**New Adaptive Method to Speed Up Training of Back Propagation Algorithm**

Zahraa Elsayed Mohamed

Zagazig University, Faculty of Science

Zagazig, Egypt

Elsadek Hussien Ibrahim

High Institute of Engineering and Technology Al-Obour

Zagazig, Egypt

**Abstract**

Using Back Propagation (BP) algorithm in training of multilayer feedforward networks is a very effective learning approach. It finds the best weights of Artificial Neural Network (ANN) by computing the weight change. There are many drawbacks of BP algorithm but the main problems are slow training and reaching local minima easily. Through the last years, a lot of algorithms were proposed to improve and modify the BP algorithm. Overcoming these problems requires adding new parameters as learning rate and momentum. In this research, a new adaptive BP algorithm is proposed by introducing a new function for adaptive learning rate and adaptive momentum which depends on the error gradient at every layer. In the learning samples, the simulation results mention the convergence action of proposed algorithm. Comparing with classic BP algorithm, the proposed algorithm gives better convergence rates and finds a good solution efficiently. Three popular benchmark classification problems are used to explain the improvement in convergence rates.

**Key words**

Artificial neural network; back propagation algorithm; learning rate; momentum parameter.

**1. Introduction**

Actually, there are a lot of algorithms for training ANN but the most common algorithm is the error BP algorithm which has received much attention in recent years. The BP algorithm or its variation on Multilayered Feedforward Networks is widely used in many applications. BP learns the ANN by calculating the errors in output layer to discover the error in hidden layers, where it uses the gradient descent learning rule which requires a good selection of parameters such as initial weights, network topology, activation function, learning coefficient value, momentum coefficient value, and performance function. In some practical applications of BP, fast response to external events within an extremely short time are highly insisted and expected. However, the extensively used gradient descent method clearly cannot satisfy large scale applications and when higher learning performances are required. However, this algorithm is well-known to have problems with local minima problem particularly caused by neuron saturation in the hidden layer Furthermore, this type of algorithm has the uncertainty in finding the global minimum of the error criterion functions. Most existing approaches modify the learning model in order to add a random factor to it, which overcomes the tendency to sink into local minima. However, the random perturbations of the search direction and various kinds of stochastic adjustment to the current set of weights are not effective in enabling a network to escape from local minima which in a way cause the network to fail to converge to a global minimum within a reasonable number of iterations. To overcome those problems, the current work depends on adaption of two parameters learning rate ( and momentum term (α) for solving the slow training BP algorithm, increasing the speed up of BP algorithm, and improving the training efficiency of conventional BP algorithm. These parameters are the main reason of speeding up the BP algorithm by improving the convergence time and convergence rates; best validation, number of epochs, and error gradient.

A review of the different works for studying the problems of Back Propagation algorithm was proposed by Shao, H.M. and Zheng, G.F. [6]. They presented a new BP algorithm with adaptive momentum, where the momentum coefficient is adjusted iteratively based on the current descent direction and the weight increment in the last iteration. The Simulation results have shown that this new algorithm has a distinct superiority in fast convergence and smoothing oscillation over the conventional BP method. Yu, C. and Liu, B. [8] discussed an efficient acceleration BP method with momentum term and adaptive learning rate. Wang, X.G. *et al.* [7] proposed a new algorithm to ignore the local minima problem by changing the sigmoid activation functions in the hidden layer for each node so that it modifies both the weights between the hidden layer and the output layer dynamically. Ooyen, A.V. and Nienhuis, B. [5] increased the convergence speed by defining a new energy function based on the cross entropy function for calculating the error in the output layer. Burse, K. *et al.* [2] avoided the local minimum problem by defining anew form for new weight by adding momentum term. Zhang, N. [9] attempted to increase the convergence of BP algorithm by using activation function through adding gain parameter to the activation function. Al-Duais, M. S. *et al.* [1] discussed the convergence problem from another side; they attempted to make a dynamic BP algorithm by determining the learning rate and momentum coefficient as a dynamic form which depends on the output of the ANN. This effect on the new weight in output layer and hidden layers affected the convergence rates of error BP algorithm. Zhixin, S. and Bingqing [10] improved the BP algorithm, where they used the adaptive momentum term to improve the algorithm.

