Enhancement Processing Time and Accuracy Training via Significant Parameters in the Batch BP Algorithm

Mohammed Sarhan Al _ Duais
Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Terengganu, Malaysia
E-mail: Sarhan2w@gmail.com

Fatma Susilawati Mohamad
Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Terengganu, Malaysia
E-mail: Fatma@unisza.edu.my

Mumtazimah Mohamad
Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Terengganu, Malaysia
E-mail: Mumtazimah@unisza.edu.my

Mohd Nizam Husen
Malaysian Institute of Information Technology, Universiti Kuala Lumpur (UniKL), Malaysia
E-mail: mnizam@unikl.edu.my

Received: 29 January 2019; Accepted: 22 November 2019; Published: 08 February 2020

Abstract—The batch back procreation algorithm is a new style for weight updating. The drawback of the BBP algorithm is its slow learning rate and easy convergence to the local minimum. The learning rate and momentum factor are the two most significant parameters for increasing the efficiency of the BBP algorithm. We created the dynamic learning rate and dynamic momentum factor for increasing the efficiency of the algorithm. We used several data set for testing the effects of the dynamic learning rate and dynamic momentum factor that we created in this paper. All the experiments for both algorithms were performed on Matlab 2016 a. The stop training was determined ten power -5. The average accuracy training is 0.9909 and average processing time improved of dynamic algorithm is 430 times faster than the BBP algorithm. From the experimental results, the dynamic algorithm provides superior performance in terms of faster training with highest accuracy training compared to the manual algorithm. The dynamic parameters which created in this paper helped the algorithm to escape the local minimum and eliminate training saturation, thereby reducing training time and the number of epochs. The dynamic algorithm was achieving a superior level of performance compared with existing works (latest studies).

I. INTRODUCTION

An artificial neural network (ANN) is a mathematical model inspired by biological node systems [1-2]. This is a description that offers great flexibility in modeling quantitative methods. The ANN has a strong ability to complete tasks many and is considered an active tool for training and classification[3-6]. An ANN provides a supervised learning algorithm which implements a nonlinear model within [0, 1] or [-1, 1], depending on the activation function [7, 8]. An ANN involves parallel processing, which consists of several parameters which need to be adjusted to minimize error training.

Heuristic techniques are the most current methods for improving the training of the BBP algorithm. These methods include learning rate and momentum factor; these are significant parameters for adjusting the weights in training [9-10]. In general, the values of learning rate and momentum factor are fixed and are chosen from the interval [0, 1]. Manually set values for each training rate and momentum factor do not help the BBP algorithm escape from local minima or to meet the requirements of speeding up the training of the BBP algorithm.

Generally, there are two techniques for selecting the values for each learning rate and momentum factor. The first is set to be a small constant value from interval [0,1], the second the selected series value from[0,1], [11,12]. Therefore, one of the requirements for speeding up the BBP algorithm is adaptive learning rate and momentum.
factor together [13,14]. The learning rate should be sufficiently large to allow for escaping the local minimum to facilitate fast convergence to minimize error training. But the biggest value of the training rate leads to fast training with oscillation error training [15,16]. On the contrary, the small value of learning rate leads to reflex the weight that is lead to flat spot which makes BBP algorithm slow training [17,18]. The big values or small values are not suitable for smooth training. It is difficult to select the best or suitable values in training [19, 20].

The heuristic method is a current method for improving the BP algorithm, which covers two parameter training rate and momentum factor. The weaknesses, the values of training, rate and momentum are selected manual values. On the other hand, improves BP algorithm through creating dynamic training rate. But the weaknesses of these dynamic training depend on the initial value of training rate. Manually set values for learning rate and momentum factor do not help the BBP algorithm avoid local minima or meet the requirements of increasing training speed and difficult to choose a suitable value for learning rate and momentum factor. One way of avoiding localminima and speeding up the training of the BBP algorithm is to use a large value for training rate.

However, a small adjustment to the training weight slows the training of the BBP algorithm, while large adjustment to the weight results in unsmooth training. Most previous studies have tried to escape the local minimum through adaptive learning rates or momentum factors to improve the BBP algorithm. The persisting major problem, though, is the accumulation of weight in the BBP, which decreases accuracy and increases the time of training.

To fill the gap, we need to avoid the gross weight training, through creating dynamic parameter each learning rate and momentum factor with boundary to control the weight updated.

The remainder of this paper is organized as follows: Section II, literature review; Section III, theoretically of training batch of back-Propagation algorithm; Section IV materials and method; Section V Created the dynamic learning rate and momentum factor; Section VI Experimental results; Section VII Discussion to validate the performance of improve algorithm; Section VIII Evaluation of the Performance of Improved Batch BP algorithm; Section XI Conlusion.

II. RELATED WORKS

Currently, works have been carried out, such as author, In [21] Improved back propagation Algorithm by adaptive the momentum factor by dynamic function. The dynamic function which proposed is depend on the two manual factors parameters. The learning rate selected by manual value to control the weight updated. Each factor and learning rate selected different value for each data set. To validate, this study through compares the performance of improved algorithm with BP algorithm. The proposed algorithm gave superior training algorithm than other with whole data set. In [22] improved the BP algorithm through two techniques, the training rate and momentum factor, values of training rate were fixed at different values. The idea of this study is to set the value of training, the rate to be large initially, and then to look at the value of error training after iteration. If the error (e) training is increased, the fit produced changes the value of training, rate multiplied by less than one and then recalculated in the original direction. If the iteration error can be reduced, this integration is effective. Therefore, by changing the training rate multiplied by a constant greater than one, the next iteration is calculated continuously.

