OPTIMUM LEARNING RATE FOR CLASSIFICATION PROBLEM WITH MLP IN DATA MINING

Lalitha Saroja Thota[1] and Suresh Babu Changalasetty[2]
[1]Department of Computer Science, King Khalid University, Abha, KSA
[2] Department of Computer Engineering, King Khalid University, Abha, KSA

ABSTRACT

Neural networks have been used effectively in a number of applications including remote sensing, expert systems, image processing, data mining, decision systems etc. Back Propagation networks are a popular type of network that can be trained to recognize different patterns. Most of these applications have used the MLP back-propagation algorithm as the learning algorithm. One of the major problems with this algorithm is that its convergence time is usually very long since the training set must be presented many times to the network. If the learning rate is too low, the network will take longer to converge. On the other hand, if high, the network may never converge. The learning rate has to be selected very carefully. Up to now, designers of neural network applications had to find an appropriate learning rate for their systems by trial and error. We carry experiments on classification in data mining WEKA toolbox with multi-layer perception model using back propagation algorithm with diabetic dataset. In this paper, we try to find an optimum learning rate which is stable and takes less time for convergence.

KEYWORDS: *learning rate, classification, data mining, neural network, back propagation, WEKA, MLP multi-layer perception, neuron, weight, threshold*

I. INTRODUCTION

Back-propagation is a supervised learning method, and is a generalization of the delta rule. It requires a dataset of the desired output for many inputs, making up the training set. It is most useful for feed-forward networks. Back propagation networks are necessarily multilayer perceptrons. A multilayer perceptron (MLP) is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate output. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called back propagation for training the network

Multilayer perceptron using a back propagation algorithm are the standard algorithm for any supervised learning pattern recognition process and the subject of ongoing research in computational neuroscience and parallel distributed processing. They are useful in research in terms of their ability to solve problems stochastically, which often allows one to get approximate solutions for extremely complex problems like fitness approximation

A back-propagation neural network is one of the most widely used supervised training algorithms among neural networks. The back-propagation algorithm requires that the weights of each unit be adjusted so that the total error between the actual output and the desired output is reduced. A big problem with back propagation networks is that its convergence time is usually very long.

The learning rate is crucial for the training process of a two-layer neural network (NN). Therefore, many researches have been done to find the optimal learning rate so that maximum error reduction can be achieved in all iterations.

Selecting a good learning rate reduces training time but can require a lot of trial and error. Trying to find a universal learning rate which fits all needs is unrealistic. To illustrate what an optimal learning rate is, the root relative squared error % versus the learning rate (Fig. 1) and root mean squared error versus learning rate (Fig 2) has been computed for weight changes calculated by the back-propagation algorithm. The point where the root relative squared error % and root mean squared error is minimal is called the optimal leaning rate. The paper describes a method to compute the optimal learning rate by using the NOLR method.

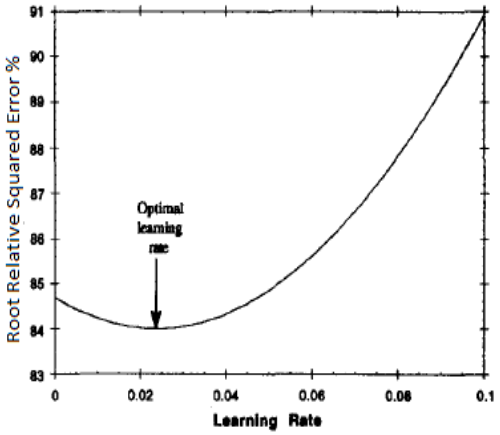

Fig 1. Root relative squared error vs learning rate

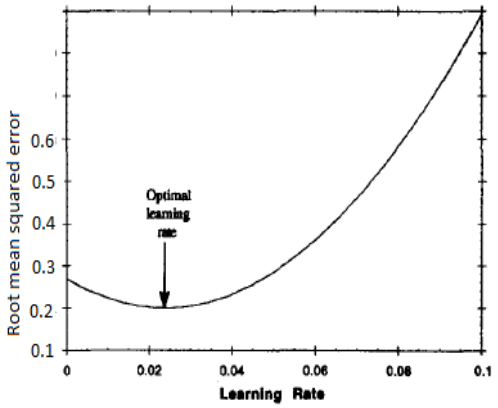

Fig 2. Root mean squared error vs learning rate

II. BACKGROUND

An introduction to neural networks [1] quickly defining neurons, neural networks, and the back propagation algorithm is presented below.

2.1 Neuron

Neuron has one or more inputs and produces one output. The output is calculated by multiplying each input by a different number (called weight), adding them all together, then scaling the total to a number between 0 and 1. Fig 3 shows a simple neuron with:

- Three inputs [x1, x2, x3]. The input values are usually scaled to values between 0 and 1.
- Three input weights [w1, w2, w3]. The weights are real numbers that usually are initialized to some random number between 0 and 1.
- One output z. A neuron has one (and only one) output. Its value is between 0 and 1, it can be scaled to the full range of actual values.

