Adaptive modified backpropagation algorithm based on differential errors

S.Jeyaseeli Subavathi[a] and T.Kathirvalavakumar[b]

[a]Department of Information Technology, Sri Kaliswari College, Sivakasi – 626130, Tamilnadu, India
[b]Department of Computer Science, V.H.N.S.N. College, Virudhunagar – 626001, Tamilnadu, India

Abstract

A new efficient modified back propagation algorithm with adaptive learning rate is proposed to increase the convergence speed and to minimize the error. The method eliminates initial fixing of learning rate through trial and error and replaces by adaptive learning rate. In each iteration, adaptive learning rate for output and hidden layer are determined by calculating differential linear and nonlinear errors of output layer and hidden layer separately. In this method, each layer has different learning rate in each iteration. The performance of the proposed algorithm is verified by the simulation results.

Keywords

Adaptive learning rate, Differential error, Linear error, Modified standard back propagation, Nonlinear error.

1. Introduction

The classical method for training feedforward neural network (FNN) is the backpropagation algorithm (BP) [9] which is based on the steepest descent optimization technique. **Training is usually carried out by iterative updating of weights based on the error signal. BP is a descent algorithm which attempts to minimize the error at each iteration. The weights of the network are adjusted by the algorithm such that the error is decreased along a descent direction [18]. Traditionally two parameters called learning rate and momentum factor are used for controlling the weight adjustment along the descent direction.** Finding initial learning rate and fixed learning rate must be done with great care. If the learning rate is very large, then the learning may become unstable. If it is small, then often it is very slow for practical applications which leads to finding of fast learning algorithms [13].

DOI : 10.5121/ijcsea.2011.1503

Many techniques have been proposed to increase the convergence speed. Abid et al. [1] described modified BP algorithm (MBP) based on sum of linear and nonlinear errors of output neurons to improve the speed of convergence in minimum iterations. The algorithm converges faster than the standard BP algorithm. Some researchers focused on selection of better energy function [2,14] and selection of suitable learning rate and momentum [6,9,16,17]. Learning rate adaptation by sign changes will adapt the step size by having a separate learning rate for each connection [12]

A problem with all of these techniques is their convergence to local minima. To solve this problem, global search algorithm like genetic algorithm have to be applied [4]. But searching for the global minimum may be trapped at local minima during gradient descent. Also if the network is trained with disturbances in the input, then global minimum point can not be found. So fast convergence and strong robustness may not be guaranteed. To solve these problems adaptive learning algorithms have been developed recently.

Jeong and Lee [7] have proposed an adaptive algorithm based on first and second order derivatives of neural activation at hidden layers which results in hybrid learning rules. Sha and Bajic [13] have proposed an adaptive learning rate algorithm for I/O identification based on two ANNs using convergence analysis of the conventional gradient descent method. Xie and Zhang [15] have proposed variable learning rate LMS algorithm using Lyapunav method especially when there is noise in the input signal. Behera et al. [3] have described new learning algorithms LFI and LF II based on Lyapunov function for the training of feeforward neural networks. In this algorithm fixed learning parameters are replaced with adaptive learning parameters using convergence theorem based on Lyapunov stability theory.]. **Zhihong Man et al [19] proposed a new adaptive backpropagation algorithm based on lyapunov stability theory for neural networks. They showed that the candidate of a lyapunov function of the tracking error between the output of a neural network and the desired reference signal is chosen first, and the weights of a neural network are then updated from the output layer to input layer.**

Our previous work [8] describes a modified backpropagation algorithm in neighborhood based network by replacing fixed learning parameters by adaptive learning parameters. Here the parameters are calculated using convergence theorem based on Lyapunov stability theory. Iranmanesh and Mahdavi [11] have proposed a learning method using differential adaptive learning rate. In each iteration, the learning rate is updated according to the error of the output layer. The learning rate of the output layer is computed by differentiating the error of the output layer. The differentiation of the sigmoidal function of the sum of multiplication of error of each output layer neuron with corresponding weights is divided by the number of hidden neurons is used as an adaptive learning rate of hidden layer.