This paper is organized as follows. In section 1, ANN architecture and error BP algorithm are shown. The proposed method is presented in section 2. The proposed algorithm for BP is mentioned in section 3. Implementation of the proposed algorithm with data sets isdiscussed in section 4. Experiments and results are proposed in section 5. In section 6, the conclusions are provided.

1. **Artificial neural network architecture**

In this paper, the ANN is a proposed model that consists of three main layers. input layer, hidden layer and output layer. The input layer is (X1, X2, ……., Xn) nodes. The hidden layer consists of two layers each layer consists of (n) nodes. The output is one layer as (O1, O2, ……, Ok). for every layer there exist one bias node. Each layer combined with another layer by weights, the network direction is divided to two types feed forward and feed backward directions. In first time, it goes forward until it reaches the output layer and calculate the error after that it goes backward to make learning.

|  |
| --- |
|  |

**FIGURE 1.** Artificial neural network model.

In Fig.1, X1, X2, …, Xn are represented the input layer neurons. Y11, Y12, …, Y1n are the first hidden layer nodes. YY1, YY2, …, YYn are the second hidden layer nodes. O1, O2, …, Ok are the output layer nodes. V11, V21, …, Vnn are the input and hidden weights. h11, h12, …, hnn are the hidden layer weights. U11, U12, …, U1n are the bias weight. W11, W12, …, Wnk are the output weight.

1. **The proposed method**

In this method, the BP algorithm performance is improved (Eq. (1)) for ignoring the local minimum and removing the late saturation training which and reaching the global minimum.

Where: α the momentum coefficient, the learning rate coefficient

A new form of learning rate coefficient () is proposed to accelerate the error BP that depends on the error function gradient (i.e. dynamic value) instead of the fixed value in Eq. (2).

The new form of learning rate coefficient:

Where:

the derivative of activation function ,  absolute value

A new momentum term (α) is used to accelerate the error BP that depends on the learning rate coefficient in Eq. (3). The momentum term:

Substituting from Eq. (2) into Eq. (3) to obtain

By substituting η and α from Eq. (2) and Eq. (4) getting

This form is generated to increase the convergence rate, the speed up of BP algorithm, and to avoid the gross weight of the all equations which depend on learning rate coefficient and momentum coefficient while the weight is updated.

The objective of the BP algorithm is to obtain the minimum error during the training between the target and input data. The entropy function (Eq. (6)) is used to compute the error instead of the mean square function. The entropy is a nonlinear function which represents information by learning unknown data. The simulation results, using the error entropy function, show a better network performance with shorter stagnation period.

**3.1 The proposed algorithm for BP**

The proposed neural network is defined as four tuples: I, T, W, A, where I denotes the set of input nodes represented and T shows the topology of the net, which indicates the number of hidden layers and the number of neurons. W illustrates the set of weights and finally A is the activation function as in figure (1). An activation function is one of the important parameter in the ANN. This function not only determines the decision borders and the value of the activation function but also demonstrates the total signal strength of the node. Therefore, the selection of activation function cannot arbitrarily be selected because it has a huge impact on the ANN performance. Sigmoid activation function is used in this paper where it is monotonic function.

The algorithm can be applied in both directions forward propagation and the backward propagation.

**3.2 Forward propagation:**

The input unit Xn is sent to the next layer as input signal after calculating its output until it reaches the final layer. in the first hidden layer each node calculated as

(7)

Using Eq. (7) and activation function for calculating the actual output for each node of first hidden layer where

(8)

In the second hidden layer each node can be calculated as

(9)

Using Eq. (9) and activation function for calculating the actual output of second hidden layer where, (10)

The input signal for output layer, (11)

Using Eq. (11) and activation function for calculating the actual output of output layer where, (12)

**3.3 Backward propagation:**

Reaching the output of last hidden layer to the last node in output layer, the BP starts. The main goal of error BP algorithm is minimizing the error between the target value and the actual value. The error between target value and actual value is calculated by using Eq. (6) then comes the calculations through the following steps:

