Partial and Random Updating Weights in Error Back Propagation Algorithm

Nasim Latifi, Ali Amiri
Computer Engineering Group
University of Zanjan
Zanjan, Iran
Nsm.latifi@gmail.com, A_amiri@znu.ac.ir

ABSTRACT: *Multi-Layered Perceptron (MLP) is a useful supervised neural network for data classification. Error Back Propagation (EBP) algorithm is the common technique for training MLP. Standard EBP algorithm has challenges for large-scale and heterogeneous data such as lack of memory and low–speed convergence, besides, computational load is high. In this paper, to overcome these drawbacks, a modified version of EBP has been proposed. It decreases the time and space complexity, and somewhat increases convergence speed of the standard EPB by partial and random updating of some of weights instead of all of them. Result of experiments on two standard dataset, confirms the effectiveness of the proposed algorithm.*

Keywords: MLP, EBP, Neural Networks, Algorithm

Received: 12 October 2011, Revised 2 December 2011, Accepted 7 December 2011

1. Introduction

MLP Neural Network (MLP NN) is one of the most important machine learning tools that is used in the various fields such as, solving complex classification problems, function approximation and time series prediction. MLP NN is a feed forward multilayer network with non-linear activation with supervised learning algorithm. Usually, it uses gradient-based algorithms and specially EBP algorithm to train network.

EBP algorithm or generalized delta rule is a gradient descent method that minimizes the squared output error of the network [3]. Generally, the learning process in the EBP algorithm describes as a step-by-step and iterative weight correction, in the negative gradient of Mean Squared Errors (MSE) function [1]. This error is calculated as the squared difference between the desired value and the output value from the last layer. This error is back propagated to the neurons of the previous layers [1]. Some parameters have high effect in performance of MLP NN including: initializations, learning rates selection, topology of the network, number of hidden layers, number of neurons, amount of training data, convergence speed, type of activation function, etc [1].

When we use MLP NN in some of scopes such as video data classification with so high data volumes and dimensions and heterogeneous training samples, there are two principle problems: 1) low convergence speed 2) high network size and so lack of memory to maintain and update the network weights. Therefore, time/space complexity of algorithm may be so high and sometimes implementation of network is impossible [2].

To overcome the mentioned challenges and specially to decrease the number of computational operations, the amount of consumption memory and the number of weights that evaluate in each step, in this paper the random selection and updating some of weights instead of all them is proposed. The experimental results on two standard datasets demonstrate that the proposed algorithm achieves high convergence speed and low space complexity in comparing to the standard EBP algorithm.

The reminder of this paper is organized as follows. In section 2, a brief description of MLP NN architecture and standard EBP algorithm is given. Section 3 presents the proposed modified EBP algorithm. Section 4 illustrates empirical evaluation of our solutions and implementations using two standard test beds. Section 5 presents our conclusions.

2. MLPNN and EBP algorithm

Figure. 1 shows a MLP NN with 3 layers (an input layer, a hidden layer and an output layer); each layer includes a set of neurons which the input of a neuron is being from external sources or outputs of neurons of previous layer. In this network each neurons of an arbitrary layer are connected to neurons of the next layer through adjustable weighted links. MLP NN utilizes the EBP algorithm for training network. The EBP learning algorithm adjusts the weights of network by back propagating the computed error from output to previous layers, as, by iterating this operation decrease error gradually.

