Paper

Dynamically-adaptive Weight in Batch Back Propagation Algorithm via Dynamic Training Rate for Speedup and Accuracy Training

Mohammed Sarhan Al_Duais and Fatma Susilawati Mohamad

Department of Information Technology, Universiti Sultan Zainal Abidin, Terengganu, Malaysia

https://doi.org/10.26636/jtit.2017.113017

Abstract— The main problem of batch back propagation (BBP) algorithm is slow training and there are several parameters need to be adjusted manually, such as learning rate. In addition, the BBP algorithm suffers from saturation training. The objective of this study is to improve the speed up training of the BBP algorithm and to remove the saturation training. The training rate is the most significant parameter for increasing the efficiency of the BBP. In this study, a new dynamic training rate is created to speed the training of the BBP algorithm. The dynamic batch back propagation (DBBPLR) algorithm is presented, which trains with adynamic training rate. This technique was implemented with a sigmoid function. Several data sets were used as benchmarks for testing the effects of the created dynamic training rate that we created. All the experiments were performed on Matlab. From the experimental results, the DBBPLR algorithm provides superior performance in terms of training, faster training with higher accuracy compared to the BBP algorithm and existing works.

Keywords— artificial neural network (ANN), batch back propagation algorithm, dynamic training rate, speed up training, accuracy training.

1. Introduction

The batch back propagation (BBP) algorithm is commonly used in many applications, including robotics and automation. It has been used successfully in neural network training with a multilayer feed-forward network [1], [2]. The BP algorithm led to a tremendous breakthrough in the application of multilayer perceptions [3]. This method has been applied successfully in applications in many areas, and has an efficient training algorithm for the multilayer perception [4], [5]. Gradient descent is commonly used to adjust the weight through a change the error training, but it is not guaranteed to find the global minimum error, because the training is slow and converges easily to a local minimum [6]–[8].

The main problem of the BP algorithm is slow training; it requires a long learning time to obtain a result and there are several parameters that need to be adjusted manually, with highest saturation training [9], [10].

Current research on solving the slow training of the back-propagation algorithm is focused on the adaptation of parameters like the training rate that controls the weight adjustment along the descent direction [11]. We have improved the speed of the back propagation algorithm through adapting the training rate [12]. A new algorithm uses the square error function with a penalty for escaping from local minima. The weight is updated beside the penalty and the relationship between the training rate and penalty. The training rate is a fixed learning rate at 0.013 and the penalty parameter is set as 0.001. The results are compared with (standard back propagation) SBP.

The remaining portion of this paper is organized as follows. In Section 2 related work is presented. Section 3 presents proposed methods, while Section 4 shows experimental results. Section 5 covers discussion to validate the performance of the improved algorithm. Finally, Section 6 contains the conclusions.

2. Related Works

Abbas in [13] proposed a novel back propagation algorithm of ANN NBPNN that has a self-adaptive training rate. The experimental results show that NBPNN gave a more accurate result than the BP algorithm. In [14] a specific penalty to obtain the proportion of the norm of the weight or to prove the boundedness of the weights in the network training process is presented. The learning rate is set by an equation to be a small constant or an adaptive series. The initial weight is chosen in the range $[-0.5, 0.5]$; and the training rate is fixed to be a small constant: 0.05 or an adaptive series. The penalty factor is set as 0.001. The results show better convergence compared to existing work.

Authors in [15] improved the batch BPAP algorithm through their proposed dynamic training rate with a penalty. The structure of the algorithm is 2:2:1, using the sigmoid as the activation function. The weight was updated in the batch BPAP algorithm with bounded during training. From the experimental result, BPAP reaches a global minimum after the 1000th iteration. [16] provides the dynamic BP algorithm for training with a boundary. In this case,

the weight is updated under the effect of this boundary. The sigmoid function is used as the activation function. The boundary helps the BP algorithm for control of the weight update.

3. The Proposed Method

The data set is very important for verification to improve the BBP algorithm. In this study, all data are taken from UCI Machine Learning Repository [17].

3.1. Neural Network Model

The proposed ANN model is a three-layer neutral 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 data types or attributes. The hidden layer is made of two layers of four nodes. The output layer is made of one layer with one neuron. Three biases, two of them, which is 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 [18]. The 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. The set of weights by the activation function is as follows:

- L_h – first hidden layer for neuron h, $h = 1, \ldots, q$,
- LL_k – second hidden layer for neuron j, $j = 1, \ldots, p$,
- Y_r – output layer for neuron r,
- u_{ih} – the weight between neuron I in the put year 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 LL,
- v_{0j} – the weight of the bias for neuron j,
- w_{jr} – the weight between neuron k from the hidden layer LL and neuron r from the output layer L,
- w_{0r} – the weight of the bias for neuron r from the output layer,
- Δw – the difference between the current and new value in the next iteration,
- γ – the manual of training rate,
- γ_{dmic} – the dynamic training rate,
- $|e|$ – an absolute value of the error training,
- BBP – batch back propagation algorithm,
- DBBPLR – dynamic batch back propagation algorithm with dynamic training rate.

