MODIFIED BACK PROPAGATION ALGORITHM FOR LEARNING ARTIFICIAL NEURAL NETWORKS

Waleed A. Maguid Ahmed (Eng., Cairo University), El Sayed M.Saad (Prof., Helwan University), El Sayed A. Aziz (Prof., Cairo University)

Abstract:

Back Propagation is now the most widely used tool in the field of Artificial Neural Networks. Many attempts try to enhance this algorithm to get minimum mean square error, less training time and small number of epochs. This paper first reviews the disadvantages of the Back Propagation algorithm. Next, the new modified Back Propagation is explained. Finally, comparison between the two algorithms is made through many examples.

1. Introduction:

For many years there was no theoretically sound algorithm for training multilayer Artificial Neural Networks. Since single-layer networks proved to be severely limited in what they could represent (hence, in what they could learn).

1.1 The Back Propagation is one of the most powerful methods used to train Artificial Neural Networks, but there exist many disadvantages and limitations such as:

1.1.1 The optimal number of hidden layer that are required to design the network to solve certain problem can not determined exactly by mathematical rule, but are be determined by trials (this is the common problem for multi-layer network using any training algorithm).

1.1.2 Necessary number of hidden neurons:

The size of a hidden layer is one of the most important considerations when solving actual problems using multilayer feedforward networks. This problem is under intensive study with no conclusive answer available thus far for many tasks. The exact analysis of the issue is rather difficult because of the complexity of the network mapping and due to the non-deterministic nature of many successfully completed training procedures.

1.1.3 The initial value of weights and biases are not determined exactly by mathematical rule, but are determined by trials.

1.1.4 The optimal value of learning rate is not determined exactly.

1.1.5 Scaling problem: The output of the Neural Network is limited by the activation function or preprocessing operation.

1.1.6 The Back Propagation algorithm is a first-order approximation of the steepest-descent technique, which is not the best type of the steepest descent method for finding the minimum of the mean square error [1].

1.1.7 Local minimum problem [2]:

Another peculiarity of the error surface that impacts the performance of the Back Propagation algorithm is the presence of local minima (i.e., isolated valleys) in addition to global minima. Since Back Propagation is basically a hill-climbing technique, it runs the risk of being trapped in a local minimum, where every small change in synaptic weights increases the cost function. But somewhere else in the weight space there exists another set of synaptic weights for which the cost function is smaller than local minimum in which the network is stuck. Clearly, it is undesirable to have the learning process terminating at a local minimum, especially if it is locate far above a global minimum.

1.1.8 Convergence problem [1]: The Back Propagation algorithm depends on the gradient of the instantaneous error surface in the weight space. The algorithm is therefore stochastic in nature; that is, it has a tendency to zigzag its way about the true direction to a minimum on the error surface. Indeed, Back Propagation learning is an application of a statistical method known as stochastic approximation and there exists two fundamental causes for this property:

1.1.8.1 The error surface is fairly flat along the weight dimension, which means that the derivative of the error surface with respect to that weight is small in magnitude. In such a situation, the adjustment applied to the weight is small and, consequently, many iterations of the algorithm may be required to produce a significant reduction in the error performance of the network. Alternatively, the error surface is highly curved along a weight dimension, in which case the derivative of the error surface with respect to that weight is

large in magnitude. In this second situation, the adjustment applied to the weight is large, which may cause the algorithm to overshoot the minimum of the error surface.

1.1.8.2 The direction of the negative gradient vector may point away from the mininum of the error surface; hence the adjustments applied to the weights may induce the algorithm to move in the wrong direction. Consequently, the rate of convergence in Back Propagation learning tends to be relatively slow.

2. Modified Back Propagation Algorithm:

To overcome some of the disadvantages of the Back Propagation algorithm, two modification techniques are proposed in this paper. The first modification technique helps in reducing the mean square error and also reducing the number of epochs required for training the Artificial Neural Networks, hence speeding up the training of the Neural Networks. The second modification technique computes the near optimum value of the learning rate for each training pattern that also reduces the number of training epochs. In the following, the two modification techniques are presented.

2.1 **First Modification Technique:** Second-Order Approximation Of The Steepest- Descent.

The Back Propagation method depends on the first-order approximation of the steepest-descent method. In the first modification technique the second-order approximation of the steepest-descent method is proposed to be used instead of the first–order approximation.