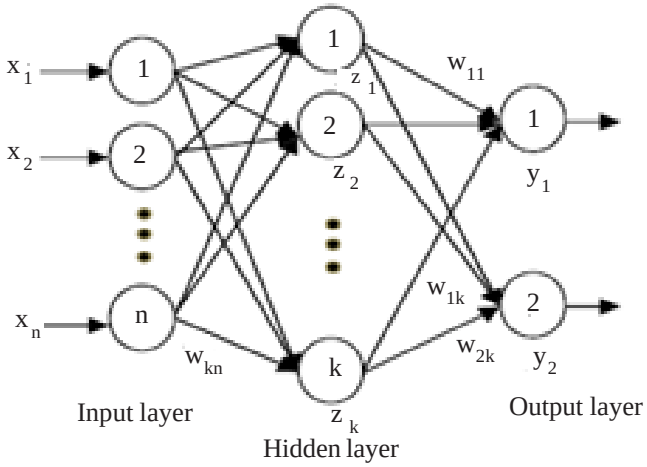


Figure 1. A three layer MLP NN

Let $\{X^i, D_i\}_{i=1}^{h}$ be the training set of the network where is $X^i = [x_{i1}, x_{i2}, ..., x_{in}]^T$ the i -th input pattern and d_i is its desired output. In the EBP algorithm, for output layer neurons, the error is calculated as [1], [2]:

$$e^i_c(n) = D_i(n) + Y_c(n) \quad (1)$$

Where Y_c is output of neuron *c* in output layer. The weights are initialized randomly and will be adjusted during the learning step. The EBP algorithm uses the following equation for adjusting the weights between neurons of output layer (Y_c, c =1, …, m) and neurons of hidden layer (Z_b, b=1, …, k) at *n* -th iteration:

$$W_{bc}(n+1) = W_{bc}(n) + Y_b(n)\,\delta_c(n) \quad (2)$$

Where μ is the learning rate and δ is the back propagated error that it is computed as follows:
For output layer:

$$\delta_c(n) = e_c(n)\, f'(Y_c(n)) \quad (3)$$

For hidden layer:

$$\delta_b(n) = f'(Y_b(n)) \sum_{c=1}^{m} \delta_c(n)\, W_{bc}(n) \quad (4)$$

Where *m* is the number of neurons in the output layer. If MLP NN includes more than one hidden layer, therefore in the next step this equation should be compute for other hidden layer.

Whereas in the EBP algorithm with momentum parameter, variation of weight is combining of present gradient and previous gradient, in fact, this method is a sample of a change in decrease gradient method. Sometimes by adding momentum parameter to updating weight equation, convergence is been faster and it allows to the network to perform the weights adjustment by large step when variation of weights be in same direct for some training pattern, at the same time, network able to use smaller learning rate until don't show large reaction to probable errors in training patterns. So we have

$$W_{bc}(n+1) = W_{bc}(n) + \mu Y_b(n)\,\delta_c(n) + \eta[\,W_{bc}(n) - W_{bc}(n-1)] \tag{5}$$

Or

$$\Delta W_{bc}(n+1) = \mu Y_b(n)\,\delta_c(n) + \eta\,\Delta W_{bc}(n) \tag{6}$$

Where η is momentum parameter and $0 < \eta < 1$.

In this paper our means of standard EBP are the EBP algorithm with momentum parameter.

3. Proposed Method

In standard EBP algorithm, when learning process is applied, all of the weights in the MLP NN are updated at each iteration cycle. It is evident that for large-scale and heterogeneous data sets, the convergence speed of the learning algorithm is low, besides, computational load rate, the space complexity and the number of weights that are evaluated in each step is high. We know that one of the most significant parameter in evaluation performance and improvement artificial neural networks is convergence speed that in this filed different research is done [1], [3], [6], [7], etc. But, despite to the successful researches, these problems are not solved efficiently. To overcome these challenges, in this paper, we propose a modified version of EBP algorithm that only selects and updates some of weights instead of all them in each iteration cycle of algorithm.

