Research Journal of Applied Sciences, Engineering and Technology 9(3): 189-200, 2015
DOI:10.19026/rjaset.9.1394
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: June 25, 2014 Accepted: July 19, 2014 Published: January 25, 2015

**Research Article**

# A Novel Strategy for Speed up Training for Back Propagation Algorithm via Dynamic Adaptive the Weight Training in Artificial Neural Network

Mohameed Sarhan Al_Duais, AbdRazak Yaakub, Nooraini Yusoff and Faudziah Ahmed
Department of Computer Science, University Utara Malaysia, 06010 Sintok, Kedah, Malaysia

**Abstract:** The drawback of the Back Propagation (BP) algorithm is slow training and easily convergence to the local minimum and suffers from saturation training. To overcome those problems, we created a new dynamic function for each training rate and momentum term. In this study, we presented the (BPDRM) algorithm, which training with dynamic training rate and momentum term. Also in this study, a new strategy is proposed, which consists of multiple steps to avoid inflation in the gross weight when adding each training rate and momentum term as a dynamic function. In this proposed strategy, fitting is done by making a relationship between the dynamic training rate and the dynamic momentum. As a result, this study placed an implicit dynamic momentum term in the dynamic training rate. This $\alpha_{dmic} = f\left(\frac{1}{\eta_{dmic}}\right)$. This procedure kept the weights as moderate as possible (not to small or too large). The 2-dimensional XOR problem and buba data were used as benchmarks for testing the effects of the 'new strategy'. All experiments were performed on Matlab software (2012a). From the experiment's results, it is evident that the dynamic BPDRM algorithm provides a superior performance in terms of training and it provides faster training compared to the (BP) algorithm at same limited error.

**Keywords:** Artificial neural network, dynamic back propagation algorithm, dynamic momentum term, dynamic training rate, speed up training

## INTRODUCTION

The Back Propagation (BP) algorithm is commonly used in robotics, automation and Global positioning System (GPS) (Thiang and Pangaldus, 2009; Tieding *et al*., 2009). The BP algorithm is used successfully in neural network training with a multilayer feed forward (Bassil, 2012, Abdulkadir *et al*., 2012, Kwan *et al*., 2013, Shao and Zheng, 2009). The back propagation algorithm led to a tremendous breakthrough in the application of multilayer perceptions (Moalem and Ayoughi, 2010, Oh and Lee, 1995). It has been applied successfully in applications in many areas and it has an efficient training algorithm for multilayer perception (Iranmanesh and Mahdavi, 2009). Gradient descent is commonly used to adjust the weight through the change training errors, but the gradient descent is not guaranteed to find the global minimum error, because training is slow and converges easily to the local minimum (Kotsiopoulos and Grapsa, 2009, Nand *et al*., 2012, Shao and Zheng, 2009, Zhang, 2010). The main problem of the *BP* algorithm is slow training; it needs a long learning time to obtain the result (Scanzio *et al*., 2010). However, stuck at a local minimum when $O_r$, the output training of hidden layers and $O_r$, the output training of output layer, extremely approaches 1 or 0 (Dai and Liu, 2012, Shao and Zheng, 2009, Zakaria *et al*., 2010).

To overcome this problem, there are techniques for increasing the learning speed of the BP algorithm or escaping the local minimum, such as the flat spots method, the gradient descent method through magnifying the slope, or changing the value of gain in the activation function, respectively. In addition, the heuristics approach is one of them, which focuses on the parameter training rate and momentum term. In this study, we propose a dynamic function for each training rate and momentum term.

However, this problem has been discussed thoroughly by many researchers. More specifically, to give the BP algorithm faster convergence through modifying it by using some parameter as a modified gain in the sigmoid function in back propagation Zhang *et al*. (2008). In addition, the $\Delta w_{jk}$ is affected by the slope value. The small value of the slope makes back propagation very slow during training. In addition, the large value of the slope may make it faster in training. The value of the gain and momentum parameter directly influences the slope of the activation function, so Nawi *et al*. (2011), adapts each parameter gain and momentum to remove the saturation, but (Oh and Lee,

**Corresponding Author:** Mohammed Sarhan, Al_Duais, Department of Computer Sciences, University Utara Malaysia, 06010 Sintok, Kedah, Malaysia

1995), focuses on magnifying the slope. The objectives of this study involve improving the speed of training of the back propagation algorithm through adapting each training rate and momentum by using a dynamic function.

Current work for solving the slow training back propagation algorithm is through adaptation of a parameter (e.g., training rate and the momentum term), which controls the weight of the adjustment along the descent direction (Iranmanesh and Mahdavi, 2009), Asaduzzaman *at el*., 2009). Improving the speed of the back propagation algorithm through adapting each training rate and momentum by dynamic function Xiaozhong and Qiu (2008) has improved the back propagation algorithm by adapting the momentum term. For a new algorithm tested by XOR -2 dimensions, the experiment results demonstrated that the new algorithm is better than the BP algorithm. Burse *et al*., (2010) proposed a new method for avoiding the local minimum by adding the momentum term and PF term. Shao and Zheng (2009) proposed new algorithm, PBP, is based on adaptive momentum. The simulation result has shown that the new algorithm has faster convergence and smoothing oscillation. Zhixin and Bingqing (2010) have improved the back propagation algorithm has improved based on the adaptive momentum term. A new algorithm was tested using the 2-dimensional XOR. The simulation results show that the new algorithm is better than the BP algorithm. On the other hand, some studies focus on the adaptive training rate Latifi and Amiri (2011) presented in a novel method based on adapting the variable steep learning rate to increase the convergence speed of the EBP algorithm. The proposed convergence is faster than the back propagation algorithm. Gong (2009) proposed a novel algorithm (NBPNN) beside this is on the self-adaptive training rate. From the experiment results, the NBPNN gives more accurate results than the others. Iranmanesh and Mahdavi, (2009) proposed different training rate for different location for output layer. Yang and Xu (2009) have proposed to modify the training rate by a math formula based on a two-step function. From the experiment results, the new algorithm gives a superior performance compared to the back propagation algorithm. Al-Duais *et al*. (2013) improved BP algorithm by created the mathematic formula of the training rate. The experiments results show that the Dynamic BP algorithm gave a faster training rate than the BP algorithm.