2.1.1 Derivation of Back Propagation Algorithm after the First Modification Technique:

Iteration (1):

i node is an output unit (at time t).

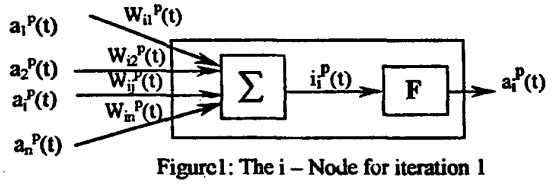


Figure1: The i – Node for iteration 1

$$W_{ij}(t+1) = W_{ij}(t) + \Delta W_{ij}(t)$$

$$\Delta_p W_{ij}(t) = \gamma\, \delta_i^P(t)\, a_j^P(t)$$

If i node is an output node:

$$\delta_i^P(t) = (d_i^P - a_i^P(t))\, F_i'(i_i^P(t))$$

If i node is not an output node:

$$\delta_i^P(t) = F'(i_i^P(t)) \sum_{h=1}^{N_o} \delta_h^P(t)\, W_{hi}(t)$$

Where

γ : The learning rate of the network.

$a_j^P(t)$: The output of the node j when the input of order P entered to the network.

d_i^P : The desired output of the i node when the input of order P entered to the network.

$i_i^P(t)$: The input to the activation function of the i node

F : The activation function of the i – node.

Iteration (2):

i node is an output unit (at time t+1).

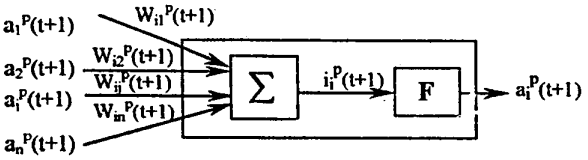


Figure 2: The i – Node is an output Node for iteration 2

$$W_{ij}(t+2) = W_{ij}(t+1) + \Delta W_{ij}(t+1)$$

$$\Delta_p W_{ij}(t+1) = \gamma\, \delta_i^p(t+1)\, a_j^p(t+1)$$

If i node is an output node:

$$\delta_i^p(t+1) = (d_i^p - a_i^p(t+1))\, F_i'(i_i^p(t+1))$$

If i node is not an output node:

$$\delta_i^p(t+1) = F'(i_i^p(t+1)) \sum_{h=1}^{N_o} \delta_h^p(t+1)\, W_{hi}(t+1)$$

Where

γ : The learning rate of the network.

$a_j^p(t+1)$: The output of the node j when the input of order P entered to the network.

d_i^p : The desired output of the i node when the input of order P entered to the network.

$i_i^p(t+1)$: The input to the activation function of the i node.

F :The activation function of the i node

2.2 **Second Modification Technique:** Computation of Learning Rate per Pattern Using One Dimensional Optimization.

This technique tries to get the near optimum learning rate within the specified range for each training input, grid search method is used, this method requires only the objective function values but not the partial derivatives of the function (non-gradient method).

2.2.1 Grid Search Method:

This method involves setting up a suitable grid in the design space, evaluating the objective function at all grid points, and finding the grid point corresponding to the lowest function value. The best value of the learning rate γ for most applications exists between 0.1 to 1.

2.2.1.1 Begin by $\gamma = 0.1$ and compute E.

2.2.1.2 Repeat step 2.2.1.1. 5 times, for each time increase γ by 0.2 and compute E

2.2.1.3 Take the value of γ that gives the smallest value of E. Where E is the mean square error.

This method of choosing near optimal learning rate γ within the depicted region for each input pattern will increase the time duration of each epoch. On the other hand, it will reduce the number of epochs required for training the Neural Networks in comparison with using Back Propagation algorithm with randomly selected learning rate

2.2.2 Note that: the effect of using the two modification techniques in the Back Propagation algorithm is to get the general rule of having the near optimum value of learning rate (second modification) with minimum training time (first modification). Another modification technique may be added for the Neural Networks that are used in classification. (The output is binary) and also, this is done in the test phase and not in the training phase. This third modification technique can be called rounding.

2.3 **Third Modification Technique:** Rounding Technique.

This technique is used only for Neural Networks that are used in the classification, and tries to train the Networks with virtually zero mean square error, This is done in the test phase and not in the training phase.

