Performance Analysis of Neural Networks Training using Real Coded Genetic Algorithm

Partha Pratim Sarangi
Dept. of CSE
Seemanta Engg. College
Mayurbhanj, Odisha, INDIA

Banshidhar Majhi
Dept. of CSE
NIT, Rourkela, INDIA

Madhumita Panda
Dept. of MCA
Seemanta Engg. College
Mayurbhanj, Odisha, INDIA

ABSTRACT
Multilayer perceptrons (MLPs) are widely used for pattern classification and regression problems. Backpropagation (BP) algorithm is known technique in the training of multilayer perceptrons. However for its optimum training convergence, the learning and momentum parameters need to be tuned on trial and error method. Further, sometimes the backpropagation algorithm fails to achieve global convergence. To alleviate these problems we suggest a genetic algorithm based training for MLP network. Both binary coded and real coded genetic algorithm are used and a comparative training performance analysis has been studied. It is observed from simulation results that both the schemes outperform backpropagation algorithm and achieve global convergence. Further the real coded GA based training shows a faster convergence than binary coded GA based training. For simulation datasets are taken from UCI based machine learning repository. Hence real-coded genetic algorithm finds an alternative for back propagation based training algorithm.

Keywords:
Multilayer perceptron Backpropagation Gradient descent Binary-coded GA Read-coded GA

1. INTRODUCTION
Classification is one of the fundamental problems in pattern recognition tasks. Major classification methodologies include statistical methods, neural networks and other machine learning approaches. However, when prior data distributions are known then statistical approach provides good or satisfactory classification results. One of the remarkable achievements of artificial neural network (ANN) is classification using multilayer perceptron (MLP), which represents a generalization of single-layer perceptron. The major objective of pattern classification system is the recognition of data objects of different classes with a minimal rate of misclassification. In designing a classifier initially, a set of patterns with known class labels are used for training (learning). The system is then asked for to classify an unseen pattern based on the information acquired during training. Multilayer perceptron have been applied successfully to model classifier using backpropagation algorithm for approximating all linearly and non-linearly separable complex functions. Multilayer perceptron has popularity in solving several business and scientific problems that involve prediction, and has also a wide range of applications in the classification problems (Denton, Hung, & Osyk [7], Holmstrom, Koistinen, Laaksonen, & Oja [8]; Mangiameli West, [9]; Patwo, Hu, & Hung [10]; Pendharkar [11]). But MLP has major problem of getting stuck at the local optimal solutions during training and the weights of the network layers are updated through error backpropagation is time consuming. Another drawback of the existing backpropagation learning is that the proper decision boundary depends on the sequence of the input data of the training set but not by considering the global effect of the training set. In order to avoid these disadvantages, researchers have proposed many alternatives. Genetic algorithms are efficient alternatives to train neural networks described by XIN YAO [2], Bornholdt and Graudenz [3]. H.Muhlenbein [4] proposed limitation of multi-layer perceptron network – steps towards genetic neural networks. Whitley et. al. [5] developed a genetic algorithms approach for optimizing connection and connectivity of neural networks. Sankar K. Pal and Dinabandhu Bhandari [1] used binary-coded GA for selection of optimal weights in MLP. Gupta and Sexton [6], Huang and Chang [12] proposed genetic algorithms that outperform significantly than backpropagation in training the MLP.

In this paper, we have proposed a real-coded GA for training the weights of a multilayer perceptron. We have used for illustration purposes the well defined benchmark exclusive-or (XOR) problem and some frequently referred datasets from UCI based machine learning repository. Rate of classification of real-coded GA based training MLP has been compared with conventional backpropagation and binary-coded GA based training MLP.

The rest of the paper is organized as follows. Section 2 deals with basic of neural networks and backpropogation algorithm. Section 3 outlines the genetic algorithms in a nutshell. Section 4 describes the real-coded GA based MLP training algorithm. In section 5 experimental results are presented for comparison. Section 6 concludes the paper.