3.2. Dynamic Training Rate

One way to escape the local minimum and save training time in the BBP algorithm is by using a large value of γ in the first training. On the contrary, a small value of γ leads to slow training, but a smaller value of γ leads to the BBP algorithm having a slow convergence [19], [20]. Even a large γ is unlikely for training the BBP algorithm. The weight update between neuron k from the output layer and neuron j from the hidden layer is as follows:

$$\Delta w_{jk}(t+1) = w_{jk}(t) - \gamma \frac{\partial E}{\partial W_{jk}(t)}, \tag{1}$$

where $\Delta w_{jk}(t)$ is a weight change, the weight is updated for each epoch from Eq. 1, and slow training or fast training depends on a parameter that affects the updating of the weight. To enhance the BBP algorithm, which is given by the Eq. 1, to avoid the local minima and to avoid saturation training, we created the dynamic training rate:

$$\gamma_{\text{dmic}} = \sec(Y_r) + (1+k)^{|e|}, \tag{2}$$

where k is an average of the activation function, in this study is a sigmoid function.
The main idea is to keep the value of dynamic training rate positive for every epoch, to avoid the vibration of the value of training error e. We substitute γ_{dmic} from Eq. 2 into Eq. 1 to obtain:

$$\Delta w_{jk}(t+1) = w_{jk}(t) - \left[\sec(Y_r) + (1+k)^{|e|}\right] \frac{\partial E}{\partial W_{jk}(t)}. \tag{3}$$

The weight update is automatic for every layer under the effect of the dynamic training rate γ_{dmic}.

3.3. DBBPLR Algorithm

There are three stages of training BBP algorithm: forward phase, backward phase and feedback phase. 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 output layer of the system. The backward pass phase is starting when the output of the last hidden layer reaches to end step then the start. The goal of the BBP algorithm is to get the minimum error training between the desired output and actual data, the all steps recorded as follows:

$$e_r = \sum_{r=1}^{n} (t_r - Y_r). \tag{4}$$

The local gradient for the output derivative of the activation function of Y is:

$$e_r = e_r f'(Y_{-inr}) f'(Y_{-inr}) = Y_{-inr}(1 - Y_{-inr}). \tag{5}$$

The weight correction term, used to update w_{jr} later is:

$$\Delta w_{jr} = -\left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_r Y Y_j. \tag{6}$$

The bias correction term, used to update w_{0r} later is:

$$\Delta w_{0r} = -\left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_r, \tag{7}$$

sending δ_r to the hidden units ($YY_j,\ j = 1,\ldots,p$) in the layer above we obtain:

$$\delta_{-inj} = \sum_{r=1}^{m} \delta_r w_{jr}\,. \tag{8}$$

The local gradient for the hidden layer (YY_j):

$$\delta_j = \delta_{-inj} f'(YY_{-inj})\,. \tag{9}$$

the weight correction term, used to update v_{hj} is:

$$\Delta v_{hj} = -\left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_j Y_h\,. \tag{10}$$

The bias collection term to update v_{0j} later:

$$\Delta v_{0j} = -\left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_j\,, \tag{11}$$

by sending δ_j to the hidden unit ($L_h,\ h = 1,\ldots,a$) in the layer above we obtain:

$$\delta_{-inh} = \sum_{j=1}^{b} \delta_j v_{hl}\,. \tag{12}$$

The local gradient of the hidden layer L_h:

$$\delta_h = \delta_{-inh} f'(L_{-inh}),\ f'(L_{-inh}) = L_{-inh}(1 - L_{-inh})\,. \tag{13}$$

The weight correction:

$$\Delta u_{ih} = -\left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_h x_i\,, \tag{14}$$

and collate the bias weight corrective term used to update u_{0h}:

$$\Delta u_{0h} = -\left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_h\,. \tag{15}$$

In feedback phase all the layers are adjusted simultaneously. The weight update is as follows: For each output layer $j = 0,1,2,\ldots,p,\ \ r = 1,\ldots,m$:

$$w_{jr}(t+1) = w_{jr}(t) + \left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_r LL_j\,. \tag{16}$$

For bias:

$$w_{0r}(t+1) = w_{jr}(t) + \left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_r\,. \tag{17}$$

For each hidden layer $LL_j,\ \ h = 0,\ldots,q,\ \ j = 1,\ldots,p$:

$$v_{hj}(t+1) = v_{hj}(t) + \left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_j Y_h\,. \tag{18}$$