In [23] compared several techniques such as BP with momentum, BP with the adaptive learning rate, BP with adaptive learning rate and momentum, Polak-Ribikre CGA, Powell-Beale CGA, scaled CGA, resilient BP (RBP) conjugate gradient algorithm (CGA), and Fletcher-Reeves. The epochs of training are fixed with different values. The BP algorithm with adaptive learning rate and momentum gave superior accuracy training at 1000 epochs.

III. THEORETICALLY OF TRAINING BATCH OF BACK-PROPAGATION ALGORITHM

In this section we will focuses on the theoretical of BBP algorithm which cover the mathematical framework with significant parameters such as learning rate and momentum factor, and the gradient descent method. The BBP algorithm was designed to train models to achieve a balance between the actual data and the target correctly in the input patterns; this is done using the gradient descent method to minimize the error training of the BBP algorithm.
A. Mathematical Framework BBP Algorithm with γ and μ

The training BP algorithm is consist of three stages, namely forward propagation, in the feed forward phase, each input unit \( x \) receives an input signal \( x \) and broadcasts this signal to the next layer until the end layer in the system. The backward propagation this step is starting when the output of the last layer reach to end step then the start feedback. The end steep is update the weight, in the batch BP algorithm the weight adjustment stage for all layer adjusted simultaneously. The training of BP algorithm shown in the figure below.

Compute the output of first hidden layer of \( L_n \)

\[
L_n = f(L_{-inh})
\]  
(2)

Send the result to \( LL_j \) ( \( LL_j \), \( j = 1, 2, ..., p \)) and Calculate the input signal

\[
LL_j = v_{th} + \sum_{i=1}^{k} v_{ih} L_i
\]  
(3)

Calculate the output layer of second hidden \( LL \)

\[
LL_j = f(LL_{-inh})
\]  
(4)

Send the output \( LL \) to input \( Y_{inr} \) and then get the input layer of \( Y_{inr} \)

\[
Y_{inr} = w_{th} + \sum_{i=1}^{k} LL_j w_{jr}
\]  
(5)

Compute the output layer of \( Y_r \)

\[
Y_r = f(Y_{inr})
\]

Backward Pass Phase: when the weight reaches to the output of the last hidden layer to end step then the start all steps as follow

\[
e_r = \sum_{i=1}^{n} (t_i - Y_r)
\]  
(6)

The local gradient of output layer \( Y_r \) at neuron \( r \) is defined by below equation

\[
\delta_r = e_r f(Y_{inr}) \cdot f'(Y_{inr}) = Y_{inr} f(1-Y_{inr})
\]  
(7)

Compute weight correction term (update later)

\[
\Delta w_{pr} = -\gamma \delta_r LL_j + \mu \Delta w_{pr} (t-1)
\]  
(8)

Calculate, bias correction term (update \( w_{0r} \) later)

\[
\Delta w_{0r} = -\gamma \delta_r + \mu \Delta w_{0r} (t-1)
\]  
(9)

Send \( \delta_r \) to hidden layer, each hidden unite \( LL_j, \ j = 1, ..., p \)

The single error or error back propagation from units in layer above to given as below

\[
\delta_{-nj} = \sum_{r=1}^{m} \delta_r W_{jr}
\]  
(10)
The local gradient of hidden layer $L_L$ at neuron $j$ is defined by below equation:

$$\delta_j = -\delta_{o_L} f'(LL_{o_L})$$  \hspace{1cm} (11)

Compute weight correction term (update $v_{hj}$ later)

$$\Delta v_{hj} (t + 1) = -\gamma \delta_j L_h + \mu \Delta w_{hj} (t - 1)$$  \hspace{1cm} (12)

Calculate, bias correction (update $v_{ij}$ later)

$$\Delta v_{ij} (t + 1) = -\gamma \delta_i + \mu \Delta w_{ij} (t - 1)$$  \hspace{1cm} (13)

Send $\delta_j$ to first layer $h=1,…,q$.

The single error or error backpropagation is given as below

$$\delta_{-o_h} = \sum_{j=1}^{h} \delta j_{o_d}$$  \hspace{1cm} (14)

The local gradient of hidden layer $L_h$ compute as follow

$$\delta_h = \delta_{-o_h} f(L_{-o_h}), \hspace{1cm} f(Y_{o_h}) = Y_{o_h} f(1-Y_{o_h})$$  \hspace{1cm} (15)

Compute weight correction (update $u_{ah}$ later)

$$\Delta u_{ah} = -\gamma \delta_h x_i + \mu \Delta u_{ah} (t - 1)$$  \hspace{1cm} (16)

Calculate bias weight corrective (used to update $u_{ah}$ later)

$$\Delta u_{ah} = -\gamma \delta_h + \mu \Delta u_{ah} (t - 1)$$  \hspace{1cm} (17)

Update Weight Phase: the weight is update of the output layer $f = (0,1,2,...,p; r = 1,...,m)$ as follows

$$w_{jr} (t + 1) = w(t) - \gamma \Delta w_{jr} (t) + \mu \Delta w_{jr} (t - 1)$$  \hspace{1cm} (18)