2. NEURAL NETWORKS:
Neural Networks are simplified models of the biological neural system, they are massively parallel distributed processors made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It is very sophisticated modeling technique capable of modeling extremely complex functions. Neural network are also referred to in literature as neurocomputers, connectionist networks, parallel distributed processors, etc [21]. For two class linearly separable problem, the perceptron algorithm is guaranteed to find a separating hyper plane in finite epochs. However if the pattern space is not linearly separable, the perceptron fails. Because of that a single layer perceptron is inadequate for situations with multiple classes and non-linear separating boundaries. This motivated the invention of a multi-layer network is called multilayer perceptron (MLP) with non-linear learning algorithm known as backpropagation (BP) algorithm. This algorithm minimizes an error term by using gradient descent technique. The MLP can produce boundaries for complex non-linearly separable classes [13].

A schematic representation of multi-layer feedforward neural network is shown in Fig 1. The neurons of this network are organized in three layers: the neurons of the input layer receive inputs from external source, usually in the form of a data file; intermediate neurons constitute one or more hidden layers allow non-linearity in processing; the neurons of the output layer recognize the inputs class level.

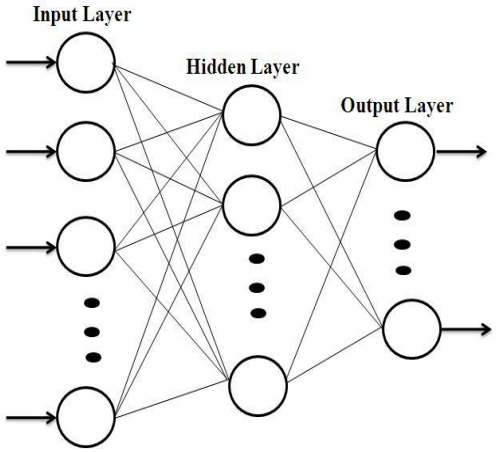

Fig 1: Schematic representation of multi layer perceptron

We have used in our experiment, a three layer feedforward neural network with *m* inputs, *k* outputs and *l* hidden neurons. In addition, each neuron posses a bias of 1 as input with associated weight as threshold value. The neurons output of both hidden and output layer are outputs of the activation function, here hyperbolic tangent function has used as the activation function. In general, the input is passed on to the output layer via the hidden layers through connection weights and activation functions to produce output. Let us define total error is obtained by summing instantaneous value of the error E_j over all neurons in the output layer:

$$E = \frac{1}{2}\sum_{j=1}^{k}(d_j - y_j)^2$$

The mean square error is obtained by summing E over all input training patterns N:

$$E_{MSE} = \frac{1}{Nk}\sum_{i=1}^{N} E_i = \frac{1}{2Nk}\sum_{i=1}^{N}\sum_{j=1}^{k}(d_{ij} - y_{ij})^2$$

The learning in multilayer perceptron is supervised through the well known backpropagation algorithm based on gradient descent technique. In this method correct set of weights are obtained by reducing the error function. To achieve minimum error, it is required to move in the direction of negative gradient of the error function. By minimizing E, the connection weights will update such that the squared error between the output of the network and the corresponding target value is minimized over all the training set. The algorithm uses a pattern-by-pattern also known as on-line training for adjusting connection weights in the network.

3. OUTLINES OF GENETIC ALGORITHMS

Genetic Algorithm is an evolutionary search technique based on the mechanism of random selection. The characteristic of evolutionary algorithms are population-based evolution, survival of the fittest, directed stochastic, derivative-free. GAs performs the search process in four stages: initialization, selection, crossover, and mutation [14]. GA is capable of solving wide range of complex optimization problems using genetic operators (reproduction, crossover and mutation) on the coded solutions in a generation fashion. The set of strings are the solutions in a generation called a population. In binary-coded GA, each solution is a binary string can be considered as a chromosomal representation of the parameter set. Goldberg [15] has shown that for large string lengths, larger population size is required which in turn increase the computational cost. To overcome these difficulties related to binary encoding of continuous parameter optimization problems, real coding of chromosomes is used. A real-coded GA treats chromosomes as vectors (points) of real valued numbers and it adapts the genetic operators [16]. In a continuous searching space, real-coded genetic algorithm is more natural than binary-coded genetic algorithm when we make the optimization process of decision variables. In this study, the artificial neural networks are trained with binary-coded genetic algorithm and real-coded genetic algorithm that has been used for the pattern classification problems [17], [18]. The block diagram of the GA is depicted in Fig. 2.

