ISSN 2277-3061

Enhancing the Performance of the BackPropagation for Deep Neural Network

Ola Mohammed Surakhi[1], Walid A. Salameh[2]
[1] Department of Computer Science
Princess Summaya University for Science and Technology
Amman, Jordan
Ola.surakhi@gmail.com
[2] Department of Computer Science
Princess Summaya University for Science and Technology
Amman, Jordan
walid@psut.edu.jo

ABSTRACT

The standard Backpropagation Neural Network (BPNN) Algorithm is widely used in solving many real problems in world. But the backpropagation suffers from different difficulties such as the slow convergence and convergence to local minima. Many modifications have been proposed to improve the performance of the algorithm such as careful selection of initial weights and biases, learning rate, momentum, network topology and activation function. This paper will illustrate a new additional version of the Backpropagation algorithm. In fact, the new modification has been done on the error signal function by using deep neural networks with more than one hidden layers. Experiments have been made to compare and evaluate the convergence behavior of these training algorithms with two training problems: XOR, and the Iris plant classification. The results showed that the proposed algorithm has improved the classical Bp in terms of its efficiency.

KEY WORDS: Neural Networks; Deep Neural Network; Backpropagation; Momentum; learning Rate; Optical Backpropagation; Extended Optical Bp; performance analysis

ISSN 2277-3061

1.INTRODUCTION

An artificial neural network, ANN, is a software system that loosely models biological neurons. It consists of small processing units known as Artificial Neurons, which can be trained to perform complex calculations. Neural networks have greatest potential in many complex real problems such as speech and image recognition. A typical multilayer feed-forward neural network consists of an input layer, hidden layer and an output layer. Every node in a layer is fully connected to every node in the next layer. Multi-layer neural networks use a variety of learning techniques; the most popular one is the *backpropagation* which proposed by Rumelhart, Hinton and Williams [8, 9].

Since the standard BackPropagation uses gradient descent learning rule, an improper choice of parameters such as the Learning Rate value, activation function, and initial weights and biases values may lead to slow network convergence, network error or failure.

A variety of researches have been applied to accelerate the learning process and improve the training efficiency [3, 4, 5, and 6]. An OBP algorithm is designed to overcome some of the problems associated with standard BP training using non-linear function, which applied on the output units to escape from local minima with high speed of convergence during the training period. [7]

This paper presents an extended version of an Optical Backpropagation (OBP) algorithm. The proposed algorithm improve the performance of the Optical Backpropagation algorithm (OBP) on deep neural network, the experimental results show that the proposed algorithm converges to a reasonable range of error after a few number of training epochs.

2. STANDARD BACKPROPAGATION (BP)

The Backpropagation BP is designed to minimize the mean square error between the actual output and the desired output . For a given set of input patterns applied to the first layer in the neural network, it propagated through each upper layer until an output is generated. This output is then compared to the known and desired output and the error value is calculated. Based on the error, the connection weights are adjusted backward from the output layer to each unit in the network.

Algorithm for a 3-layer network with m input units, n hidden units, and p output units can be described as follows [2, 7]:

1. Initialize network weights (often small random values)
2. Apply the input vector to the input units

$$X_p = (X_{p1}, X_{p2}, \ldots X_{pn})$$

3. Compute the net- input values to the hidden layer units:

$$net^h_{pj} = \left(\sum_{i=1}^{N} W^h_{ji} \bullet X_{pi} + \theta\right) \quad (2.1)$$

4. Calculate the outputs from the hidden layer:

$$i_{pj} = f^h{}_j(net^h{}_{pj}) \quad (2.2)$$

5. Calculate the net-input values to each output unit:

$$net^o{}_{pk} = \left(\sum_{j=1}^{L} W^o{}_{kj} \bullet i_{pj}\right) \quad (2.3)$$

6. Calculate the outputs:

$$O_{pk} = f^o{}_j(net^o{}_{pk}) \quad (2.4)$$

7. Calculate the error terms for the output units:

$$\delta^o{}_{pk} = (Y_{pk} - O_{pk}) \bullet f^{o'}{}_k(net^o{}_{pk}) \quad (2.5)$$