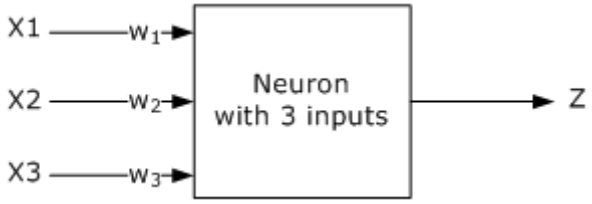

Fig 3 : Neuron - Input Layer

Let d= (x1 * w1) + (x2 * w2) + (x3 * w3) (1)

In a more general fashion, for n number of inputs: (∑ means the *sum of*)

$$d = \sum x_i * w_i \ldots\ldots\ldots \text{ for } i=1 \text{ to } n \quad (2)$$

Let θ be a real number which we will call *Threshold*. Experiments have shown that best values for θ are between 0.25 and 1.

$$z = s(d + \theta) \quad \text{…….. } \textit{Apply the sigmoid to get a number between 0 and 1} \qquad (3)$$

This says that the output *z* is the result of applying the sigmoid function on $(d + \theta)$. In NN applications, the challenge is to find the right values for the weights and the threshold.

2.2 Back Propagation Network

Fig 4 shows a Back Propagation NN:

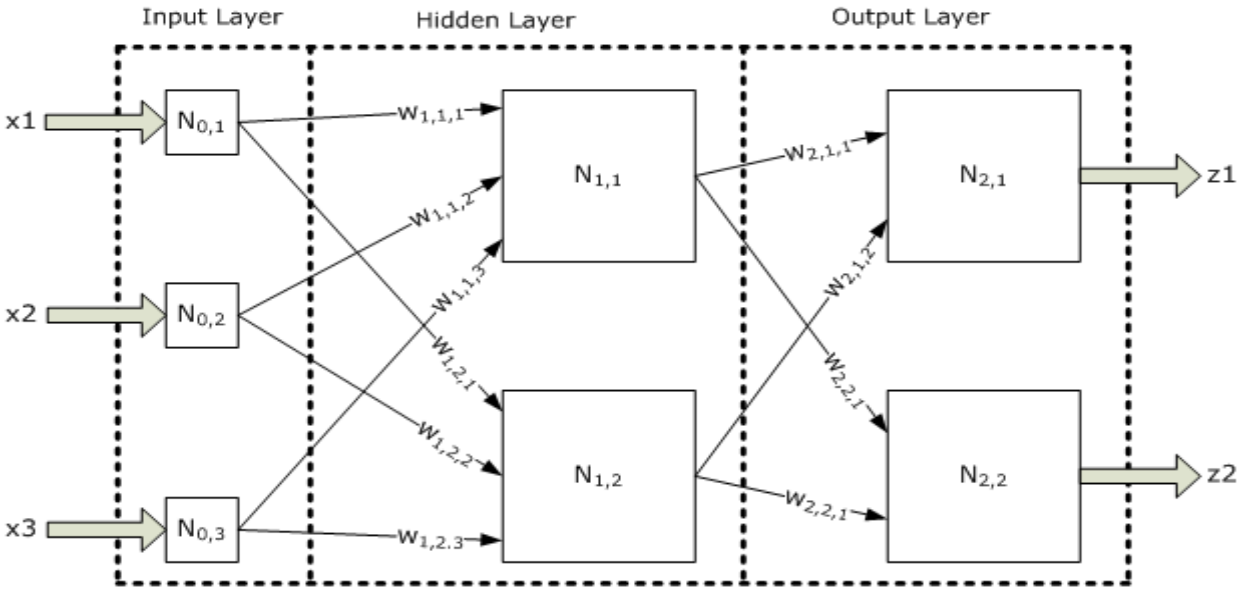

Fig 4 : A Neural network model

This NN consists of three layers:

- Input layer with three neurons.
- Hidden layer with two neurons.
- Output layer with two neurons.

- The output of a neuron in a layer goes to all neurons in the following layer.
- Each neuron has its own input weights.
- The weights for the input layer are assumed to be 1 for each input. In other words, input values are not changed.
- The output of the NN is reached by applying input values to the input layer, passing the output of each neuron to the following layer as input.
- The Back Propagation NN must have at least an input layer and an output layer. It could have zero or more hidden layers.

The number of neurons in the input layer depends on the number of possible inputs we have, while the number of neurons in the output layer depends on the number of desired outputs. The number of hidden layers and how many neurons in each hidden layer cannot be well defined in advance, and could change per network configuration and type of data. In general the addition of a hidden layer could allow the network to learn more complex patterns, but at the same time decreases its performance. You could start a network configuration using a single hidden layer, and add more hidden layers if you notice that the network is not learning as well as you like.