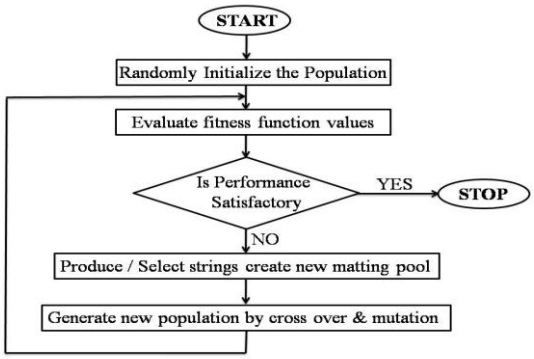

Fig 2: Basic steps of genetic algorithm

4. PROPOSED GA BASED NEURAL NETWORK TRAINING ALGORITHM

In this study, we have used GA to optimize the connection weights of the neural networks, so as to overcome the disadvantage of getting struck in local minima easily. Traditional method uses gradient searching technique that tends to get trapped at local minima. GA has good global searching ability and can learn the near optimal solution without using local gradient information of error function. Exploiting global searching feature of GA, we replace back propagation algorithm by GA for training the neural networks. Proposed algorithm consists of three major phases. The first phase is to decide the chromosomal representation of the parameter set i.e., connection weights, either using a binary string form or directly use a real number form. The second phase is the evaluation of the fitness of these connection weights by computing fitness function, here as mean square error function; The third one is applying the evolutionary process such as selection, crossover, and mutation operations by a genetic algorithm according to its fitness.

4.1 Encoding connection weights and initial population

In [1] proposed, number of parameters P in the network for L number of layers is

$$P = \sum_{i=1}^{L-1} p_{i+1}(p_i + 1)$$

Each parameter is represented by fixed size bits, let's consider q, hence length of each string is P*q. Each parameter of

length q is then decoded into [-1, 1]. String length must be decided a prior to get a desired degree of precision in the final solution. Larger the string length results in more precise solution. In addition to that more number of chromosomes in the population results in lesser number of generations to be executed for convergence but takes higher processing time for each generation.

4.2 Determine fitness function

Decode each chromosome in a population and arrange gene segments of that chromosome into each neuron's connection weights and bias of the neural network. In real-coded GA, the gene segments value of a chromosome are arranged directly as connection weights of the neural networks. So each chromosome in a population represents a neural network structure. Now compute mean square error of each neural network structure of the corresponding chromosome in a population. Here, the network with more error, the fitness value is less and vice versa. To do this, decreasing overall error function F (E_{net_i}) may be represented as increasing function

$$F(E_{net_i}) = {}^{1}/_{1 + E_{MSE}}$$

where E_{MSE} is the total mean square error of individual chromosome. A MSE normalization operation applied on that chromosome's total mean square error.

4.3 Fitness scaling or fitness shifting

In order to make GAs work effectively on finite populations, we must modify the way we select individuals for reproduction. The basic idea is to control the number of reproductive opportunities each individual gets, so that it is neither too large, nor too small. One of the ways to control the reproductive opportunity of each individual is fitness scaling or fitness shifting. In this, the maximum number of reproductive trials allocated to an individual is set to a certain value typically 2.0.

4.4 Genetic operators

Genetic operators are the basic search mechanisms of the GA for creating new points based on the existing population. Selection operator performs exploitation of promising candidates in every generation. Reproduction of new points (children) from the selected points (parents) using the two genetic operations: crossover and mutation that leads to exploration of new parts of the search space. Crossover produces two new individuals (children) from two existing individuals (parents), whereas mutation alters one individual to produce a single individual. In this work, arithmetic crossover [22] and non-uniform-mutation [22] operators have been used in real-coded genetic algorithms.