For Bias

$$w_{0r} (t + 1) = w(t) - \gamma \delta_j + \mu \Delta w_{0r} (t - 1)$$  \hspace{1cm} (19)

For each hidden layer $(LL_h, h=0,...,q; j=1,...p)$

$$v_{hj} (t + 1) = v_{hj} (t) - \gamma \Delta w_{hj} (t) + \mu \Delta w_{hj} (t - 1)$$  \hspace{1cm} (20)

For Bias

$$v_{0j} (t + 1) = v_{0j} (t) - \gamma \Delta w_{0j} (t) + \mu \Delta w_{0j} (t - 1)$$  \hspace{1cm} (21)

For each hidden layer $L_h (i=0,...,n; h=1,...,q)$

$$u_{ih} (t + 1) = u_{ih} (t) - \gamma \Delta u_{ih} (t) + \mu \Delta u_{ih} (t - 1)$$  \hspace{1cm} (22)

For Bias

$$u_{0h} (t + 1) = u_{0h} (t) - \gamma \Delta u_{0h} (t) + \mu \Delta u_{0h} (t - 1)$$  \hspace{1cm} (23)

The flowchart of the BP algorithm with trail value for each training rate and

B. Gradient Descent Backpropagation Algorithm

The gradient descent method is one of the most popular methods used to adjust the weight for minimizing the training error in the BP algorithm [24, 25]. Unfortunately, the gradient descent method is not guaranteed to find the global error [26, 27]. It may, therefore, result in leading to a local minimum, as the gradient descent method is not powerful enough to adjust for the weight, apart from limited changes [28]. The batch gradient descent method computes gradients using pattern training and updates the weight in the final stages of training [29]. This method focuses on modifying one parameter or more than one as adaptive of activation function or modifying the gain. The value of the gain is seen in the effect of the behavior of the descent algorithm [30]. The author [31] proposed a new algorithm through the modified gain in the activation function. The proposed algorithm gives superior training compared with the BBP an algorithm.

Gradient descent $\Delta E$ taken partial of the differentiate of error with respect the weight $w_{ij}$ in each layer. ANN which is consist one input, one hidden layer and one output also focus for one neuron as $K$ from output layer and one neuron $j$ from hidden layer, as follows

$$\frac{\partial E}{\partial w_{ij}} = \frac{1}{2} \sum_{k=1}^{K} (t_k - o_k)^2 \cdot \frac{\partial f(x)}{\partial x}$$  \hspace{1cm} (24)

Let the error training is given by $E = \frac{1}{2} \sum_{k=1}^{K} (t_k - o_k)^2$, activation function of training is sigmoid function $f(x) = \frac{1}{1 + \exp(-x)}$. When the gain equals is one. The weight update between the neuron $K$, from output and neuron $j$ from hidden layer as follow

Copyright © 2020 MECS
\[
\Delta w_{jk} (new) = \Delta w_{jk} (old) - \gamma \Delta w_{jk} + \mu \Delta v_{ij} (t-1) \quad (24)
\]

The gradient between neuron \( K \) from output layer and neuron \( j \) from hidden layer

\[
\Delta w_{jk} = \frac{\partial E}{\partial W_{jk}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \frac{\partial net_j}{\partial w_{jk}} \quad (25)
\]

\[
\frac{\partial E}{\partial o_j} = 1 \frac{\partial}{\partial o_j} (t_j - o_j)^2
\]

\[
\frac{\partial E}{\partial o_j} = -(t_j - o_j) \quad (26)
\]

\[
\frac{\partial o_j}{\partial net_j} = \frac{\partial}{\partial net_j} \left( \frac{1}{1 + e^{-\theta_j}} \right)
\]

\[
\frac{\partial o_j}{\partial net_j} = o_j (1 - o_j) \quad (27)
\]

\[
\frac{\partial net_j}{\partial w_{jk}} = x_j \quad (28)
\]

Insert equation 25, 26, 27 into equation (25) it becomes as follow

\[
\Delta w_{jk} = -(t_j - o_j) o_j (1 - o_j) x_j \quad (29)
\]

Insert equation (29) into equation (24) to get a new equation for update the weight as below

\[
\Delta w_{jk} (new) = \Delta w_{jk} (old) - \gamma (t_j - o_j) o_j (1 - o_j) x_j \quad (30)
\]

Put the \( \delta_j = (t_j - o_j) o_j (1 - o_j) \) then get the equation

\[
\Delta w_{jk} (new) = w_{jk}(old) + \eta \delta_j x_j \quad (31)
\]

For calculate the local gradient between neuron \( j \) from hidden layer and neuron \( i \) from input layer. In this case the weight update as follow

\[
\Delta w_{ij} (t+1) = w_{ij} (t) - \gamma \Delta w_{ij} (t) \quad (32)
\]

The gradient between neuron \( j \) from hidden layer and neuron \( i \) from input layer given as follow

\[
\frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial net_i}{\partial o_j} \frac{\partial o_j}{\partial net_j} \quad (33)
\]

\[
\frac{\partial E}{\partial net_j} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \quad (34)
\]

\[
\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial o_j} \frac{\partial net_j}{\partial o_j} \frac{\partial o_j}{\partial net_j} \quad (35)
\]