For bias:

$$v_{0j}(t+1) = v_{0j}(t) + \left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_j\,. \tag{19}$$

For each hidden layer $L_h,\ i = 0,\ldots,n,\ h = 1,\ldots,q$:

$$u_{ih}(t+1) = u_{ih}(t) + \left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_h x_i\,. \tag{20}$$

For the biases:

$$u_{ih}(t+1) = u_{ih}(t) + \left[\sec(Y_r) + (1+k)^{|e|}\right] \delta_h\,. \tag{21}$$

3.4. Implementation DBBPLR Algorithm

The BBP algorithm was implemented with a fixed value of the training rate from 0 to 1, and DBBPLR trained with the dynamic function of the training rate. There are no theories to determine the value of the limited error or condition. Anyway, the range of the limited error affects the training time [21]. In [22] the stop training is set by l to 10^{-5}. The convergence rate is very slow. It takes 500,000 epochs. In [23] the limited error by less than $3 \cdot 10^{-4}$. The convergence rate was very slow. It took 10,000 epochs.

0 : Read the initial weights.

1 : Read the number of neurons in the hidden layer.

2 : Read the pattern XOR 2 bit, obtain the target and limit the error E to 10^{-6}.

3 : Read the dynamic rate.

4 : While MSE $>$ limited error, repeat steps 4–15.

5 : For each training pair, repeat steps 5–15.

6 : Calculate the error training using Eq. 4.

7 : Compute the error signal δ_k at neuron k from Eq. 5.

8 : Calculate the weight correction for each Δw_{jr} and bias Δw_{0r} using Eq. 6 and 7.

9 : Send δ_k to LL_j and calculate the error signal δ_{-inj} and the local gradient of the error signal δ_j using Eq. 8 and 9.

10 : Calculate the weight correction for each Δv_{hj} and the bias Δv_{0j} using Eq. 10 and 11.

11 : Send δ_j to L_h and calculate the error signal δ_{-inh} and the local gradient of the error signal δ_h, using Eq. 12 and 13.

12 : For layer L_h calculate the weight correction for each Δu_{ih} and bias Δu_{0h} using Eq. 14 and 15.

13 : Update weight for each layer:

- output layer Y_r using Eq. 16 and 17,
- hidden layer LL_j using Eq. 18 and 19,
- hidden layer L_h using Eq. 20 and 21.

14 : Calculate the error training, time training and accuracy training.

15 : Test the conditional.

4. The Results

The accuracy training is measured by the following [25]:

$$\text{Accuracy} = \frac{1 - \text{absolut}(\text{T}_\text{i} - \text{O}_\text{i})}{\text{UP} - \text{LW}} \cdot 100\ [\%]\,,$$

where UP and LW are the upper bound and lower bound of activation function. Because the sigmoid function is used, the UP = 1 and LW = 0.

4.1. DBBPLR Algorithm Using XOR Problem

We run 10 experiments with DBBPLR algorithm given in Eq. 2 in Matlab 2012a. The experimental results are shown in Table 1.

Table 1
Average the performance of DBBPLR algorithm with XOR

Experiments	Time [s]	Epoch	Accuracy of training
Average	8.119	4426	0.9847
St. dev.	0.6614	0	$1.112 \cdot 10^{-16}$

As shown in Table 1 the back propagation algorithm enhances the performance of the training, and also reduces the training time. The average time of training is $t = 8.119$ s with the average epoch is 4426. The dynamic training rate has highest effects for increasing the accuracy of the training, whereas the average of accuracy training is 0.9807, it is close to 1. The training is shown in Fig. 1.

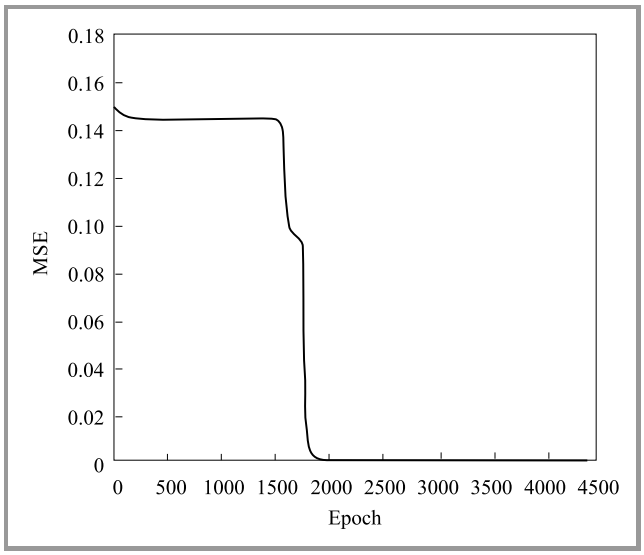

Fig. 1. Training curve for the DBBPLR algorithm.