4.4.1 Crossover operator

Let us assume that $C_1 = (c_1^1, \ldots, c_i^1, \ldots, c_n^1)$ and $C_2 = (c_1^2, \ldots, c_i^2, \ldots, c_n^2)$ are two chromosomes selected for the application of crossover operator. For the arithmetic crossover operator then two offspring, $H_K = (h_1^k, \ldots, h_i^k, \ldots, h_n^k)$, k = 1, 2, are generated, where $h_i^1 = \lambda c_i^1 + (1 - \lambda)c_i^1$ and $h_i^2 = \lambda c_i^2 + (1 - \lambda)h_i^2$. λ is a positive constant and in the experiments, λ is set to 0.28.

4.4.2 Mutation operator

Let us assume the nonuniform mutation operator is applied in a generation r, and g_{max} is the maximum number of generations, new offspring c_i' can obtain as

$$c_i' = \begin{cases} c_i + \Delta(r, b_i - c_i) \; if \; \tau = 0, \\ c_i + \Delta(r, c_i - a_i) \; if \; \tau = 1, \end{cases}$$

$$\Delta(r, y) = y \left(1 - rand^{\left(1 - \frac{r}{g_{max}}\right)^b}\right)$$

Where **τ** is a random number either 0 or 1, b is a parameter that determines the degree of dependence on the number of iterations, in our experiments, b is set to 0.5. The random value rand lies in a range [0, 1].

4.5 Training algorithm to optimize the connection weights of the neural network

Step 0: Initialize the population of real values in the domain of [-1, +1] for each neuron's connection weights and bias to its correspondent gene segments
Step 1: While (new_gen is less than equal to Max_Gen) {
Step 2: The fitness of a chromosome is determined by MSE
Step 3: Sort minimum fitness values
Step 4: If first fitness value is less than equal to min. error then select optimal weights for MLP
Step 5: Select parents for reproduction based on their fitness
Step 6: new population by crossover and mutation
Step 7: Go to step 1 for new generation}

5. EXPERIMENTS

In this section numbers of experiments were conducted with standard benchmark data sets of the University of California Irvine (UCI) machine learning repository [23] to test the performance of our proposed algorithm. Table 1 depicts description of benchmark datasets from Proben1 benchmark [19] used in the experiments. We used neural network structure containing one hidden layer for all dataset used to train the network. In the realm of pattern classification problem, Prechelt [19] proposed three different partitions out of the total set of patterns in the dataset, forming the training, validation, and test sets. The validation set is used for testing to evaluate the performance of a network during training. Such evaluation is called cross validation, it is necessary to avoid overtraining phenomenon. In our experiments, classification performances of backpropagation, binary-coded and real-coded genetic algorithms to train neural networks are compared with 10-fold cross validation technique as proposed by Salzberg [20]. The data set was divided into ten subsets, out of these ten different combinations of nine subsets are used for training and one for testing and finally classification accuracy is calculated by taking average of ten testing subsets accuracy results.

Table 1: Summary of the datasets

Dataset	Number of Attributes	Number of classes	Number of Instances
XOR	2	2	4
Breast cancer	9	2	683
Pima diabetes	8	2	768
Heart	13	2	270
Iris	4	3	150
Wine	14	3	178

5.1 Experimental preparation:

1. Normalize the patterns to the range [-1 to 1]
2. The binary coded genetic algorithms used binary encoding
3. The real coded genetic algorithms used real encoding
4. The length of the chromosome is the total number of connection of the neural networks
5. Real parameters of the chromosome are scaled into a domain (-1 to 1)
6. Population size varies from 30 to 50
7. Fitness scaling or fitness shifting are used to compress the range of fitness values
8. Roulette Wheel selection used for selection in the mating pool
9. Crossover rate: 0.8 in both binary and real GA
10. Single-point crossover operator in binary coded GA
11. Arithmetic crossover operator used in real coded GA
12. Mutation rate: 0.03 and 0.07 in binary and real GA respectively
13. Non-uniform mutation operator used in real coded GA
14. Elitism strategy used in both GA
15. The feed forward neural network has only one hidden layer, number of neurons in this layer is optimized by trial and error method that mentioned in table 2. The transfer function of the hidden layer and output layer are bipolar sigmoid functions. The learning rate and momentum of the BP algorithm is 0.01 and 0.9.
16. To compare the performance of classification rate, 10-fold cross validation technique is used.