We propose a new adaptive learning rate algorithm to speed up the learning process of the neural network. In the proposed algorithm separate adaptive learning rate is used in both hidden and output layers. In this, linear and nonlinear errors for each neuron in the output layer are multiplied with derivative of the corresponding neuron's activation function, added and then differentiated to get the adaptive learning rate for the output layer. Linear and nonlinear error of each hidden neuron is multiplied with its corresponding output layer weights separately and then added. Then the value is divided by number of hidden neurons. The differentiation of the sigmoidal function of this value is used as a learning rate for the hidden layer. The efficiency of the proposed algorithm in terms of time and epochs shown by simulating the benchmark problems such as XOR, 3-bit parity, nonlinear function approximation problem and iris data sets.

The remaining of the paper is organized as follows: section 2 describes adaptive learning rate algorithm, section 3 describes the proposed algorithm and section 4 discusses the simulation results.

2. Training of Neural network

Consider a single hidden layer feedforward neural network shown in Figure 1. A bias node is included in the input layer. Let X = (x_i) be the input vector, Y = (y_j) be the output vector and $w_{ji}^{[s]}$ be the weight of the i^{th} unit in the $(s-1)^{th}$ layer to the j^{th} unit in the s^{th} layer. The activation function of both hidden and output layer neurons are assumed to be sigmoidal. Sequential mode training is applied here.

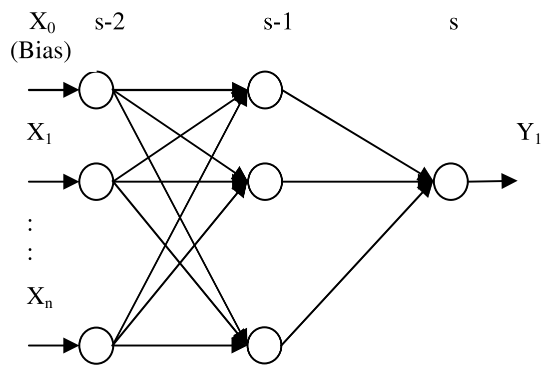

Figure 1. Single hidden layer neural network

.

Standard BP (SBP)

For each input pattern nonlinear output of the j^{th} neuron of the output layer network is calculated as follows:

$$u_j^s = \sum_{i=1}^{n_{(s-1)}} w_{ji}^s y_i^{s-1} \qquad (1)$$

$$f(u_j^s) = \frac{1}{\left(1 + e^{-u_j^s}\right)} = d_j^s \qquad (2)$$

where $n_{(s-1)}$ represents number of neurons in the $(s-1)^{th}$ layer.

SBP minimizes the following criterion equals to the sum of the squares of the errors between the actual y_j^s and the desired d_j^s outputs for a pattern p.

$$E_p = \sum_{j=1}^{n_s} \left(e_{1j}^s\right)^2 \qquad (3)$$

where the nonlinear error signal is

$$e_{1j}^s = (y_j^s - d_j^s) \qquad (4)$$

The weight update rule is

$$\Delta w_{ji}^{s} = -\mu \frac{\partial E_p}{\partial w_{ji}^{s}} \qquad (5)$$

where μ is the fixed learning rate selected by trial and error. Substituting (3) in (5), the weight update rule becomes,

$$\Delta w_{ji}^{s} = \mu e_{1j}^{s} \frac{\partial y_j^s}{\partial w_{ji}^{s}}$$

$$\Delta w_{ji}^{s} = \mu e_{1j}^{s} \frac{\partial y_j^s}{\partial u_j^s} \frac{\partial u_j^s}{\partial w_{ji}^{s}}$$

$$\Delta w_{ji}^{s} = \mu e_{1j}^{s} f'\left(u_j^s\right) y_i^{s-1} \qquad (6)$$

The estimated nonlinear error of the hidden layer (s-1) is as follows:

$$e_{1j}^{s-1} = \sum_{r=1}^{n_s} f'\left(u_r^s\right) e_{1r}^{s} w_{rj}^{s} \qquad (7)$$