1. calculate the error gradient for an output using:

(13)

1. Computing the weight correction term. for updating the weights of output layer (using: (14)
2. Calculate the bias correction term. For updating the weight bias of output layer () using:

(15)

1. After updating the weights for output layer and weight bias, the output layer sends to the hidden layer. Each hidden node’s (YYn) sum weighted inputs from the nodes in above layers to get: (16)
2. Calculate the actual gradient for hidden layer (YYn) by using Eq. (16) and derivative of activation function to get: (17)
3. Compute the weight correction term for hidden layer to update (hnn):

(18)

Step 7: Calculate the bias weight correction term to update (u2n):

(19)

Step 8: After updating the weights and weight bias for the second hidden layer, the hidden layer sends to hidden unit. Each hidden node’s (Yn) sum is the input weighted from the unit in previous layer and get: (20)

Step 9: Calculate the local gradient for hidden layer (Yn) by using Eq. (20) and derivative of activation function to get: (21)

Step 10: Compute the weight correction term to update (vnn):

(22)

Step 11: Calculate the bias weight correction term to update (u1n):

(23)

**3.4 Proposed algorithm updating weight**

Updating the weights takes strategy that all weights adjusted simultaneously for every layer. The weight adjustment depends on the effect of Eq. (5) as follows:

New weight and bias weight for output layer:

(24)

(25)

New weight and bias weight for second hidden layer:

(26)

(27)

New weight and bias weight for first hidden layer:

(28)

(29)

1. **Implementation of the proposed algorithm with iris and cancer data sets**

The algorithm can be implemented by using a lot of data sets, but for example it used for training iris, cancer, and wine data sets. Iris data set represents 150 patterns, cancer data set represents 216 patterns, and wine data set represents 178 sample. The structure of BP and the proposed algorithm is (input: hidden: output). In this case the structure of iris data set (4: 10: 3), the structure of cancer data set (100: 10: 2), and the structure of wine data set (13: 10: 3).

**The steps of implementation for the proposed algorithm:**

1. For iris data set, weights are initialized W1= rand (10, 4, ‘double’); W2 = rand (10, 10, ‘double’); W3 = rand (10, 3, ‘double’); b1 = rand (10, 1, ‘double’); b2 = rand (10, 1, ‘double’); b3 = rand (3, 1, ‘double’).

For cancer data set, weights are initialized W1 = rand (10, 100, ‘double’); W2 = rand (10, 10, ‘double’); W3 = rand (10, 2, ‘double’); b1 = rand (10, 1, ‘double’); b2 = rand (10, 1, ‘double’); b3 = rand (2, 1, ‘double’).

1. Read the number of neurons in hidden layers; read the inputs Xn; after that find the target value and the determined error value = 10-6;
2. Read the proposed learning rate and momentum term and compute the error value by using error entropy function. If the error value > the determined error value, apply steps 4 to 13, and steps 4 to 6 (forward propagation).
3. Compute the nodes () of first hidden layer using Eq. (7), and the actual output ( for each node using Eq. (8).
4. Calculate the nodes ( of second hidden layer using Eq. (9), and the actual output ( for every node using Eq. (10).
5. Compute the nodes (of output layer using Eq. (11), and the actual output (for every node using Eq. (12). After that back other direction (backward propagation).
6. Compute the error training using the error entropy function using Eq. (6).
7. Compute the error gradient ( for output layer using Eq. (13).
8. Compute the weight correction ( and bias correction (term using Eq. (14) and Eq. (15).
9. After updating the weight and bias of output layer, the output layer sends to second hidden layer then compute the gradient (for second hidden layer, weight correction term ( , and bias correction term ( using equations Eq. (17), Eq. (18), and Eq. (19) respectively.
10. After update the weight and bias of second hidden layer, the second hidden layer sends to first hidden layer then calculate the gradient (for first hidden layer, weight correction term (, and bias correction term ( using equations Eq. (21), Eq. (22), and Eq. (23) respectively.
11. Update weights and bias for each layer, output layer using Eq. (24) and Eq. (25), second hidden layer using Eq. (26) and Eq. (27), first hidden layer using Eq. (28) and Eq. (29) respectively.
12. Compute the error using error function and test the condition of the determined error.
13. **Experiments and Results**