Insert equation (35) into equation (34) it becomes as follow

\[
\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial o_j} \quad (36)
\]

\[
\Delta w_{ij} = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial net_i} \frac{\partial net_i}{\partial o_j} \frac{\partial o_j}{\partial net_j} \quad (37)
\]

\[
\frac{\partial E}{\partial net_j} = -(t_j - o_j) o_j (1 - o_j) \quad (38)
\]

Put \( \delta_k = (t_k - o_k) o_k (1 - o_k) \) then get

\[
\frac{\partial net_j}{\partial o_j} = \frac{\partial}{\partial o_j} \left( \frac{1}{1 + e^{-\theta_j}} \right)
\]

\[
\frac{\partial net_j}{\partial o_j} = \frac{\partial}{\partial net_j} \left( \frac{1}{1 + e^{-\theta_j}} \right)
\]

\[
\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left( \sum_{i=1}^{k} x_i w_{ij} + \theta_j \right)
\]

\[
\frac{\partial net_j}{\partial w_{ij}} = x_i \quad (42)
\]

Insert equation (26, 27, 28) into equation (25) it becomes as follow

\[
\Delta w_{ij} = \frac{\partial E}{\partial W_{ij}} = -\sum_{i=1}^{k} (t_i - o_i) o_i (1 - o_i) w_{ij} \delta_i (1 - \delta_i) x_i
\]

From equation (29) replace \( (t_i - o_i) o_i (1 - o_i) \) by \( \delta_i \)
\[ \Delta w_j = \frac{\partial E}{\partial w_j} = -\sum_{i=1}^{k} \delta_i w_{ji} (1 - \sigma_j) x_i \]  \hspace{1cm} (43)

Put \( \delta_j = \sum_{i=1}^{j} \delta_i w_{ji} (1 - \sigma_j) \) in (42) to get

\[ \Delta w_j = \frac{\partial E}{\partial w_j} = \delta_j x_i \]  \hspace{1cm} (44)

Insert (43) into equation (32), then the weight update between neuron \( j \) from hidden layer and neuron \( i \) from input layer as below

\[ \Delta W_{ij} (\text{new}) = w_{ij} (\text{old}) + \gamma \delta_j x_i \]  \hspace{1cm} (45)

IV. MATERIALS AND METHOD

This kind of this research belongs to the heuristic method. This method includes the learning rate and momentum factor. To investigate the aims of this study there are many steps as follows:

A. Data set

The data set is very important for verification of improved DDBPLM algorithm. In this study, all data are taken from UCI Machine Learning Repository through the link https://archive.ics.uci.edu/ml/index.html. All real data set to change to become nomination data set between [0,1] and divided into two set training set and testing set.

B. Neural Network Model

We propose an ANN model, which is a three-layer neural network that has an input layer, hidden layer, and output layer. The input layer is considered to be \( \{ x_1, x_2, \ldots, x_i \} \), which represents the nodes; the nodes depend on the types or attributes of the data. The hidden layer is made of two layers with four nodes. The output layer \( Y \) is made of one layer with one neuron. Three basis, two of them are used in the hidden layers and one in the output layer. Finally, the sigmoid function is employed as an activation function.

V. CREATED THE DYNAMIC THE PRAMETERS LEARNING RATE AND MOMENTUM FACTOR

The weight update between neuron \( k \) from the output layer and neuron \( j \) from the hidden layer ( \( w_{jk} \) ) in equation (1) which recorded as above.

To enhance the BBP algorithm, which is given by the Equation above (1) to avoid the local minima and to avoid saturation training. The learning rate should be sufficiently large to allow escape from the local minimum and to facilitate fast convergence to minimize error training. However, the value of the learning rate which is too large leads to fast training with an oscillation in error training. To ensure a stable learning BP algorithm, the learning rate must be small. Depend on above, we will create the dynamic training rate with boundary to keep the weight as moderate as follows:

\[ \gamma_{\text{mic}} = \frac{\sec (\theta (1 - \theta))}{2} + \exp (\theta \epsilon) \]  \hspace{1cm} (46)

Where the \( \sec \theta \) is Hyperbolic functions, \( \theta \) is angle, and the \( \sec \theta = 2 / (e^\theta + e^{-\theta}) \). The \( \theta (1 - \theta) \) is the first derivative of the activation function of the output of second hidden layer and \( \epsilon \) is the error training. We can see the value of \( (\theta \epsilon) \) is located between 0 and 1, the value of \( (\theta \epsilon) \) is a boundary. In the same way, the training error \( \epsilon \) is boundary value \( 0 \leq \epsilon \leq 1 \). Each \( \theta \) and \( \epsilon \) boundary function, that is lead to shx is boundary function, \( 0 \leq \theta \leq \epsilon \leq 1 \). Depend on the above we can find the \( 1 \leq \leq 2.78 \) the dynamic \( \gamma_{\text{mic}} \) boundary function.