The weight update rule of the hidden layer is

$$\Delta w_{ji}^{(s-1)} = -\mu \frac{\partial E_p}{\partial w_{ji}^{(s-1)}} \qquad (8)$$

$$\Delta w_{ji}^{(s-1)} = \mu e_{1j}^{(s-1)} f'\left(u_j^{(s-1)}\right) y_i^{(s-2)} \qquad (9)$$

Now the weights of both hidden and output layer are updated using

$$w_{ji}(t-1) = w_{ji}(t) + \Delta w_{ji} \qquad (10)$$

Modified BP

For each input pattern the linear and nonlinear outputs of the j^{th} neuron in output layer s of the network are calculated respectively as follows:

$$u_j^s = \sum_{i=1}^{n_{(s-1)}} w_{ji}^{s} y_i^{s-1} \qquad (11)$$

$$f(u_j^s) = \frac{1}{\left(1 + e^{-u_j^s}\right)} = d_j^s \qquad (12)$$

where $n_{(s-1)}$ represents number of neurons in the $(s-1)^{th}$ layer. The MBP approach minimizes modified form of criterion E_p used in standard BP algorithm. The criteria E_p is sum of the linear and nonlinear quadratic errors of the output neuron for the current pattern p.

$$E_p = \sum_{j=1}^{n_s} \frac{1}{2}\left(e_{1j}^s\right)^2 + \sum_{j=1}^{s_s} \lambda \frac{1}{2}\left(e_{2j}^s\right)^2 \qquad (13)$$

where the nonlinear error signal is

$$e_{1j}^s = (y_j^s - d_j^s) \qquad (14)$$

and the linear error signal is

$$e_{2j}^s = (ly_j^s - u_j^s) \qquad (15)$$

Here

$$ly_j^s = f^{-1}\left(y_j^s\right) \qquad (16)$$

where y_j^s and d_j^s respectively are desired and current output for j[th] unit in the s[th] layer. p in (13) denotes the p[th] pattern and λ is the weighting coefficient. In the output layer the linear and nonlinear errors are known [1]. So the weight update rule [1] for the output layer is

$$\Delta w_{ji}^s = -\mu \frac{\partial E_p}{\partial w_{ji}^s} \qquad (17)$$

where μ is the fixed learning rate selected by trial and error. Substituting (13) in (17), the weight update rule becomes,

$$\Delta w_{ji}^s = \mu e_{1j}^s \frac{\partial y_j^s}{\partial w_{ji}^s} + \mu \lambda e_{2j}^s \frac{\partial u_j^s}{\partial w_{ji}^s}$$

$$\Delta w_{ji}^s = \mu e_{1j}^s \frac{\partial y_j^s}{\partial u_j^s} \frac{\partial u_j^s}{\partial w_{ji}^s} + \mu \lambda e_{2j}^s y_i^{s-1}$$

$$\Delta w_{ji}^s = \mu e_{1j}^s f'\left(u_j^s\right) y_i^{s-1} + \mu \lambda e_{2j}^s y_i^{s-1} \qquad (18)$$

In the hidden layer, the linear and nonlinear errors are unknown and must be calculated [1]. The estimated nonlinear and linear error [1] of the hidden layer (s-1) are respectively as follows:

$$e_{1j}^{s-1} = \sum_{r=1}^{n_s} f'\left(u_r^s\right) e_{1r}^s w_{rj}^s \qquad (19)$$

$$e_{2j}^{s-1} = f'\left(u_j^{s-1}\right) \sum_{r=1}^{n_s} e_{2r}^s w_{rj}^s \qquad (20)$$

The weight update rule of the hidden layer is

$$\Delta w_{ji}^{(s-1)} = -\mu \frac{\partial E_p}{\partial w_{ji}^{(s-1)}} \qquad (21)$$

$$\Delta w_{ji}^{(s-1)} = \mu e_{1j}^{(s-1)} f'\left(u_j^{(s-1)}\right) y_i^{(s-2)} + \mu \lambda e_{2j}^{(s-1)} y_i^{(s-2)} \qquad (22)$$

Now the weights of both hidden and output layer are updated using

$$w_{ji}(t-1) = w_{ji}(t) + \Delta w_{ji} \qquad (23)$$

where t represents iteration. In order to increase the convergence speed and to make the learning rate μ adaptive , we propose a new technique based on differential linear and nonlinear errors of output layer and hidden layer.

