International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(4): 866-878
The Society of Digital Information and Wireless Communications, 2011 (ISSN: 2220-9085)

Improvements of Back Propagation Algorithm Performance by Adaptively Changing Gain, Momentum and Learning Rate

Norhamreeza Abdul Hamid*, Nazri Mohd Nawi,
Rozaida Ghazali, Mohd Najib Mohd Salleh.
Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia,
86400 Parit Raja, Batu Pahat, Johor, MALAYSIA.
gi090007@siswa.uthm.edu.my, {nazri, rozaida, najib}@uthm.edu.my

ABSTRACT

In some practical Neural Network (NN) applications, fast response to external events within enormously short time is highly demanded. However, by using back propagation (BP) based on gradient descent optimisation method obviously not satisfy in several application due to serious problems associated with BP which are slow learning convergence velocity and confinement to shallow minima. Over the years, many improvements and modifications of the BP learning algorithm have been reported. In this research, we modified existing BP learning algorithm with adaptive gain by adaptively change the momentum coefficient and learning rate. In learning the patterns, the simulation results indicate that the proposed algorithm can hasten up the convergence behaviour as well as slide the network through shallow local minima compare to conventional BP algorithm. We use five common benchmark classification problems to illustrate the improvement of the proposed algorithm.

KEYWORDS

Back propagation, convergence speed, shallow minima, adaptive gain, adaptive momentum, adaptive learning rate

1 INTRODUCTION

Multilayer Feedforward Neural Network (MLFNN) also referred to as Multilayer Perceptron (MLP) is one of the most popular and most frequently used type of Neural Network (NN) models due to its clear architecture and comparably simple algorithm. It can unravel classification problems implicating non-linearly separable patterns and can be used as a comprehensive function generator [1]. Due to its ability to solve some problems with relative ease of use, robustness to noisily input data, execution speed and analysing complicated systems without accurate modelling in advance, MLP has successfully been implemented across an extraordinary range of problem domains that involves prediction and a wide ranging usage area in the classification problems [2-9].

The MLP is composed by a set of sensorial units organised in three hierarchical of layers comprise of the input layer of neurons, one or more intermediary or hidden layer of neurons and the output layer of neurons. The consecutive layers are fully connected. The connections between the neurons of adjacent layers relay the output signals

from one layer to the next. Throughout the learning phase, the interconnections are optimised to minimise the predefined function.

Among the existing paradigms, Back Propagation (BP) algorithm is a supervised learning procedure for training MLP which is based on the gradient descent (GD) optimisation method that endeavors to minimise the error of the network by moving down the gradient of the error curve [1]. This algorithm mapping the input values to the desired output through the network. This output pattern (actual output) is then compared to the desired output and the error signal is computed for each output unit. The signals are then transmit backward from the output layer to each unit in the transitional layer that contributes directly to the output and the weights are adjusted iteratively during the learning process.

In some practical NN applications, fast response to external events within tremendously short time are highly demanded and expected. However, the comprehensively used of BP algorithm based on GD optimisation method obviously not satisfy in many applications especially large scale application and when higher learning accuracy as well as generalisation performances are obligatory. The reason for this unsatisfaction is due to the slow learning convergence velocity though the network has achieved stopping criteria. Moreover, it also frequently confinement to shallow minima.

It is noted that many local minima complications are closely associated to the neuron saturation in the hidden layer. When such saturation exists, neuron in the hidden layer will lose their sensitivity to the input signals and propagation chain is blocked severely. In some situation, the network can no longer learn. Furthermore, the convergence behaviour of the BP algorithm also depends on the selection of network architecture, initial weights and biases, learning rate, momentum coefficient, activation function and value of the gain in the activation function.

In the recent years with the progress of researches and applications, the NN technology has been enhanced and sophisticated. Research has been done on modification of the conventional BP algorithm in order to improve the efficiency and performance of MLP network training. Much work has been devoted to improve the generalization ability of the networks. These implicated the development of heuristic techniques, based on properties studies of the conventional BP algorithm. These techniques include such idea as varying the learning rate, using momentum and gain tuning of activation function. Lera *et al.* [10] described the use of Levenberg-Marquardt algorithm for training multi-layer feed forward neural networks. Though, the training times required strongly depend on neighbourhood size. Meanwhile, Ng *et al.* [11] localised generalisation error model for single layer perceptron neural network (SPLNN). This is an extensibility of the localised generalisation model for supervised learning with mean squared error minimisation. Though, this approach serves as the first step of considering localised generalisation error models of ANN.Meanwhile, Wang *et al.* [12]