2.3.1 Rounding Technique:

In the test phase after computing the actual output, the technique proceeds as follows:

If the actual output is greater or equal to 0.8 then make it 1.

If the actual output is less or equal to 0.3 then make it 0.

If the actual output is less than 0.8 and greater than 0.3, its value remains the same.

The result of this rounding is to make the mean square error virtually zero. That should improve the performance of the Artificial Neural Networks without affecting the recognition of the network.

3. Modified Back Propagation Algorithm:

Step 0: Initialize weights (set to small random values).

Step 1: While stopping condition is false. Do steps 2 – 9.

Step 2: For each training pair, do steps 3 –9.

Step 3: Each input unit (X_i , i = 1,.....N_i) receives input signal and broadcasts this signal to all units in the layer above (the hidden units).

Step 4: Each hidden unit sums its weighted input signals,

$$i_i^p(t) = \theta_i(t) + \sum_{j=1}^{N_i} a_j^p(t)\, W_{ij}(t)$$

Apply this activation function to compute the output signal,

$$a_i^p(t) = F(i_i^p(t))$$

Send this signal to all units in the layer above (output units).

Step 5: Each output unit sums its weighted input signals,

$$i_i^p(t) = \theta_i(t) + \sum_{j=1}^{N_h} a_j^p(t)\, W_{ij}(t)$$

Apply the activation function to compute the output signal.

$$a_i^p(t) = F(i_i^p(t))$$

$$\delta_i^p(t) = (d_i^p - a_i^p(t))\, F'(i_i^p(t))$$

Step 6: Each output unit receives a target pattern corresponding to the input training pattern. Compute the error term.

Step 7: Calculate the near optimal learning rate using the second modification technique, which is mentioned above.

Step 8: Calculate the weight correction term and its bias correction. Send error term to units in the layer below.

$$\Delta W_{ij}(t) = \gamma\, \delta_i^p(t)\, a_j^p(t)$$

$$\Delta\theta_i(t) = \gamma\, \delta_i^p(t)$$

Step 9: Each hidden unit sum its delta input (from the unit in the layer above). Multiply by the derivative of its activation function to calculate its error term. Calculate the weight correction term, and calculate the bias correction term

$$\delta_i^p(t) = F'(i_i^p(t)) \sum_{k=1}^{N_h} \delta_k^p(t)\, W_{ik}(t)$$

$$\Delta W_{ij}(t) = \gamma\, \delta_i^p(t)\, a_j^p(t)$$

$$\Delta\theta_i(t) = \gamma\, \delta_i^p(t)$$

Step 10: Each output unit updates its bias and weights, and each hidden unit updates its bias and weights.

$$W_{ij}(t+1) = W_{ij}(t) + \Delta W_{ij}(t)$$

$$\theta_i(t+1) = \theta_i(t) + \Delta\theta_i(t)$$

Step 11: Enter the same input unit (X_i , i = 1,.....N_i) and repeat again steps from step 4 to step 10 except step 7
Step 12: (used in the test phase for classification only)

$$\text{If } a_i^p(W_{final}) \le 0.3 \quad \text{then } a_i^p(W_{final}) = 0$$

$$\text{If } a_i^p(W_{final}) \ge 0.8 \quad \text{then } a_i^p(W_{final}) = 1$$

Step 13: **Test stopping condition.**
The Back Propagation algorithm is terminated at the weight vector W_{final} when

$$-\tau \le (d_i^p - a_i^p(W_{final})) \le \tau$$

Where τ : is sufficient small error threshold

4. The advantages of The Modified Back Propagation Algorithm:

4.1 Minimizing the training time and the number of epochs required for training.
4.2 Searching for the optimum learning rate for each input pattern.
4.3 Give virtually zero mean square error in the case of classification problem.