Adaptive Modified BP

In the proposed technique first linear and nonlinear errors of j^{th} neuron in the output layer s are calculated using (14), (15) and (16). Then all the linear and nonlinear errors of the neurons are multiplied with the derivative of the corresponding neuron's activation function and added separately as shown below:

$$\delta_{o1} = \sum_{j=1}^{n_s} e_{1j}^{s} f'(u_j^s) \qquad (24)$$

$$\delta_{o2} = \sum_{j=1}^{n_s} e_{2j}^{s} f'(u_j^s) \qquad (25)$$

Then δ_{o1} and δ_{o2} are added to get the total error

$$\delta_o = \delta_{o1} + \delta_{o2} \qquad (26)$$

Now the total error is divided by the total number of output neurons known as δ^a

$$\delta^a = \frac{\delta^a}{n_s} \qquad (27)$$

and the μ_{out} of the output layer s is computed as follows:

$$\mu_{out} = f'\left(\delta^a\right) \qquad (28)$$

where f is a sigmoidal activation function given by

$$f\left(\delta^a\right) = \frac{1}{\left(1 + e^{-\delta^a}\right)} \qquad (29)$$

with property

$$f'\left(\delta^a\right) = f\left(\delta^a\right)\left(1 - f\left(\delta^a\right)\right) \qquad (30)$$

Then the change of weights are calculated using

$$\Delta w_{ji}^{s} = \mu_{out} e_{1j}^{s} f'\left(u_j^s\right) y_i^{s-1} + \mu_{out} \lambda e_{2j}^{s} y_i^{s-1} \qquad (31)$$

Similarly for the hidden layer (s-1) the same procedure is applied to calculate adaptive learning rate μ_{hid} . First nonlinear errors $e_{1j}^{(s-1)}$ and linear errors $e_{2j}^{(s-1)}$ of all hidden neurons are calculated using (19) and (20). Then nonlinear errors δ_{h1} and δ_{h2} respectively are

$$\delta_{h1} = \sum_{i=1}^{n_{(s-1)}} \sum_{j=1}^{n_s} e_{1j}^{(s-1)} w_{ji}^{s} \qquad (32)$$

$$\delta_{h2} = \sum_{i=1}^{n_{(s-1)}} \sum_{j=1}^{n_s} e_{2j}^{(s-1)} w_{ji}^{s} \qquad (33)$$

and then both δ_{h1} and δ_{h2} are added to get the total error δ_h as below:

$$\delta_h = \delta_{h1} + \delta_{h2} \qquad (34)$$

Now the total error is divided by the total number of hidden neurons known as δ^b

$$\delta^b = \frac{\delta_h}{n_{(s-1)}} \qquad (35)$$

and then μ_{hid} is computed as follows:

$$\mu_{hid} = f'\left(\delta^b\right) \qquad (36)$$

where f is a sigmoidal activation function. Then the change of weights are calculated using the following equation.

$$\Delta w_{ji}^{(s-1} = \mu_{hid} e_{1j}^{(s-1)} f'\left(u_j^{(s-1)}\right) y_i^{(s-2)} + \mu_{hid} \lambda e_{2j}^{(s-1)} y_i^{(s-2)} \qquad (37)$$

Now the weights of both hidden and output layer are updated using (23).

3. Algorithm

1. Define network structure and assign initial weights randomly.
2. Select a pattern to be processed in the network.
3. For each node in the hidden layer, compute
   a. Net value using Eq (11).
   b. Output value using Eq (12).
4. For the output layer, compute
   a. Net value using Eq (11) and output value using Eq (12).
   b. Non Linear and linear errors using Eq (14), Eq (15) and Eq (16).
   c. Adaptive learning rate μ_{out} using Eq (24) to Eq (30).
   d. Change of weight using Eq (31).
5. For the hidden layer, compute
   a. Non Linear error using Eq (19).
   b. Linear error using Eq (20).

   c. Adaptive learning rate μ_{hid} using Eq (32) to Eq (36).
   d. Change of weight using Eq (37)
6. Update weights of output and hidden layer using Eq (23).
7. Repeat the steps 2 to 6 for all the patterns.
8. Evaluate network error with new weights.
9. Stop training if termination condition is reached. Otherwise repeat the steps 3 to 9.