For example, suppose we have a bank credit application with ten questions, which based on their answers, will determine the credit amount and the interest rate. To use a Back Propagation NN, the network will have ten neurons in the input layer and two neurons in the output layer.

2.3 Supervised Training

The Back Propagation NN works in two modes, a supervised training mode and a production mode. The training can be summarized as follows:
Start by initializing the input weights for all neurons to some random numbers between 0 and 1, then:

- Apply input to the network.
- Calculate the output.
- Compare resulting output with desired output for the given input. This is called the *error*.
- Modify the weights and threshold θ for all neurons using the *error*.
- Repeat the process until *error* reaches an acceptable value (e.g. *error* < 1%), which means that the NN was trained successfully, or if we reach a maximum count of iterations, which means that the NN training was not successful.

The challenge is to find a good algorithm for updating the weights and thresholds in each iteration to minimize the *error*. Changing weights and threshold for neurons in the output layer is different from hidden layers. Note that for the input layer, weights remain constant at 1 for each input neuron weight. Before we explain the training, let's define the following:

- η the Learning Rate: a real number constant, usually 0.2 for output layer neurons and 0.15 for hidden layer neurons.
- Δ the change: For example Δx is the change in x. Note that Δx is a single value and not Δ multiplied by x.

2.4 Output Layer Training

- Let z be the output of an output layer neuron as shown above
- Let y be the desired output for the same neuron, it should be scaled to a value between 0 and 1. This is the ideal output which we like to get when applying a given set of input.
- Then e (the *error*) will be:

$$e = z * (1 - z) * (y - z) \quad (4)$$

$$\Delta\theta = \eta * e \quad \text{..... } \textit{The change in } \theta \quad (5)$$

$$\Delta w_i = \Delta\theta * x_i \quad \text{..... } \textit{The change in weight at input i of the neuron} \quad (6)$$

In other words, for each output neuron, calculate its error e, and then modify its threshold and weights using the formulas above.

2.5 Hidden Layer Training

Consider a hidden layer neuron as shown in the following Fig 5.

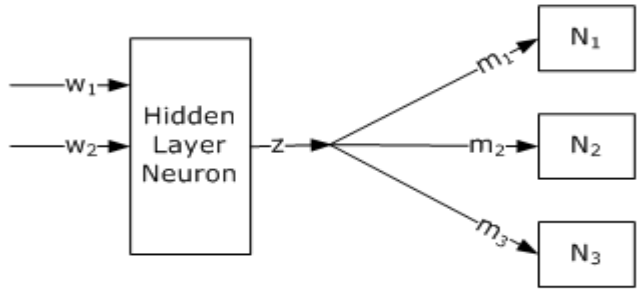

Fig 5 : Neuron - Hidden Layer

- Let z be the output of the hidden layer neuron as shown above
- Let m_i be the weight at neuron N_i in the layer following the current layer. This is the weight for the input coming from the current hidden layer neuron.

- Let e_i be the error (e) at neuron N_i.
- Let r be the number of neurons in the layer following current layer. (In above diagram r=3).

$$g = \sum m_i * e_i \text{ ... } \textit{(for i=1 to r)} \quad (7)$$
$$e = z * (1 - z) * g \text{ ... } \textit{Error at the hidden layer neuron} \quad (8)$$
$$\Delta\theta = \eta * e \text{ ... } \textit{The change in } \theta \quad (9)$$
$$\Delta w_i = \Delta\theta * x_i \text{ ... } \textit{The change in weight i} \quad (10)$$

Notice that in calculating g, we used the weight m_i and error e_i from the following layer, which means that the error and weights in this following layer should have already been calculated. This implies that during a training iteration of a Back Propagation NN, we start modifying the weights at the output layer, and then we proceed backwards on the hidden layers one by one until we reach the input layer. It is this method of proceeding backwards which gives this network its name *Backward Propagation*.

III. RELATED WORK

When neural network is trained using error back propagation algorithm, the learning rate determines how much the weights can change in response to an observed error on the training set. Almost anyone who has used such training algorithm has been faced with the problem of choosing the learning rate, but there is seldom much guidance on what value to use, since the best value to use depends on the particular task. Several algorithms exist for automatically tuning the learning rate [2,3], but such methods typically concentrate on improving the speed of convergence and fail to focus on generalization performance. However, the choice of learning rate can have a dramatic effect on generalization performance as well as the speed of training.

The choice of the learning rate is still a non-trivial problem in BP theory. A low value of η results in slow learning, while a large value of η corresponds to rapid learning. This might also result in oscillations which leads to no learning at all. In addition, if the NN is to be trained on-line, the choice of η becomes more critical. Applications have shown that the stability of the total system depends heavily on the choice of this value [4].

