World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:4, No:12, 2010

Improved Back Propagation Algorithm to Avoid Local Minima in Multiplicative Neuron Model

Kavita Burse, Manish Manoria, Vishnu P. S. Kirar

***Abstract*—**The back propagation algorithm calculates the weight changes of artificial neural networks, and a common approach is to use a training algorithm consisting of a learning rate and a momentum factor. The major drawbacks of above learning algorithm are the problems of local minima and slow convergence speeds. The addition of an extra term, called a proportional factor reduces the convergence of the back propagation algorithm. We have applied the three term back propagation to multiplicative neural network learning. The algorithm is tested on XOR and parity problem and compared with the standard back propagation training algorithm.

***Keywords*—**Three term back propagation, multiplicative neural network, proportional factor, local minima.

I. Introduction

ARTIFICIAL Neural Network (ANN) consists of a number of interconnected processors known as neurons, which are identical to the biological neural cells of the human brain. Neural network is defined by its architecture, neuron model and the learning algorithm. Architecture refers to a set of neurons and the weighted links connecting the layers of neurons. Neuron model refers to information processing unit of the neural network. The weights are adjusted during the training process. A learning algorithm is used to train the NN by modifying the weights in order to model a particular learning task correctly on the training examples. Learning is a fundamental and essential characteristic of ANN. ANN training usually updates the weights iteratively using the negative gradient of a Mean Squared Error (MSE) function. The error signal is then back propagated to the lower layers.

The back propagation (BP) algorithm was developed by Rumelhart, Hinton and Williams in 1986. Efficient learning by the BP algorithm is required for many practical applications. The BP algorithm calculates the weight changes using a two-term algorithm consisting of a learning rate and a momentum factor. The major drawbacks of the two-term BP learning algorithm are the problems of local minima and slow convergence speeds. The addition of an extra term, called a proportional factor (PF), to the two-term BP algorithm was proposed in 2003 by Zweiri and has outperformed standard two- term BP in terms of low complexity and computational cost [1]. BP is a method for calculating the first derivatives, or gradient, of the cost function required by some optimization methods. It is certainly not the only method for estimating the gradient. However, it is the most efficient [2]. The major limitations of this algorithm are the existence of temporary, local minima resulting from the saturation behavior of the activation function. A number of approaches have been implemented to avoid the local minima which are based on selection of dynamic variation of learning rate and momentum, selection of better activation function and better cost function.

In [3] the learning rate and momentum coefficient are adapted according to the coefficient of correlation between the downhill gradient and the previous weight update. In [4] modification is based on the solving of weight matrix for the output layer using theory of equations and least squares techniques. Drago *et al.* have proposed an adaptive momentum BP for fast minimum search [5]. A randomized BP algorithm is proposed by Chen *et al.* It is obtained by choosing a sequence of weighting vectors over the learning phase [6]. A new generalized BP algorithm is proposed in [6] to change the derivative of the activation function so as to magnify the backward propagated error signal, thus the convergence rate can be accelerated and the local minimum can be escaped. An adaptive BP algorithm is proposed in [7] which can update learning rate and inertia factor automatically based on dynamical training error rate of change. In [8] an improved BP is proposed where each training pattern has its own activation function of neurons in hidden layer to avoid local minima. In [9] new learning procedure for training single hidden layer feedforward network is proposed where the output layer and the hidden layer is trained separately. Wang *et al.* have proposed an individual inference adjusting learning rate technique to enhance the learning performance of the BP neural network [10].

This paper is organized as follows. In the next section we propose the learning rule with improved BP algorithm to avoid local minima in multiplicative neuron model. In section III, we illustrate the basic results of our paper. Section IV concludes the paper.

II. Learning with Improved BP Algorithm

The McCulloch-Pitts model initiated the use of summing units as the neuron model, while neglecting all possible

Kavita Burse is with Department of Electrical and Electronics, Truba Institute of Engineering and Information Technology, Bhopal, India. (phone: 9893141968, e-mail: kavitaburse14@gmail.com).

Dr Manish Manoria is Director, Truba Institute of Engineering and Information Technology, Bhopal, India (e-mail: manishmanoria@rediffmail.com).