4. Simulation Results and discussions

The performance of the proposed algorithm is verified by simulating the benchmark problem such as XOR, 3-Bit parity, Nonlinear function approximation function problem and Iris data set. All the problems are simulated using language C on a Pentium IV with 2.40 GHz. The convergence property of the proposed algorithm is compared with MBP [1], Backpropagation with momentum (BPM) [9] and backpropagation algorithm [10]. Each time all the patterns in the problem have been used once in the network during training is called an epoch. Mean squared error (MSE) of the network is calculated by dividing the sum of squared linear error in each epoch by twice the number of patterns. Network structure, parameter values and termination condition are considered as constant for all the algorithms to have better comparison. Network weights are randomly and uniformly generated from the range [-5, +5]. The weighting coefficient λ is assigned the value 3.7. **The convergence of the proposed algorithm is shown by the learning curve.**

XOR

The network structure considered in this problem has 3 input neurons including bias, 4 hidden neurons and one output neuron. The termination condition fixed for convergence is MSE 0.001. The results obtained are tabulated in Table 1.

Table 1: Comparison table for XOR problem

ALGORITHM	PARAMETERS	EPOCHS	TRAINING MSE	TIME IN MSECS
BP	μ=1.15	754	0.000998	176
BPM	μ=1.15 α=0.01	710	0.001	151
MBP	μ=0.25 λ=0.01	501	0.001	115
Proposed	λ=3.7	237	0.000987	49

It has been observed that the BP algorithm takes 176 msecs and 754 epochs to reach the minimum error. The proposed algorithm converges faster even the learning rate is not fixed in the beginning. Since the learning rate is adapted based on the error of output and hidden layers it takes minimum time of 49 msecs and minimum epochs of 237 for convergence. The learning curve obtained is shown in Figure 2 for the proposed algorithm. The adaptive learning rate obtained based on the error of output layer and hidden layer are shown in Figure 3 and Figure 4.

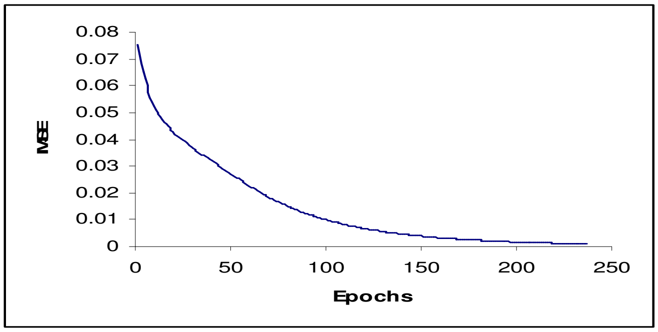

Figure 2. Learning curve based on MSE and Epochs of XOR problem for the proposed algorithm

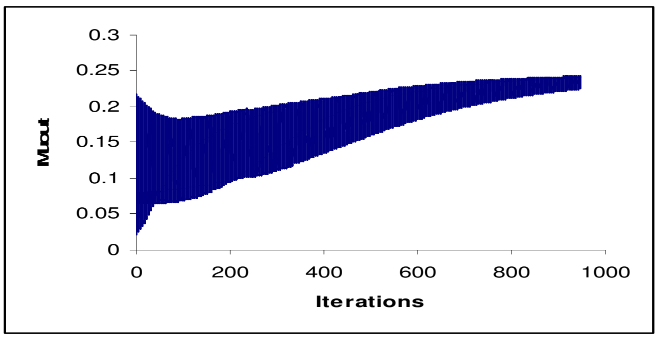

Figure 3. Adaptive learning rate of hidden layer.