proposed an improved BP algorithm caused by neuron saturation in the hidden layer. Each training pattern has its own activation function of hidden nodes in order to prevent neuron saturation when the network output has not acquired the desired signals. The activation functions are adjusted by the adaptation of gain parameters during the learning process. However, this approach not performed well on the large problems and practical applications. Otair and Salameh [13] designed the optical back propagation (OBP) algorithm which is applied on the output units. This kind of algorithm used for training process that depends on a multilayer NN with a very small learning rate, especially when using a large training set size. Conversely, it does not guarantee to converge at global minima because if the error closes to maximum, the OBP error grows increasingly. While Ji *et al.* [14] proposed a BP algorithm that improved conjugate gradient (CG) based. In the CG algorithm, a search is performed along conjugate directions which usually lead to faster convergence compared to gradient descent directions. Nevertheless, if it reaches a local minimum, it remains forever, as there is no mechanism for this algorithm to escape.

Nazri *et al.* [15] demonstrated that by adaptively change the 'gain' value for each node can significantly reduce the training time without modifying the network topology. Therefore, this research proposed a further improvement on [15] by adjusting activation function of neurons in the hidden layer in each training patterns. The activation functions are adjusted by the adaptation of gain parameters together with adaptive momentum and adaptive learning rate value during the learning process. The proposed algorithm, back propagation gradient descent with adaptive gain, adaptive momentum and adaptive learning rate (BPGD-AGAMAL) significantly can obviate the network from trapping into shallow minima that caused by the neuron saturation in the hidden layer as well as hasten up the convergence behaviour. In order to verify the efficiency of the proposed algorithm, the performance of the proposed algorithm will be compared with the conventional BP algorithm and back propagation gradient descent with adaptive gain (BPGD-AG) proposed by [15]. Some simulation experiments were performed on three classification problems including glass [16], soybean [17], breast cancer Wisconsin [18], card [19] and Iris [20].

The remaining of the paper is organised as follows. In Section 2, the effect of using activation function with adaptive gain is reviewed. While in Section 3 presents the proposed algorithm. The performance of the proposed algorithm is simulated on benchmark dataset problems in Section 4. This paper is concluded in the final section.

2 THE GAIN OF ACTIVATION FUNCTION IN BACK PROPAGATION ALGORITHM

An activation function is used for limiting the amplitude of the output neuron. It generates an output value for a node in a predefined range as the closed unit interval $[0,1]$ or alternatively $[-1,1]$ which can be a linear or non-linear function. This value is a function of the

weighted inputs of the corresponding node. The most commonly used activation function is the logistic sigmoid activation function. Alternative choices are the hyperbolic tangent, linear, step activation functions. For the j^{th} node, a logistic sigmoid activation function which has a range of $[0,1]$ is a function of the following variables, viz:

$$o_j = \frac{1}{1+e^{-c_j a_{net,j}}} \quad (1)$$

where,

$$a_{net,j} = \left(\sum_{i=1}^{l} w_{ij} o_i \right) + \theta_j \quad (2)$$

- o_j output of the j^{th} unit.
- o_i output of the i^{th} unit.
- w_{ij} weight of the link from unit i to unit j.
- $a_{net,j}$ net input activation function for the j^{th} unit.
- θ_j bias for the j^{th} unit.
- c_j gain of the activation function.

The value of the gain parameter, c_j, directly influences the slope of the activation function. For large gain values $(c > 1)$, the activation function approaches a 'step function' whereas for small gain values $(0 < c \leq 1)$, the output values change from zero to unity over a large range of the weighted sum of the input values and the sigmoid function approximates a linear function.

Most of the application oriented papers on NN tend to advocates that NN operate like a 'magic black box', which can simulate the "learning from example" ability of our brain with the help of network parameters such as weights, biases, gain, hidden nodes, and so forth. Also, a unit value for gain has generally being used for most of the research reported in the literature, though a few authors have researched the relationship of the gain parameter with other parameters which used in BP algorithms.