Where

$$f^{o'}{}_k(net^o{}_{pk}) = f^o{}_k(net^o{}_{pk}) \bullet (1 - f^o{}_k(net^o{}_{pk})) \quad (2.6)$$

8. Calculate the error terms for the hidden units:

$$\delta^h{}_{pj} = f^{h'}{}_j(net^h{}_{pj}) \bullet \left(\sum_{K=1}^{M} \delta^o{}_{pk} \bullet W^o{}_{kj}\right) \quad (2.7)$$

9. Update weights on the output layer:

$$W^{o}kj(t+1) = W^{o}kj(t) + (\eta \bullet \delta^{o}pk \bullet i_{pj}) \tag{2.8}$$

10. Update weights on the Hidden layer:

$$W^{h}ji(t+1) = W^{h}ji(t) + (\eta \bullet \delta^{h}pj \bullet X_{i}) \tag{2.9}$$

3. PROPOSED ALGORITHM

3.1 Extended Optical Bp (EOBP)

The difficulty encountered in the standard Backpropagation algorithm is when the actual value $f^{o}{}_{k}(net^{o}{}_{pk})$ approaches either extreme value, the factor $f^{o}{}_{k}(net^{o}{}_{pk}) \bullet (1 - f^{o}{}_{k}(net^{o}{}_{pk}))$ in equation (2.6) makes the error signal very small. This implies that an output unit can be maximally wrong without producing a strong error signal with which the coupling strengths could be significantly adjusted.[7] The Optical Backpropagation algorithm (an enhanced Backpropagation) focused on this delay of the convergence that is caused by the derivative of the activation function.

The convergence speed of the training process was improved significantly by OBP through maximizing the error signal, which was transmitted backward from the output layer to each unit in the intermediate layer.

The error at a single output unit in adjusted OBP is defined as [6], [7]:

$$New\delta_{pk} = (1 + e^{(Y_{pk} - O_{pk})^2}) \bullet f'(\sum w_{ij} x_i)$$

$$, \; if\,(Y - O) > zero. \tag{3.1}$$

$$New\delta_{pk} = -(1 + e^{(Y_{pk} - O_{pk})^2}) \bullet f'(\sum w_{ij} x_i)$$

$$, \; if\,(Y - O) < zero. \tag{3.2}$$

$$New\delta_{pk} = 0$$

$$, \; if\,(Y - O) = zero. \tag{3.3}$$

Where the subscript "P" refers to the p_{th} training vector, and "K "refers to the k_{th} output unit. In this case, Y_{pk} is the desired output value, and O_{pk} is the actual output from k_{th} unit, then δpk will propagate backward to update the output-layer weights and the hidden-layer weights.

The error signal will minimize the errors of each output unit more quickly than the Backpropagation error signal and so the weights on certain units change very large from their starting values. [7]

The error function defined in Optical Backpropagation is proportional to the square of the distance between the desired output and the actual output of the network for a particular input pattern.

As an alternative, any other error functions whose derivatives exist and can be calculated at the output layer can replace the traditional square error criterion [4]. In this paper, a new error function had been adopted to replace the error function used in Optical Backpropagation. The equations of the new function are given as:

$$New\delta_{pk} = \frac{1}{(1 + e^{(Y_{pk} - O_{pk})^2})} \bullet f'(\sum w_{ij} x_i) \quad if\,(Y - O) > zero \tag{3.4}$$

$$New\delta_{pk} = \frac{-1}{(1 + e^{(Y_{pk} - O_{pk})^2})} \bullet f'(\sum w_{ij} x_i) \quad if\,(Y - O) < zero \tag{3.5}$$

$$New\delta_{pk} = 0$$

$$, \; if\,(Y - O) = zero. \tag{3.6}$$

Note the equations in step 7,8,9 and 10 are changed, but all other equations and steps will be as outlined in the standard BP algorithm.

