I.J. Modern Education and Computer Science, **2014, 12, 34-40**
Published Online December 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2014.12.05

Comparative Study of High Speed Back-Propagation Learning Algorithms

Saduf
Dept. of computer sciences, University of Kashmir, Srinagar, j&k, India
Sadaf.afzal.123@gmail.com

Mohd.Arif Wani
Dept. of computer sciences, University of Kashmir, Srinagar, j&k, India
awani@uok.edu.in

Abstract—Back propagation is one of the well known training algorithms for multilayer perceptron. However the rate of convergence in back propagation learning tends to be relatively slow, which in turn makes it computationally excruciating. Over the last years many modifications have been proposed to improve the efficiency and convergence speed of the back propagation algorithm. The main emphasis of this paper is on investigating the performance of improved versions of back propagation algorithm in training the neural network. All of them are assessed on different training sets and a comparative analysis is made. Results of computer simulations with standard benchmark problems such as XOR, 3 BIT PARITY, MODIFIED XOR and IRIS are presented. The training performance of these algorithms is evaluated in terms of percentage of accuracy, and convergence speed.

Index Terms—ANN, gain, momentum, error saturation, Local minima.

I. Introduction

Neural networks are information processing networks that mimic the human nervous system. Sigmoid activation function which models the actual behaviour of a neuron is mainly used, even though there exist various other functions that represent the characteristics of neuron. A neural network can be trained to imitate any function by using a training set that contains the input output pairs of the desired function. The popular structure of neural networks is multi layer perceptron (MLP) in which the neurons are placed in some layers and signal is transmitted in one direction. MLP consists of an input layer, one or more hidden layer and an output layer. The hidden layer is used to capture the non linear relationships among input variables and the output layer is used to obtain the predicted output. Back propagation algorithm (BP) introduced by [1] is a supervised learning method based on the gradient descent of the quadratic error function and is considered as the universal function approximator. Back propagation based MLP (BPNN) could approximate any smooth function to an arbitrary degree of accuracy, when the tuning parameters could be optimized properly. Supervised learning of a neural network can be viewed as a curve fitting process. A training vector pairs that consist of an input vector from the input space and a target vector as the neural response are presented to the network. Based on the learning algorithm, the neural network performs the weight adjustment so that the error between the actual output vectors and the target vector is minimized relative to some optimization criterion. Once trained the neural network performs the interpolation in the output vector space which is then referred to as the generalization capability. While as in unsupervised learning the target output pattern is not known and the system learns of its own by discovering the features in the input patterns. This type of learning is based on clustering technique. Reinforcement learning is based upon both the supervised learning as well as unsupervised learning. In this type of learning although a teacher is present but the correct answer is not presented to the network. The teacher only indicates whether the computed answer is correct or not.

This paper is divided into two prime areas. The second section provides a general idea of BP and presents the brief idea of some of the improvements of BP. In the third section a performance comparative analysis of the results is made that are obtained by implementing some of improved versions of BP on a number of benchmark problems.

II. Bp Algorithm

The MFNN is presented with a set of exemplar cases consisting of input pattern and target pattern. The input pattern is fed directly into the input layer. The activations of the input nodes are multiplied by the weighted connections and are passed through a transfer function at each node in the first hidden layer. The activations from the first hidden layer are then passed to the neurons in the next layer, and this process is repeated until the output activations are obtained from the output layer. The output activation values and the target pattern are compared and the error signal is calculated based on the difference

between target and calculated pattern. This error signal is then propagated backwards to adjust network weights so that network will generate correct output for the presented input pattern. The training patterns are presented repeatedly until the error reaches an acceptable value or other convergence criteria are satisfied. As this technique involves performing computation backwards it is named as backpropagation.