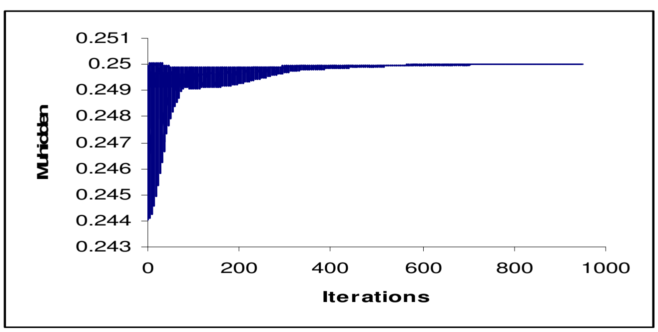

Figure 4. Adaptive learning rate of output layer.

3-bit parity

We used 4-9-1 ANN including one bias in input layer to simulate the 3-bit parity problem. The results obtained are tabulated in Table 2.

Table 2. Comparison table for the 3-bit parity problem

ALGORITHM	PARAMETERS	EPOCHS	TRAINING MSE	TIME IN MSECS
BP	μ=1.15	1570	0.000999	379
BPM	μ=1.15 α=0.01	1450	0.000998	364
MBP	μ=0.25 λ=0.01	520	0.000995	126
Proposed	λ=3.7	298	0.000997	77

From the table it has been observed that the proposed algorithm converges quickly within 77 msecs in 298 epochs. But the algorithm BP, BPM and MBP require 1570, 1450 and 520 epochs for convergence respectively. Also they require 379 msecs, 364 msecs and 126 msecs time to reach the termination condition MSE 0.001. All the algorithm except proposed algorithm take time to fix the learning rate. The best performance of the proposed algorithm is shown in Figure 5.

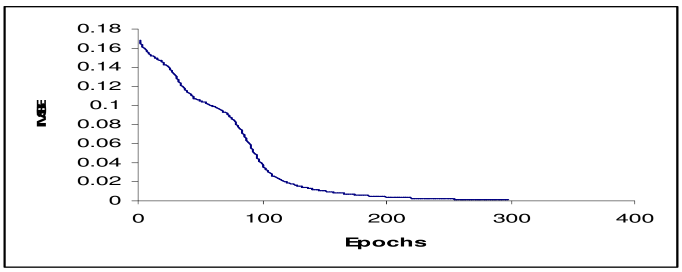

Figure 5. Learning curve based on MSE and Epochs of 3-bit parity problem for the proposed Algorithm

Nonlinear function approximation problem

A nonlinear function approximation with 8 input values x_i is defined in this problem. The three output quantities y_i are defined by the following equations

$$y_1 = (x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8)/4$$

$$y_2 = (x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8)/8$$

$$y_3 = (1 - y_1)^{1/2}$$

500 number of input values x_i ε (0,1) are randomly generated and the corresponding y_i are calculated using the above equation. All the algorithms taken for comparison are assumed to have the network structure with 9 neurons in the input layer including bias, 5 neurons in the hidden layer and 3 neurons in the output layer. All the algorithms including proposed is fixed with the minimum error of MSE 0.004. The results obtained are tabulated in Table 3. It shows that the algorithms BP and BPM converge to MSE 0.004 in 590 epochs and 389 epochs within 612 msecs and 487 msecs respectively. But MBP converges to the termination condition with the maximum

of 75 epochs within 89 msecs. The proposed algorithm converges quickly in 25 epochs within 51 msecs.

Table 3. Comparison table for the nonlinear function approximation problem.

ALGORITHM	PARAMETERS	EPOCHS	TRAINING MSE	TESTING MSE	TIME IN MSECS
BP	μ=1.15	590	0.003990	0.004285	612
BPM	μ=1.15 α=0.01	389	0.003995	0.004125	487
MBP	μ=0.25 λ=0.01	75	0.003574	0.003913	89
Proposed	λ=3.7	25	0.003796	0.003835	51

The learning curve of the proposed algorithm is shown in Figure 6. Another set of 500 patterns are generated for testing. The testing MSE obtained for the proposed is 0.003835 and for the MBP is 0.003913.