The second significant parameter is momentum. To get smooth training and avoid inflation in the gross weight of the added values for momentum factor, the fitting producer through creating dynamic momentum factor and implicate the Depend above we can create the dynamic momentum factor \( \mu_{\text{mic}} \) as follow

\[ \mu_{\text{mic}} = \frac{1}{\sec (\theta (1 - \theta)) + \exp (\theta \epsilon)} \]  \hspace{1cm} (47)

We Insert the equation (46) and (47) into equation (1), then the weight is updated between any layer as below

\[ w_{jk}(t+1) = w_{jk}(t) - \frac{\sec (\theta (1 - \theta))}{2} + \exp (\theta \epsilon) \frac{\partial E}{w_{jk}(t)} \]

\[ + \frac{1}{\sec (\theta (1 - \theta)) + \exp (\theta \epsilon)} \Delta w_{jk}(t-1) \]  \hspace{1cm} (48)

For the equation (48) the dynamic algorithm will be updated the weight under affected the dynamic learning rate and dynamic momentum factor.

VI. EXPERIMENTAL RESULTS

The accuracy of training is calculated as follows [32,33]

\[ \text{Accuracy} = \frac{1 - \text{abs} \left( T_{\text{P}} - O_{\text{L}} \right)}{T_{\text{P}} - L_{\text{W}}} \times 100 \]  \hspace{1cm} where \( UP \) and \( LW \) are the upper bound and lower bound of the activation function.

A. Experimental Results for the DDBPLM Algorithm Using the XOR Problem

Ten experiments were carried out using Matlab, and the average (AV) was taken for several criteria used for
measurement of the training performance. Ten experimental has been done for DBBPLM with XOR and then taken the average and S.D. The average and S.D of experimental results are tabulated in Table 1.

Table 1. Average the Performance of DBBPLM algorithm with XOR

<table>
<thead>
<tr>
<th>Value of</th>
<th>times</th>
<th>MSE</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>52.4760</td>
<td>9.999e-05</td>
<td>67490</td>
</tr>
<tr>
<td>0.0001</td>
<td>6530</td>
<td>0.5011</td>
<td>194345</td>
</tr>
<tr>
<td>0.004</td>
<td>5670</td>
<td>1.615e-04</td>
<td>153329</td>
</tr>
<tr>
<td>0.1</td>
<td>55.9070</td>
<td>9.999e-05</td>
<td>76190</td>
</tr>
<tr>
<td>0.2</td>
<td>31.0900</td>
<td>9.997e-05</td>
<td>37839</td>
</tr>
<tr>
<td>Average</td>
<td>2467.8784</td>
<td>1.0033127</td>
<td>105838.6</td>
</tr>
<tr>
<td>S.D</td>
<td>2978.0706</td>
<td>0.2003937</td>
<td>58416.301</td>
</tr>
</tbody>
</table>

From Table 2., the range of the training time is 31.0090 ≤ t ≤ 6530 seconds. We consider the value 31.0090 second as minimum training time and the value 6530 second as maximum training time. The best performance of the BP algorithm is achieved at γ = 0.2 and at μ = 0.2 whereas the training time is 21.0090 seconds, the worst performance of the BP algorithm is achieved at γ = 0.0001 and μ = 0.0001 for first, whereas the training time is 6530 second.

C. Experiments result of the DBBPLM algorithm with Balance-Train set

From Table 1., the formulas proposed in equations (46 and 47) help the back-propagation algorithm to enhance the performance of the training. Whereas the average time training is t = 2.2463 seconds and the epoch is 1681 epoch. The smallest value of standard deviation S.D indicates the scatter of time or error training is very close for every experiment, whereas the S.D = 0.36127166 second. The accuracy rate is 0.9859.

B. Experiment result of the BBP algorithm with XOR Problem

For the dynamic DBBPLM algorithm we run the dynamic algorithm 10 time with XOR problem and then take the average for time, epoch and accuracy rate as follow.

Table 2. Performance of BBP algorithm with XOR

<table>
<thead>
<tr>
<th>Value of</th>
<th>times</th>
<th>MSE</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>52.4760</td>
<td>9.999e-05</td>
<td>67490</td>
</tr>
<tr>
<td>0.0001</td>
<td>6530</td>
<td>0.5011</td>
<td>194345</td>
</tr>
<tr>
<td>0.004</td>
<td>5670</td>
<td>1.615e-04</td>
<td>153329</td>
</tr>
<tr>
<td>0.1</td>
<td>55.9070</td>
<td>9.999e-05</td>
<td>76190</td>
</tr>
<tr>
<td>0.2</td>
<td>31.0900</td>
<td>9.997e-05</td>
<td>37839</td>
</tr>
<tr>
<td>Average</td>
<td>2467.8784</td>
<td>1.0033127</td>
<td>105838.6</td>
</tr>
<tr>
<td>S.D</td>
<td>2978.0706</td>
<td>0.2003937</td>
<td>58416.301</td>
</tr>
</tbody>
</table>

From Table 2., the range of the training time is 31.0090 ≤ t ≤ 6530 seconds. We consider the value 31.0090 second as minimum training time and the value 6530 second as maximum training time. The best performance of the BP algorithm is achieved at γ = 0.2 and at μ = 0.2 whereas the training time is 21.0090 seconds, the worst performance of the BP algorithm is achieved at γ = 0.0001 and μ = 0.0001 for first, whereas the training time is 6530 second.

C. Experiments result of the DBBPLM algorithm with Balance-Train set

We implement the DBBPLM algorithm ten time with the balance data training set and then the average was taken for several criteria. The experiment's result is tabulated in the Table 3.