3.2 The EOBP Steps

The EOBP steps are described as follows:

1. Apply the input example to the input units.
   $x_p = (x_{p1}, x_{p2}, \ldots x_{pn})$
2. Calculate the net-input values to the hidden layer units.
3. Calculate the outputs from the hidden layer.
4. Calculate the net-input values to the output layer units.
5. Calculate the outputs from the output units.
6. Calculate the error term for the output units, using the Newδ°pk described in equations (3.4),(3.5) and (3.6)
7. Calculate the error term for the hidden units, through applying Newδ°pk.
8. Update weights on the output layer.
9. Update weights on the hidden layer.
10. Repeat steps from step 1 to step 9 until the error (Ypk – Opk) is acceptably small for each training vector pairs.

The proposed algorithm stops as the standard Bp when the squares of the differences between the actual and target values summed over the output units at all patterns are relatively small.

The new error function gives a rapid reaction to changes in the weights value by increasing the speed with less number of iterations and without loss of learn-ability.

The new algorithm was tested on different neural network architecture, with one or more hidden layers.

A deep neural network (DNN) is a feed-forward, artificial neural network that has more than one layer of hidden units between its inputs and its outputs. Each hidden unit typically uses the logistic function which was the hypertan function to map its total input from the layer below, then to sends them to the layer above. The output unit then converts its total input to produce the output, by using the “softmax” non-linearity.

In DNNs with full connectivity between adjacent layers, the initial weights are given small random values to prevent all of the hidden units in a layer from getting exactly the same gradient [10].

The normalization is important because of the multiplicative effect through layers, to maintain the activation variances and back-propagated gradients variance as one move up or down the network [10].

DNNs with many hidden layers and many units per layer are very flexible model. This makes them capable of modeling very complex and highly non-linear relationships between inputs and outputs, such as the acoustic modeling.

DNN’s can be discriminatively trained by the EOBP to measures the difference between the target outputs and the actual outputs produced for each training case.

At the beginning of the training, the back-propagated gradients gets smaller as it is propagated downwards using the New error signal which minimize the errors of each output unit faster than the old one, and the weights on certain units change relatively large from their starting values. Weights are either increased or decreased according to the sign of the term (Y – O).

4. EXPERIMENTAL EVALUATION

4.1 XOR Problem (XOR 2-2-1)

The XOR problem will be solved using neural network which consists of two input units, two hidden units, and single output unit, with biases for hidden unit and the output unit, without direct connection from input to the output layers. The architecture of this network is shown in figure 4.1.

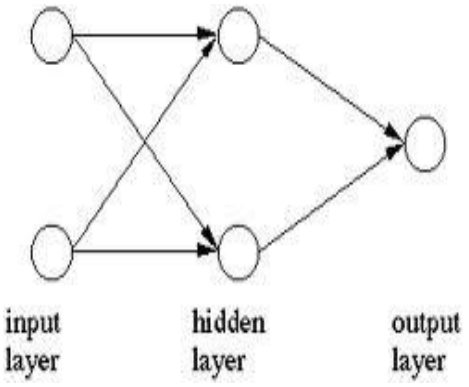

Figure 4.1: XOR neural network architecture 2x2x1

Table 4.1 shows the parameters to solve the XOR (2-2-1) problem using the BP and EOBP algorithms.

Table 4.1: The parameters to solve the XOR (2-2-1) problem

	EOBP	BP
Initial weights	**0.001 - 0.0001**	**0.001 - 0.0001**
Learning Rate	**0.1 - 1.0**	**0.1 - 1.0**
Momentum	**0.2**	**0.2**
Epoch limits	**1000**	**10000**
MSE	**0.0001**	**0.0001**

The results of the training processes using the EOBP and BP algorithms will be explained in table 4.2 and figure 4.2.

