

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