## MATERIALS AND METHODS

This kind of this research belong the heuristic method. Heuristic method included two parameter such training rate and momentum term. This study will be

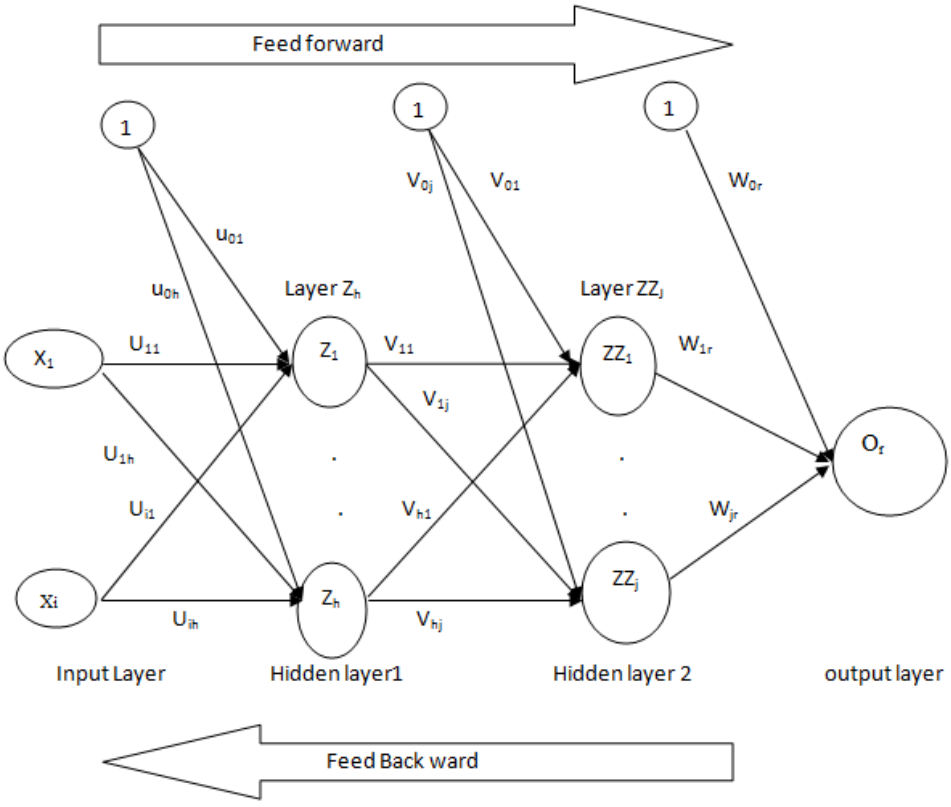


Fig. 1: Training of back propagation

creating dynamic function for each training rate and momentum term to increase speeding up back propgation algorithm. There are many steps which appear in follows:

## NEURAL NETWORKS MODEL

In this section, we will propose the ANN model, which consists of a three-layer neural network composed of an input layer, a hidden layer and an output layer. The input layer is considered as $\{x_1, x_2, ..., x_i\}$ nodes, which depends on the kind or attribute of the data. The hidden layer is made of two layers with four nodes. The output layer is made of one layer with one neuron. Of the three biases, two are used in the hidden layers and one in the output layer, denoted by $u_{0j}$, $v_{0k}$ and $w_{0r}$. Finally, the sigmoid function is employed as an activation function, which is linear for the output layer in (Hamid *et al*., 2012). The proposed neural network can be defined as {I, T, W, A}, where, I denotes the set of input nodes and T denotes the topology of NN, which covers the number of hidden layers and the number of neurons. $W_{jr}$ denoted the set of weight and A, denoted by the activation function as Fig. 1.

Before presenting the BPDRM algorithm, let us bri efly define some of the notations used in the algorithm as follows:

| | | |
|---|---|---|
| $Z_h$ | : | First hidden layer for neuron h, h = 1, …, q |
| $ZZ_r$ | : | Second hidden layer for neuron j, j = 1,..., p |
| $O_r$ | : | Output layer for neuron r |
| $u_{ih}$ | : | The weight between neuron i in the input layer and neuron h in the hidden layer |
| $u_{0h}$ | : | The weight of the bias for neuron j |
| $v_{hj}$ | : | The weight between neuron h from hidden layer z and neuron j from the hidden layer ZZ |
| $v_{0j}$ | : | The weight of the bias for neuron j |
| $w_{jr}$ | : | The weight between neuron k from the hidden layer ZZ and neuron r from the output layer O |
| $w_{0r}$ | : | The weight of the bias for neuron r from the output layer |
| $\Delta w$ | : | The difference between the current and new value in the next iteration |
| $\eta$ | : | The manual of training rate |
| $\alpha$ | : | The manual of momentum term |
| $\eta_{dmic}$ | : | The dynamic training rate |
| $\alpha_{dmic}$ | : | The dynamic momentum term |
| $\acute{f}(O_r)$ | : | Differential of activation functions for output layer $O_r$ at neuron r |
| $\varepsilon$ | : | Absolute value |
| e | : | Error training |
| 1.0 E-n | : | 1 power -n, n = 1, …, i, $\forall$ i $\in$ N |
| $\delta_r$ | : | The error back propagation at neuron r |
| $\delta_j$ | : | The error back propagation at neuron j |

## CREATING THE DYNAMIC FUNCTIONS FOR THE TRAINING RATE AND MOMENTUM TERM

One way to escape the local minimum and save training time in the BP algorithm is by using a large value of $\eta$ in the first training instance. On the contrary, the small value of $\eta$ leads to slow training (Huang, 2007). In the BP algorithm, the training rate is selected by depending on experience and a trail value between (0, 1) in (Li *et al*., 2010, 2009). Despite this, there are studies that have proposed techniques to increase the value of $\eta$ to speed up the BP algorithm through creating a dynamic function. However, the increasing value of $\eta$ becomes too large; it leads to oscillated output training in (Negnevitsky, 2005). Even a large value of $\eta$ is unlikely for the training BP algorithm. The weight update between neuron k from the output layer and neuron j from the hidden layer is as follows:

$$\Delta w_{jk}(t+1) = w_{jk}(t) - \eta \frac{\partial E}{\partial W_{jk}(t)} + \alpha \Delta w(t-1) \quad (1)$$

where, the $\Delta w_{jk}(t)$ changes, the weight is updated for each epoch from equation 1, slow training or fast depends on some parameter, which affects updating the weight. The key for the convergence of the error training is monotonicity function in (Zhang, 2009). Many studies adapt the training rate and momentum by using a monotonicity function such as (Shao and Zheng, 2009, Yang and Xu, 2009), used exponentially to increase the speed of the BP algorithm. The exponential function is a monotonic function. We propose a dynamic training rate as follows:

$$\eta_{dmic} = \frac{\sin(1-o_r)}{1-f'(o_r)} + \left| \tan(\log(\varepsilon e)) \right| \quad (2)$$

Substituting $\alpha_{dmic}$ from Eq. (2) into Eq. (1) to obtain:

$$\Delta w_{jk}(t+1) = w_{jk}(t) - [\frac{\sin(1-o_r)}{1-f'(o_r)} + \left|\tan(\log(\varepsilon e))\right|] \Delta w_{jk}(t) \quad (3)$$

Alternatively, we can extend the Eq. (1) by adding a momentum term to become as follows:

$$w_{jk}(t+1) = w_{jk}(t) - \eta \Delta w_{jk}(t) + \alpha \Delta w_{jk}(t-1) \quad (4)$$

In the back propagation algorithm, the value of the momentum term and training rate are selected as a trial value from the interval [0, 1] or $0<\alpha\leq 1$.