We assume that MLP NN has K layers and the number of nodes in i^{th} layer is:

$$N_i, \text{ for } i = 1, ..., K \tag{7}$$

And M_i, the number of weights between i^{th} and $i + 1^{th}$ layers is calculated as:

$$M_i = N_i \times N_{i+1} \text{ for } i = 1,...K-1 \tag{8}$$

Let R be a finite set of integers. We define a parameter as $S \in R$ where in each weight adjustment step of algorithm, the number of weights that are randomly selected and updated in each layer is:

$$P_i = \frac{1}{S} \times M_i \tag{9}$$

So, in n^{th} iteration of algorithm the number of selected weights is

$$P(n) = \sum_{j=1}^{K-1} p_i(n) \tag{10}$$

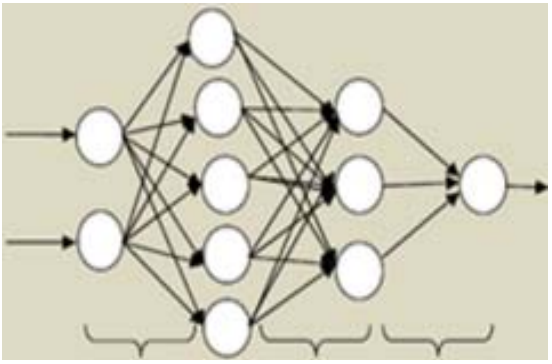
Figure. 2 shows a sample of proposed method performance.

The following pseudo code indicates the details of this process using MATLAB:

```
For k = l - 1 :- 1 : 1
        M = num of nodes (k)*number of nodes (k + 1);
        P =ceil (1/S * M);
    For f =1 : P
       m = randint (1, 1, [1, number of nodes (k)]);
       n = randint (1, 1, [1, number of nodes (k + 1)]);
    Δw (m, n, k) =μ*δ (n, k + 1)*Y(m,k) + η * Δw (m, n, k);
     W(m, n, k) = W(m, n, k) + Δw (m, n, k);
```

End
End

In this paper we assume $\mu = 0.04$ and $\eta = 0.4$.



$M_1 = 10$ $M_2 = 15$ $M_3 = 3$

$P_1 = \frac{1}{S} M_1$ $P_2 = \frac{1}{S} M_2$ $P_3 = \frac{1}{S} M_3$

Figure 2. A sample of proposed method performance

4. Experimental Results

The experiments are performed over Vehicle and Iris datasets [10]. the features and assumptions related to datasets and structures of MLP NN used for every data set are explained as below:

- Vehicle dataset: this dataset including about 250 samples of 2 classes. Each sample has 18 attributes. We use 4-layers MLP NN, the input layer has 18 neurons and there are 2 hidden layers, first layer including 10 neurons and second layer 8 neurons. We assume dataset is noisy with SNR= 10 db.
- Iris dataset: this dataset including about 150 samples and 1 class. Each sample has 4 attributes. We have 4-layers MLP NN, the input layer has 4 neurons and there are 2 hidden layers, both of them including 5 neurons.

Table 1 shows that the different values of S have evident influence in the rate of performed computations, also the convergence speed have the same efficiency.

Algorithm	Number of Iteration	Rate of Calculation
PRU-EBP with S=1	1422	98049000
PRU-EBP with S=2	1826	62962500
PRU-EBP with S=3	3405	79143000
PRU-EBP with S=4	2258	38933250
PRU-EBP with S=5	3765	52696000

Table 1: Rate of calculations and convergence speed of PRU-EBP algorithms over Vehicle dataset (10-8)

The results in figure. 3 demonstrate that different values of *S* have effect to convergence speed of algorithm.

The result of experiments on Iris dataset are in Table 2, as shown in all cases (S = 2, 3, 4, 5) rate of calculations are decreased except in S = 1. In S = 2, 5 convergence speed is increased in addition to rate of calculations decreased. As for Table 2, it is better to select a PRU-EBP algorithm increase convergence speed as is possible and too decrease computational load. Of course we can prefer one to another among the two efficiency parameters, depending to algorithms application in various fields. For example in large-scale data classification as video data, it is suitable to select an algorithm which have both high convergence speed and low memory usage rate.