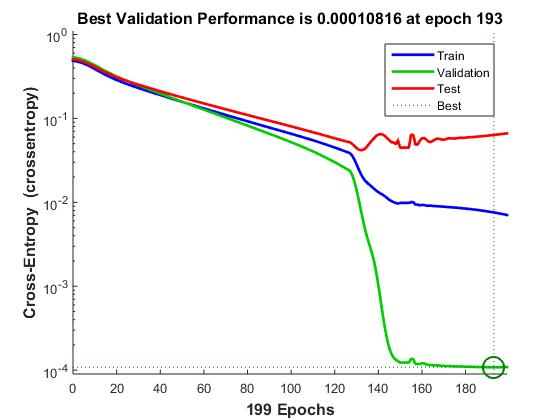
This section reports the results obtained from training the proposed algorithm by using Matlab software R2015 for three types of training sets: Iris data, cancer data, and wine data as a benchmark. The main aim of this section is to reach the minimum error between targets and actual outputs, where there is no hypothesis to define the value of bounded error, but the domain of bounded error effects the training time is defined in Grapsa *et al.* [4] locates the error possibility by 1 to a power of -5. The convergence rate is very slow and it takes 500000 epochs, but Cheung *et al.* [3] located the bounded error by minimal than 3 to a power of -4. The convergence rate is very slow and it takes 1000 epochs.

**First: Iris data set**

This section attempts to build a neural network that trains iris flowers into natural classes, such that the similar classes are grouped together. Each iris is described by four features. It represents 150 sets of iris flower attributes. The proposed algorithm results are tabulated as number of epochs and best validation in table 1. From Fig. 2 the training curve for iris data set using the proposed algorithm begins with large value for cross entropy error; after that it is smoothed and converged quickly at global minimum where the best validation is 0.000010816 at 193 epochs. Although the training curve passed through a lot of minimum values (0.0000366 at 146 epochs, 0.0000301 at 160 epochs, and 0.0000211 at 180 epochs) but all of them are considered local minima but the best validation is the global minimum.

**TABLE 1.** The best validation of proposed algorithm for iris data set.

|  |  |  |
| --- | --- | --- |
| data set | Best validation | No of epochs |
| iris data set | 0.000010816 | 193 |



**FIGURE 2.** Training curve of iris data set.

The proposed algorithm and BP algorithm with adaptive learning rate using MSE results are tabulated in table 2. The best validation and no. of epochs for each algorithm are shown in Fig. 2.

**TABLE 2.** The best validation of proposed algorithm and BP algorithm with learning rate using MSE for different epochs for iris data set.

|  |  |  |  |
| --- | --- | --- | --- |
| Proposed algorithm | | Bp with adaptive learning rate | |
| No. of epochs | Best validation | No. of epochs | Best validation |
| 129 | 0.048213 | 129 | 0.064447 |
| 131 | 0.030339 | 145 | 0.039531 |
| 134 | 0.019246 | 147 | 0.029612 |
| 145 | 0.005174 | 149 | 0.021439 |
| 152 | 0.008609 | 150 | 0.014636 |
| 155 | 0.000738 | 155 | 0.011353 |
| 158 | 0.008071 | 156 | 0.012291 |
| 159 | 0.000151 | 157 | 0.006711 |
| 168 | 0.000227 | 170 | 0.006711 |
| 227 | 0.000398 | 228 | 0.002795 |

From Fig. 3 the convergence rates as best validation and no. of epochs for proposed algorithm are better than the BP algorithm with adaptive learning rate using MSE. It reaches the global minimum at smallest time and lowest number of epochs.