Vishnu P S Kirar is a pass out student Department of Electronics and Communication, Truba Institute of Engineering and Information Technology, Bhopal, India. (e-mail: vishnupskirar@live.com).

nonlinear capabilities of the single neuron and the role of dendrites in information processing in the neural system. It is widely agreed that there is only minor correspondence between these neuron models and the behavior of real biological neurons. In particular, the interaction of synaptic inputs is known to be essentially nonlinear. In search for biologically closer models of neural interactions, neurobiologists have found that multiplicative-like operations play an important role in single neuron computations. For example, multiplication models nonlinearities of dendritic processing and shows how complex behaviour can emerge in simple networks. In recent years evidence has accumulated regarding specific neurons in the nervous system of several animals compute in a multiplicative manner. Multiplication increases the computational power and storage capacity of neural networks is well known from extensions of ANN where this operation occurs as higher order units [11].

The back-propagation learning algorithm with multiplicative neural networks (MNN) has to be more efficient as both single-units and also in networks. The goal of learning is to update the network weights iteratively to minimize globally the difference between the actual output vector of the network and the desired output vector. The rapid computation of such a global minimum is a rather difficult task since, in general, the number of network variables is large and the corresponding non convex multimodal objective function possesses multitudes of local minima and has broad flat regions adjoined with narrow steep ones. The architecture of the MNN is described as follows. The basic building block of the MNN is a single neuron or node as depicted in Fig. 1[12].

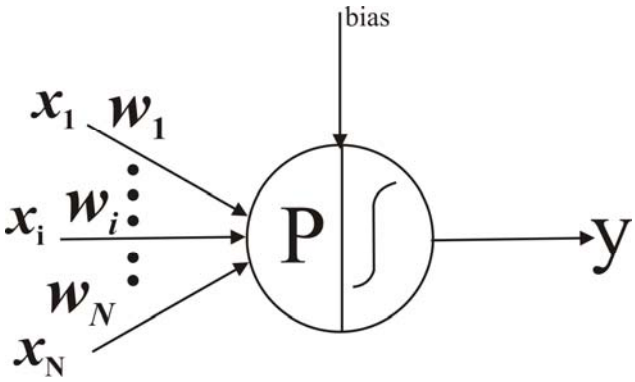

Fig. 1 Node structure of MNN

A node receives a number of real inputs $x_1, x_2 \ldots x_n$ which are then multiplied by a set of weights $w_1, w_2 \ldots w_n$ and bias terms $b_1, b_2 \ldots b_n$ are added. The resultant values are multiplied to form a polynomial structure. This output of the node is further subjected to a nonlinear function f defined as

$$f = \frac{1-e^{-x}}{1+e^{-x}}$$

In the MNN a number of nodes described above are arranged in layers. A multidimensional input is passed to each node of the first layer. The outputs of the first layer nodes then become inputs to the nodes in the second layer and so on. The output of the network is the output of the nodes of the final layer. Weighted connections exist from a node to every node in the succeeding node but no connections exist between nodes of the same layer. As the number of layers in the MNN is increased decision regions are formed which are considerably more complex and have highly nonlinear boundaries. Fig. 2 shows a general model of a feed forward MNN where the summation at each node is replaced by the product unit.

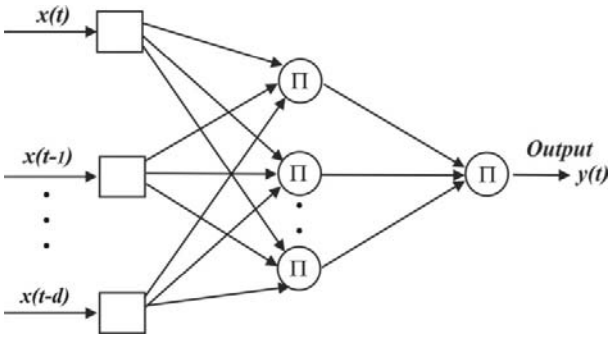

Fig. 2 Architecture of feed forward MNN

The MNN is trained using supervised learning where a set of input and the corresponding target vector is used to adjust the scalar parameters weight and bias. The network is trained incrementally so that the weights and biases are updated after each input is presented. The MNN is trained using supervised learning where a set of input and the corresponding target vector is used to adjust the scalar parameters weight and bias. The network is trained incrementally so that the weights and biases are updated after each input is presented. As the inputs are applied to the network, the network outputs are compared to the targets. The learning rule is then used to adjust the weights and biases of the network in order to move the network outputs closer to the targets.