The purpose of the training process is to reduce the sum squared error function (MSE) which is given as:

$$E(n) = \frac{1}{2}\sum_{c\in N}(t_k(n) - o_k(n)^2 \quad (1)$$

t_k and o_k are the desired and actual outputs of neuron k. The averaged squared error over the total number of patterns N is given by:

$$E_{av} = \frac{1}{2}\sum_{n=1}^{N} E(n) \quad (2)$$

The output of the unit j in any layer after applying the sigmoid function is given by:

$$o_j = \frac{1}{1 + e^{a_j}}$$

And the activation a_j is given by

$$a_j = \sum_i w_{ij} o_i$$

w_{ij} refers to the weighted connection between neuron i and neuron j. A weight change Δw_{ij} is calculated as:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

η is the learning rate. Once the initial weights are known the formula for the weight updation takes the following form:

$$W^{k+1} = W^k - \eta E_W(W^k) \quad (3)$$

Where $E_W(W^k)$ is partial derivative of E with respect to weight vector W Updation of weights in BP can be done in two ways: online updating and batch updating. In batch updating the weight adjustments are made only at the end of epoch i.e. after the presentation of the entire training set. On the other hand in online updating the weights are updated after every pattern presentation. In both the schemes the learning process continues until the sum squared error reaches a predefined value.

BP algorithm has many considerable advantages, such as it is computationally efficient, simple and easy to implement. However it has some disadvantages also such as it may converge to local minima and convergence to the global minima is not always guaranteed. Even if the specified termination criterion is achieved it takes long time to converge. However most of the researchers have found that most of the limitations of BP are because of the flat spot problem as well as due to choice of the initial values of the network weight connections and the parameters that are used in the algorithm such as learning rate and momentum. The flat spot problem generally occurs when the actual output is in the saturation areas i.e. '0' or '1'. Flat spot problem makes it very difficult for a neural network to learn. It results in slow learning speed and slight weight update of the neuron network and thus taking long time for a neural network to converge. Many modifications have been proposed to improve the performance of BP, many of which include 1) momentum strategy 2) using error saturation prevention function 3) using proper weight initialisation methods 4) adjusting the steepness of sigmoid function. These are summarized below.

A. Momentum Strategy

The learning strategy that is used in original BP is gradient descent, considering the effect of learning rate on BP reveals that smaller the learning rate, smaller will be the changes to the synaptic weights in the network from one iteration to the next. On the other hand if the learning rate is made too large, it will result in large changes to the synaptic weights and in turn makes network unstable. A simple method of increasing the learning rate nonetheless avoiding the risk of instability of network is to add momentum coefficient to the weight updation rule. The momentum strategy [2] adds a fraction of the last weight change to the current direction of weight change. The weight updating rule for BP with momentum is given as:

$$W^{k+1} = W^k - \eta E_W(W^k) + \alpha \Delta W^{k-1} \quad (4)$$

But it was found that a fixed momentum coefficient accelerates the learning only when the current downhill gradient of the error function and the last change in weight have the similar direction. However, when the current gradient is in an opposing direction to the previous weight change, the momentum causes the weight to be adjusted up the slope instead of down the slope. In order to make the learning more effective a number of methods have been proposed by researchers to dynamically vary the momentum coefficient.

Reference [3] proposed an adaptive momentum approach by considering the current negative gradient and the last weight change. If the current weight update vector is in a similar direction to previous update, the momentum term is increased. However, if the current weight update is in an opposing direction to previous update then the momentum is reduced to zero. The weight update vector is given as:

$$W^{k+1} = W^k - \eta E_W(W^k) + \alpha(1 + \cos\theta)\,\Delta W^{k-1} \quad (5)$$

Reference [4] proposed an efficient acceleration technique BP with adaptive learning rate and momentum term. This algorithm has faster convergence rate than those of using a fixed parameter, but at the price of extra computation. In this algorithm each weight has its own

learning rate and momentum factor, which are adapted at each iteration. The learning rate and momentum are adapted on the basis of the relative factor which is defined as: $e_k = \frac{\mathrm{E}(W^K)-E(W^{k-1})}{E(W^k)}$.