Table 4.2: Solve XOR (2x2x1) problem Using EOBP and BP

Learning Rate	EOBP	BP
0.1	**804**	**17140**
0.2	**368**	**7867**
0.3	**227**	**4844**
0.4	**158**	**3376**
0.5	**118**	**2523**
0.6	**92**	**1974**
0.7	**74**	**1594**
0.8	**62**	**1319**
0.9	**52**	**1112**
1.0	**44**	**953**

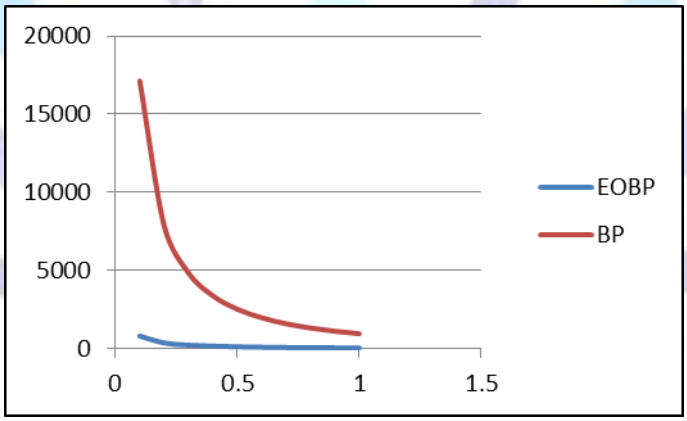

Figure 4.2: Solve XOR (2x2x1) problem Using EOBP and BP

Table4.2 and figure4.2 show that the EOBP algorithm is more efficient than the BP algorithm because; it can speed up the convergence rate. Also, as the learning rate become near to the 1.0 value, the performance of the EOBP become better.

From the table4.2, the number of epochs needed to train the network is equal to **953** through a learning rate of 1.0, and it is equal to **804** if we used the EOBP with the learning rate equal to 0.1.

So the EOBP speeds up the training process when use a very small learning rate, while using a small value of the learning rate in the BP will leads to slow down the training process.

4.2 Iris Plant Classification

The iris plant classification problem is a well-known benchmark problem concerning classification of flowers. Three iris flowers classes are known: Iris setosa, Iris versicolor and Iris virginica. The classification is based on four leaf attributes namely, sepal length and width, and petal length and width. These attributes denoted by x1, x2, x3, and x4 were measured in millimeters and collected in the Iris database which consists of 150 items as shown below.

5.1, 3.5, 1.4, 0.2, Iris setosa

7.0, 3.2, 4.7, 1.4, Iris versicolor

6.3, 3.3, 6.0, 2.5, Iris virginica

4.9, 3.0, 1.4, 0.2, Iris setosa

There are 50 data points for each species. Because neural networks work with numeric data, the categorical species information must be converted to numeric data. When performing neural network classification, the classes to be predicted is stored directly in 1-of-N encoded dependent-variable data located in the last three columns:

5.1, 3.5, 1.4, 0.2, 0, 0, 1

7.0, 3.2, 4.7, 1.4, 0, 1, 0

6.3, 3.3, 6.0, 2.5, 1, 0, 0

4.9, 3.0, 1.4, 0.2, 0, 0, 1

...

The experiments starts by splitting the data set, which consists of 150 items, into a training set of 120 items (80 percent) and a test set of 30 items (20 percent). Next, the experiments create a neural network with four input nodes (one for each numeric input), seven hidden nodes and three output nodes (one for each possible output class). The neural network's weights and bias values are initialized to small (between 0.001 and 0.0001) random values. As the weights and biases determine the output values for a given set of input values, the two algorithms (EOBP and BP) are used to search for weights and bias values that generate neural network outputs that most closely match the output values in the training data then the two results are compared to check the performance of each algorithm. Table 4.3 shows the parameters to solve the Iris plant classification problem using the BP and EOBP algorithms.