Table 3. Average the Performance of DBBPLM algorithm with Iris-Train set

<table>
<thead>
<tr>
<th>Value of</th>
<th>times</th>
<th>MSE</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>52.4760</td>
<td>9.999e-05</td>
<td>67490</td>
</tr>
<tr>
<td>0.0001</td>
<td>6530</td>
<td>0.5011</td>
<td>194345</td>
</tr>
<tr>
<td>0.004</td>
<td>5670</td>
<td>1.615e-04</td>
<td>153329</td>
</tr>
<tr>
<td>0.1</td>
<td>55.9070</td>
<td>9.999e-05</td>
<td>76190</td>
</tr>
<tr>
<td>0.2</td>
<td>31.0900</td>
<td>9.997e-05</td>
<td>37839</td>
</tr>
<tr>
<td>Average</td>
<td>2467.8784</td>
<td>1.0033127</td>
<td>105838.6</td>
</tr>
<tr>
<td>S.D</td>
<td>2978.0706</td>
<td>0.2003937</td>
<td>58416.301</td>
</tr>
</tbody>
</table>

From Table 4., the best performance of the BP algorithm was at γ = 0.01 and μ = 0.25 whereas the time training is 61.5210 ≈ 52 second while the best performance of the BP algorithm is achieved at γ = 0.004 and μ = 0.95 whereas the time training is 220.0530 ≈ 220 seconds. In addition, the range of the average training time is 61.5210 ≤ t ≤ 220.0530 seconds.

E. Experiments result of the DBBPLM Algorithm with Balance-Testing set

From Table 5., the average training time is 3.6801 seconds at an average epoch is 88 epoch. The average S.D of time is 0.98730577. The accuracy rate is very highest about 0.9887.
F. Experiments result of the BBP Algorithm with Balance-Testing set

We tested the BBP algorithm with several optimum values. The experiments result recorded in the Table 6., with iris - testing set

From Table 6., the best performance of the BBP algorithm was at $\gamma = 0.1$ and $\mu = 0.1$, whereas the time training is $79.9910 \approx 80$ seconds, while the worst performance of the BBP algorithm is achieved at $\gamma =0.0001$ and $\mu =0.0001$ whereas the time training is $5600$ seconds. The range of the average training time is $79.9910 \leq t \leq 5600$ seconds. The average S.D of time is greater than one.

Table 6. Performance of BBP Algorithms with Iris-Testing set

<table>
<thead>
<tr>
<th>Value of $\gamma$ (times)</th>
<th>$\mu$ (times)</th>
<th>MSE</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.004</td>
<td>79.9910</td>
<td>5.5408e-05</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.0001</td>
<td>5600</td>
<td>0.2049</td>
</tr>
<tr>
<td>0.005</td>
<td>0.09</td>
<td>4868</td>
<td>0.8193</td>
</tr>
<tr>
<td>0.005</td>
<td>0.001</td>
<td>1.422e+3</td>
<td>5.0238e-05</td>
</tr>
<tr>
<td>0.01</td>
<td>0.25</td>
<td>591.4530</td>
<td>5.0937e-05</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>70.0050</td>
<td>5.6064e-05</td>
</tr>
<tr>
<td>Average</td>
<td>2105.2415</td>
<td>0.1707354</td>
<td>27142.167</td>
</tr>
<tr>
<td>S.D</td>
<td>2267.46456</td>
<td>0.2995366</td>
<td>33674.904</td>
</tr>
</tbody>
</table>

G. Experiments result of the DBBPLM Algorithm with Iris-Testing set

Ten experimental has been done for DBBPLM with balance data set and then taken the average and S.D. The average and S.D of experimental results are tabulated in Table 7.

Table 7. Average the Performance of DBBPLM Algorithm with Balance - Test set

<table>
<thead>
<tr>
<th>Time-sec</th>
<th>Epoch</th>
<th>MSE</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>9.9023</td>
<td>77</td>
<td>2.9658e-04</td>
</tr>
<tr>
<td>S.D</td>
<td>0.52140945</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

From Table 7., the experiments result indicates the dynamic learning rate and momentum factor helps the BBP algorithm for reducing the time training. The average time is $9.9023 \approx 10$ seconds with 77 epoch. The average S.D of time is $0.52140945$.

H. Experiments result of the BP Algorithm with Balance - Train set

We test the manual algorithm at several values for each learning rate and momentum factor. The result are recoded in the Table 8.

From Table 8., the range of the training time is $76.7300 \leq t \leq 105.5750$ seconds, the range of training time is very widely.

I. Experiments result of the DBBPLM Algorithm with iris-Testing set

Ten experimental has been done for DBBPLM with Balance - Testing set and then taken the average and S.D. The average and S.D of experimental results are tabulated in Table 9.

Table 9. Average the Performance of DBBPLM Algorithm with Balance - Test set

<table>
<thead>
<tr>
<th>Time-sec</th>
<th>Epoch</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>6.482</td>
<td>76</td>
</tr>
<tr>
<td>S.D</td>
<td>0.4571277</td>
<td>0</td>
</tr>
</tbody>
</table>

From Table 9., the average time is $6.482$ 6 seconds with 76 epochs. The accuracy rate is $0.9860$, the accuracy rate is the nearest one

J. Experiments result of the manually BBP Algorithm with Iris-Testing set

We test the manual algorithm at several optimum values for each learning rate and momentum factor. The result are recoded in the Table 10.