The learning rate (LR) is one of the most effective means to accelerate the convergence of BP learning. It is a crucial factor to control the variable of the neuron weight adjustments at each iteration during the training process and therefore affects the convergence rate. In fact, the convergence speed is highly depending on the choice of LR. The LR values need to be set appropriately since it dominate the performance of the BP algorithm. The algorithm will take longer time to converge or may never converge if the LR is too small. On the contrary, the network will accelerate the convergence rate significantly and still possibly will cause the instability whereas the algorithm may oscillates on the ideal path if the LR value is too high. The value of LR usually set to be constant which means that the selected value is employed for all weights in the whole learning process. Later, Ye [21] stated that the constant learning rate of the BP algorithm fails to optimise the search for the optimal weight combination. Hence, a search methodology has been classified as a "blind-search".

Another effective approach regarding to hasten up the convergence and stabilise the training procedure is by adding some momentum coefficient (MC) to the network. Moreover, with MC, the network can slide through shallow local

minima. Formerly, the MC is typically preferred to be constant in the interval $[0,1]$. In spite of that, it is discovered from simulations that the fixed momentum coefficient value seems to hasten up learning only when the recent downhill gradient of the error function and the last change in weight have a parallel direction. When the recent negative gradient is in a crossing direction to the previous update, the MC may cause the weight to be altered up the slope of the error surface as opposed to down the slope as preferred. This leads to the emergence of diverse schemes for adjusting the MC value adaptively instead of being kept constant throughout the training process.

Results in [22] demonstrate that the LR, MC and gain of the activation function have a significant impact on training speed. Thimm *et al.* also proved that a relationship between the gain value, a set of initial weight values, and LR value exists. Eom *et al.* proposed a method for automatic gain tuning using a fuzzy logic system. Nazri *et al.* [15] proposed a method to change the gain value adaptively on other optimisation method such as CG. Norhamreeza *et al.* demonstrated that adaptive momentum coefficient and adaptive gain of the activation function significantly improved the training time.

3 THE PROPOSED ALGORITHM

In this section, a further improvement on the current working algorithm proposed by [15] for improving the training efficiency of BP is proposed. The proposed algorithm modifies the initial search direction by changing the three terms adaptively for each node. Those three terms are; gain value, MC and LR. The advantages of using an adaptive gain value together with MC and LR have been explored. Gain update expressions as well as weight and bias update expressions for output and hidden nodes have also been proposed. These expressions have been derived using same principles as used in deriving weight updating expressions.

The following iterative algorithm is proposed for the batch mode of training. The weights, biases, gains, LRs and MCs are calculated and updated for the entire training set which is being presented to the network.

```
For a given epoch,
For each input vector,
    Step 1.
    Calculate the weight and bias values using the previously converged gain, MC and LR value.
    Step 2.
    Use the weight and bias value calculated in Step (1) to calculate the new gain, MC and LR value.
  Repeat Steps (1) and (2) for each input vector and sum all the weights, biases, LR, MC and gain updating terms
Update the weights, biases, gains, MCs and LRs using the summed updating terms and repeat this procedure on epoch-by-epoch basis until the error on the entire training data set reduces to a predefined value.
```

The gain update expression for a gradient descent (GD) method is calculated by differentiating the following error term E with respect to the corresponding gain parameter.

The network error E is defined as follows

$$E = \frac{1}{2}\sum\left(t_k - o_k\left(o_j, c_k\right)\right)^2 \quad (3)$$

For output unit, $\frac{\partial E}{\partial c_k}$ needs to be calculated whereas for hidden units $\frac{\partial E}{\partial c_j}$ is also required. The respective gain values would then be updated with the following equations:

$$\Delta c_k = \eta\left(-\frac{\partial E}{\partial c_k}\right) \quad (4)$$

$$\Delta c_j = \eta\left(-\frac{\partial E}{\partial c_j}\right) \quad (5)$$

$$\frac{\partial E}{\partial c_k} = -\left(t_k - o_k\right) o_k\left(1-o_k\right)\left(\sum w_{jk} o_j + \theta_k\right) \quad (6)$$

Therefore, the gain update expression for links connecting to output nodes is:

$$\Delta c_k(n+1) = \eta\left(t_k - o_k\right) o_k\left(1-o_k\right)\left(\sum w_{jk} o_j + \theta_k\right) \quad (7)$$

$$\frac{\partial E}{\partial c_j} = \left[-\sum_k c_k w_{jk} o_k\left(1-o_k\right)\left(t_k - o_k\right)\right] o_j\left(1-0_j\right)\left(\left(\sum_j w_{ij} o_i\right) + \theta_j\right) \quad (8)$$