The output of the node *u* before applying activation function is given by [13]-[14]:

$$u = \prod_{i=1}^{n} (w_i x_i + b_i) \tag{1}$$

The bipolar sigmoidal activation function *f* is given by

$$y = f(u) = \frac{1-e^{-u}}{1+e^{-u}} \tag{2}$$

An error back propagation based learning rule is used for training. The MSE is given by

$$E = \frac{1}{2N} \sum_{p=1}^{N} (y^p - y_d^p)^2 \tag{3}$$

Where, p is the number of input patterns. The weight update equation for single layer algorithm is given by

$$\Delta w_i = -\eta \frac{dE}{dw_i}$$
$$= -\frac{1}{2}\eta(y-d)(1+y)(1-y)\frac{u}{(w_i x_i + b_i)}x_i \quad (4)$$

Where, η is the learning rate and d is the desired signal. If η is large, learning occurs quickly, but if it is too large it may lead to instability and errors may even increase.

The bias is updated as

$$\Delta b_i = -\eta \frac{dE}{db_i}$$
$$= -\frac{1}{2}\eta(y-d)(1+y)(1-y)\frac{u}{(w_i x_i + b_i)} \quad (5)$$

The standard BP algorithm calculates the new weights and biases as

$$w_i^{new} = w_i^{old} + \Delta w_i \quad (6)$$

$$b_i^{new} = b_i^{old} + \Delta b_i \quad (7)$$

The standard algorithm is further modified by adding the momentum term and PF term. The momentum term is a fraction of the previous weight change. The momentum term prevents extreme changes in the gradient due to anomalies and suppresses oscillations due to variations in the slope of the error surface [15] and prevents the network to fall into shallow local minima. The convergence still remains relatively slow because of the saturation behavior of the activation function. In the saturation area of the output activation function, the corresponding gradient descent takes very small value leading to small changes in weight adjustments. The problem of slow convergence is solved by adding a term proportional to the difference between the output and the target. The improved BP weight update is calculated as

$$\Delta w_i^{improved} = \Delta w_i + \beta \Delta w_i^{old} + \gamma(y-d) \quad (8)$$

$$\Delta b_i^{improved} = \Delta b_i + \beta \Delta b_i^{old} + \gamma(y-d) \quad (9)$$

β is the proportional term

Δw_i^{old} is the previous weight change

γ is the proportional term

$(y-d)$ is the difference between the output and the target at each iteration

Δb_i^{old} is the previous bias change

III. Simulation And Results

We have tested the convergence of the three term BP algorithm for the MNN network on the XOR problem which is the most used nonlinear pattern classification problem as compared to other logic operations. The architecture of the network is a single layer MNN with 2 inputs, 3 hidden layer neurons and 1 output neuron. The convergence curve for the three term BP algorithm is compared with the standard BP algorithm in Fig. 3. The convergence of the improved BP algorithm with momentum and PF factor is five times faster as compared to the standard BP algorithm. Table 1 compares the testing performance for the XOR problem.

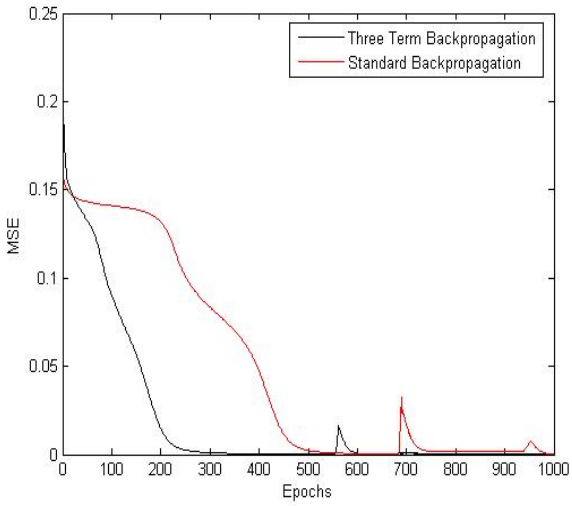

Fig. 3 Convergence curves for XOR problem

TABLE I TESTING PERFORMANCE FOR XOR PROBLEM

Input	Target	Output with MNN trained with standard BP algorithm	Output with MNN trained with three term BP algorithm
0 0	0	0.0004	0.0001
0 1	1	0.9841	0.9942
1 0	1	0.9835	0.9997
1 1	0	0.0002	0.0001