Table 4.3: The parameters to solve the Iris plant classification problem

	EOBP	BP
Initial weights	**0.001 - 0.0001**	**0.001 - 0.0001**
Learning Rate	**0.05**	**0.05**
Momentum	**0.1**	**0.1**
Epoch limits	**2000**	**2000**
MSE	**0.001**	**0.001**

The 4-7-3 neural network will have 4*7 + 7*3 = 49 weights and 7+3 = 10 biases. The initial weights selected randomly, and the same initial weights have been used for the two algorithms. After the neural network has been trained, result displays the final weights that were determined by the training process. Table 4.4 shows a sample of twenty values for the final weights.

Figure 4.3 shows that the differences between the final weights from input layer to the output layer using an EOPB and BP are very small.

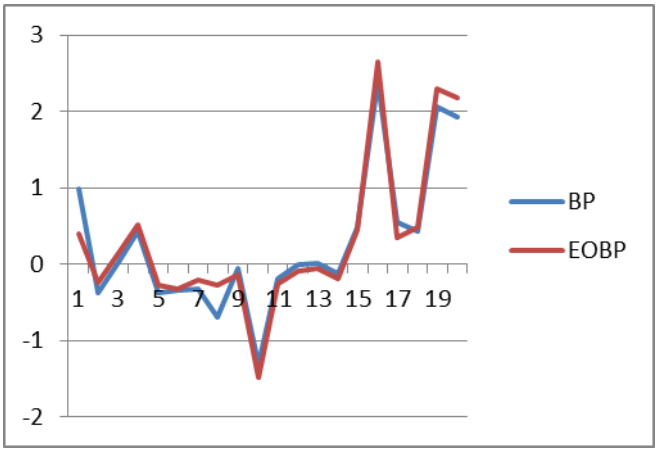

Figure 4.3: Final weights using the BP and an EOBP.

After a neural network has been trained, the prediction accuracy on the training data set and the prediction accuracy on the test data are computed and shown in table 4.5.

Table 4.4: initial and final weights using BP and EOBP

Initial weights	BP	EOBP
0.0008	0.989	0.405
0.0008	-0.373	-0.246
0.0008	0.033	0.14
0.0006	0.425	0.511
0.0003	-0.379	-0.283
0.0006	-0.335	-0.321
0.0009	-0.329	-0.201
0.0005	-0.7	-0.278
0.001	-0.054	-0.143
0.0003	-1.31	-1.484
0.0004	-0.184	-0.257
0.0005	-0.004	-0.093
0.0007	0.003	-0.065
0.0005	-0.123	-0.185
0.001	0.488	0.441
0.0001	2.403	2.646
0.0009	0.549	0.346
0.001	0.436	0.487
0.0007	2.066	2.298
0.0004	1.928	2.171

Table 4.5: accuracy on the training data and test data

	BP	EOBP
Training data	0.9833, 118 correct out of 120	0.9917,119 correct out of 120
Test data	0.9667, 29 out of 30	0.9667, 29 out of 30

The result shows that the prediction accuracy on the training data set and the prediction accuracy on the test data for the EOBP are larger than the prediction accuracy on the training data set and the prediction accuracy on the test data for the BP.

4.3 Iris Plant Classification (Deep Learning)

The next problem to be described is the Iris plant classification problem with different neural network architecture. The network consists of 4 units in the input layer, and 3 units in the output layer, and two hidden layers with 4 units for the first hidden layer and 8 units for the second hidden layer. And use the softmax logistic function for the output layer and the hypertan function for the hidden layers.

Table 4.6 shows the parameters to solve the Iris plant classification problem (Deep Learning) using the BP and EOBP algorithms.

In this experiment, the EOBP was tested to solve it. The results show that EOBP needed 1000 epochs to train the network, which is less that number of epochs needed to train the network using the standard BP which equals to 2120.

Table 4.6: The parameters to solve the Iris plant classification problem (Deep Learning)

	EOBP	BP
Initial weights	0.001 - 0.0001	0.001 - 0.0001
Learning Rate	0.01	0.01
Epoch limits	10000	10000
MSE	0.001	0.001

4.4 Compare Two Results Using Different Neural Network Architecture

For a neural network with 3 units for input layer, 3 hidden layers with 3, 4 and 7 nodes for each respectively, and 2 units for output layer, the final weights from input to the first hidden layer and from the first hidden layer to the second hidden after using the EOBP and the BP are summarized in table 4.7.