The weight does not change before 1500 epochs, meaning that the DBBPLR algorithm is saturated, after which the training curve converges quickly to obtain the minimum error.

4.2. BBP Algorithm Using XOR

The simulation result of the BBP algorithm, given in Eq. 1 with trial or manual values for each training rate is tabulated in Table 2. The best performance of the BBP algorithm is achieved at $\gamma = 0.5$ when the training time is 35.7590 s. The worst performance of the BBP algorithm is achieved at $\gamma = 0.034$.

Meanwhile for Fig. 2 one can see that the BBP algorithm has the highest saturation training because the weight training slightly changes until 6,000 epoch and then starts to change.

Table 2
Average the performance of BBP algorithm with XOR

γ	Time [s]	Epoch
0.1	193.7690	86954
0.2	215.4940	43310
0.3	103.1070	28988
0.4	91.4450	21894
0.5	35.7590	17665
0.6	57.7380	14858
0.07	330.2740	124845
0.08	311.1590	109017
0.09	212.3080	96745
0.034	646.9530	260273
Average	219.8006	80455
St. dev.	171.4614783	71506.58607

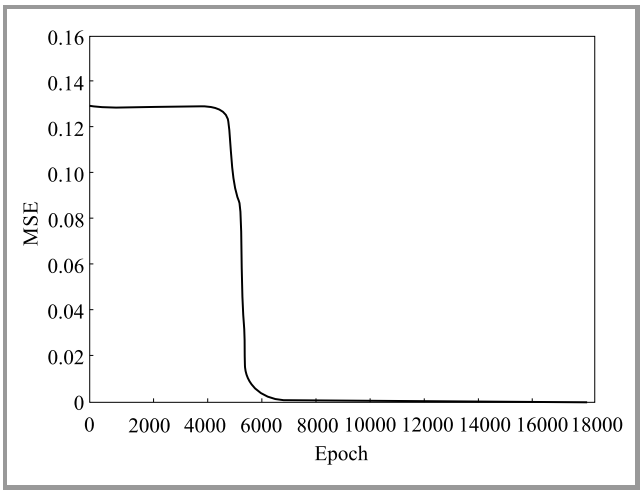

Fig. 2. Training curve of the BBP algorithm.

4.3. DBBPLR Algorithm with Balance Training Set

The balance data set is one of the best-known databases in pattern recognition. The data set has 625 patterns, which are used for training, and 375 patterns used for testing. Ten experiments were performed. The experimental results are given in Table 3.

The average training time is 11.284 s with 144 epochs. The average of the training accuracy is 0.999, it is close to one. This high accuracy indicates that the dynamic training rate helps the BBP algorithm to remove saturation training, to obtain faster training and to reach the global minimum training of the balance training set.

Table 3
The training performance of DBBPLR algorithm with balance

Experiments	Time [s]	Epoch	Accuracy of training
Average	11.284	144	0.999
St. dev.	0.925871	0	0

The DBBPLR algorithm has a flat spot until 600 epochs. After 600 epochs, the training curve convergence reduces the error training quickly. This observation means that the formulae created help the DBBPLR algorithm to reach the global minimum after 50 epochs.

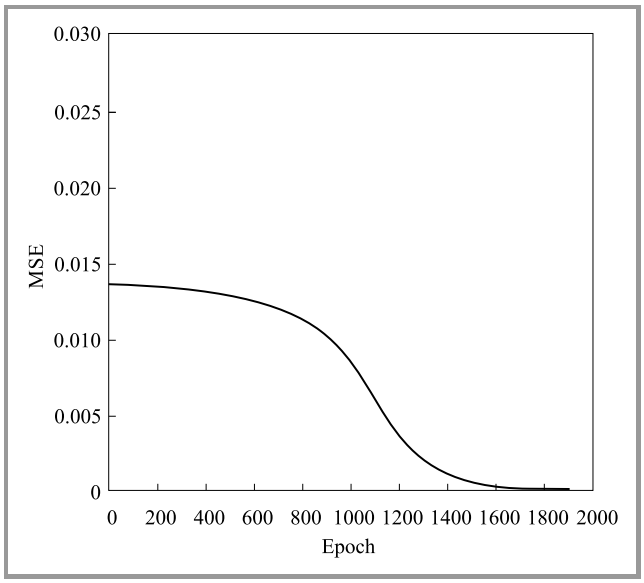

Fig. 3. Training curve of DBBPLR algorithm with the balance testing set.