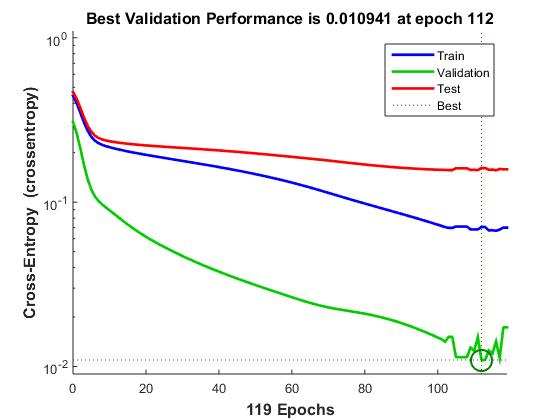
**FIGURE 3.** The relation between no. of epochs and best validation for proposed algorithm and Bp with adaptive learning for iris data set.

**Second: cancer dataset**

This section attempts to build a neural network that distinguishes between cancer and control patients from the mass spectrometry data. There are 216 columns representing 216 patients, out of which 121 are ovarian cancer patients and 95 are normal patients, where there exist 100 specific mass-charge values for each patient. The results of training data set using proposed algorithm are tabulated in table 3. From Fig. 4 the training curve shows stability of algorithm and better convergence rates (i.e. The best validation and no of epochs). The cancer data set is trained by using error BP algorithm and the results tabulated in table 4. From Fig. 5 the results show that the proposed algorithm is more stable, speed convergence than BP algorithm with adaptive learning rate using MSE, and the epoch number is very small. The minimum global error is reached faster without any saturation in the training process. Although the training curve passed through a lot of minimum values (0.0321 at 105 epochs, 0.0201 at 107 epochs, and 0.011 at 110 epochs), all of them are considered local minima but the best validation is the global minimum.

**TABLE 3.** The best validation and number of epochs for cancer data set using proposed algorithm.

|  |  |  |
| --- | --- | --- |
| Data set | Best validation | No of epochs |
| Cancer data set | 0.010941 | 112 |



**FIGURE 4**. Training curve of cancer data set.

**TABLE 4.** The best validation and epoch number for proposed algorithm and BP with adaptive learning rate using MSE.

|  |  |  |  |
| --- | --- | --- | --- |
| Proposed algorithm | | Bp with adaptive learning rate | |
| No of epochs | Best validation | No of epochs | Best validation |
| 101 | 0.07537 | 103 | 0.088464 |
| 102 | 0.070481 | 105 | 0.076529 |
| 106 | 0.065205 | 107 | 0.065932 |
| 107 | 0.053516 | 109 | 0.055226 |
| 115 | 0.047996 | 119 | 0.048154 |
| 116 | 0.047926 | 120 | 0.046196 |
| 122 | 0.036361 | 122 | 0.037543 |
| 128 | 0.026021 | 130 | 0.027295 |
| 140 | 0.009461 | 142 | 0.024786 |

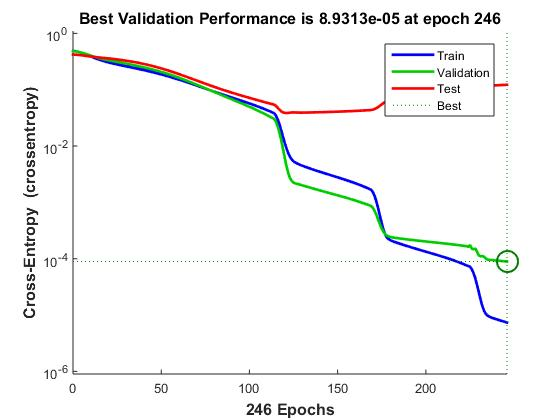
**FIGURE 5.** The relation between no. of epochs and best validation for proposed algorithm and Bp with adaptive learning for cancer data set.

**Third: wine dataset**

This section attempt to build a neural network that can classify wines. It represents 178 wine sample attributes (inputs) and associated winery class vectors (targets). The data set is divided to training, validation, and testing. The results of training data set using the proposed algorithm are tabulated in table 5. From Fig. 6 the training curve begin with large value for cross entropy error. It oscillated stationary until reach to the global minimum at epoch 246.