In this study, we proposed a new strategy, which consists of two steps to avoid inflation in the gross

weight when added for each training rate and momentum term as a dynamic function. We proposed a new strategy to avoid the gross weight of the fitting producer by creating a relationship between the dynamic training rate and the dynamic momentum, so we placed an implicit momentum function in the training rate $\alpha_{dmic} = f(\eta_{dmic})$, which was defined as the implicit training rate proposed in Eq. 2. From the previous decoction, we can propose the dynamic function of the momentum term as follows:

$$\alpha_{dmic} = \frac{1}{\eta_{dmic}} \tag{5}$$

From Eq. 5 we see the relationship between $\alpha_{dmic}$ and $\eta_{dmic}$ are inverse. By having this the weight is moderator (no large value, no small value) for avoid the gross the weight or according the overshooting of training. Substituting $\eta_{dmic}$ from Eq. (2) into Eq. (5), the dynamic of the momentum term is defended by Eq. (6) as follows:

$$\alpha_{dmic} = \frac{1}{\frac{\sin(1\text{-}o_r)}{1\text{-}f'(o_r)} + \left|\tan(\log(\varepsilon e))\right|} \tag{6}$$

The value of dynamic of $\alpha_{dmic}$ is located (0, 1) for epoch. The small value of $\alpha_{dmic}$ avoids the gross weight for each equation (25, 26, 27, 28, 29, 30), while the weight is updated.

## BACK PROPAGATION WITH DYNAMIC TRAINING RATE AND MOMENTUM (BPDRM) ALGORITHM

The back propagation algorithm, BP, is trained with a trial value of the training rate between a range of $0<\eta\leq 1$ *l* and $0<\alpha\leq 1$. Many techniques for enhancing the BP algorithm neglect speeding up the training, using flat-spot, gradient descent and the heuristics technique, which include the training rate and the momentum term. The weight update for every epoch or iteration in the new algorithm BPDRM between any neurons {*j*, *k*, …, *r*} from any hidden layer or output layer is as follows:

**Forward propagation:** In the feed forward phase, each input unit $x_i$ receives an input signal $x_i$ and broadcasts this signal to the next layer until the end layer in the system.

Equation 6 indicted the update to the weight for a new algorithm that we denote as BPDRM. The best value of the ε at ε = 0.0042:

$$z_{-inh} = u_{oh} + \sum_{i=1}^{n} x_i u_{ih} \tag{7}$$

Then, each hidden unit computes its activation to get the signal $Z_h$:

$$z_h = f(z_{-inp}) \tag{8}$$

It then sends its output signal to all the units in the second hidden layer. And each hidden unit ($zz_j$ $j$ = 1, 2, …, $p$) calculates the input signal:

$$ZZ_{-inj} = v_{0j} + \sum_{i=1}^{i} z_h v_{hj} \tag{9}$$

It also calculates the output layer of hidden *zz*:

$$zz_j = f(zz_{-in_j}) \tag{10}$$

It sends out layer *zz* to output layer $o_r$ then calculates the input layer for the out layer:

$$O_{-inr}$$

$$O_{-inr} = w_{0r+} \sum_{j=1}^{p} zz_j w_{jr} \tag{11}$$

Finally, it computes the output layer signal:

$$o_r = f(o_{-inr}) \tag{12}$$

**Backward propagation:** This step starts when the output of the last hidden layer or feed forward reaches the end step then starts the feedback that is obvious in Fig. 1. The information provides feedback to allow the adjustment of the connecting weights between each layer. The goal of the BP is to get the minimum error training between the desired output and actual data, as Eq. (13):

$$e_r = \sum_{r=1}^{n} (t_r - o_r) \tag{13}$$

Calculate the local gradient for an output derivative of the activation function of $O_r$ to get:

$$\delta_r = e_r f'(o_{-inr}) \ , \ f'(o_{-inr}) = o_{-inr}(1 - o_{-inr}) \tag{14}$$

Calculates the weight correction term (used to update $w_{jr}$ latter):

$$\Delta w_{jr} = -[\ \frac{\sin(1\text{-}o_r)}{1\text{-}f'(o_r)} + |\ \tan(\log(\varepsilon e))\ |]\delta_r zz_j + [\frac{1}{\frac{\sin(1\text{-}o_r)}{1\text{-}f'(o_r)} + |\ \tan(\log(\varepsilon e))\ |}]\Delta w_{jr}(t-1) \tag{15}$$

Calculate, the bias correction term (used to update the news $w_{0r}$:

$$\Delta w_{0r} = -[\ \frac{\sin(1\text{-}o_r)}{1\text{-}f'(o_r)} + |\ \tan(\log(\varepsilon e))\ |]\delta_r + [\frac{1}{\frac{\sin(1\text{-}o_r)}{1\text{-}f'(o_r)} + |\ \tan(\log(\varepsilon e))\ |}]\Delta w_{0r}(t-1) \tag{16}$$

And then sends $\delta_r$ to hidden units ach hidden unite ($zz_j, j = 1, \ldots, p$)
Sums weighted input from the units in the layer above to get:

$$\delta_{-inj} = \sum_{r=1}^{m} \delta_r w_{jr} \tag{17}$$

Calculate the local gradient for hidden layer ($zz_j$) to get:

$$\delta_j = \delta_{-inj} f'(zz\text{-}_{-inj}) \tag{18}$$

Calculate weight correction term (used to update the news $v_{hj}$):

$$v_{hj} = -[\frac{1}{\frac{\sin(1\text{-}o_r)}{1\text{-}f'(o_r)} + |\tan(\log(\varepsilon e))\ |}]\Delta v_{hj}(t) + [\frac{\sin(1\text{-}o_r)}{1\text{-}f'(o_r)} + |\ \tan(\log(\varepsilon e))\ |]\Delta v_{hj}(t-1) \tag{19}$$

Calculates the bias collection term (used to update $v_{0_j}$ newest):

$$\Delta v_{0j} = -[\ \frac{\sin(1\text{-}o_r)}{1\text{-}f(o_r)} + |\ \tan(\log(\varepsilon e))\ |]\delta_j + [\frac{1}{\frac{\sin(1\text{-}o_r)}{1\text{-}f(o_r)} + |\tan(\log(\varepsilon e))\ |}]\Delta v_{0j}(t-1) \tag{20}$$

It then sends $\delta_j$ to hidden unit, each hidden unit's ($Z_h$ h = 1, …, q) sum is the weighted input from the unit in the layer above and gets:

$$\delta_{-inh} = \sum_{j=1}^{b} \delta_j v_{hl} \tag{21}$$

Calculate the local gradient of hidden layer $z_h$ (expressed in terms of $x_i$):

$$\delta_h = \delta_{-inh} f'(z_{-inh}) \ , \ f'(z_{-inh}) = z_{-inh}(1 - z_{-inh}) \tag{22}$$