A new method to compute dynamically the optimal learning rate is proposed. By using this method, a range of good static learning rates could be found. Furthermore, some studies have been done on factors influencing learning rates [5].

Some techniques have previously been developed to find a good leaning rate [6],[7] but these methods are usually based on heuristics and do not provide an optimal learning rate. A method has been developed to compute dynamically the near optimal learning rate (NOLR) for back-propagation neural networks [8]. In [9], an adaptive learning rate algorithm is introduced. This procedure increases n, but only to the extent that the network can learn without large error increases. Thus a near optimum learning rate IS obtained for the local terrain. When a larger n could result in stable learning, n is increased. Otherwise, it is decreased until stable learning is resumed.

Authors in [10] proposed dynamic optimization of the learning rate using derivative information. In [10], it was shown that the relatively large or small learning rates may affect the progress of BP algorithm and even may lead to failure of the learning process. However, the analysis of stable learning rates was not discussed in [10]. A New Dynamic Optimal Learning Rate for A Two-Layer Neural Network is proposed [11]. In this the direction to search is revised for a new dynamic optimal learning rate, which can have a better convergence in less iteration count than previous approach.

Fuzzy Neural Network (FNN) is transformed into an equivalent fully connected three layer neural network, or FFNN. Based on the FFNN, BP training algorithm is derived. To improve convergent rate, a new method to find near optimal learning rates for FFNN is proposed [12].

The learning rate was analysed using BP algorithm of Artificial Neural Network (ANN) for hand written digit recognition application. Various parameters such as learning rate, number of hidden neurons in hidden layer, momentum term and number of training runs are used during the analysis of the BP algorithm .The parameters of BP algorithm are used to analyze the learning rate which shows its impact for hand written digit recognition application. Simulation results show that the learning rate affects the performance of ANN [13].

A network focuses on the determination of an optimum learning rate is proposed for the classification of satellite images. An optimum learning rates between the ranges 0.001 – 0.006 was determined for training the algorithm [14].

IV. EXPERIMENTAL STUDY

The data mining tool kit WEKA 3.6 is used to simulate the system and experiment with the multi layer perception with back propagation neural network analysis in classification module. The diabetic's dataset with 768 records supplied by the National Institute of Diabetes and Digestive and Kidney Diseases USA was utilised in the experiment [15]. There are nine attributes describing various properties of diabetics like Number of times pregnant (preg), Plasma glucose concentration a 2 hours in an oral glucose tolerance test (plas), Diastolic blood pressure (mm Hg (pres), Triceps skin fold thickness (mm) (skin), Hour serum insulin (mu U/ml) (insu), Body mass index (weight in kg/(height in m)^2) (mass), Diabetes pedigree function (pedi), Age (years) (age), Class variable (0 or 1) <tested_positive, tested_negative>. The classification experiment is run with various learning rate on same diabetic dataset. Fig 6 shows the neural network classifier model build in WEKA.