4.4. Average the Performance of BBP Algorithm with Balance Training

The performance was tested using 375 patterns as a form of training. The experiments result is given in Table 4. The average training time is in the interval $30.9480 \leq t \leq 431.8660$ s, with 30.9480 s as the minimum training time and 431.8660 s as the maximum training time. The best performance of the BBP algorithm is achieved at $\gamma = 0.08$, whereas the average training time is 30.9480 s. The worst performance of the BBP algorithm is achieved at $\gamma = 0.01$. The BBP algorithm suffers the highest saturation when γ is 0.01.

4.5. DBBPLR Algorithm With Balance Testing Set

Table 5 shows results of testing the DBBPRL algorithm using the balance data testing set. The dynamic approach for training rate reduces the time required for training and

Table 4
Average the performance of BBP algorithm with XOR

γ	Time [s]	Epoch
0.01	431.8660	1907
0.02	105.5980	964
0.33	68.8110	655
0.045	55.5870	453
0.05	44.2000	414
0.068	43.7580	324
0.07	45.6230	317
0.08	30.9480	288
0.09	34.2550	258
Average	95.627	620
St. dev.	120.767	501.87

enhances the convergence of the time training. The average training time is 13.879 s at an average epoch of 273.

Table 5
Average the performance of DBBPLR algorithm with balance data testing set

Experiments	Time [s]	Epoch	Accuracy of training
Average	13.879	273	0.9963
St. dev.	0.570648	0	0

The training curve at approximately 10–50 epochs is a straight line with a flat spot, which means that the weight does not change for each epoch. In addition, the curve training begins to fall quickly.

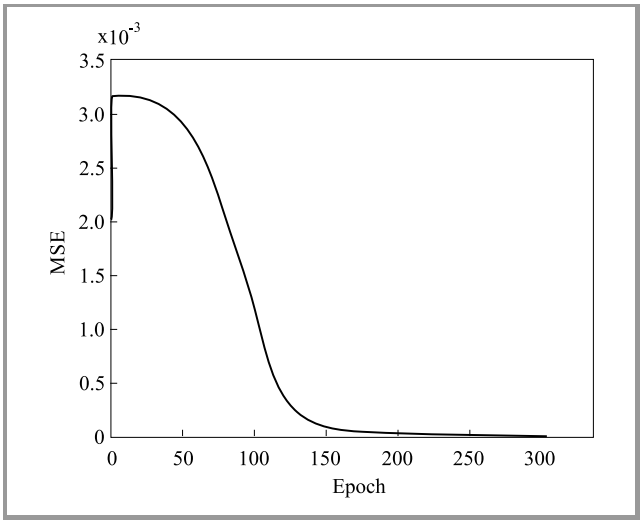

Fig. 4. Curve training of DBBPLR algorithm with the balance training set.

4.6. BBP Algorithm with Balance Testing Set

The BBP algorithm was tested using 250 patterns. The results are given in Table 6.

Table 6
The performance of training of BBP algorithm with balance testing set

γ	Time [s]	Epoch
0.1	448.9520	4084
0.2	213.3720	2080
0.3	181.1260	1418
0.4	68.5580	1091
0.5	54.6830	896
0.6	45.2700	768
0.7	40.7950	676
0.8	38.8610	609
0.9	71.4690	556
1	61.7610	514
Average	122.4847	1269
St. dev.	123.1680033	1043.628459

The best performance of the BBP algorithm was at $\gamma = 0.8$, where the BBP algorithm gives fast training at the same point. The range of the average training time is $38.8610 \leq t \leq 448.9520$ s. 38.8610 is a minimum training time; 448.9520 is the maximum.

4.7. DBBPLR Algorithm with Iris Training Set

The DBBPRL algorithm, given in Eq. 2, was run 10 times in Matlab 2012a. The experimental results are given in Table 7.

Table 7
Average the performance of DBBPLR algorithm with iris training set

Experiments	Time [s]	Epoch	Accuracy of Training
Average	1.637	121	0.99442
St. dev.	0.832629	60.93242	0.0004069

The average training time is $1.637 \approx 2$ s, with 121 epochs. The average of the accuracy training is 0.99455 the value of accuracy is close to 1. This high accuracy indicates that the dynamic training rate helps the back-propagation algorithm to remove saturation training, to obtain faster training and to reach the global minimum training. The standard deviation for time training is high value.

4.8. BBP Algorithm with Iris Training Set

The algorithm was tested with trial or manual values for each training rate. The results are shown in Table 8. The best performance of the BP algorithm is achieved at $\gamma = 0.5$ where the training time is 25.1360 s. The worst performance is achieved at $\gamma = 0.08$ when the training time is 191.7140 s.