Calculates the weight correction (update $u_{ih}$ newest):

$$\Delta u_{ih} = -[\ \frac{\sin(1\text{-}o_r)}{1\text{-}f'(o_r)} + |\ \tan(\log(\varepsilon e))\ |]\delta_h x_i + [\frac{1}{\frac{\sin(1\text{-}o_r)}{1\text{-}f'(o_r)} + |\tan(\log(\varepsilon e))\ |}]\Delta u_{ih}(t-1) \tag{23}$$

Calculates the bias weight corrective term (used to update the news $u_{0h}$):

$$\Delta u_{0h} = -[\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|]\delta_h + [\frac{1}{\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|}]\Delta u_{0h}(t-1) \quad (24)$$

**Update the weight:** The weight adjustment stage for all the layers are adjusted simultaneously. The adjustment of the weight is based on the above calculated factor in this cases the formal of update the weight is given by as below:
For each output layer (j = 0, 1, 2, … p; r = 1…, m):
The weight update for every layer according of the equations below:

$$w_{jk}(t+1) = w_{jk}(t) + (-\eta)\Delta w_{jk}(t) + \alpha \Delta w_{jk}(t-1)$$

Then the weight update dynamically for every layer under effect of the Eq. (2) and (6), as follows:

$$W_{jr}(t+1) = w_{jr}(t) + [\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|]\delta_r z_j + [\frac{1}{\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|}]\Delta w_{jr}(t-1) \quad (25)$$

For the bias:

$$W_{0r}(t+1) = w_{jr}(t) + [\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|]\delta_r + [\frac{1}{\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|}]\Delta w_{0r}(t-1) \quad (26)$$

$$v_{hj}(t+1) = v_{hj}(t) + [\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|]\delta_j z_h + [\frac{1}{\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|}]\Delta v_{hj}(t-1) \quad (27)$$

For the bias:

$$v_{0j}(t+1) = v_{0j}(t) + [\frac{1}{\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|}]\delta_j + [\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|]\Delta v_{0j}(t-1) \quad (28)$$

For each hidden layer, $Z_h$ ( $i = 0,\ldots,n$ ; $h = 1,\ldots,q$ ) :

$$u_{ih}(t+1) = u_{ih}(t) + [\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|]\delta_h x_i + [\frac{1}{\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|}]\Delta u_{ih}(t+1) \quad (29)$$

For the bias:

$$u_{0h}(t+1) = u_{0h}(t) + [\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|]\delta_h + [\frac{1}{\frac{\sin(1-o_r)}{1-f'(o_r)} + |\tan(\log(\varepsilon e))|}]\Delta u_{0h}(t-1) \quad (30)$$

## IMPLEMENTATION OF THE BPDRM ALGORITHM WITH XOR-2BIT AND BUBA DATE SET

In this section, we implement the Dynamic BPDRM algorithm with the XOR problem is famous use of training in BP algorithm. XOR problem gives the response true if exactly one of them in put value is true otherwise the response is false. XOR problem it has two input with four patterns. Also buba data is famous data which consist 6 inputs with 345 patterns. In this case, the structure of the BP and BPDRM algorithm is 2:2:1 with XOR problem. However, the structure of the BPDRM algorithm and BP algorithm is 6:2:1 with buba data.

**Steps of implementation for the BPDRM algorithm:**

**Step 0:** For XOR problem, read initialize weight $W_1$ = [0.5-0.1; -0.3 0.2], $W_2$= [0.3-0.5; -0.4 0.3] $W_3$ = [0.3; 0.7], $b_1$ = [0.1; -0.2], $b_2$ = [-0.3; -0.2], $b_3$ = [0.4].

For buba data set, read weight W1 = rand (2, 6, 'double'); W2 = rand (2, 2, 'double'), W3 = rand (2, 1, 'double') b1 = rand (2, 1, 'double'), b1 = rand (2, 1, 'double'), b3 = rand (1,'double').

**Step 1:** Read the number of the neuron hidden layer
**Step 2:** Read the pattern from XOR-2Bit, get to find the target and the limited error = 10 power-6
**Step 3:** Read the dynamic training rate and momentum
**Step 4:** While (MSE>limited error), do steps 5-18
**Step 5:** For each training pair, do steps 6-17 Forward Propagation
**Step 6:** Compute the input layer of hidden layer $Z$ using Eq. (7) and output value using Eq. (8).
**Step 7:** Compute the input layer of hidden layer $ZZ$ using Eq. (9) and output value using Eq. (10).
**Step 8:** Compute the input layer of hidden layer $O_r$ using Eq. (11) and output value using Eq. (12).

**Back propagation:**

**Step 9 :** Calculate the error training using Eq. (13)
**Step 10:** Computing the error signal $\delta_r$ at neural r using Eq. (14).
**Step 11:** Calculate the weight correction for each $\Delta w_{jr}$ and bias $\Delta w_{0r}$ using Equations 15 and 16, respectively.
**Step 12:** Send $\delta_k$ to $zz_j$ and calculate the error signal $\delta_{-in_j}$ and local gradient of error signal $\delta_j$ using Eq. (17) and (18), respectively.
**Step 13:** Calculate the weight correction for each $\Delta v_{hj}$ and bias $\Delta v_{0j}$ using Eq. (19) and (20), respectively
**Step 14:** Send $\delta_j$ to $z_h$ and calculate the error signal $\delta_{-in_h}$ and local gradient of error Signal $\delta_h$, using Eq. (21) and (22), respectively.
**Step 15:** For layer $z_h$, calculate the weight correction for each $\Delta u_{ih}$ and bias $\Delta u_{0h}$ using Equations 23 and 24, respectively.
**Step 16:** The weight update for each layer:

Output layer $O_r$ using Eq. (25) and Eq. (26), respectively

Hidden layer $zz_j$ using Eq. (27) and Eq. (28), respectively

Hidden layer $z_h$ using Eq. (29) and Eq. (30), respectively

**Step 17:** Calculate the Mean Square error, $MSE$ =

$$0.5\frac{1}{p}\sum_{p=1}^{n}\sum_{k}^{i}(t_{kp}-o_{kp})^2$$

**Step 18:** Test the conditional

## EXPERIMENTS RESULTS

In this section, we report the results obtained when experimenting with our proposed method with the 2-bit XOR parity problem and the iris data as a benchmark. We use Matlab software R2012a running on a Windows machine. There are no theories to determine the value of the limited error, but the range of the limited error effects the training time (Kotsiopoulos and Grapsa, 2009) determines the error tolerance by l to a power of -5. The convergence rate is very slow and it takes 500000 epochs, but (Cheung *et al*., 2010) determined the limited error by less than 3 to a power of -4. The convergence rate is very slow and it takes 1000 epochs.