TABLE II TESTING PERFORMANCE FOR 3 BIT PARITY PROBLEM

Input	Target	Output with MNN trained with standard BP algorithm	Output with MNN trained with three term BP algorithm
0 0 0	0	0.0312	0.0032
0 0 1	1	0.8921	0.9978
0 1 0	1	0.9876	0.9886
0 1 1	0	0.0214	0.0021
1 0 0	1	0.8953	0.9778
1 0 1	0	0.0021	0.0041
1 1 0	0	0.0032	0.0034
1 1 1	1	0.9873	0.9921

We have further tested the three term BP algorithm for the MNN network on the three bit parity problem which maps 3 bit binary numbers onto its parity. The parity bit output is 1 for odd number of 1 else it is 0. Table 2 compares the testing performance for parity problem.

IV. Conclusion

In this paper we have proposed an improved BP algorithm to avoid local minima and for faster convergence of multiplicative neural network training algorithm. We have tested the algorithm for XOR and three bit parity problem and compared the result with standard BP multiplicative neural network algorithm. The addition of PF term helps in convergence of the algorithm five times faster.

References

[1] Y. H. Zweiri, J. F. Whidborne, K Althoefer and L. D. Seneviratne, "A three term back propagation algorithm," *Neurocomputing*, vol. 50, pp. 305-318, 2003.

[2] R. J. Edward: An Introduction to Neural Networks. A White paper. United States of America, Visual Numerics Inc. 2004.

[3] Y. F. Yam and T.W.S. Chow, "Extended back propagation algorithm," *Electronics Letters*, vol. 29(19), pp. 1701-1702, 1993.

[4] B. K. Verma and J. J. Mulawka, "A modified back propagation algorithm," in *Proc. IEEE World Congress on Computational Intelligence*, pp. 840-844, 1994.

[5] G. P. Drago, M. Morando and S. Ridella," An adaptive momentum back propagation, *Neural Computing and Application*, vol. 3, pp. 213-221, 1995.

[6] Y. Q. Chen, T. Yin and H. A. Babri, "A stochastic back propagation algorithm for training neural networks," in proc. *International Conference on Information, Communications and Signal Processing*, pp. 703-707, 9-12 September, 1997, Singapore.

[7] S. C. Ng, S.H. Leung and A. Luk, "Fast convergent generalized back propagation algorithm with constant learning rate", *Neural Processing Letters*, vol. 9, pp. 13-23, 1999.

[8] J. W. Wen, J. L. Zhao, S. W. Luo and Z. Han, "The improvements of BP neural network learning algorithm," in *Proc. of ICSP2000*, pp. 1647-1649, 2000.

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

[10] C. H. Wang, C.H. Kao, and W. H. Lee, "A new interactive model for improving the learning performance of back propagation neural network," *Automation in Construction*, vol. 16, no. 6, pp.745-758, 2007.

[11] B. Mel, "Information processing in dendritic trees," *Neural Computing*, vol.6, pp.1031-1085, 1994.

[12] B. Mel, "Information processing in dendritic trees," *Neural Computing*, vol.6, pp.1031-1085, 1994.

[13] R. N. Yadav, P. K. Kalra and J. John, "Time series prediction with single multiplicative neuron model," *Applied soft computing*, vol. 7 pp 1157-1163, 2007.

[14] R. N. Yadav, V. Singh and P. K. Kalra, "Classification using single neuron," in *Proc. IEEE Int. Conf. on Industrial Informatics*, pp. 124-129, 21-24 August, 2003, Banff, Alberta, Canada.

[15] C-C Yu, and B-D Liu, "A back propagation algorithm with adaptive learning rate and momentum coefficient," in *Proc. of the International Joint Conference on Neural Networks, IJCNN 2002*, pp.1218-1223, May 2007.