Table 8
The performance of training of BBP algorithm with iris training set

γ	Time [s]	Epoch
0.1	74.2100	1458
0.2	25.1360	527
0.03	46.5020	1086
0.4	37.0980	861
0.06	75.9100	1890
0.7	29.0920	708
0.07	82.1590	2057
0.08	191.7140	4826
0.09	66.0930	1680
0.034	134.4380	3398
Average	71.43136364	1849
St. dev.	49.2239	1267.7981

4.9. DBBPLR Algorithm with Iris Testing Set

The performance of a proposed dynamic algorithm was tested using balance data from the iris data set. The data set has 150 patterns. 90 patterns were used for training and 60 patterns for testing. The structure of the algorithm considered is 4:2:1. The results are shown in Table 9.

Table 9
Average the performance of DBBPLR algorithm with iris testing set

Experiments	Time [s]	Epoch	Accuracy
Average	1.894	226	0.993
St. dev.	0.973410	98.77049154	0.002761159

The average training time is 1.894 s with average 226 epochs. The average of accuracy is 0.993. It indicates that the dynamic training rate helps the BBP algorithm to remove saturation training, to obtain faster training and to reach the global minimum training of the balance training set.

4.10. BBP Algorithm with Iris Testing Set

The performance was tested with 60 patterns (Table 10, next page). The best performance of the BBP algorithm is

achieved at $\gamma = 0.1$ whereas the training time is 21.2480 s. The worst performance of the BBP algorithm is achieved at $\gamma = 0.5$ when the training time is 108.355 s.

Table 10
The performance of training of BBP algorithm with iris testing set

γ	Time [s]	Epoch
0.1	21.2480	674
0.2	96.8510	3068
0.03	51.7230	1534
0.04	85.2280	1487
0.5	108.355	2069
0.06	59.7260	1826
0.07	39.7960	1157
0.08	61.0250	1776
0.09	77.9630	2217
0.034	48.6180	1499
Average	65.0533	1731
St. dev.	25.5128	612.2052

5. Discussion on Performance of the DBBPLR Algorithm

To verify or to validate the efficiency of the proposed algorithm, the performance of the improved DBBPLR algorithm was compared to the BBP algorithm based on certain criteria such as MSE, average time of training, and the number of epochs. The performance of the dynamic algorithm has been compared to the BBP algorithm in [26], [27]. The speed up training is calculated using the formula [28]–[30]:

$$\text{Speedup} = \frac{\text{Execution time of BBP}}{\text{Execution time of DBBPLR}}.$$

The number of an epoch is considered as a criterion used to compare the performance of the training. The comparison between the DBBPLR algorithm and BBP algorithm is presented in Table 11.

The dynamic algorithm provides superior performance over the BBP algorithm for all data sets. The range of the training time of the DBBPLR algorithm is $1.63686 \leq t \leq 13.8791$ s. This is a narrow interval, meaning that the DBBPLR algorithm reaches the global minimum in a short time and with few epochs. The range of training times of the BBP algorithm is $65.0533 \leq t \leq 122.4847$ s. This is a wide interval, meaning that the BBP algorithm has a long training time and a high level of training saturation. The DBBPLR algorithm is ≈ 44 times faster than the BBP algorithm at its maximum, and also the DBBPLR algorithm is ≈ 8 times faster than the BBP algorithm at its minimum.

Table 11
Speed up the DBBPLR algorithm versus BBP with various data set

Data set	DBBPLR		BBP		
	Average time [s]	Average epoch	Average time [s]	Average epoch	Speed up rate
XOR	8.119	4426	219.801	80455	27.074
Balance training	11.284	144	95.627	620	8.474
Balance testing	13.879	273	122.485	1269	8.825
Iris training	1.637	121	71.431	1849	43.639
Iris testing	1.894	226	65.053	1731	34.351

6. Conclusions

The DBBPLR algorithm gives superior training than BBP algorithm for all data set, whereas, the DBBPLR is 44 times faster than the BBP algorithm as maximum, and also the DBBPLR algorithm is 8 times faster than BBP algorithm. The dynamic training rate affected the weight for each hidden layer and output layer and eliminated the saturation training in the BBP algorithm. The dynamic DBBPLR algorithm provides superior performance of training, with higher accuracy compared to the BBP algorithm.

References

[1] H. Shao, J. Wang, L. Liu, L. D. Xu, and W. Bao, "Relaxed conditions for convergence of batch BPAP for feed forward neural networks", *Neurocomputing*, vol. 153, pp. 174–179, 2015 (doi: 10.1016/j.neucom.2014.11.039).

[2] H. H. Örkcü and H. Bal, "Comparing performances of back propagation and genetic algorithms in the data classification", *Expert Sys. with Applic.*, vol. 38, no. 4, pp. 3703–3709, 2011 (doi: 10.1016/j.eswa.2010.09.028).