5.2 Experimental results

The neural networks are built with one hidden layer and optimized number of neurons in this layer is decided by trial and error method. Maximum number of iterations is chosen as 10000 but it is found that both binary-coded genetic algorithm and real-coded genetic algorithm to train neural networks converged earlier than backpropagation. As genetic algorithm is not a gradient based optimization technique, error convergence of backpropagation algorithm has not compared in our experiments. Number of generations of GA and epochs of backpropagation are different in training neural networks. In backpropagation training weights of the network varies with respect to errors in every epoch but weight parameters of a chromosome are same in a generation. In GA based training, the parameters of the chromosome are changed in new generation by crossover and mutation operators of GA. Hence MSE of both BP and GA based training are different, it cannot be shown together in Fig. 3 to Fig. 8. The 10-fold cross validation average accuracy rates and average error are mentioned in Table 2. Out of ten runs, best run which had highest classification rates for all three training methods were selected to show mean square error convergence with respect to iterations in figures 3,4,5,6, 7 and 8.

It is observed that the BP is relatively fast in training and GA-based training consumes two times of BP. But when BP gets stuck in local minima it takes more iteration to converge. In case of XOR problem BP takes more epochs to converge, when it gets stuck in local minimum. In this XOR experiment both genetic algorithms based training converged earlier than backpropagation algorithm. In breast cancer data, the real-coded genetic algorithm achieved higher accuracy in faster rate than other two algorithms with less training mean square error. We also get same results for wine and pima diabetes data also. Except XOR, all dataset in the experiments, real-coded genetic algorithm based training has converged with less error than binary-coded genetic algorithm based training. Ten-fold average rate of classification results are shown in Table 2. The results from table 2 shows that training using genetic algorithms is more effective for simple problem like XOR problem as well as complex real world benchmark datasets also. It can be observed from Table 2 that the classification results of real-coded genetic algorithm based training are better than other two training methods in all dataset. When data sets are complex due to overlapping, BP based neural networks training is hard to converge; hence real-coded genetic algorithm based training is indeed an alternative method to conventional backpropagation training of neural networks.

Table 2: Results of 10-fold performance of multilayer perceptron training using three methods

Datasets	Network Architecture	Backpropagation Classification Rate (%)	BGA Classification Rate (%)	RGA Classification Rate (%)	MSE		
					BP	BGA	RGA
XOR	2-2-1	100	100	100	0.001	0.00001	0.0312
Breast cancer	9-10-2	96.33	98.21	98.74	0.01218	0.0245	0.0074
Pima diabetes	8-10-2	68.18	81.64	83.11	0.0455	0.0774	0.0617
Heart	13-14-2	73.45	89.62	90.74	0.01609	0.07239	0.02519
Iris	4-5-3	95.33	93.33	97.07	0.0050	0.0366	0.0259
Wine	14-10-3	98.88	90.00	98.88	0.120	0.0572	0.00701