Training is discontinued when the MSE falls below 0.00001. The initial weights selected randomly, and the same initial weights have been used for the two algorithms, the Learning rate was set to 0.5 and the momentum value was equal to 0.1.

Figure 4.4 shows that the differences between the final weights from input layer to the first hidden layer and from the first hidden layer to the second hidden layer using an EOPB and BP are very small.

Table 4.7: Initial and final weights from input to the first hidden layer and from the first hidden layer to the second hidden

Weights	Initial weights	BP	EOBP
W1	0.0324	0.04501	0.04497
W2	0.0199	0.04298	0.04298
W3	0.0520	0.07850	0.07852
W4	0.0794	0.05418	0.05426
W5	0.0691	0.02314	0.02314
W6	0.0489	-0.00398	-0.00403
W7	0.0418	0.07977	0.07965
W8	0.0949	0.16400	0.16400
W9	0.0191	0.09851	0.09858

W10	**0.0678**	**0.07501**	**0.07477**
W11	**0.0125**	**0.02369**	**0.02344**
W12	**0.0323**	**0.04334**	**0.04303**
W13	**0.0388**	**0.04889**	**0.04861**
W14	**0.0990**	**0.11938**	**0.11906**
W15	**0.0713**	**0.10325**	**0.10295**
W16	**0.0689**	**0.10027**	**0.09976**
W17	**0.0354**	**0.06416**	**0.06378**
W18	**0.0653**	**0.07989**	**0.07944**
W19	**0.0733**	**0.09583**	**0.09536**
W20	**0.0731**	**0.09541**	**0.09481**
W21	**0.0954**	**0.11582**	**0.11530**

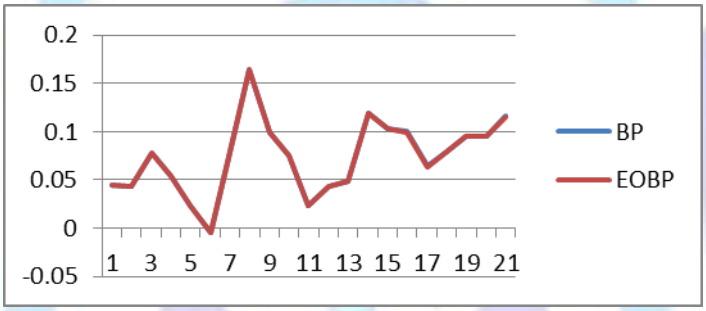

Figure 4.4: Final weights using the BP and an EOBP.

The adapting process of weights from input layer to the first hidden layer and from the first hidden layer to the second hidden layer using the standard BP and EOBP is shown in figures 4.5 and 4.6.

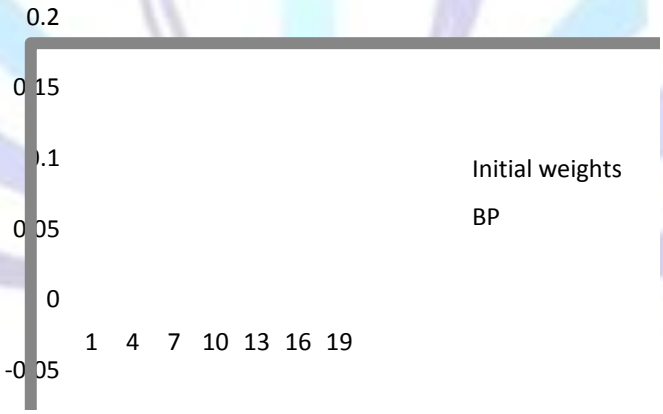

Figure 4.5: Adapting process of weights from input layer to the first hidden layer and from the first hidden layer to the second hidden layer using BP.