[3] P. Moallem and S. A. Ayoughi, "Improving back-propagation VIA an efficient combination of a saturation suppression method", *Neural Network World*, vol. 2, pp. 207–223, 2010.

[4] S. M. Shamsuddin and F. I. Saman, "Three Term backpropagation algorithm for classification backpropagation algorithm (BP)", *Neural Network World*, vol. 4, no. 10, pp. 363–376, 2007 (doi: 10.1109/NABIC.2009.5393407).

[5] J. Ge, J. Sha, and Y. Fang, "A new back propagation algorithm with chaotic learning rate", in *Proc. 2010 IEEE Int. Conf. on Softw. Engin. and Serv. Sci.*, Beijing, China, 2010, pp. 404–407 (doi: 10.1109/ICSESS.2010.5552353).

[6] D. Xu, H. Shao, and H. Zhang, "A new adaptive momentum algorithm for split-complex recurrent neural networks", *Neurocomputing*, vol. 93, pp. 133–136, 2012 (doi: 10.1016/j.neucom2012.03.013).

[7] G. Yang and F. J. Qian, "A fast and efficient two-phase sequential learning algorithm", *Appl. Soft Comp.*, vol. 25, no. C, pp. 129–138, 2014 (doi: 10.1016/j.asoc.2014.09.012).

[8] S. Nandy, W. Bengal, and P. P. Sarkar, "An improved Gauss-Newtons method based back-propagation algorithm for fast convergence", *Int. J. of Comp. Appl. in Technol.*, vol. 39, no. 8, pp. 1–7, 2012.

[9] M. S. Al_Duais and F. S. Mohamad, "A review on enhancements to speed up training of the batch back propagation algorithm", *Ind. J. of Sci. and Technol.*, vol. 9, no. 46, pp. 1–10, 2016 (doi: 10.17485/ijst/2016/v9i46/91755).

[10] E. Noersasongko, F. T. Julfia, A. Syukur, R. A. Pramunendar, and C. Supriyanto, "A tourism arrival forecasting using genetic algorithm based neural network", *Ind. J. of Sci. and Technol.*, vol. 9, no. 4, pp. 1–55, 2016 (doi: 10.17485/ijst/2016/v9i4/78722).

[11] L. Wang, Y. Zeng, and T. Chen, "Back propagation neural network with adaptive differential evolution algorithm for time series forecasting", *Expert Sys. with Applic.*, vol. 42, no. 2, pp. 855–863, 2015 (doi: 10.1016/j.eswa.2014.08.018).

[12] L. Rui, Y. Xiong, K. Xiao, and X. Qiu, "BP neural network-based web service selection algorithm in the smart distribution grid", in *Proc. 16th Asia-Pacific in Netw. Oper. and Manag. Symp. APNOMS*, Hsinchu, Taiwan, 2014 (doi: 101109/APNOMS.2014.6996111).

[13] Q. Abbas, F. Ahmad, and M. Imran, "Variable learning rate based modification in backpropagation algorithm (MBPA) of artificial neural network for data classification", *Sci. Int.*, vol. 28, no. 3, pp. 2369–2378, 2016.

[14] H. Zhang, W. Wu, and M. Yao, "Boundedness and convergence of bach back-propagation algorithm with penalty for feed-forward neural networks", *Neurocomputing*, vol. 89, pp. 141–146, 2012 (doi: 10.1016/j.neucom.2012.02.029).

[15] H. Shao, J. Wan , L. Liu, D. Xu, and W. Bao, "Relaxed conditions for convergence of batch BPAP for feed forward neural networks", *Neurocomputing*, vol. 153, pp. 174–179, 2015 (doi: 10.1016/j.neucom.2014.11.039).

[16] Q. Feng and G. Daqi, "Dynamic learning algorithm of multi-layer perceptrons for letter recognition", *The Int. Joint Conf. in Neural Networks IJCNN*, pp. 1-6, Dallas, TX, USA, 2013.

[17] UCI Machine Learning Repository [Online]. Available: https://archive.ics.uci.edu/ml/index.html (accesed Oct. 11 2017).

[18] N. A. Hamid, N. M. Nawi, R. Ghazali, M. Najib, and M. Salleh, "Improvements of back propagation algorithm performance by adaptively changing gain, momentum, and learning rate", *Int. J. of New Comp. Archit. and their Applic. IJNCAA*, vol. 1, no. 2, pp. 889–90, 2011.

[19] G. Yang and F. J. Qian, "A fast and efficient two-phase sequential learning algorithm", *Appl. Soft Comput.*, vol. 25, pp. 129–138, 2013 (doi: 10.1016/j.asoc.2014.09.012).

[20] Y. Huang, "Advances in artificial neural networks – methodological development and application", *Algorithms*, vol. 2, no. 3, pp. 973–1007, 2009 (doi: 10.3390/algor2030973).