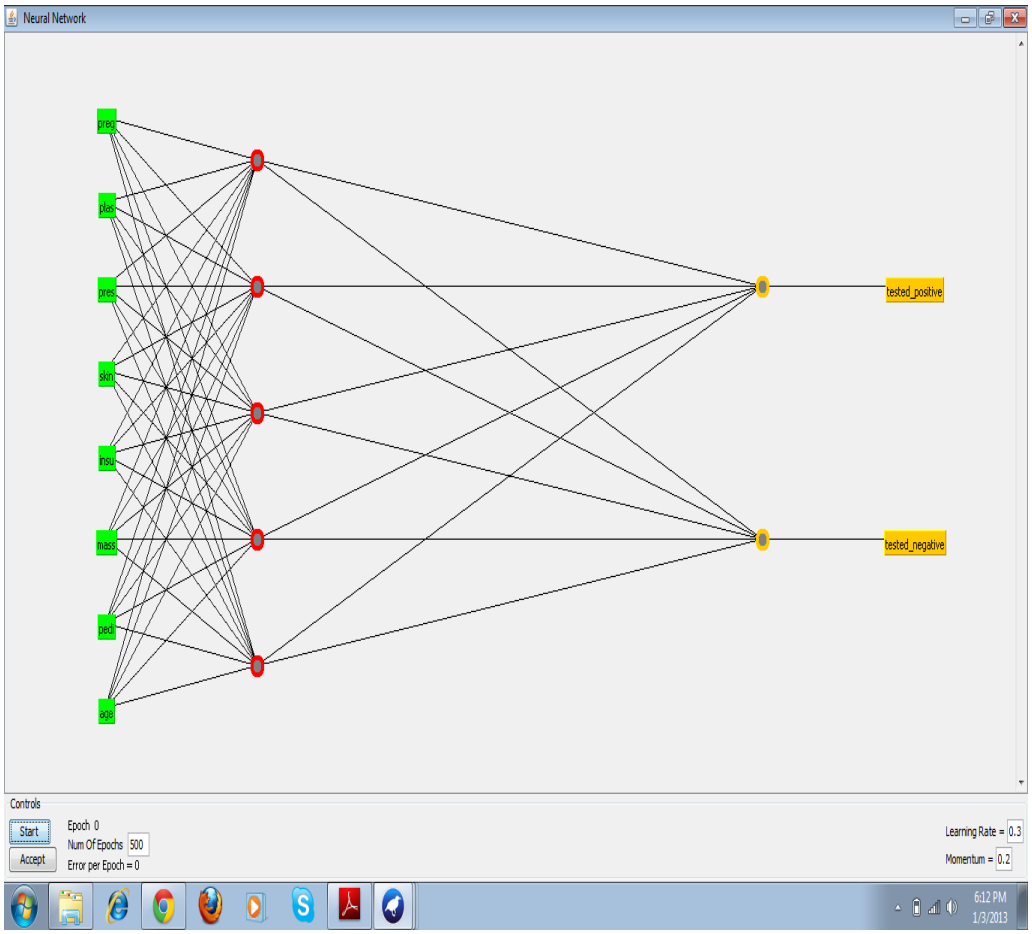

Fig 6 : Neural network classifier model build in WEKA

V. RESULTS AND DISCUSSION

The results obtained from the classification experiment with various learning rate on the diabetic dataset is tabulated in table 1. From the table we see that, as the learning rate increase from 0.0001 to 0.02 the Mean absolute error, Root mean squared error, Relative absolute error % and Root relative squared error % values decreases and then these values increases as the learning increase from 0.02 to 1.00. The values of correct classified instances % increases with the learning rate increase from 0.0001 to 0.02 in beginning and then decrease as learning rate increase from 0.02 to 1.00. On the other hand the values

of incorrect classified instances % decreases with the learning rate increase from 0.0001 to 0.02 in beginning and then increase as learning rate increase from 0.02 to 1.00. Fig 7 to Fig 10 shows the above results in graph form.

Table 1 : Classification results for different Learning Rate values

Learning rate	Correct classified instances %	Incorrect classified instances %	Mean absolute error	Root mean squared error	Relative absolute error %	Root relative squared error %
0.0001	65.0	35.0	0.453	0.476	99.7	99.8
0.0005	65.0	35.0	0.446	0.470	98.0	98.0
0.001	66.0	34.0	0.419	0.443	91.7	93.0
0.0015	76.4	23.5	0.370	0.412	81.4	86.5
0.002	76.6	23.4	0.343	0.401	75.4	84.0
0.0025	76.8	23.2	0.329	0.397	72.3	83.2
0.003	77.3	22.7	0.321	0.396	70.6	82.9
0.005	77.4	22.6	0.311	0.395	68.4	82.8
0.01	77.3	22.6	0.308	0.394	67.8	82.7
0.02	78.0	22.0	0.301	0.393	66.3	82.4
0.04	76.4	23.6	0.299	0.399	65.8	83.7
0.06	75.9	24.1	0.380	0.404	66.0	84.7
0.08	76.3	23.7	0.298	0.407	65.6	85.4
0.10	75.0	25.0	0.296	0.410	65.2	86.0
0.20	76.0	24.0	0.295	0.421	65.0	88.3
0.30	75.0	25.0	0.294	0.424	64.6	88.9
0.40	74.3	25.7	0.297	0.427	65.3	89.6
0.50	73.3	26.7	0.291	0.423	64.2	88.8
0.60	75.0	25.0	0.295	0.434	64.9	91.1
0.70	75.3	24.6	0.287	0.428	63.2	89.9
0.80	74.6	25.4	0.294	0.435	64.8	91.4
0.90	75.4	24.6	0.287	0.427	63.2	89.6
1.00	75.2	24.7	0.289	0.433	63.6	90.9