**Experiments the BPDRM algorithm:** We run the BPDRM algorithm, which is given in Eq. (2) and (6). Ten experiments have been done at the limited error 1.0E-05. The average time for training and the epoch for all experiment results are tabulated in Table 1.

From Table 1 above, the formula proposed in Eq. (5) and (6) helps the back propagation algorithm to reduce the time for training. Whereas $t$ = 1.0315 sec, the average value of the MSE performance is a very small value for every epoch training. Training is shown in Fig. 2.

From Fig. 2 the training curve as a beginner is a slightly vibrating curve during the first training, then the curve decays with an inverse of the index of epoch. From the figure above, the training curve is smooth and convergence is quickly at global minimum.

**Experiments on the BP algorithm:** We are going to run the BP algorithm, which is given in equation 1 with trial or manual values for each training rate and momentum term. The value of $\eta$ and $\alpha$ are chosen $\in$ [0, 1]. The experiments' result is tabulated in Table 2.

From Table 2, the best performance of the BP algorithm is a achieved at $\eta = \alpha = 0.9$, whereas the time training was 9.3020 sec. The worst performance of the BP algorithm was achieved at $\eta = \alpha = 1$, whereas the training time was 1920 sec. The range of the time training is located $1920 \leq t \leq 9.3020$ sec. We consider the 1920 sec as the maximum training time and the value 9.3020 as the minimum training time. In addition,

Table 1: Average time training of BPDRM algorithm with XOR problem

| Average time-sec | Average MSE | Average Epoch |
|---|---|---|
| 1.0315 | 5.05E-07 | 504 |

Table 2: Average time training of BP algorithm with XOR problem

| Value of | | | | |
|---|---|---|---|---|
| $\eta$ | $\alpha$ | Time-Sec | MSE | Epoch |
| 0.1 | 0.1 | 904.1690 | 1.00 E-05 | 538028 |
| 0.2 | 0.2 | 398.0140 | 1.00 E-05 | 237289 |
| 0.3 | 0.3 | 227.9380 | 1.00 E-05 | 137566 |
| 0.4 | 0.4 | 150.8910 | 1.00 E-05 | 87969 |
| 0.5 | 0.5 | 103.1740 | 9.99 E-06 | 58389 |
| 0.6 | 0.6 | 62.71000 | 9.99 E-06 | 38783 |
| 0.7 | 0.7 | 45.36400 | 1.00 E-05 | 24877 |
| 0.8 | 0.8 | 24.90300 | 9.99 E-06 | 14492 |
| 0.9 | 0.9 | 9.302000 | 9.99 E-06 | 6512 |
| 1.0 | 1.0 | 1920.000 | 0.096000 | 1259443 |

Table 3: Average time training of BPDRM algorithm with buba training set

| Limited error | Average Time-Sec | Average MSE | Average Epoch |
|---|---|---|---|
| 0.000001 | 4.8689 | 9.96 E-07 | 119 |

Table 4: Average Time Training of BP algorithm with buba-training set

| Value of | | | | |
|---|---|---|---|---|
| $\eta$ | $\alpha$ | Average Time-Sec | Average MSE | Average Epoch |
| 0.1 | 0.1 | 196.3241 | 1.00 E-06 | 4591 |
| 0.2 | 0.2 | 80.15617 | 1.00 E-06 | 2017 |
| 0.3 | 0.3 | 47.94164 | 1.00 E-06 | 1164 |
| 0.4 | 0.4 | 32.77282 | 1.00 E-05 | 747 |
| 0.5 | 0.1 | 36.00864 | 9.99 E-07 | 907 |
| 0.6 | 0.5 | 16.61182 | 9.99 E-07 | 410 |
| 0.7 | 0.2 | 22.43927 | 9.99 E-07 | 581 |
| 0.8 | 1.0 | 21.66527 | 9.99 E-07 | 571 |
| 0.9 | 0.99 | 21.66527 | 0.500000 | 124769 |
| 1.0 | 1.0 | 17.72250 | 9.99 E-07 | 459 |

Table 5: Average Time Training of BPDRM algorithm with buba-testing set

| Limited error | Average Time-Sec | Average MSE | Average Epoch |
|---|---|---|---|
| 0.000001 | 7.2813 | 9.96328 E-07 | 180 |

the value of MSE and the number of epochs at $\eta = \alpha = 1$ whereas the value of MSE = 0.0960 and number or epoch is 1259443. The large value of MSE at $\eta = 0.9$, $\eta = 0.1$. That means the weight change is very slight or equal for every epoch. The figure training is shown in Fig. 3.

**BPDRM Algorithm experiments using the data training set:** We test the performance of our contribution, created in Eq. (2) and (6), by using 178 patterns as a form of training. Ten experiments have been done; the simulation results are tabulated in Table 3.

From Table 3, we are shown the average of the training time is very short and also the epoch number is very small. That indicates the dynamic training rate and momentum term to help the back propagation algorithm to remove the saturation training and reach the global minimum training. The training curve of the BPDRM algorithm on buba data is as shown in Fig. 4.

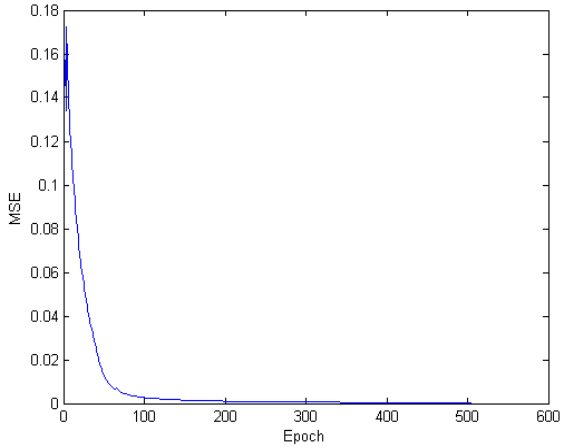


Fig. 2: Training curve of BPDRM algorithm

From Fig. 4, we can see that the BP starts training with a small value for a training error, whereas MSE is 4.5 power-3, then the MSE decays quickly with an inverse index of the epoch number. At around 10 epochs, the value of MSE = 0.005 then reaches the global minimum.

**Experiments of the BP algorithm with buba-training set:** In this part, we test the performance using 180 patterns as a form of training. 100 experiments have been done and then taken average of the experiments. The results are tabulated in the Table 4.

From Table 4, the best performance of the BP algorithm is achieved at, $\eta = 0.6$, $\alpha = 0.5$, whereas the training time is 16.61482 sec. The worst performance of the BP algorithm was achieved at $\eta = 0.9$, $\alpha = 0.99$, whereas the training time is 4750.909 sec. The range of the average training time is located between $16.61482 \leq t \leq 4750.909$ sec. We consider that 4750.909 sec is the maximum amount of training time and the value 16.61482 is the minimum amount of training time. The BP algorithm suffers the highest saturation at a value for each η and momentum term α at a value of 1. The curve of training as shown in Fig. 5.