**Table 5.** The best validation and no of epochs for wine dataset by using proposed algorithm

|  |  |  |
| --- | --- | --- |
| Data set | Best validation | No of epochs |
| wine data set | 8.9313e-05 | 246 |



**FIGURE 6.** training curves of wine data set

The training results of wine data set using proposed algorithm and BP algorithm with adaptive learning rate using MSE are shown in table 6 for different epochs. From figure 8 the convergence rates and satiability of proposed algorithm is better than the BP algorithm with adaptive learning rate using MSE. The convergence of proposed algorithm increased by increasing the training epochs.

**TABLE 6.** Best validation and epoch number for proposed algorithm and BP with adaptive learning rate using MSE

|  |  |  |  |
| --- | --- | --- | --- |
| Proposed algorithm | | Bp with adaptive learning rate | |
| No of epochs | Best validation | No of epochs | Best validation |
| 148 | 0.01422 | 123 | 0.026865 |
| 155 | 0.008815 | 124 | 0.021712 |
| 161 | 0.005245 | 142 | 0.02009 |
| 173 | 0.0048852 | 145 | 0.017274 |
| 181 | 0.0046325 | 179 | 0.012549 |
| 184 | 0.0042745 | 184 | 0.003618 |
| 246 | 0.0014836 | 233 | 0.002371 |
| 247 | 0.0001164 | 243 | 0.002335 |
| 254 | 8.47E-06 | 254 | 0.001525 |
| 325 | 3.91E-07 | 260 | 0.000183 |

**FIGURE 7.** relation between no of epochs and best validation for proposed algorithm and Bp with adaptive lr for wine data set

1. **Conclusion**

There exist a lot of branches using BP algorithm such as image processing and artificial intelligence, but these branches interface some of challenges as slow training, large numbers of iterations, and slow convergence rates. In this paper, the convergence rates increase by using cross-entropy error function with adaptive learning rate and momentum; where best validation and error gradient decrease and so number of epochs. It reaches to the global minima quickly and does not stuck in local minima. The simulation results are recommended that the proposed algorithm is better than the error BP algorithm.

**References**

1. Al-Duais, M. S., Yaakub, A., Yusoff, N., and Ahmed, F. (2015) A Novel Strategy for Speed up Training for Back Propagation Algorithm via Dynamic Adaptive the Weight Training in Artificial Neural Network Feed forward direction. Research Journal of Applied Sciences, Engineering and Technology 9(3), 189-200.
2. Burse, K., Manoria, M., and Kirar, V.P. (2010) Improved back propagation algorithm to avoid local minima in multiplicative neuron model. World Acad. Sci. Eng. Technol., 72, 429-432.
3. Cheung, C.C., Ng, S.C., Lui, A.K., and Xu, S.S. (2010) Enhanced two-phase method in fast learning algorithms. Proceeding of the International Joint Conference on Neural Networks, pp. 1-7.
4. Kotsiopoulos, A.E. and Grapsa, N. (2009) Self-scaled conjugate gradient training algorithms. Neurocomputing, 72, 3000-3019.
5. Ooyen, A.V. and Nienhuis, B. (1992) Improving the learning convergence of the back propagation algorithm. Neural Networks, 5, 465-471.
6. Shao, H.M. and Zheng, G.F. (2009) A new BP algorithm with adaptive momentum for FNNs training. WRI Global Congress on Intelligent Systems (GCIS’09), 4, 16–20.
7. Wang, X.G., Tang, Z., Tamura, H., Ishii, M., and Sun, W.D. (2004) An improved backpropagation algorithm to avoid the local minima problem. Neurocomputing, 56, 455 – 460.
8. Yu, C. and Liu, B. (2002) A backpropagation algorithm with adaptive learning rate and momentum coefficient. Proceeding of Int. Conf. on Neural Networks (IJCNN’02), vol 2, pp 1218-1223.
9. Zhang, N. (2009) An online gradient method with momentum for two-layer feed forward neural networks. q. Appl. Math. Computat, 2, 488-498.
10. Zhixin, S. and Bingqing. (2011) Research of improved back-propagation neural network algorithm. Proceeding of 12th IEEE International Conference on Communication Technology (ICCT), pp: 763-766.