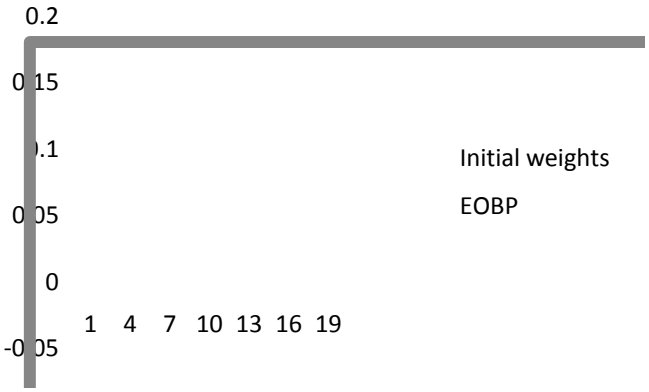

Figure 4.6: Adapting process of weights from input layer to the first hidden layer and from the first hidden layer to the second hidden layer using EOBP.

Table 4.8 shows the initial and final weights from the last hidden layer to the output layer, figure 4.7 and figure 4.8 show the adapting process of weights from hidden to output layer using BP and EOBP respectively.

Table 4.8: Initial and final weights from the last hidden layer to the output layer.

Weights	Initial weights	BP	EOBP
W1	0.0380	0.03286	0.03270
W2	0.0593	-0.32729	-0.34368
W3	0.0375	0.37077	0.35736
W4	0.0998	-0.18947	-0.20195
W5	0.0888	0.22326	0.21397
W6	0.0816	0.01536	0.00022
W7	0.0298	0.09392	0.10220
W8	0.0190	-0.27993	-0.29982
W9	0.0261	0.35315	0.34141
W10	0.0486	0.01765	0.01765
W11	0.0960	0.09763	0.09763
W12	0.0462	0.05392	0.05392
W13	0.0796	0.06751	0.06751
W14	0.0952	0.01422	0.01422

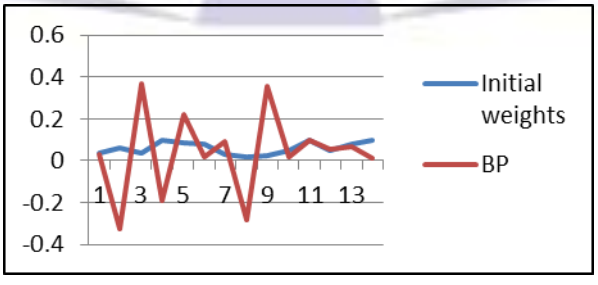

Figure 4.7: Adapting process of weights from hidden to output layer using BP.

ISSN 2277-3061

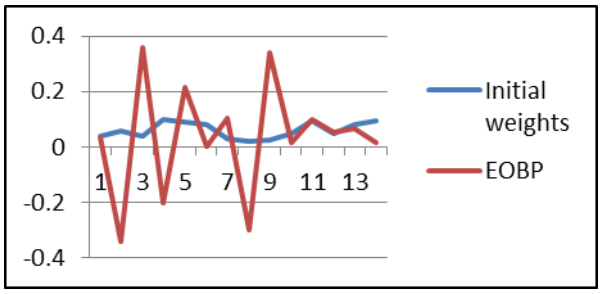

Figure 4.8: Adapting process of weights from hidden to output layer using EOBP.

Figure 4.9 shows the differences between the final weights from input layer to the first hidden layer and from the first hidden layer to the second hidden layer using BP and EOBP.

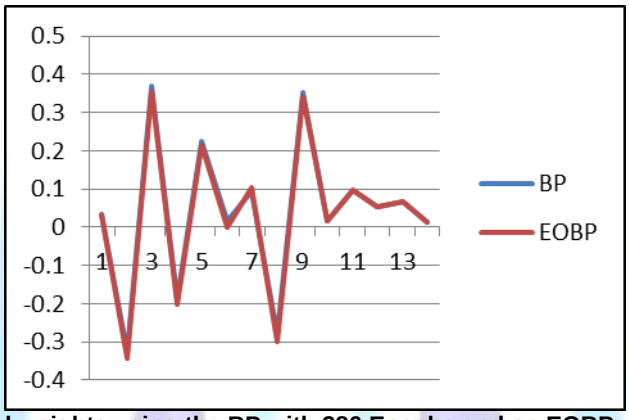

Figure 4.9: Final weights using the BP with 286 Epochs and an EOBP with 32 Epochs.