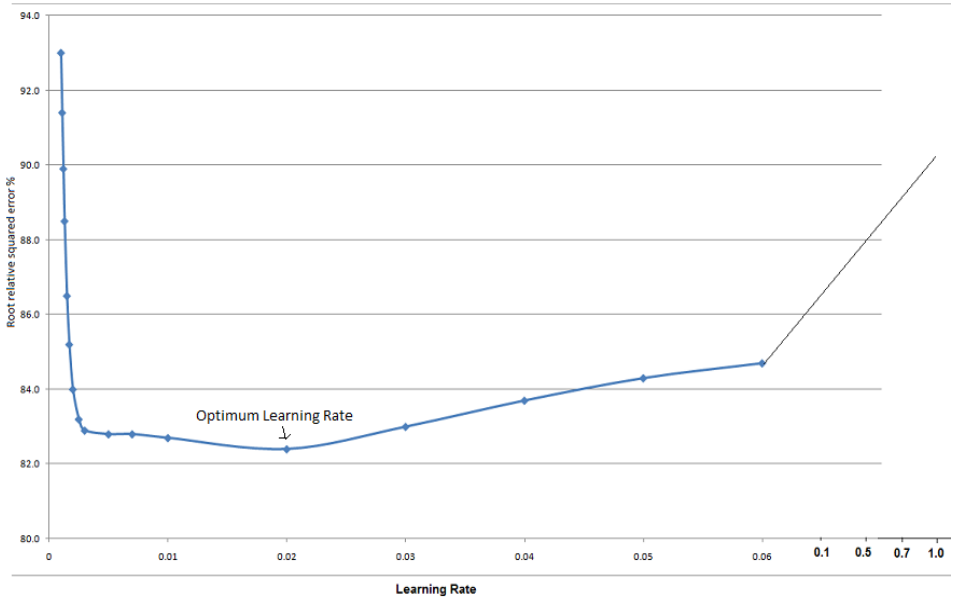

Fig 7 : Learning rate vs Root relative squared error %

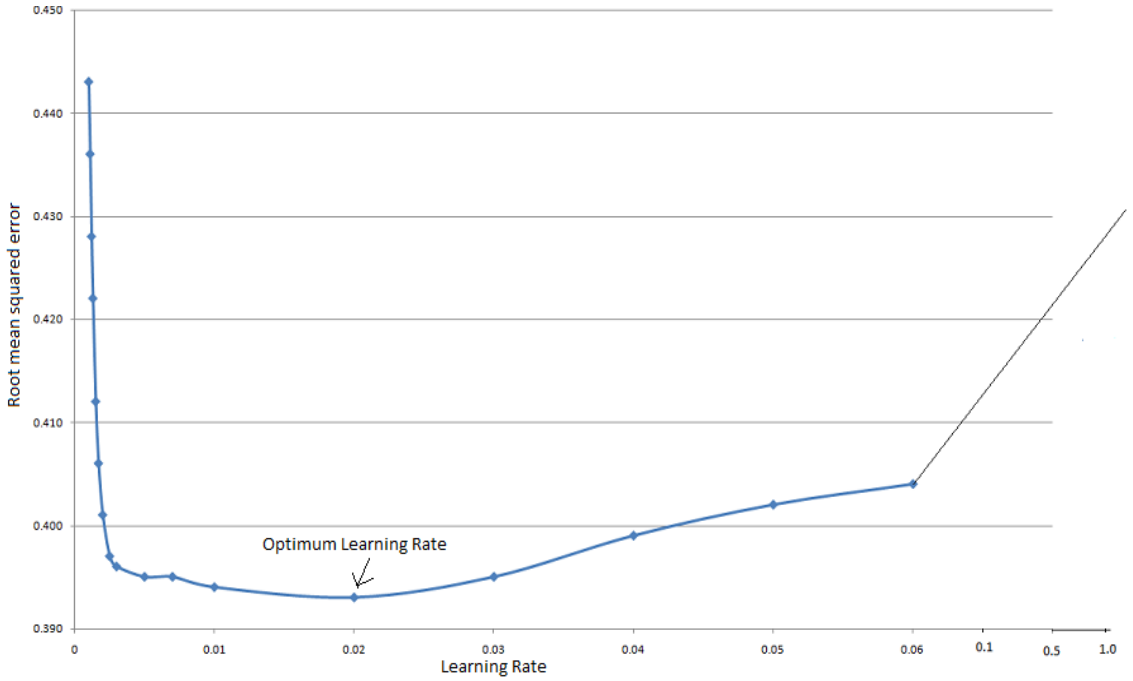

Fig 8 : Learning rate vs Root mean squared error

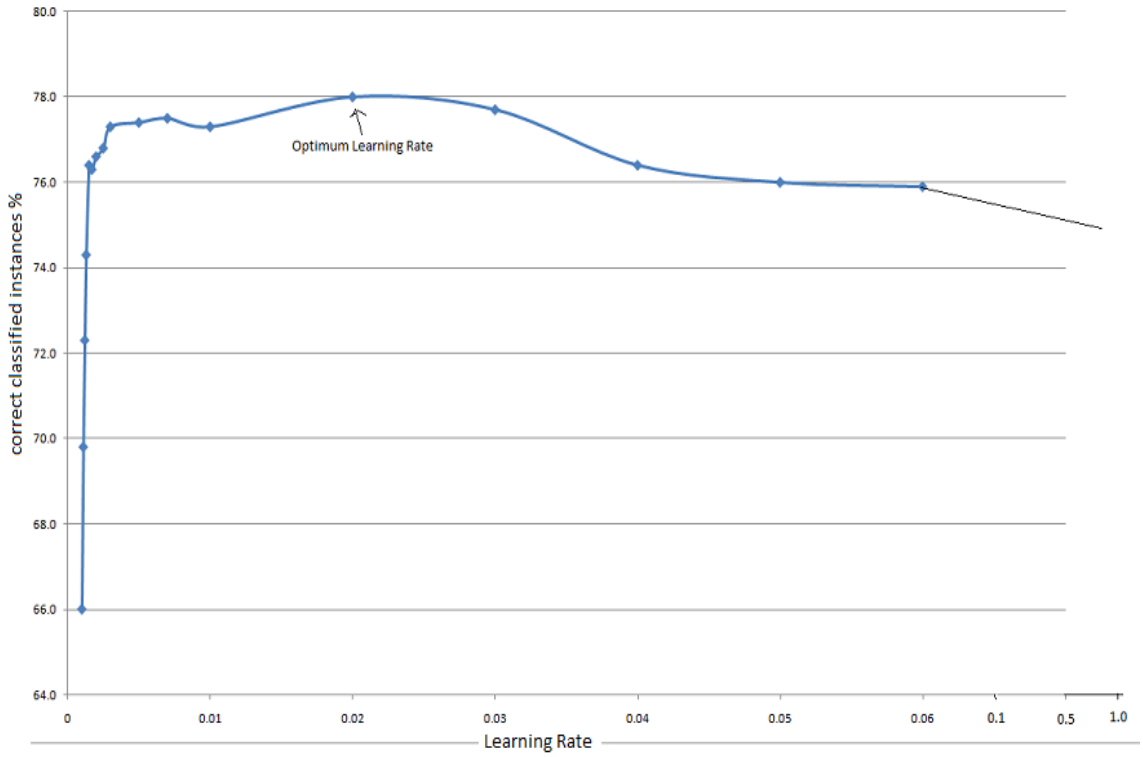

Fig 9 : Learning rate vs correct classified instances %

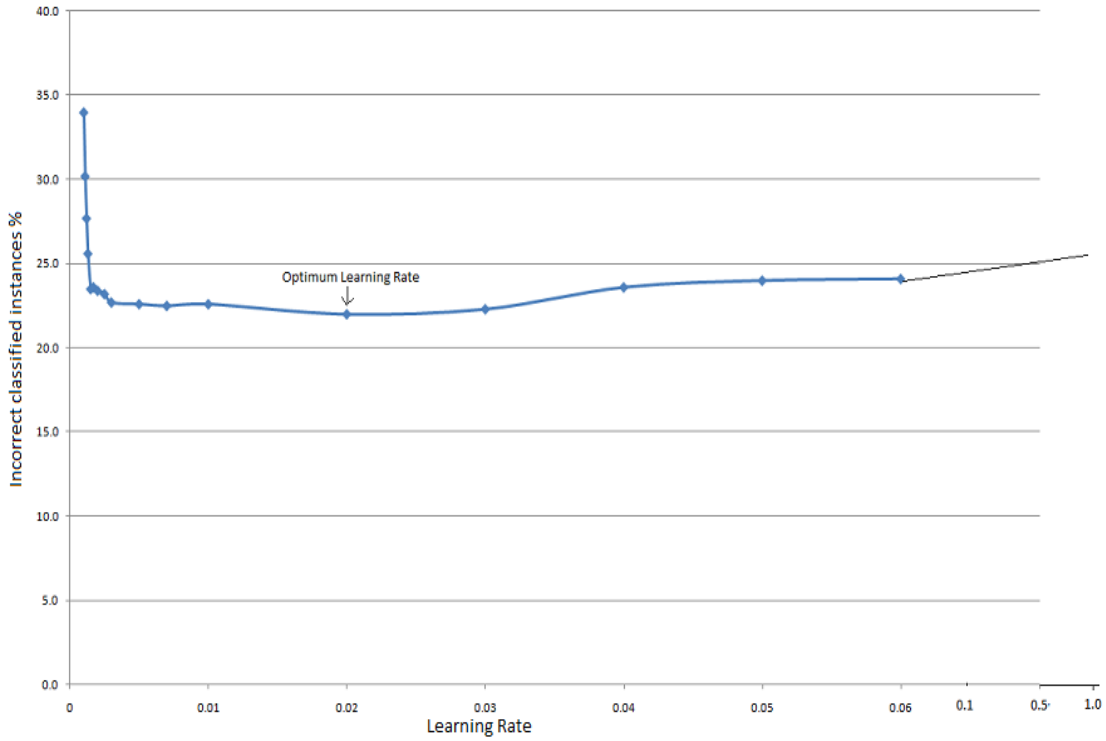

Fig 10 : Learning rate vs incorrect classified instances %

VI. CONCLUSIONS AND FUTURE WORK

The classification module in data mining WEKA toolbox with multi layer perception model using back propagation algorithm with diabetic dataset was run in incremental steps by varying the learning rate parameter in each step. The results obtained are plotted in graph form between learning rate vs Root relative squared error %, Root mean squared error, correct classified instances % and incorrect classified instances %. From all the graphs the optimum learning rate was found as 0.02. The learning rate parameter has to be carefully selected for any classification problem for achieving maximum percent correct classified instances. The learning rate at corresponding to maximum correct classified instances % is the optimum learning rate. We made an attempted to find the optimum learning rate which is stable and takes less time for convergence for classification problem. The optimum learning rate was found with only one diabetic dataset. The work may be extended with more number of datasets from multiple areas to verify the optimum learning rate.