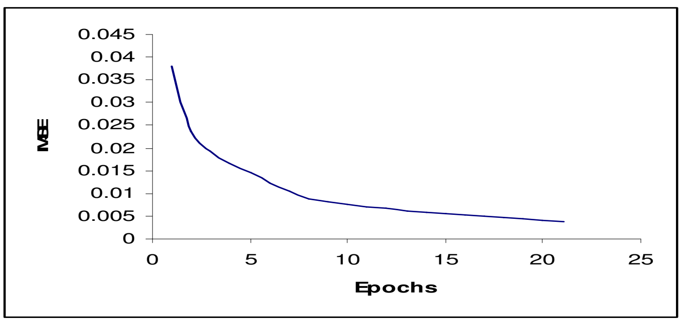

Figure 6. Learning curve based on MSE and Epochs of Non linear function approximation problem for the proposed algorithm

Iris data set

The Iris data [5], is one of the best known databases in the pattern recognition literature. The data set contains three classes. Each class has 50 instances, totally 150 patterns are used. Among 75 patterns are used for training and the remaining for testing. All the values are normalized by dividing the value by 10. The network structure considered is 5-10-1 including one bias in the input layer. Table 4 shows the results obtained for all the algorithms taken for comparison.

Table 4. Comparison table for the Iris data set problem.

ALGORITHM	PARAMETERS	EPOCHS	TRAINING MSE	TESTING MSE	TIME IN MSECS
BP	μ=1.15	491	0.0003	0.008172	193
BPM	μ=1.15 α=0.01	414	0.0003	0.008155	165
MBP	μ=0.25 λ=0.01	368	0.0003	0.006869	143
Proposed	λ=3.7	95	0.00029	0.006654	33

The proposed algorithm and MBP take minimum epochs of 95 and 368 and minimum time of 33 msecs and 143 msecs respectively. But BP and BP with momentum require 491 and 414 epochs and 193 msecs and 165 msecs respectively to reach the termination condition MSE 0.0003. Also the testing MSE obtained for the proposed algorithm is minimum. The learning curve drawn against epochs and MSE for the proposed algorithm is shown in Figure 7.

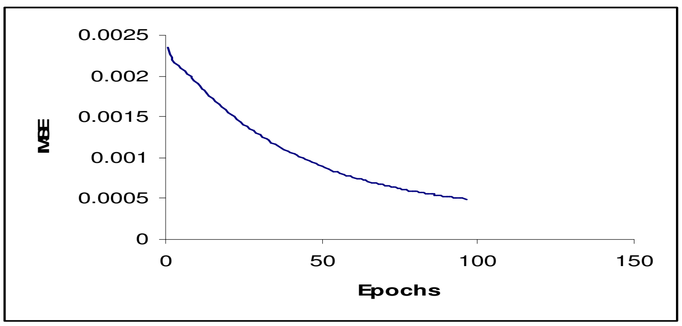

Figure 7. Learning curve based on MSE and epochs of Iris data set problem for the proposed algorithm.

4. Conclusion

An efficient technique for adapting the learning rate in modified backpropagation algorithm for training sequential FNN is proposed. Here, the learning rate is adapted based on the differential linear and nonlinear errors of output and hidden layers. Separate adaptive learning rate is used for both hidden and output layer in each iteration. The time required to fix the learning rate by trial and error is saved. The proposed algorithm improves the convergence speed in terms of time and epochs which is shown by simulating four different problems. The main advantage of the proposed algorithm is no need to put effort to tune the learning parameter to obtain optimal convergence. The proposed algorithm is easy to implement and easy to compute learning rate for both hidden and output layer which modifies the values of weights and increases the convergence speed. The learning curve show that the convergence is guaranteed.

References

[1] Abid S, Fnaiech F, Najim M, (2001), “A fast feedforward training algorithm using a modified form of the standard backpropagation algorithm,” IEEE Trans. Neural Networks 12 424-430.

[2] Ahmad M, Salam F.M, (1992) “Supervised learning using cauchy energy function,” Proc. 2nd Int. Conf. Fuzzy logic neural networks, lizuka, Japan, 721-724.

[3] Behera L, Kumar S, Patnaik A, (2006) “On adaptive learning rate that guarantees convergence in feedforward networks,” IEEE Trans. Neural Networks 17 1116-1125.