**Experiments of the BPDRM algorithm with buba-testing set:** In this section, we implement the BPDRM algorithm using the buba data testing set. A hundred and twenty patterns were used as a test set. The input layer equals the attribute of the data. The structure of the BPDRM algorithm becomes 6:2:1. All experiments are illustrated in the Table 5.

From Table 5, the dynamic training rate and momentum reduces the time for training and enhancing the convergence of MSE. The average training time is 7.2813 sec at an epoch of 180. The curve of training as shown in Fig. 6.

From Fig. 6 we can see the training curve of the back propagation as it starts training with a small value of training error. The average value of MSE decays fast

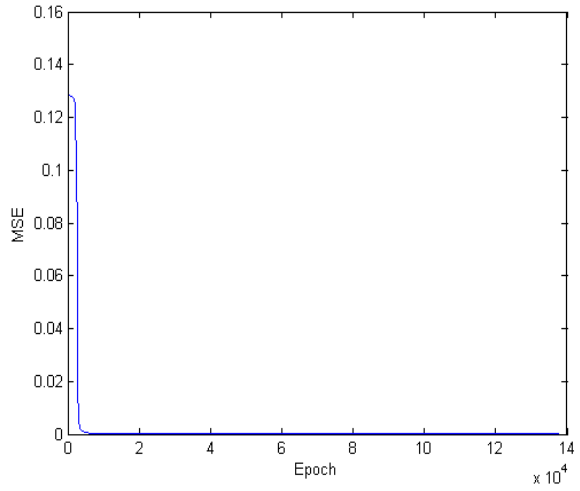


Fig. 3: Training curve of BP algorithm

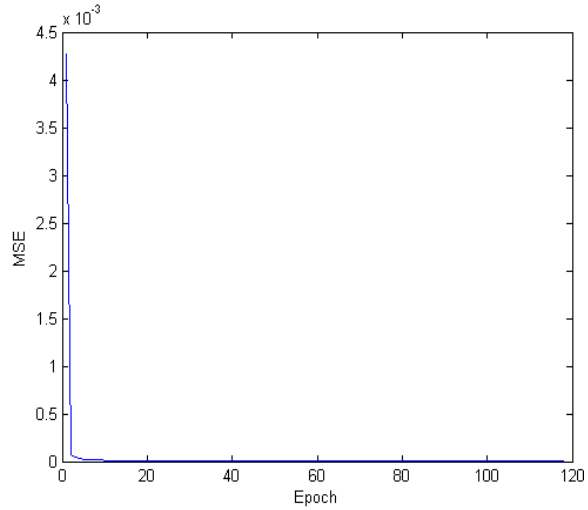


Fig. 4: Training curve of the BPDRM algorithm with buba-training set

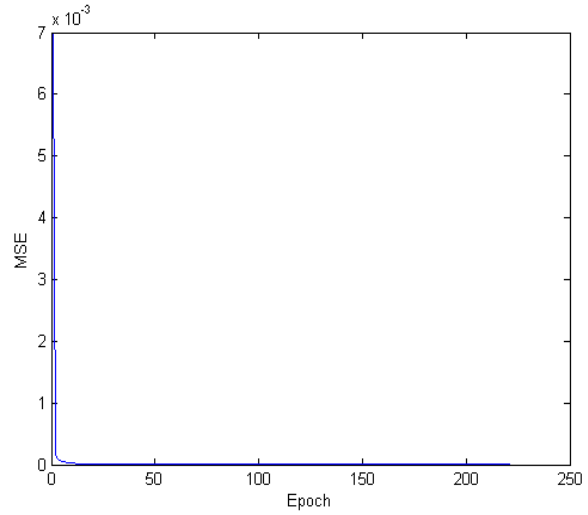


Fig. 5: Training curves of BP algorithms

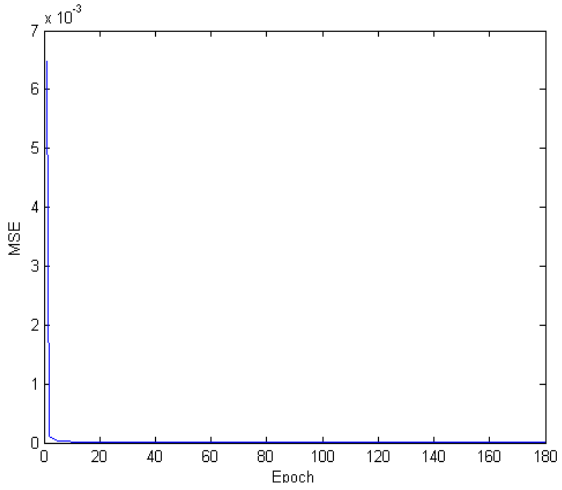


Fig. 6: Training curve for the BPDRM algorithm for the buba-testing set

Table 6: Average Time Training of BP algorithm with buba-testing set

| Value of | | | | |
|---|---|---|---|---|
| $\eta$ | $\alpha$ | Average Time-Sec | Average MSE | Average Epoch |
| 0.1 | 0.1 | 268.4603 | 1.00 E-06 | 6937 |
| 0.2 | 0.2 | 125.2125 | 9.998 E-07 | 3216 |
| 0.3 | 0.3 | 96.71273 | 1.00 E-06 | 1804 |
| 0.4 | 0.4 | 45.75273 | 1.00 E-06 | 1133 |
| 0.5 | 0.5 | 30.50627 | 9.99 E-07 | 763 |
| 0.6 | 0.5 | 24.69000 | 9.99 E-07 | 636 |
| 0.7 | 0.4 | 26.58618 | 9.99 E-07 | 687 |
| 0.8 | 0.3 | 26.44518 | 9.99 E-07 | 674 |
| 0.9 | 1.0 | 4330.909 | 0.500000 | 100183 |
| 1.0 | 1.0 | 3834.545 | 0.500000 | 100617 |

with the inverse index of number epoch. Around 5 epochs are the value of MSE = 0.001, which then reaches global minimum.

**Experiments on the BP algorithm for the buba-testing set:** We implement the BP algorithm using 120 patterns, which represents the test data set. A hundred experiments have been done on matlab. The experiment results are tabulated in Table 6.

From Table 6, the best performance of the BP algorithm was achieved at $\eta$ = 0.6 and $\alpha$ = 0.5. In addition, the BP algorithm provides fast training at the same point, whereas the training time = 24.69 sec. The worst performance of the BP algorithm is achieved at $\eta$ = 0.9, $\alpha$ = 1 whereas the training time = 4330.909 and MSE = 0.5. The range of the average training time location is $24.69 \leq t \leq 4330.909$ sec. We consider that 4330.909 sec is the maximum of training time and the value 24.69 is the minimum training time. The BP algorithm suffers the highest saturation at a value for each $\eta$ = 0.9 and momentum term $\alpha$ = 1.