The weight update vector is given as:

$$W^{k+1} = W^k - \eta_k E_W(W^k) + \alpha_k \Delta W^{k-1} \qquad (6)$$

The learning term and the momentum are adjusted as follows:

$$\eta_k = \begin{cases} \eta_{k-1}[1 + u.e^{-e_{k(n)}}] & if\ e_k(n) < 0 \\ \eta_{k-1}[1 - u.e^{-e_{k(n)}}] & if\ e_k(n) \geq 0 \end{cases} \quad u\ \varepsilon(0,1)$$

$$\alpha_k = \begin{cases} \alpha_{k-1}[1 + u.e^{-e_{k(n)}}] & if\ e_k(n) < 0 \\ \alpha_{k-1}[1 - u.e^{-e_{k(n)}}] & if\ e_k(n) \geq 0 \end{cases} \quad u\ \varepsilon(0,1)$$

Reference [5] adjusted the momentum coefficient dynamically. They adjusted the momentum coefficient iteratively based on the inner product between the current descent direction and the last weight increment. When the angle between the current negative gradient and the last weight change is less than 90^0, the momentum is defined as a positive value to accelerate the learning. On the other hand when the angle between them is more than 90, the momentum coefficient is set to 0. The momentum coefficient is changed according to:

$$\alpha_i^k = \begin{cases} \alpha \frac{-\eta \mathrm{EW_i}(W^k).\Delta \mathrm{W_i^{k-1}}}{||\Delta \mathrm{W_i^{k-1}}||^2} & if\ \mathrm{EW_i}(W^k)\Delta \mathrm{W_i^{k-1}} < 0 \\ else \quad 0 \end{cases} \qquad (7)$$

B. Using Error Saturation Prevention Function

The sigmoid derivative that is used in the original BP method can sometimes cause slow learning when the output of a unit is near '0' or '1' (error saturation). As a result of which the learning term will be very small leading to a very little progress in the weight adjustments. In order to avoid error saturation phenomena various modifications to the error function has been proposed. Reference [6] found that if the actual value of the output node is under error saturation condition, it makes the learning term small which in turn results in little change to the weight adjustments. In order to avoid this problem he proposed a modified energy function. The proposed energy function will make the learning term quite good no matter what is the distribution of the actual output value. Reference [7] proposed a modified energy function that could scale up the partial derivatives of the activation function and proposed a new weight evolution algorithm based on the modified energy function. Reference [8] improved the convergence speed by proposing a new energy function based on the cross entropy algorithm. Reference [9] found that the error saturation condition is the main reason for the premature saturation (PS). They handled the PS problem by proposing an error saturation prevention function (ESP) which is a parabolic function to the learning term in the nodes of the output layer to overcome the error saturation that is caused by the gradient descent method. The error saturation prevention function scales up the learning term when the actual output is near the extreme value of 0 or 1.

The ESP function for the output node is given as:

$$\mathrm{ESP}\ (o_i) = \alpha(o_k - 0.5)^n \qquad (8)$$

Where α is a scale term, n is an exponent value and o_i is the actual output of the i^{th} node.This ESP function is then incorporated in the learning term for output nodes. The learning term for the output nodes after including the ESP function is given as:

$$\delta_k = (t_k - o_k)o_k(1 - o_k) + \mathrm{ESP}\ (o_k) \qquad (9)$$

However it was found that the learning term for the hidden layer will also be small when the actual output of a unit arrives in the saturation area of 0 or 1. So in order to handle the error saturation condition in the hidden layer they used a constant factor (real value) to the ESP function of the output layer to make the enhancement of the learning term in the hidden layer nodes more reasonable, as it was found that learning term in the hidden layer nodes may be enlarged hundreds of times if the ESP function is used directly as is done with the output nodes. The learning term for the hidden nodes including the ESP function is as follows:

$$\delta_j = \sum_k^N w_{jk}\ \delta_k\ o_j(1 - o_j) + \mathrm{cESP}\ (o_k) \qquad (10)$$

C is the small real factor (e.g. 0.01).

C. Using Weight Initialisation Methods

BP is sensitive to the initial weight values, choosing optimal values for the initial weights reduces the number of training iterations and the initial error is substantially reduced. A training period can be considerably reduced if the initial weights chosen are close to the true minimum. An inappropriate choice of initial weights is one of the main reasons of getting stuck in minima. Over years many weight initialisation techniques have been proposed.