The remarkable result obtained from previous figure is the number of epochs using the two algorithms, which proves that the EOBP gives a better result compared with the standard BP with a fewer number of iterations.

In addition, this network was tested using the two algorithms with different learning rate, Table 4.9, shows the results:

The results of EOBP are much faster than the BP for all training processes with different learning rate.

Table 4.9: Training processes using different learning rate.

Learning Rate	BP	EOBP
0.1	1418	155
0.2	710	78
0.3	474	53
0.4	356	41
0.5	286	32

5. CONCLUSION

The Back-propagation Neural Network (BPNN) is a supervised learning neural network model highly applied in different applications around the globe. Although it is widely implemented in the most practical ANN applications and performs relatively well, it is suffering from a problem of slow convergence and convergence to local minima and there still exist areas where improvements can be made. This makes Artificial Neural Network's application very challenging when dealing with large problems. This paper introduced an extended version of the Optical Backpropagation algorithm, EOBP, for training the Deep Neural Network in order to improve the learning speed.

The EOBP is an enhanced version of the Optical Backpropagation algorithm. The study shows that EOBP is beneficial in speeding up the learning process by using a modified error function which enhances the whole training process in terms of requiring less number of epochs to converge.

The characteristics of the EOBP are:

- Able to reach a very small MSE

- Work with deep neural network with more than one hidden layers
- Work with multilayer neural networks with one hidden layer
- Work with a small value for the learning rate
- Work with biases
- Work with small range for the initial weights
- can perform well for small training samples

The effectiveness of the EOBP algorithm has been compared with the standard BP algorithm and verified by means of simulation on two real problems. The experimental results show that the EOBP algorithm converges to a reasonable range of error after a few number of training epochs and this enforce the usage of it as alternative training algorithm of standard BP for Deep Neural Network.

6. REFERENCES

[1] F. M. Silva and L. B. Almeida,"Speeding up backpropagation", in Advanced Neural Computers, 1990,151-158.

[2] Freeman, J. A., Skapura, D. M., Backpropagation. Neural Networks Algorithm Applications and Programming Techniques, 1992, 89-125.

[3] J. Leonard and M. A. Kramer, "Improvement of the backpropagation algorithm for training neural networks", Computer Chem. Engng., 1990, 14(3),337-341.

[4] Haykin, S. (2003): Neural Networks: A Comprehensive Foundation, PHI, New –Delhi, India

[5] Minai, A.A., Williams, R.D., Acceleration of back-propagation through learning rate momentum adaptation, Proceedings of the International Joint Conference on Neural Networks, 1990, 1676-1679.

[6] M.A. Otair and W.A. Salameh, An Improved Back-Propagation Neural Networks using a Modified Non-linear Function, Proceedings of the IASTED International Conference, 2004,442-447.

[7] M.A. Otair and W.A. Salameh, Efficient Training of Backpropagation Neural Networks, Neural Network World, Vol. 6, 2006, 291-311. http://www.nnw.cz/obsahy06.html

[8] Rumelhart, D. E., Hinton, G.E., Williams, R. J.: Learning Internal Representations by error Propagation. J. Parallel Distributed Processing: Exploration in the Microstructure of Cognition. (1986).

[9] Rumelhart, D. E., Richard Durbin, Richard Golden, and Yves Chauvin, Backpropagation: theoretical foundations. In Y.Chauvin and D. E Rumelhart (eds). Backpropagation and Connectionist Theory. Lawrence Erlbaum, 1992.

[10] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in Proceedings of AISTATS, 2010,pp. 249–256.