## DISCUSSION

In this part, we discuss and compare the BPDRM algorithm with consider the BP algorithm on three criteria: the training time, MSE and the number of

epoch. According to (Saki *et al*., 2013; Nasr and Chtourou, 2011; Scanzio *et al*., 2010) we calculate speed up training by formulae as follow:

$$\text{Speed up} = \frac{\text{Execution time of } BP \ \ algorithm}{\text{Execution time of } BPDRM \ algorithm}$$

For XOR problem the dynamic propagation provides better training which is show in the Table 7.

From Table 7, it is evident that the BPBRM algorithm provides superior performance over the BP algorithm. The BPDRM algorithm is better for training, whereas the BPDRM algorithm is $1861.36699 \approx 1862$ times faster than the BP algorithm as a maximum training. In addition, the BPDRM algorithm is $9.019 \approx 9$ times faster than the BP algorithm as a minimum training time.

For buba training set we compare the BPDRM algorithm and the BP algorithm on three criteria: training time, MSE and number of epochs to discover which gives the superior training. The comparison between them is Table 8.

From Table 8, it is clear that the BPDRM algorithm has superior performance over the BP algorithm, whereas the BPDRM algorithm is $40.32 \approx 40$ times faster than the BP algorithm as a maximum training in the same way as the BPDRM algorithm is $3.6398 \approx 4$ time faster than the BP algorithm as a minimum training time.

For iris testing set the dynamic propagation provides better training that is show in the Table 9.

From Table 9, it is evident that the BPDRM algorithm has a superior performance compared to the BP algorithm. Whereas the BPDRM algorithm is

Table 7: Speeding up BPDRM versus BP algorithm with XOR

| | Value of | | | | | |
|---|---|---|---|---|---|---|
| Algorithm | $\eta$ | $\alpha$ | Time-Sec | MSE | Epoch | Speed up rate BP/BPDRM |
| BDRM | | | 1.031500 | 5.05 E-07 | 504 | |
| BP | 0.1 | 0.1 | 904.1690 | 1.00 E-05 | 538028 | 876.55 |
| | 0.2 | 0.2 | 398.0140 | 1.00 E-05 | 237289 | 385.85 |
| | 0.3 | 0.3 | 227.9380 | 1.00 E-05 | 137566 | 220.97 |
| | 0.4 | 0.4 | 150.8910 | 1.00 E-05 | 87969 | 74.27 |
| | 0.5 | 0.5 | 103.1740 | 9.99 E-06 | 58389 | 100.02 |
| | 0.6 | 0.6 | 62.71000 | 9.99 E-06 | 38793 | 60.79 |
| | 0.7 | 0.7 | 45.36400 | 1.00 E-05 | 24877 | 43.97 |
| | 0.8 | 0.8 | 24.90300 | 9.99 E-06 | 14492 | 12.25 |
| | 0.9 | 0.9 | 9.302000 | 9.99 E-06 | 6512 | 9.01 |
| | 1.0 | 1.0 | 1920.000 | 0.096000 | 1259443 | 186136 |

Table 8: Speed up BPDRM versus BP Algorithm with bub-Training set

| | Value of | | | | | |
|---|---|---|---|---|---|---|
| Algorithm | $\eta$ | $\alpha$ | Average Time-Sec | Average MSE | Average Epoch | Speed up rate BP/BPDRM |
| BDRM | | | 4.869000 | 9.96 E-07 | 119 | |
| BP | 0.1 | 0.1 | 196.3241 | 1.00 E-06 | 4591 | 40.32 |
| | 0.2 | 0.2 | 80.15617 | 1.00 E-06 | 2017 | 16.46 |
| | 0.3 | 0.3 | 47.94164 | 1.00 E-06 | 1164 | 9.84 |
| | 0.4 | 0.4 | 32.77782 | 9.99 E-07 | 747 | 6.73 |
| | 0.5 | 0.1 | 36.00864 | 9.99 E-07 | 907 | 7.39 |
| | 0.6 | 0.5 | 16.61182 | 9.99 E-07 | 410 | 3.41 |
| | 0.7 | 0.2 | 22.43927 | 9.99 E-07 | 581 | 4.60 |
| | 0.8 | 1.0 | 21.66527 | 9.99 E-07 | 571 | 4.44 |
| | 0.9 | 0.99 | 4750.909 | 0.5000000 | 124769 | 975.74 |
| | 1.0 | 1.0 | 17.72250 | 9.99 E-07 | 459 | 3.63 |

Table 9: Speeding BPDRM versus BP algorithm with buba-Testing set

| | Value of | | | | | |
|---|---|---|---|---|---|---|
| Algorithm | $\eta$ | $\alpha$ | Average Time-Sec | Average MSE | Average Epoch | Speed up rate BP/BPDRM |
| BDRM | | | 7.281300 | 9.96328E-07 | 180 | |
| BP | 0.1 | 0.1 | 268.4603 | 1.00 E-06 | 6937 | 36.86 |
| | 0.2 | 0.2 | 125.2125 | 9.998E-07 | 3216 | 125.12 |
| | 0.3 | 0.3 | 69.71273 | 1.00 E-06 | 1804 | 9.57 |
| | 0.4 | 0.4 | 45.75273 | 9.99 E-07 | 1133 | 6.28 |
| | 0.5 | 0.1 | 30.50627 | 9.99 E-07 | 763 | 4.18 |
| | 0.5 | 0.5 | 24.69000 | 9.99 E-07 | 636 | 3.39 |
| | 0.4 | 0.2 | 26.58618 | 9.99 E-07 | 687 | 3.65 |
| | 0.3 | 1.0 | 26.44518 | 9.99 E-07 | 674 | 3.63 |
| | 1.0 | 0.99 | 4330.909 | 0.5000000 | 100183 | 594.79 |
| | 1.0 | 1.0 | 3834.545 | 0.5000000 | 100617 | 526.62 |

$594.7988 \approx 595$ times faster than the BP algorithm maximum training time, on the other hand, the BPDRM algorithm is $3.3908 \approx 3.4$ time faster than the BP algorithm as a minimum training time.

## CONCLUSION

The back propagation BP algorithm is widely used in many tasks such as robot control, GPS and image restoration, but it suffers from slow training. To overcome this problem, there are many techniques for increasing the speed of the back propagation algorithm. In this study, we focused on a heuristic method, which included two parameters, the training rate and the momentum term. This study introduces the BPDRM algorithm, which is training by creating the dynamic function for each training rate and momentum. The dynamic function influenced the weight for each hidden layer and output layer. One of the main advantages of dynamic training and the momentum term is a reduction in the training time, error training and number of epochs. All algorithms were implemented on Matlab software R2012 a. The XOR problem and buba data were used as benchmarks. For the XOR problem, in the experiments result, the BPDRM algorithm is 1862 times faster than the BP algorithm at a maximum time. In addition, the BPDRM algorithm is 9 times faster than the BP algorithm at a minimum training time. For the buba data training set, the BPDRM algorithm is 976 times faster than the BP algorithm at the maximum time. For the buba data testing set**,** the BPDRM algorithm is 595 times faster than the BP algorithm at the maximum time.