Reference [10] proposed a weight initialisation technique for feed forward networks based on Cauchy's inequality and a linear algebraic method. This method ensured that the outputs of the hidden neurons are in the active region, i.e. the derivative of the activation function has a large value. From the output of the last hidden layer to the output layer, the optimal values of the weight are evaluated by using a least square method. When the optimal initial weights are determined, it takes less number of iterations to reach the prescribed tolerance.

Reference [11] used least square method for weight initialisation. For networks with one hidden layer, they initialised the weights between input and hidden layer by

using simulated annealing and genetic algorithm, the output layer weights were computed using singular value decomposition.

Reference [12] found that the weight initialisation process is to be done in such manner that all the hidden units are scattered uniformly in the input space which in turn will result in substantial improvement of the learning speed of network. This is done by distributing the initial weights and biases in such a manner that the region of interest is divided into small intervals. So that for each input pattern, it is likely that the net input to one of the hidden units will be in the range that the neuron will learn most quickly. Each hidden unit is assigned the task of approximating a portion of desired function at the start of training. The steps for this initialisation technique are:

1. For each hidden unit, initialise its weight vector (from the inputs):

w_{ij} (old)=random number between -0.5 to 0.5

2.Compute

$$\|w_j(old)\|=\sqrt{w_{ij}(old)^2 + w_{2j}(old)^2 + \cdots . + w_{nj}(old)^2}$$

Reinitialise weighs

$$w_{ij} = \frac{\beta w_{ij}(old)}{\|w_j(old)\|} \quad (11)$$

3. Set bias w_{oj} =random number between –β to +β

Where β =0.7(p)$^{1/n}$, n be the no. of input units, p be the no. of hidden units, β is the scale factor and w_{oj} is the bias for hidden neurons.

D. Adjusting the steepness (slope) of sigmoid function

Error surfaces for MLP's are generally quite severe. These surfaces consist of flat and extremely steep surfaces. When such a flat area with high error level is encountered, the outputs of neuron get temporarily trapped into high error level as the respective weights are adjusted by a small amount. Hence, no significant decrease in error occurs for some period of time, which then decreases gradually. One solution to this problem is to adjust the steepness of the sigmoid function i.e. adjusting the gain parameter of sigmoid function. Reference [13] found that local minima generally occur because of the error saturation in the hidden layer. They proposed a new algorithm to avoid this problem by adjusting the sigmoid activation functions in the hidden layer for each neuron so that the weights connecting the hidden layer and output layer are modified cordially. The sigmoid function is adjusted by varying the gain parameter of the hidden neurons. The modification of the gain parameter is done according to the degree of approximation of the desired output of the output layer. Usually, the activation function of a neuron with gain parameter is given by:

$$1/1 + e^{-c\,x}.$$

c is the gain of the activation function.

The parameter 'c' is updated by obtaining an estimate of the desired output of the output layer. This is done by introducing two parameters e and H. 'H' which denotes the average of the difference between teacher signals. It is a constant for a given learning task and $e_{=}max(|t_k - o_k|)$. The estimation of the desired output of the output layer A is given by A=$\frac{e}{H}$

And the gain parameter is updated as

$$c=\begin{cases} \frac{1}{A} & \text{if A>1.0} \\ 1.0 & else \end{cases} \quad (12)$$

Reference [14] proposed an improvement to the basic BP by adjusting the slope of the activation functions of the output layer nodes and using different learning rates for the hidden and output layer nodes. Reference [15] proposed an algorithm that modifies the gradient based search direction by adaptively varying the slope parameter (gain) of the sigmoid function. The gain updation rule for the output and hidden nodes is given as:

The gain updation formula for output nodes is:

$$\Delta c_k(\mathrm{n}+1)= \eta((t_k - o_k)o_k(1 - o_k)(\sum w_{jk}o_j + \theta_k \quad (13)$$