Table 10. Performance of BBP Algorithm with Balance - Test set

<table>
<thead>
<tr>
<th>Value of $\gamma$ (times)</th>
<th>$\mu$ (times)</th>
<th>Time-S</th>
<th>MSE</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.004</td>
<td>81.9550</td>
<td>2.772e-4</td>
<td>11490</td>
</tr>
<tr>
<td>.0001</td>
<td>0.0001</td>
<td>4976</td>
<td>0.0077</td>
<td>369071</td>
</tr>
<tr>
<td>0.004</td>
<td>0.95</td>
<td>3693</td>
<td>0.0077</td>
<td>152912</td>
</tr>
<tr>
<td>0.005</td>
<td>0.09</td>
<td>3546</td>
<td>5.35e-04</td>
<td>129706</td>
</tr>
<tr>
<td>0.005</td>
<td>0.001</td>
<td>1.97e+03</td>
<td>2.68e-04</td>
<td>273293</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>94.9780</td>
<td>2.76e-04</td>
<td>9833</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2</td>
<td>75.6650</td>
<td>2.979e-04</td>
<td>5204</td>
</tr>
<tr>
<td>Average</td>
<td>2062.514</td>
<td>0.0024363</td>
<td>135929</td>
<td></td>
</tr>
<tr>
<td>S.D</td>
<td>1893.50603</td>
<td>0.0033302</td>
<td>132031</td>
<td></td>
</tr>
</tbody>
</table>
From Table 10., the range of the average training time is in the interval $75.6650 \leq t \leq 4976$ seconds this means the range of time training is wide. The best performance of the BP algorithm is achieved at $\gamma = 0.2$ and $\mu = 0.2$ the worst performance of the BP algorithm is achieved at $\gamma = 0.0001$ and $\mu = 0.0001$. The BBP algorithm suffers the highest saturation at $\gamma = 0.0001$ and $\mu = 0.0001$.

VII. DISCUSSION TO VALIDATE THE PERFORMANCE OF IMPROVED ALGORITHM

To validate the efficiency of the improved algorithm, through comparing the performance of the DBBPLM algorithm with the performance of the batch BP algorithm based on certain criteria[34, 35]. We calculate the speed up training using the following formula [36].

$$\text{Processing Time} = \frac{\text{Execution time of BBP algorithm}}{\text{Execution time of DBBPLM algorithm}}$$

In this part we will compare the performance for each DBBPLM dynamic algorithm which listed in the from Table 1, 3, 5, 7, 9 with the performance of manually algorithm which listed in the Tables 2, 4, 6, 8, 10. The comparison result are tabulated in Table 11.

<table>
<thead>
<tr>
<th></th>
<th>Dynamic (DBBPLM) algorithm</th>
<th>A manually BP algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AV Time</td>
<td>AV Epoch</td>
</tr>
<tr>
<td>XOR</td>
<td>2.2463</td>
<td>1681</td>
</tr>
<tr>
<td>Balance Training</td>
<td>9.9023</td>
<td>77</td>
</tr>
<tr>
<td>Balance Testing</td>
<td>6.482</td>
<td>76</td>
</tr>
<tr>
<td>Iris Training</td>
<td>3.3663</td>
<td>61</td>
</tr>
<tr>
<td>Iris Testing</td>
<td>3.6801</td>
<td>88</td>
</tr>
</tbody>
</table>

From Table 11., it is evident that the dynamic algorithm provides superior performance over the Manually algorithm for all datasets. The range of the training time of the Dynamic algorithm is $2.2463 \leq t \leq 9.9023$ s; this is a narrow interval, meaning that the dynamic parameters which created help the algorithm to reaches the global minimum in a short time and with few epochs. The range of training times of the BBP algorithm is $129.157 \leq t \leq 2105$ s; this is a wide interval, meaning that the BBP algorithm has a long training time and a high level of training saturation. The dynamic algorithm is $2360.166 \approx 2360$ times faster than the BBP algorithm at its maximum.

Easily we can see in the end Column in the table above, dynamic algorithm which created in this paper gave faster training than manual algorithm. For example the dynamic algorithm is $1098.641499 \approx 1099$ time faster than manual algorithm at its maximum time training. Also dynamic algorithm which created in this paper gave faster training than manually algorithm. For example the dynamic algorithm is $38.367644 \approx 38$ time faster than manual algorithm at its minimum time training.

From Figure 3 Obviously, we can see the dynamic algorithm taken a short time to reach the global minimum, for all data set while the Manual algorithm takes a long time to reach a global minimum. The dynamic algorithm created in this study is able to escape the local minimum and remove the saturation training.

Fig.3. The performance of Dynamic algorithm versus Manually algorithm for time.
The performance of Dynamic algorithm

From Figure 4 it is evident that the dynamic algorithm which created in this study provides superior performance over the Manually algorithm for all datasets. The dynamic algorithm taken few epochs to reach to the global minimum while the Manual algorithm takes a big value of epochs to reach the global minimum.

VIII. EVALUATION OF THE PERFORMANCE OF IMPROVED BATCH BP ALGORITHM

To evaluated the performances of the improved batch BP algorithm for speeding up training the performances of the improved batch BP algorithm are compared to previous research works such, Hameed et al (2016) and Azami, (2015), the stop training for both study determined by number of iteration, the number iteration set by 2000 iteration and 1000 iteration respectively. For the purpose of the comparison between the results of this study and the previous works, the fit is re-run again with different stop training values according to previous work. The performance of the improved algorithm which proposed in this study give superior performance than exists works.