## REFERENCES


Al-Duais, M.S., A.R. Yaakub and N. Yusoff, 2013. Dynamic training rate for back propagation learning algorithm. Proceeding of IEEE Malaysia International Conference on Communications (MICC), pp: 277-282.

Asaduzzaman, M.D., M. Shahjahan and K. Murase, 2009. Faster training using fusion of activation functions for feed forward neural networks. Int. J. Neural Syst., 6: 437-448.

Abdulkadir, S.J., S.M. Shamsuddin and R. Sallehuddin, 2012. Three term back propagation network for moisture prediction. Proceeding of International Conference on Clean and GreenEnergy, 27: 103-107. Bassil, Y., 2012. Neural network model for path-planning of robotic rover systems. Int. J. Sci. Technol., 2: 94-100.

Burse, K., M. Manoria and V.P. Kirar, 2010. Improved back propagation algorithm to avoid local minima in multiplicative neuron model. World Acad. Sci. Eng. Technol., 72: 429-432.

Cheung, C.C., S.C. Ng, A.K. Lui and S.S. Xu, 2010. Enhanced two-phase method in fast learning algorithms. Proceeding of the International Joint Conference on Neural Networks, pp: 1-7.

Dai, Q. and N. Liu, 2012. Alleviating the problem of local minima in back propagation through competitive learning. Neurocomputing, 94: 152-158.

Gong, B., 2009. A novel learning algorithm of back-propagation neural network. Proceeding of International Conference on Control, Automation and Systems Engineering, pp: 411-414.

Hamid, N.A., N.M. Nawi, R. Ghazali, M. Salleh and M. Najib, 2012. Improvements of back propagation algorithm performance by adaptively changing gain, momentum and learning rate. Int. J. Database Theor. Appl., 4: 65-76.

Huang, Y., 2007. Advances in artificial neural networks-methodological development and application. Algorithms, 3: 973-1007.

Iranmanesh, S. and M.S. Mahdavi, 2009. A differential adaptive learning rate method for back-propagation neural networks. World Acad. Sci. Eng. Technol., 50: 285-288.

Kwan, L., S. Shao and K. Yiu, 2013. YOA new optimization algorithm for single hidden layer feed forward neural networks. Appl. Soft Comput. J., 13: 2857-2862.

Kotsiopoulos, A.E. and N. Grapsa, 2009. Self-scaled conjugate gradient training algorithms. Neurocomputing, 72: 3000-3019.

Latifi, N. and A. Amiri, 2011. A novel VSS-EBP algorithm based on adaptive variable learning rate. Proceeding of 3rd International Conference on Computational Intelligence, Modelling and Simulation, pp: 14-17.

Li, J., Lian and M.E. Min, 2010. An accelerating method of training neural networks based on vector epsilon algorithm function. Proceeding of International Conferences on Information and Computing, pp: 292-295.

Li, Y., Y. Fu, H. Li and S.W. Zhang, 2009. The improved training algorithm of back propagation neural network with self-adaptive learning rate. Proceeding of International Conference in Computational Intelligence and Natural Computing, pp: 73-76.

Moalem, P. and S.A. Ayoughi, 2010. Improving backpropgation via an efficient combination of a saturation suppression method. Neural Network World, 10: 207-22.

Nand, S., P.P. Sarkar and A. Das, 2012. An improved gauss-newtons method based back-propagation algorithm for fast convergence. Int. J. Comput. Appl., 39: 1-7.

Nawi, N.M., N.M. Hamid and R.S. Ransing, 2011. Enhancing back propagation neural network algorithm with adaptive gain on classification problems. Int. J. Database Theor. Appl., 2: 65-76.

Negnevitsky, M., 2005. Multi-layer neural networks with improved learning algorithms. Proceeding of International Conference on Digital Image Computing: Techniques and Applications, (2005), pp: 34-34.

Nasr, M. B. and M. Chtourou, 2011. A self-organizing map-based initialization for hybrid training of feedforward neural networks. Applied Soft Computing, 11(8): 4458-4464.

Oh, S. and Y. Lee, 1995. A modified error function to improve the error back-propagation algorithm for multi-layer. ETRI J., 1: 1-22.

Scanzio, S., S. Cumani, R. Gemello, F. Mana and F.P. Laface, 2010. Parallel implementation of artificial neural network training for speech recognition. Pattern Recogn. Lett., 11: 1302-1309.

Shao, H. and G. Zheng, 2009. A new BP algorithm with adaptive momentum for FNNs training. Proceeding of WRI Global Congress on Intelligent Systems (GCIS '09), pp: 16-20.

Saki, F., A. Tahmasbi, H. Soltanian-Zadeh and S. B. Shokouhi, 2013. Fast opposite weight learning rules with application in breast cancer diagnosis. Computers in biology and medicine, 43(1): 32-41.

Thiang, H.K. and R. Pangaldus, 2009. Artificial neural network with steepest descent back propagation training algorithm for modeling inverse kinematics of manipulator. World Academy of Science, Engineering and Technology, 60: 530-533.

Tieding, L., Z. Shijian, G. Yunlan and T. Chengfang, 2009. Application of Improved BP Neural Network to GPS Height Conversion. In Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on IEEE, pp: 1-4.

Xiaozhong, L. and L. Qiu, 2008. A parameter adjustment algorithm of BP neural network. Proceeding of 3rd International Conference on Intelligent System and Knowledge Engineering (ISKE), pp: 892-895.

Yang, C. and R.C. Xu, 2009. Adaptation learning rate algorithm of feed-forward neural networks. Proceeding of International Conference on Information Engineering and Computer Science (ICIECS), pp: 4244-4994.

Zakaria, Z., N. Ashidi, M. Isa and S.A. Suandi, 2010. A study on neural network training algorithm for multi-face detection in static images. World Acad. Sci. Eng. Technol., 62: 170-173.

Zhang, Z., 2010. Convergence analysis on an improved mind evolutionary algorithm. Proceeding of 6th International Conference on Natural Computation (ICNC), pp: 2316-2320.

Zhang, C., W. Wu, X.H. Chen and Y. Xiong, 2008. Convergence of BP algorithm for product unit neural networks with exponential weights. Neurocomputing, 72: 513-520.

Zhang, N., 2009. An online gradient method with momentum for two-layer feed forward neural networks q. Appl. Math. Computat., 2: 488-498.

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.