The gain updation formula for hidden nodes is:

$$\Delta c_j(\mathrm{n}+1) =\eta\left[-\sum_k c_k w_{jk} o_k(1 - o_k)(t_k - o_k)\right]o_j(1 - o_j)[(\sum_j w_{ij}o_i)] \quad (14)$$

Based on the updated gain parameter the weights connecting the output nodes are updated as

$$\Delta w_{jk}(\mathrm{n}+1)= \eta((t_k - o_k)o_k(1 - o_k)c_k o_j \quad (15)$$

The weights connecting the hidden nodes are updated as

$$\Delta w_{ij}(\mathrm{n}+1)= \eta\left[-\sum_k c_k w_{jk} o_k(1 - o_k)(t_k - o_k)\right]o_j(1 - o_j)c_j o_i \quad (16)$$

III. Experiments And Results

In order to assess the performance of improved versions of BP, we simulate them on several benchmark problems XOR, 3-BIT PARITY, MODIFIED XOR and IRIS classification problems. In our simulations we have used the online versions of standard BP(BP), BP algorithm with momentum(BP-M), BP with ESP function(BP-ESP) for output nodes based on [9], BP with ESP function for hidden as well as output nodes(BP-ESP-H)[9], BP with gain(BP-G) based on[13], BP with adaptive momentum(BP-AM) based on[5], BP with adaptive gain(BP-AG) based on [15] and Ngyuen-widrow weight initialisation technique(NG-W) based on[12]. For all the methods, the weights and biases were

initialised to random values which were uniformally distributed in the range of (-0.5, 0.5) and the gain (slope) parameter of the sigmoid function for all the neurons is set to 1.0. For the hidden and output neurons sigmoid activation function is used except for the NG-W technique where tan hyperbolic function is used for all the neurons. The performance measure that is used to compare different algorithms is set to MSE (mean square error). All the algorithms are performed in 30 independent trials, each starting from random initial weights. For trial the numbers of epochs required for the convergence are recorded, which are then accumulated over 30 runs and from them mean of the epochs **(#** of epochs required to converge) is collected. A training run is considered to be a success if it converges within the specified tolerance (MSE) of 0.001. The upper limit on the epochs is set to 10000 for all the algorithms except for BP, where the upper limit is set to 20000 as it takes longer time to converge.

A. XOR problem

In order to train XOR problem we have used a neural network with 2 input, 2 hidden and 1 output node (2-2-1). The learning rate for BP, BP-M, BP-ESP, BP-ESP-H, NG-W, BP-G, BP-AG and BP-AM is set to 0.5. The momentum coefficient for BP-M and BP-AM is set to 0.2. For the algorithms with ESP functions we select α=16 and n=4 i.e. the ESP function for both the algorithms is $16(o_i - 0.5)^4$. The values for the various learning parameters are chosen on the basis of hit and trial method, there was no particular reason for using the same. The simulation results on XOR problem are given in the Table 1.

Table 1. Performance Comparison of various algorithms on XOR problem**.**

Categories	Algorithms	# of epochs required to converge
Standard BP	BP	7588
Based on Momentum	BP-M	5971
	BP-AM	4996
Error saturation prevention	BP-ESP	2220
	BP-ESP-H	2256
Weight initialisation	NG-W	1127
Based on Gain	BP-G	5805
	BP-AG	6885

B. 3-bit Parity problem

In this problem, the output required is 1 if the sum of the input patterns yields an odd number and 0 otherwise. The selected architecture for the 3 bit parity problem is 3-4-1(3- input neurons, 4-hidden neurons and 1-output neurons). The learning rate for BP, BP-M, BP-ESP, BP-ESP-H, NG-W, BP-G, BP-AG and BP-AM is set to 0.5. The momentum coefficient for BP-M and BP-AM is set to 0.2. Besides we select α=16 and n=4 for BP-ESP and BP-ESP-H i.e. the ESP function for both the algorithms is $16(o_i - 0.5)^4$. The performance comparison of different algorithms on 3-bit parity problems is given in the Table 2.