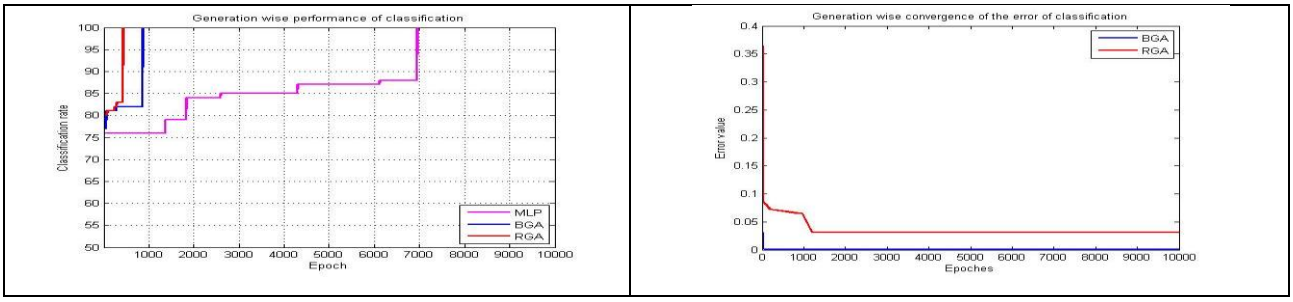

Fig 3: Rate of Classification and error convergence of XOR problem

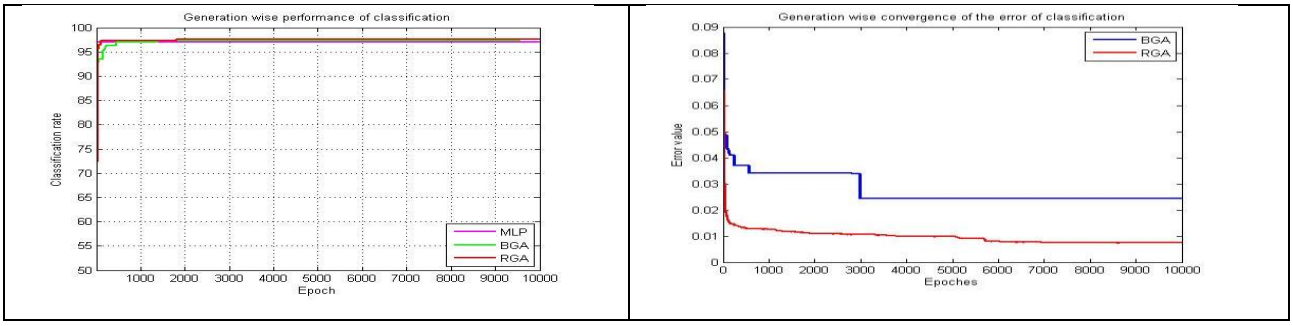

Fig 4: Rate of Classification and error convergence of breast cancer dataset

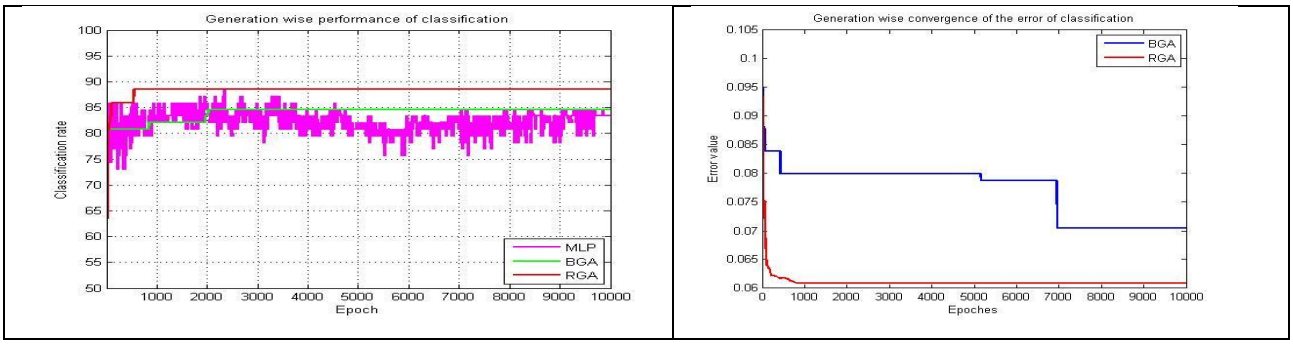

Fig 5: Rate of Classification and error convergence of pima diabetes dataset

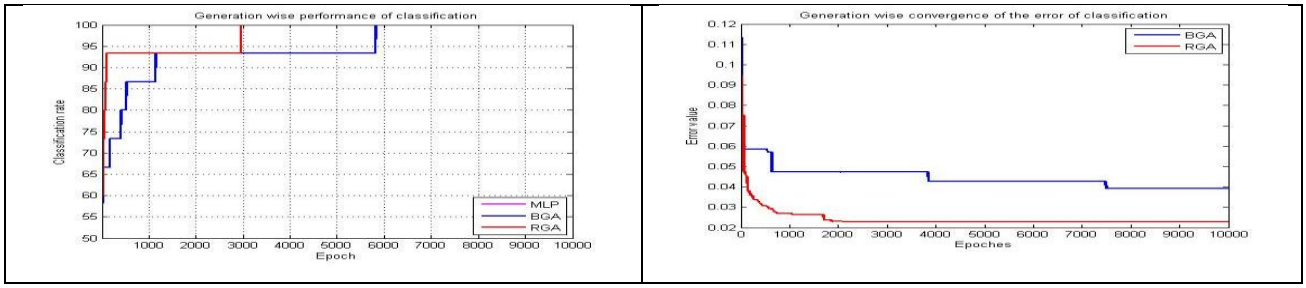

Fig 6: Rate of Classification and error convergence of iris dataset

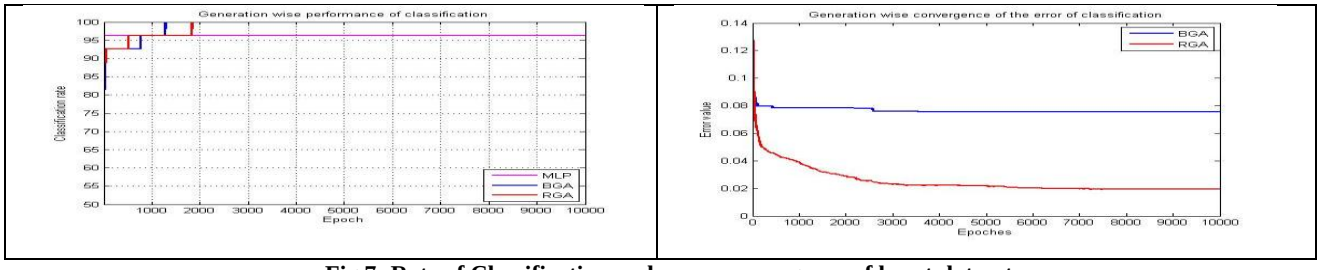

Fig 7: Rate of Classification and error convergence of heart dataset

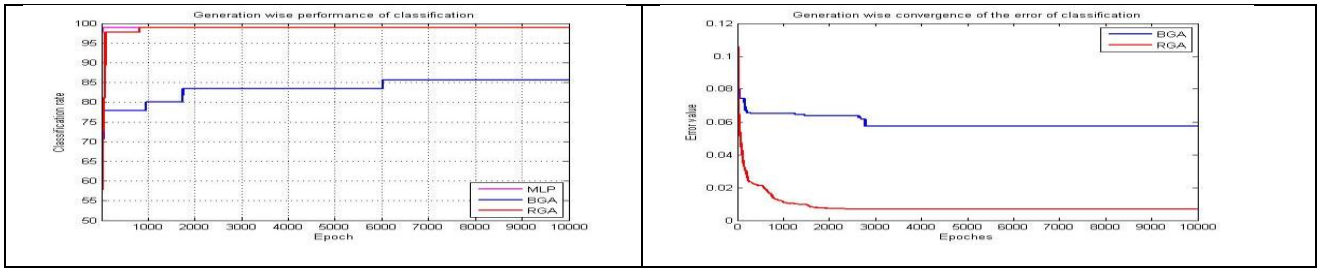

Fig 8: Rate of Classification and error convergence of wine dataset

6. CONCLUSION

This paper demonstrates the training of multilayer perceptron using backpropagation, binary-coded and real-coded genetic algorithms. For the problem of classification, rate of classifications are compared to the backpropagation and genetic algorithms based training of artificial neural networks. It is observed that the real-coded genetic algorithm is superior to other training algorithms in the experiments with six real world data. The experimental results reveal that real-coded genetic algorithm based training overcomes two problems of backpropagation algorithm. First unlike BP, the performance of the genetic algorithms is independent from the sequence of training set.

Second for complex data where BP fails because local minima problem, there genetic algorithms based training is an alternative method of training the neural networks.