5. Applications:

5.1 Logic function:

The Architecture of the Artificial Neural Network that is used to simulate AND, OR logic function is as follows:

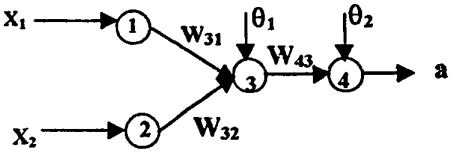


Figure 3: Architecture of neural network used to simulate AND, OR logic function

The following tables illustrates a comparison of the two techniques according to number of epochs, mean square error and duration of training, When the maximum error of the output is 0.01 (this means that the maximum difference between any desired and actual output is 0.01)

5.1.1 AND Operation:

	Back Propagation (learning rate 0.1)	Back Propagation (learning rate 0.9)	Modified Back Propagation
No. of epoch	65549	7330	89
Mean square error	0.000195	0.000196	Zero
Duration of training	37 min and 14 sec	4 min and 14 sec	3 sec

Table 1: Comparison between the MBP and BP at error 0.01 in case of AND logic function

5.1.2 OR operation:

	Back Propagation (learning rate 0.1)	Back Propagation (learning rate 0.9)	Modified Back Propagation
No. of epoch	85613	9520	39
Mean square error	0.00008459	0.000084569	ZERO
Duration of training	48 min and 45 sec	5 min and 26 sec	2 sec

Table 2: Comparison between the MBP and BP at error 0. 01
in case of OR logic function

5.1.3 XOR Operation:

The Architecture of the Artificial Neural Network that used to simulate XOR logic function is as follows:

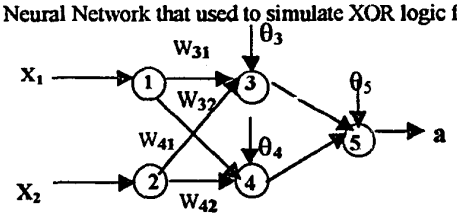


Figure 4:Architecture of neural network used for simulating XOR logic function

	Back Propagation (learning rate 0.1)	Back Propagation (learning rate 0.9)	Modified Back Propagation
No. of epoch	89299	10786	323
Mean square error	0.000160345	0.0001466	ZERO
Duration of training	51 min and 8 sec	4 min and 11 sec	13 sec

Table 3: Comparison between the MBP and BP at error 0. 01

in case of XOR logic function

5.2 **Character Recognition:**

5.2.1 Back Propagation can be used to train a multiple layer network to function as character recognizer of printed character. Each character is represented in the input data as a vector with 63 binary components, the output is a vector of 26 binary components (bit map method). Each output distinguishes one character. The Architecture of the Artificial Neural Network that is used to simulate character recognition function consists of three layers (input layer, one hidden layer and output layer). In the input layer there are 63 units, in the hidden layer there are 7 units and in the output layer 26 output units.

The following table illustrates the comparison of the two techniques according to the number of epochs, the mean square error and the duration of training, when the maximum error of the output is 0.1

	Back Propagation (learning rate 0.1)	Back Propagation (learning rate 0.9)	Modified Back Propagation
No. of epoch	5320	637	230
Mean square error	0.13707348	0.16484148	ZERO
Duration of training	1 hour and 31 min and 28 sec	10 min and 45 sec	13 min and 34 sec

Table 4: Comparison between the MBP and BP at error 0. 1

in case of character recognition function

5.2.2 Important notes:

As mentioned from the above table there is no comparison between the BP and the MBP when we use learning rate of 0.1 in the BP and as mentioned above the MBP is better than BP. This means that, when we choose the wrong learning rate the power of MBP appears. But when we compare between the two methods in case of learning rate of 0.9 with the BP (which is the best learning rate in the mentioned range), the learning time of BP is less than that of MBP. This increase of learning time is the cost paid for unprediction of any learning rate. To prove this point we will use the MBP without one-dimensional optimization and the following table illustrates the idea.

	Modified Back Propagation	Modified Back Propagation (without one-dimensional optimization)
No. of epoch	230	232
Mean square error	ZERO	ZERO
Duration of training	13 min and 34 sec	6 min and 13 sec

Table 5: Comparison between MBP and BP without One-dimensional optimization

5.2.3 Generalization phase:

In BP or MBP learning, we typically start with a training set and use one of the above two algorithms to compute the synaptic weights of a multilayer NN by loading as many of the training examples as possible into the NN. The hope is that the designed NN will become general. A network is said to be generalized well when the input-output relationship, that is computed by the network is correct (or nearly so) for input/output patterns (test data) that was never used in creating or training the network. (The test data drew from the same population, which are used to generate the training data).