[4] Bengio S, Bengio Y, Cloutier J, (1994) “Use of genetic programming for the search of a new learning rule for neural networks,” Proc.IEEE World Congr. Computational Intelligence and Evolutionary, 324-327

[5] Fisher R.A, (1936) “The use of multiple measurements in taxonomic problems,” Annual Eugenics 7 179-188.

[6] Jacobs R.A, (1988) “Increased rates of convergence through learning rate adaptation,” Neural networks 1 295-307..

[7] Jeong S.Y, Lee S.Y, (2000) “Adaptive learning algorithms to incorporate additiona functional constraints into neural networks,” Neurocomputing 35 73-90.

[8] Kathirvalavakumar T, Subavathi S.J, (2009) “Neighborhood based modified backpropagation algorithm using adaptive learning parameters for training feedforward neural networks,” Neurocomputing 72 3915-3921.

[9] Rojas R, (1996) “Neural networks : a systematic introduction,”. Berlin. Springer verlag; 424-430.

[10] Rumelhart DE, Hinton GE, Williams RJ, 1 (1986) “Learning internal representations by error propagations,” Parallel distributed processing: explorations in the microstructures of cognition, Cambridge(MA): MIT Press; 62-318.

[11] Saeid Iranmanesh, Amin Mahadevi M, (2009) “Differential adaptive learning rate method for back propagation neural networks,” World Academy of Science, Engineering and Technology 50 285-288.

[12] Sarkar D, (1995) “Methods to speed up error back propagation learning algorithm,” ACM Comput. Surv., 27 519-544

[13] Sha D, Bajic V.B, (1999) “Adaptive on-line ANN learning algorithm and application to identification of non-linear systems,” Informatica 23 521-529

[14] Van Ooyen A, Nienhuis B, (1992) “Improving the convergence of the backpropagation algorithm, Neural networks,” 5 465-471

[15] Xie S, Zhang C, (2006) “Variable learning rate LMS based linear adaptive inverse control,” Journal of information and computing science 1 139-148

[16] Yu C.C, Liu B.D, (2002) “A backpropagation algorithm with adaptive learning rate and momentum coefficient,” Proc.Int.Joint Conf. Neural networks(IJCNN'02), 2 1218-1223

[17] Yu X.H, Chen G.A, Cheng S.X, (1993) “Acceleration of backpropagation of learning using optimzed learning rate and momentum,” Electron.Lett, 29(14) 1288-1289.

[18] Yahya H. Zweiri, (2006) “ Optimization of a three term backpropagation algorithm used for neural network learning “, International journal of Computational Intelligence 3 322 – 327.

[19] Zhihong Man, Hong Ren Wu, Sophie Liu, Xinghuo Yu, (2006), “ A new adaptive backpropagation algorithm based on Lyapunov stability thory for neural networks”, IEEE Transactions on Neural Networks 17 1580-1591.

Authors

T.Kathirvalavakumar received M.Sc. degree in Mathematics from Madurai Kamaraj University in 1986, Post Graduate Diplomo in Computer Applications from Bharathidasan University in 1987, M.Phil. degree in Computer Science from Bharathiar University in 1994 and the Ph.D. degree in Computer Science from University of Madras in 2004. Since 1987 he has been working as a Lecturer, currently Associate Professor in Computer Science at V.H.N.Senthikumara Nadar College, Virudhunagar, Tamilnadu, India. His research interests include Neural Networks and Applications, Pattern recognition and Data Mining.

S.Jeyaseeli Subavathi received the MCA degree from Madurai Kamaraj University, in 1998 and M.Phil degree in Computer Science, from Mother Teresa Women's University in 2004. From January 2000 to April 2007 she worked as Lecturer in Computer Applications at SFR College, India. Since July 2007 she has been working as a Lecturer in Information Technology, Sri Kaliswari College, Sivakasi, Tamilnadu, India. At present she is a doctoral candidate in the Department of Computer Science at Madurai Kamaraj University, India. Her area of interests include Neural Networks and Data structures and algorithms.