The result of this study concludes that real-coded genetic algorithms converge faster to the desired results without trapping in local optimums and provides an alternative approach in training neural networks.

Our proposed real-coded genetic algorithm based training can explore to obtain nearly optimal weights of the neural networks by extending the search space, for complex non-linearly separable problems.

REFERENCES

[1] Pal S. K. and Bhandari D. 1994"Selection Of optimal set of weights in a layered network using genetic algorithm", Information Sciences, Vol: 80, Pages: 213-234.

[2] Yao X. 1991 "Evolving Artificial Neural Networks," in *Preprints Int. Symp. AI, Reasoning & Creativity*, Queensland, Australia,Griffith Univ.

[3] Bornholdt S. and D. Graudenz. 1992. "General asymptotic and neural networks and structure design by genetic algorithms," Neural Networks, vol. 5, no. , pp. 327-334.

[4] Muehlenbein H. 1990. "Limitations of Multi-layer Perceptron Networks - Steps Towards Genetic Neural Networks", Parallel Computing, Vol. 14, pp. 249-260.

[5] Whiteley. D. T. Starkweather & C. Bogart, 1990. "Genetic algorithms and neural networks: optimizing connections and connectivity," Parallel Computing, Vol. 4, pp. 374-361.

[6] Gupta, J. N. D., & Sexton, R. S. Comparing backpropagation with a genetic algorithm for neural network training. Omega, 27, 679–684.

[7] Denton, J. W., Hung, M. S., & Osyk, B. A. 1990. A neural network approach to the classification problem. Expert Systems with Applications, 1(4), 417–424.

[8] Holmstrom, L., Koistinen, P., Laaksonen, J., & Oja, E. 1997. Neural and statistical classifiers taxonomy and two case studies. IEEE Transactions on Neural Networks,8, 5–17.

[9] Mangiameli, P., & West, D. 1999. An improved neural classification network for the two-group problem. Computers and Operations Research, 26, 443–460.

[10] Patwo, E., Hu, M. Y., & Hung, M. S. 1993. "Two-group classification using neural networks" Decision Sciences, 24(4), 825–845.

[11] Pendharkar, P. C.2005. A threshold varying artificial neural network approach for classification and its application to bankruptcy prediction problem. Computers and Operations Research, 32, 2561–2582.

[12] Chien-Yu Huang, Long-Hui Chen, Yueh-Li Chen, Fengming M.Chang. 2009. Evaluating the process of a genetic algorithm to improve the back-propagation network: A Monte Carlo study, Expert Systems with Applications 36 1459–1465

[13] Rumelhart, D. E., Hinton, G., & Williams, R. 1986. "Learning representation by backpropagation errors". Nature, 32(9), 533–536.

[14] Gen, M., & Cheng, R. (1996). Genetic algorithms and engineering design. New York:Wiley.

[15] Goldberg, D. E. 1989. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison Wesley.

[16] Deep, K., & Thakur, M. (2007). A new crossover operator for real coded genetic algorithms. Applied Mathematics and Computation, 188, 895–911.

[17] Montana D.J. 1989. and L. Davis, "Training Feedforward Neural Networks Using Genetic Algorithms," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 762–767.

[18] Schaffer J. D. 1992. D. Whitley, and L. J. Eshelman, "Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art," Proceedings of the IEEE Workshop on Combinations of Genetic Algorithms and Neural Network.

[19] Prechelt, L.: Proben1. 1994. "A Set of Neural Network Benchmark Problems and Benchmarking Rules". Technical Report 21, FakultÄat fÄur Informatik UniversitÄat Karlsruhe, 76128 Karlsruhe, Germany.

[20] Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Mining and Knowledge Discovery, 1, 317–328.

[21] Simon Haykin, "Neural Networks: A comprehensive foundation," Pearson Education Asia, Seventh Indian Reprint, 2004.

[22] Michalewicz, Z. 1992 Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, New York.

[23] UCI repository of machine learning databases, Department of Information and Computer Sciences, University of California, Irvine, http://www.ics.uci.edu/~mlearn/MLRepositor