5.3 **Function approximation:**

5.3.1 Simulation of SINE function:

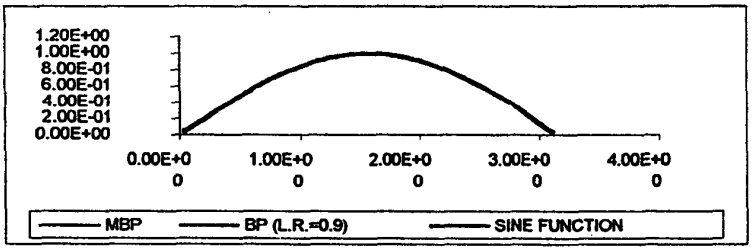
Twenty training points are used for training the network. The network is said to be learned if the total squared error over the training points is less than a specified threshold (typically 0.0005).
The architecture used in this problem is as follows:
One input layer, one hidden layer and one output layer
One input to the network, 11 nodes in the hidden layer and one node in the output layer
The following table illustrates the comparison of the two techniques according to number of epochs, mean square error and duration of training. When the maximum error of the output is 0.01 (this means that the maximum difference between any desired and actual output is 0.01)

	Back Propagation (learning rate 0.1)	Back Propagation (learning rate 0.9)	Modified Back Propagation
No. of epoch	4876322	141886	62880
Mean square error	0.0001011	0.0001514	0.0001865
Duration of training	22 hours and 30 min. and 15 sec	2 hours and 52 min. and 45 sec	2 hours and 37 min and 25 sec

Table 6: Comparison between the MBP and BP at error 0.0 1

in case of Y=sin (x) function

Figure 5 shows the output of the trained network for a test data of 100 equally spaced patterns from 0 to π , in the case of normal sine function and in BP (L.R. = 0.9) and in MBP .



5.3.2 Simulation of COS function:

Fifteen training points are used for training the network. The network is said to be learned if the total squared error over the training points is less than a specified threshold (typically 0.0005).
The architecture is the same as that used in training the SINE function.
The following table illustrates comparison of the two techniques according to number of epochs, mean square error and duration of training. When the maximum error of the output is 0.01 (this means that the maximum difference between any desired and actual output is 0.01)

	Back Propagation (learning rate 0.1)	Back Propagation (learning rate 0.9)	Modified Back Propagation
No. of epoch	424072	47741	16049
Mean square error	0.00013532	0.000121178	0.0001695
Duration of training	6 hours and 47 min and 48 sec	46 Min And 21 Sec	30 Min And 13 Sec

Table 7: Comparison between the MBP and BP at error 0. 01

in case of Y=COS (x) function

Figure 6 shows the output of the trained network for a test data of 150 equally spaced patterns from 0 to $\pi/2$, in the case of normal COS function and in BP (L.R. = 0.9) and in MBP.

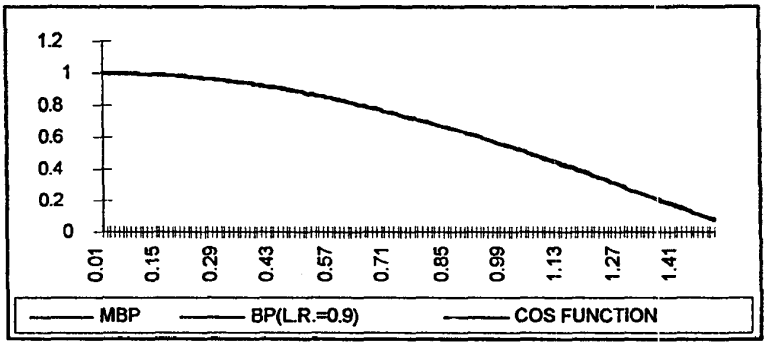


Figure 6: COS function in case of MBP and BP (L.R.=0.9)

6. Conclusion:

The modified Back Propagation Algorithm enhances the behavior of Back Propagation Algorithm as it trains the NN in a short time and with minimum error by Entering each input two successive times which has the same effect as taking two iterations for steepest descent method.

And also, it reduces the heuristic in choosing the learning rate by specifying the near optimum learning rate within the specified range for each training input by using the second modification technique.

In case of classification another modification technique may be added to the modified Back Propagation Algorithm which is called rounding, that technique results in reaching to minimum error.

7. References

[1] P. R. Adby and M. A. H. Dempster, "Introduction To Optimization Methods", chapman and hall, London, A halsted Press Book John Wiley & Sons, New York 1974.

[2] Simon Haykin, "Neural Networks: A compressive Foundation", Macmillan College, New York, 1994.