Table 2. Performance Comparison of various algorithms on 3 bit parity problem

Categories	Algorithms	# of epochs required to converge
Standard BP	BP	13080
Based on Momentum	BP-M	11442
	BP-AM	10519
Error saturation prevention	BP-ESP	5615
	BP-ESP-H	5082
Weight initialisation	NG-W	-
Based on Gain	BP-G	9389
	BP-AG	7906

C. The modified XOR problem

It's a classical XOR problem but with one or more introduced patterns such that a unique global minimum [16] and several local minima exist simultaneously [17]. The truth table for the modified XOR problem is given in the Table 3. We have used 2×2×1 architecture to solve this problem (2input neurons, 2 hidden neurons and 1 output neuron). Besides we select α=16 and n=4 for BP-ESP and BP-ESP-H i.e. the ESP function for both the algorithms is $16(o_i - 0.5)^4$ .The learning rate for BP, BP-M, BP-ESP, BP-ESP-H, NG-W, BP-G ,BP-AG and BP-AM is set to 0.5. The momentum coefficient for BP-M and BP-AM is set to 0.2. The performance comparison of various algorithms on modified XOR problem is given in the Table 4.

Table 3. Truth Table for Modified XOR problem

Pattern no	Feature1	Feature2	Desired output
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0
5	0.5	0.5	1

Table 4. Performance Comparison of various algorithms on modified XOR problem

Categories	Algorithms	# of epochs required to converge
Standard BP	BP	7742
Based on Momentum	BP-M	6621
	BP-AM	5100
Error saturation prevention	BP-ESP	2718
	BP-ESP-H	2269
Weight initialisation	NG-W	1143
Based on Gain	BP-G	5960
	BP-AG	2389

D. IRIS classification

Fisher Iris data is the fourth training database that is used for the purpose of comparison. This dataset consists of 150 samples of Iris flowers (Iris Setosa, Iris Virginica, and Iris Versicolor). Each species, in the dataset is represented by four attributes: length of sepal, width of sepal, length of petal and width of petal. In order to develop a network to solve this problem, we have used 4×5×3 neural network (4 neurons in the input layer, 5 in the hidden layer and 3 in the output layer as there are 3 different classes).we have used 75 instances as a training set and the rest are used as testing set to check the generalisation ability of the trained network. For the iris classification problem the learning rate for BP, BP-M, BP-ESP, BP-ESP-H, NG-W, BP-G, BP-AG and BP-AM is set to 0.9. The momentum coefficient for BP-M and BP-AM is set to 0.2. For BP-ESP and BP-ESP-H we set α=4 and n=2 i.e. the ESP function for both the algorithms is$4(o_i - 0.5)^2$. The performance of various algorithms for the iris classification is given in the Table 5:

Table 5. Performance Comparison of various algorithms on IRIS classification problem

Categories	Algorithms	# of epochs required to converge	Accuracy (%)
Standard BP	BP	9715	93.3
Based on Momentum	BP-M	8472	94.5
	BP-AM	8109	95.1
Error saturation prevention	BP-ESP	720	96.4
	BP-ESP-H	492	96.8
Weight initialisation	NG-W	-	-
Based on Gain	BP-G	7210	96.3
	BP-AG	210	97.3

The results clearly demonstrate that all the improved versions perform better than the standard BP with better acceleration speed and generalisation ability. However the performance of these algorithms depends on the data set used. In case of XOR and modified XOR, NG-W weight initialisation technique gives better results. However, when the same technique is used in 3 bit parity and Iris classification problem it shows divergence. Although BP-G and BP-AM shows better results for small data set problems and perform better than the standard BP and BP-M but when these are used with other data sets of higher dimensionality they show slow convergence speed. BP-ESP, BP-ESP-H and BP-AG outperform all the other improved versions. They show good convergence speed as well as the generalisation ability.

CONCLUSION

In this paper we have compared the performance of standard BP with its improved versions in order to investigate the potentials of these algorithms. The improved versions are simulated on modified XOR, XOR, Parity and iris classification problem. The training performance of these algorithms is evaluated in terms of percentage of accuracy and convergence speed. The comparative analysis done on the various algorithms proves that most of the algorithms are superior to the standard BP ,with error saturation prevention function and the adaptive gain methods showing better performance than the rest. Further research can be done in this field by simulating these algorithms on other data sets.