IX. CONCLUSION

We have introduced the dynamic DBBPLM algorithm, which trains by creating a dynamic function for each the learning rate and momentum factor. Several dataset were used, all data are taken from the UCI Machine Learning Repository through the following link: http://mir.cs.umass.edu/ml/dataset.html. All datasets should be changed to be normalized in the interval [0,1]. The dataset was divided into two sets, a training set and a testing set for validating the performance of the contribution. This function influences for the weight. Of the DBBPLM algorithm were carried out on Matlab 2012a with same goal error or limited error training. One of the main advantages of the dynamic training and dynamic momentum factor are reduces the training time, the training error, number of epochs and enhancement the accuracy of the training. The performance of dynamic DBBPLM algorithm which presented in this study gave superior performance compared with exists work.

REFERENCES


Enhancement Processing Time and Accuracy Training via Significant Parameters in the Batch BP Algorithm


Authors’ Profiles

Fatma Susilawati Mohamad is an Associate Professor in the Department of Information Technology, Faculty of Informatics and Computing, UniSZA. She services 18 years in academic field and has been actively involved in teaching, research, publications, professional services, consulting, and administration. She was also been appointed for holding various administrative positions in faculty and university level. Now she serves as a Deputy Dean at the Graduate School, UniSZA. She has graduated a Bachelor of Science (Information System) from Oklahoma City University, U.S.A in 1997. She has continued his studies at Universiti Kebangsaan Malaysia and obtained her Master of Information Technology (Computer Science) in 2004. She received his Ph.D. in Computer Science from Universiti Teknologi Malaysia in 2012. During her tenure as an academic staff, she has taught various courses for undergraduate studies.

Copyright © 2020 MECS

IJ. Intelligent Systems and Applications, 2020, 1, 43-54
(Diploma and Bachelor Degree). Her experience gained not only from the aspect of academic, but also from the aspect of research, leadership, management and administration. She has published in more than 30 articles in various conferences and refereed journals where most of her publications came from Scopus and ISI indexed journals. She has led several research grants and her work is recognized in both national and international level. She was also invited to give talk in several conferences and technical workshops national and internationally. Currently, she has more than 10 postgraduate students under supervision where 5 of them were already graduated. Her specialization is in Computer Science focusing on Pattern Recognition and Image Processing.

Associate Professor Dr Fatma Susilawati Mohamad, Department of Information Technology, Faculty of Informatics and Computing Universiti Sultan Zainal Abidin (UniSZA), 21300 Gong Badak, Kuala Nerus, Terengganu, Telephone: 09-6688796, E-mail: fatma@unisza.edu.my, Website: http://fik.unisza.edu.my/fatma

Mumtazimah Mohamad completed her B.Sc in Information Technology (2000) from University Kebangsaan Malaysia, Malaysia, and her M.Sc. in Software Engineering (2005) from University Putra Malaysia. She received her Ph.D (2014) from Universiti Malaysia Terengganu in Computer Science. She joined Kolej Ugama Sultan Zainal Abidin in 2000 as a lecturer. She then joins Universiti Sultan Zainal Abidin, Terengganu, Malaysia in 2006, where she is currently employed as the Deputy Dean for Academic & Postgraduate since

Mohammed Sarhan completed his B.S and Master from Sana’a university. He received her Ph.D (2018) from Universiti Sultan Zainal Abidin, Terengganu, Malaysia in Computer Science. The field is artificial intelligence more specially, improve the training of back propagation algorithm. He is an author in several scientific papers and publications in the field of Arterial intelligence. He is a reviewer for several journals which index under ISI journal or Scopus such as Engineering Applications of Artificial Intelligence, Neural Processing Letters (NEPL), Journal of Electrical Systems and Information Technology International Journal of Electrical and Computer Engineering (IJEECE) and 2018 IEEE International Conference on Automatic Control and Intelligent Systems and many others. E-mail:Sarhan2w@gmail.com.

Mohd Nizam Husen received the B.S. degree in computing from University of Portsmouth, United Kingdom, in 1997 and the M.S. degree in information technology from Universiti Utara Malaysia, Malaysia, in 2007. He received the Ph.D. degree in computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2017. From 1997 to 2008, he was a Lecturer with the Universiti Kuala Lumpur (UniKL), Malaysia. In 2008, he was promoted as a Senior Lecturer at the same institution. He is currently the Deputy Dean (Academic & Technology) at UniKL. He is the author of more than 30 research articles. His research interests include intelligent systems, visual attention, pattern recognition, and indoor localization. He is also a Reviewer of the IEEE Communications Letters and Springer Mobile Networks and Applications journal and many others.

How to cite this paper: Mohammed Sarhan Al Duais, Fatma Susilawati Mohamad, Mumtazimah Mohamad, Mohd Nizam Husen, "Enhancement Processing Time and Accuracy Training via Significant Parameters in the Batch BP Algorithm", International Journal of Intelligent Systems and Applications(IJISA), Vol.12, No.1, pp.43-54, 2020. DOI: 10.5815/ijisa.2020.01.05