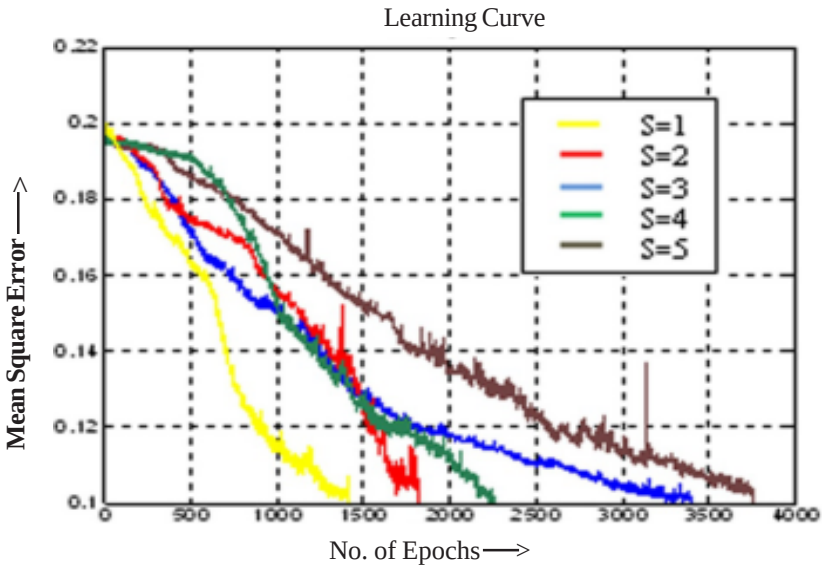


Figure 3. Convergence comparison for *PRU-EBP* algorithms over *Vehicle dataset (10-8), Emax=0.1*

Algorithm	Number of Iteration	Rate of Calculation
Standard-EBP	415	3112500
PRU-EBP with S=1	525	3937500
PRU-EBP with S=2	247	963300
PRU-EBP with S=3	980	2646000
PRU-EBP with S=4	892	1873200
PRU-EBP with S=5	277	415500
SPU-EBP	947	1775082

Table 2. Rate of calculations and convergence speed of *PRU-EBP* algorithms, *Standard-EBP* and *SPU-EBP* algorithms over *Iris dataset(5 - 5)*

In Figure. 4 and Table 2, we compare convergence speed and computational load of three algorithms; standard-EBP, SPU-EBP algorithms and PRU-EBP algorithm. In [2], SPU-EBP algorithm using the sequential and partial updating of weights. The results present our proposed algorithm has appropriate performance in similar condition and the result is acceptable.

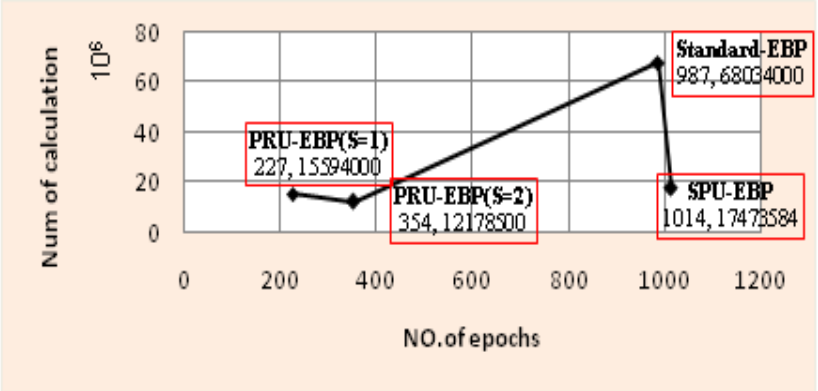


Figure 4. Comparison of 3 algorithms *(Standard-EBP, SPU-EBP and PRU-EBP for S=1, 2)* and *SNR=10db, Emax=0.18*

The gained result from Table 2 and also Figure.4 show that the convergence speed and computational load of PRU-EBP algorithm is faster than standard - EBP and SPU_EBP algorithms. And also the results of Table 1 and 2 describe that type of data and structure of neural network has direct effect on convergence speed and computational load of algorithms. In Table 1 we saw that had least rate of calculations for S = 2, 4 while in Table 2 we have least rate of calculations to S = 2, 5.