[21] A. E, Kostopoulos and T. N. Grapsa, "Self-scaled conjugate gradient training algorithms", *Neurocomputing*, vol. 72, no. 13–15, pp. 3000–3019, 2009 (doi: 10.1016/j.neucom.2009.04.006).

[22] Y. Li, Y. Fu, H. Liand, and S.-W. Zhang, "The improved training algorithm of back propagation neural network with self-adaptive learning rate", in *Proc. Int. Conf. on Comput. Intell. and Natur. Comput.*, no. 3, Wuhan, Hubei, China, 2009, pp. 1–4 (doi: 10.1109/CINC.2009.111).

[23] C.-C. Cheung, S.-C. Ng, A. K. Lui, and S. S. Xu., "Enhanced two-phase method in fast learning algorithms", in *Proc. Int. Joint Conf. on Neural Networks IJCNN*, Barcelona, Spain, 2010, pp. 1–7 (doi: 10.1109/IJCNN.2010.5596519).

[24] C. Yang and R. Xu, "Adaptation Learning rate algorithm of feed-forward neural networks", in *Proc. Int. Conf. on Inf. Engin. and Comp. Sci.*, Wuhan, Hubei, China, 2009, pp. 1–3 (doi: 10.1109/ICIECS.2009.5366919).

[25] N. M. Nawi, N. A. Hamid, R. S. Ransing, R. Ghazali, and M. N. M. Salleh, "Propagation neural network algorithm with adaptive gain on classification problems", *Int. J. of Datab. Theory and Applic.* vol. 4, no. 2, pp. 65–75, 2011.

[26] K. Wang, L. Zhuo, H. Lu, H. Guo, L. Xu, and Y. Zhang, "An improved BP algorithm over out-of-order streams for big data", in *Proc. Int. ICST Conf. on Commun. and Network.*, Guilin, Kuangsi, China, 2013, pp. 840–845 (doi: 10.1109/ChinaCom.2013.6694712).

[27] Q. Dai, Z. Ma, and Q. Xie, "A two-phased and ensemble scheme integrated ack propagation algorithm", *Appl. Soft Comput.*, no. 24, pp. 1124–1135, 2014 (doi: 10.1016/j.asoc.2014.08.012).

[28] S. Scanzio, S. Cumani, R. Gemello, R. F. Mana, and F. Laface "Parallel implementation of neural network training for speech recognition", *Patt. Recog. Lett.*, vol. 1, no. 11, pp. 1302–1309, 2010 (doi: 10.1016/j.patrec.2010.02.003).

[29] M. N. Nasr and M. Chtourou, "A self-organizing map-based initialization for hybrid training of feed-forward neutral networks", *Appl. Soft Comput.*, vol. 11, no. 8, pp. 4458–4464, 2011 (doi: 10.1016/j.asoc.2011.05.017).

[30] F. Saki, A. Tahmaasbi, H. Soltanian-Zadeh, and S. B. Shokouhi, "Fast opposite weight learning rules with application in breast cancer diagnosis", *Comp. in Biol. and Med.*, vol. 43, no. 1, pp. 32–41, 2013 (doi: 10.1016/j.compbiomed.2012.10.006).

Fatma Susilawati Mohamad is an Associate Professor in the Department of Information Technology, Faculty of Informatics and Computing, UniSZA. She has 18 years of experience in the academic field and has been actively involved in teaching, research, publications, professional services, consulting, and administration. Now she serves as a Deputy Dean at the Graduate School, UniSZA. She has graduated a B.Sc. (Information Systems) from Oklahoma City University, U.S.A. in 1997. She has continued his studies at Universiti Kebangsaan Malaysia and obtained her M.Sc. in Information Technology (Computer Science) in 2004. She received his Ph.D. in Computer Science from Universiti Teknologi Malaysia in 2012. Her specialization is in computer science focusing on pattern recognition and image processing.
E-mail: fatma@unisza.edu.my
Department of Information Technology
Faculty of Informatics and Computing
Universiti Sultan Zainal Abidin (UniSZA)
21300 Gong Badak, Kuala Nerus, Terengganu

Mohammed Sarhan Al_Duais is Ph.D. student in the Department of Information Technology, Faculty of Informatics and Computing. His field of research activity is artificial intelligence more specially, improvement the training of back propagation algorithm. He is an author in several scientific papers and publications in the field of arterial intelligence. He has completed the B.Sc. degree and master from Sana'a University.
E-mail: duais2025w@gmail.com
Department of Information Technology
Faculty of Informatics and Computing
Universiti Sultan Zainal Abidin (UniSZA)
21300 Gong Badak, Kuala Nerus, Terengganu