REFERENCES

[1] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition (D. Rumelhart and J. McClelland, editors*), pp 318-362, 1986.

[2] D.E. Rumelhart, G.E. Hinton and R.J.Williams,"Learning representations by back-propagating errors",*Nature*,vol 323,pp 533-536,1986.

[3] D.J. Swanston, J.M. Bishop and R.J. Mitchell, "Simple adaptive momentum, new algorithm for training multilayer perceptrons," *Electronics Letters*, 30(18), pp 1498 -1500, 1994.

[4] C. Yu and B. Liu, "A backpropagation algorithm with adaptive learning rate and momentum coefficient," *Proc. Int. Conf. on Neural Networks (IJCNN'02*), vol 2, pp 1218-1223, 2002.

[5] H.M. Shao, G.F. Zheng, "A new BP algorithm with adaptive momentum for FNNs training," *Proc. WRI Global Congress on Intelligent Systems (GCIS'09),* vol. 4, pp. 16–20, 2009.

[6] S.H. Oh, "Improving the error back-propagation algorithm with a modified error functions*," IEEE Trans. Neural Networks* 8 (3), pp 799-803, 1997.

[7] S.C. Ng, S.H. Leung, A. Luk, "Fast and global convergent weight evolution algorithm based on the modified back-propagation,"*IEEE International Conference on Neural Networks Proceedings*,pp. 3004-3008, 1995.

[8] A.V. Ooyen, B. Nienhuis, "Improving the learning convergence of the back propagation algorithm," *Neural Networks*, vol 5, pp 465-471, 1992.

[9] H.M.Lee, C.M.Chen, T.C.Huang, "Learning efficiency improvement of back-propagation algorithm by error saturation prevention method,"*Neurocomputing*, vol 41, pp. 125-143, 2001.

[10] Yam, J.Y. and Chow, T.W., "A Weight initialization method for improving training speed in Feedforward neural network," *Neurocomputing*, Vol. 30, pp. 219-232, 2000.

[11] T. Masters, Practical Neural Network Recipes in C + + (Academic Press, Boston, 1993).

[12] Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," *Proc. Internat. Joint Conf on Neural Networks*, San Diego Vol. 3, pp 21-26, 1990.

[13] X.G. Wang, Z. Tang, H. Tamura, M. Ishii, W.D. Sun, "An improved backpropagation algorithm to avoid the local minima problem,"*Neurocomputing*,vol 56,pp 455 – 460, 2004.

[14] Y. Bai, H. Zhang, Y.Hao, "The performance of the back propagation algorithm with varying slope of the activation function*," Chaos, Solitons and Fractals*, vol 40, pp 69–77, 2009.

[15] N. M. Nawi, R. S. Ransing and M. R. Ransing, "A new method to improve the gradient based search direction to enhance the computational efficiency of back propagation based Neural Network algorithms," *Proc.IEEE Second Asia International Conference on Modelling & Simulation*, pp.546-551 DOI 10.1109/AMS.2008.70,2008.
[16] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Trans. Pattern Anal. Mach.Intell.* 14 (1) pp 76–86, 1992.
[17] H. Ishibuchi, R. Fujioka, H. Tanaka, Neural networks that learn from fuzzy if-then rules, *IEEE Trans. Fuzzy Syst.* 1 (2), pp 85-97,1993.
[18] Saduf, M. Arif Wani, "Comparative study of adaptive learning rate with momentum and resilient back propagation algorithms for neural net classifier optimization," *International Journal of Distributed and Cloud Computing*, vol 2, pp. 1-6, 2014.

Short bio data of authors

Saduf has done MCA from university of Kashmir, India. She is currently pursuing research in the field of neural networks.

Mohd.Arif Wani he is currently working as a professor in the department of computer sciences, University of Kashmir, India. He has more than 50 publications in the journals of international repute. His area of interest are Data mining and machine learning.