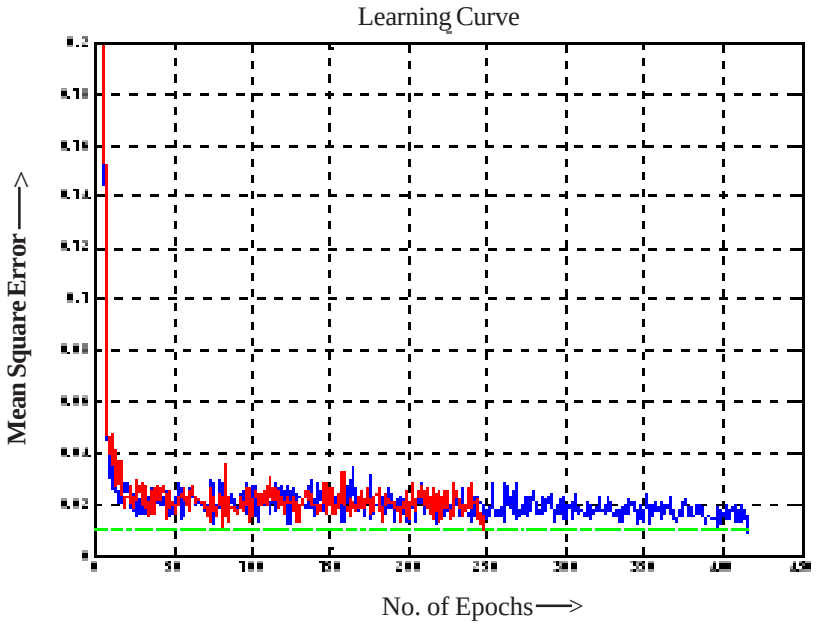


Figure 5. Convergence comparison of two algorithms *(Standard-EBP, PRU-EBP S=2), Iris (5-5)*

5. Conclusion

In this paper we are evaluated partial and random updating weights in error back propagation algorithm. The results of experiment have shown that the proposed algorithm have acceptable performance in compared to other two algorithms. We can use the proposed algorithm for working on large-scale data because the calculations rate and memory consumption have decreased. In the compared of standard-EBP although the convergence speed in some cases is improved but it is rather low and is not more significant. Our next target is significantly increasing the convergence speed of the proposed algorithm by using adaptive filtering.

References

[1] Didandeh, A., Mirbakhsh, N., Amiri, A., Fathy, M. (2011). AVLR-EBP, A Variable Step Size Approach to Speed-up the Convergence of Error Back-Propagation Algorithm. *Neural Processing Letters*, 3 (2) 201-214.
[2] Rahmani, N. M., Amiri, A., Fathy, M. (2010). Partial Update Error Back Propagation Algorithm, *In*: Proceedings of the 15th Iranian Conference on Electrical Engineering, Tehran, Iran.
[3] Sarkar, D. (1995). Methods to Speed Up Error Back-Propagation Learning Algorithm. *ACM Computing Surveys*, 27 (4) 545-644.
[4] Wilamowski, M. B. (2010). Advanced Learning Algorithms. IEEE.
[5] Rojas, R. (1996). The Back propagation Algorithm. Neural Networks, Springer-Verlag, Berlin.
[6] Abid, S., Faniech, F., Jervis, B.W., Cheriet, M. (2005). Fast training of multilayer perceptrons with a mixed norm algorithm, *In:* The IEEE Intl. Joint Conference on Neural Networks (IJCNN'05) 2, 1018 – 1022, Canada.
[7] Mulawka , J. J., Verma, B. K. (1994). Improving the Training Time of the Backpropagation Algorithm. *International Journal of Microcomputer Applications*, 13 (2) 85-89, Canada.
[8] Verma, B. (1997). Fast training of multilayer perceptrons (MLPs). *IEEE Trans Neural Netw.* 8 (6) 1314–1321.
[9] Haykin, S. (1998). Neural Networks, a comprehensive foundation, second edition. Prentice Hall International Inc., McMaster University, Hamilton, Ontario, Canada.
[10] Frank, A., Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.