REFERENCES

[1]. Abdul Habra, Neural Networks – An Introduction (To quickly define neurons, neural networks, and the back propagation algorithm) http://www.tek271.com
[2]. Jacobs, R A, Increased rates of convergence through learning rate adaptation, Neural Networks, 1988,1(4):295- 307
[3]. Tollenaere, T, SuperSAB: fast adaptive back-propagation with good scaling properties, Neural Networks, Vol. 3, No.5, pp. 561-573,1990
[4]. V.K Sood, N. Kandil, R.V. Patel, and K. Khorasani, "Comparative Evaluation of Neural Network Based Current Controllers for HVDC Transmission", Power Electronics specialists Conf., PESC'92, June 29-July 3, 1992, Toledo, Spain
[5]. Serge Roy, Factors influencing the choice of a learning rate for a back propagation neural network, 1994, IEEE World Congress on Computational Intelligence, 1994 IEEE International Conference on Neural Networks, Volume: 7
[6]. Janakiraman J. and Honavar V., 'Adaptive Leaning Rate for Increasing Leaning Speed in Back propagation Networks", F"* g s of Science of Artificial Neural Networks II, Vol. 1966, pp. 225-235.1993.
[7]. Fahlman, S. E., "An Empirical Study of Leaning Speed in Back Propagation Networks", Technical Report CMU-CS-88-162, Carnegie Mellon University, 1988.

[8]. Roy, S., "Near-Optimal Dynamic Leaning Rate for Training Back-Propagation Neural Networks", Proceedings of Science of Artificial Neural Networks II,Vol. 1966, pp. 277-283.1993
[9]. H. Demuth and M.Beal , "Neural Network ToolBox for use with MATLAB", The Math Works Inc., Natick, MA, 1992.
[10]. X. H. Yu et al., "Dynamic learning rate optimization of the back propagation algorithm," IEEE Trans. Neural Networks, vol. 6, pp. 669–677, May 1995
[11]. Tong Zhang, C. L. Philip Chen, Chi-Hsu Wang, Sik Chung Tam, A New Dynamic Optimal Learning Rate for A Two-Layer Neural Network, International Conference on System Science and Engineering, June 30-July 2, 2012, Dalian, China
[12]. Jing Wang, C. L. Philip Chen, Finding the Near Optimal Learning Rates of Fuzzy Neural Networks (FNNs) via Its Equivalent Fully Connected Neural Networks (FFNNs), International Conference on System Science and Engineering June 30-July 2, 2012, Dalian, China.
[13]. Qamar Abbas, Jamil Ahmad and Waqas Haider Bangyal, Analysis of Learning Rate using BP Algorithm for Hand Written Digit Recognition Application, International Conference on Information and Emerging technologies ICIET 14-16 June 2010, Islamabad, Pakistan
[14]. Amimi J, Optimum learning rate in back propagation network for classification of satellite images (IRS-1D), Scientia Iranica, Vol 15 No. 6 PP 558-567, Dec 2008
[15]. UCI Machine Learning Repository, Pima Indians Diabetes Data Set http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

AUTHORS

Lalitha Saroja Thota received M.Tech in Software Engineering from Jawaharlal Nehru Technological University, Hyderabad, India in 2010 and M.Sc in Computer Science from Annamalai University, Chennai, Tamilnadu in 2008. She is pursuing Ph.D in Computer Science and Engineering from Acharya Nagarjuna Univeristy, Guntur, AP, India since 2010. She has 5 years of teaching experience. Her areas of interest are bioinformatics and data mining. She has more than 4 research papers in international journals and conferences to her credit. She is currently working as Lecturer in College of Arts and Science Khamis Mushayt, Department of Computer Science, King Khalid University, Abha, KSA.

Suresh Babu Changalasetty received Ph.D in Computer Science and Engineering from Acharya Nagarjuna Univeristy, Guntur, India in 2009 and M.Tech in Computer Science and Engineering from Osmania Univeristy, Hyderabad, India in 2001. He has 12 years of teaching experience. His research areas of interest include bioinformatics, data mining and image processing. He has more than 11 research papers in international journals and conferences to his credit. He is currently Associate Professor in College of Computer Science, Department of Computer Engineering, King Khalid University, Abha, KSA.