Therefore, the gain update expression for the links connecting hidden nodes is:

$$\Delta c_j(n+1) = \eta\left[-\sum_k c_k w_{jk} o_k\left(1-o_k\right)\left(t_k - o_k\right)\right] o_j\left(1-0_j\right)\left(\left(\sum_j w_{ij} o_i\right) + \theta_j\right) \quad (9)$$

Similarly, the weight and bias expressions are calculated as follows:
The weights update expression for the links connecting to output nodes:

$$\Delta w_{jk}(n+1) = \eta\left(t_k - o_k\right) o_k\left(1-o_k\right) c_k o_j + \alpha\Delta w_{jk}(n) \quad (10)$$

Where the LR, η and MC, α are randomly generated.
Similarly, the bias update expressions for the output nodes would be:

$$\Delta \theta_k(n+1) = \eta\left(t_k - o_k\right) o_k\left(1-o_k\right) c_k + \alpha\Delta w_{jk}(n) \quad (11)$$

The weight update expression for the links connecting to hidden nodes is:

$$\Delta w_{ij}(n+1) = \eta\left[\sum_k c_k w_{jk} o_k\left(1-o_k\right)\left(t_k - o_k\right)\right] c_j o_j\left(1-o_j\right) o_i + \alpha\Delta w_{ik}(n) \quad (12)$$

Similarly, the bias update expressions for the hidden nodes would be:

$$\Delta \theta_j(n+1) = \eta\left[\sum_k c_k w_{jk} o_k\left(1-o_k\right)\left(t_k - o_k\right)\right] c_j o_j\left(1-o_j\right) + \alpha\Delta w_{ik}(n) \quad (13)$$

4 RESULTS AND DISCUSSIONS

The performance criterion used in this research focuses on the speed of convergence, measured in number of iterations and CPU time as well as accuracy. The real world classification problem datasets are obtained from UCI Machine Learning Repository at Centre for Machine Learning and Intelligent Systems have been used to verify our algorithm. Five classification have been tested including glass [16], soybean [17], breast cancer Wisconsin [18], card [19] and Iris [20].
The simulations have been carried out on a Pentium IV with 2 GHz HP Workstation, 3.25 GB RAM and using MATLAB version 7.10.0 (R2010a). On each problem, the following three algorithms were analysed and simulated.

1) The conventional Back Propagation Gradient Descent (BPGD)
2) The Back Propagation Gradient Descent with Adaptive Gain (BPGD-AG) [15]
3) The proposed algorithm which is Back Propagation Gradient Descent with Adaptive Gain, Adaptive Momentum and Adaptive Learning Rate (BPGD-AGAMAL)

To compare the performance of the proposed algorithm with conventional BPGD and BPGD-AG [15], network parameters such as network size and architecture (number of nodes, hidden layers and so forth), values for the initial weights and gain parameters were kept the same. For all problems, the NN had

one hidden layer with five hidden nodes and sigmoid activation function was used for all nodes. All algorithms were tested using the same initial weights which were randomly initialised from range $[0,1]$ and received the input patterns for training in the same sequence.

For all training algorithms, as the gain, MC and LR value were modified; the weights and biases were updated using the new value of gain, MC and LR. To avoid oscillations during training and to achieve convergence, an upper limit of 1.0 is set for the gain value. The initial value used for the gain parameter is set to one. The initial value for MC and LR is randomly generated depends on the dataset problems. For each run, the numerical data is stored in two files - the results file and the summary file. The result file lists the data about each network. The number of iterations until the network converged is accumulated for each algorithm which is the mean, the standard deviation (SD) and the number of failures is calculated. The networks that failed to converge are obviously excluded from the calculations of the mean and SD and were considered to be reported as failures. For each problem, 50 different trials were run, each with different initial random set of weights. For each run, the number of iterations required for convergence is reported. For an experiment of 50 runs, the mean of the number of iterations (mean), the SD, and the number of failures are collected. A failure occurs when the network exceeds the maximum iteration limit; each experiment is run to 10 000 iterations; otherwise, it is halted and reported as a failure. Convergence is achieved when the outputs of the network conform to the error criterion as compared to the desired outputs.

4.1 Glass Classification Problem

This dataset was collected by B. German on fragments of glass encountered in forensic work. The glass dataset is used for separating glass splinters into six classes, namely float processed building windows, non-float processed building windows, vehicle windows, containers, tableware, or head lamps [16]. The selected architecture of the network is 9-5-6 with target error was set to 0.001. The best MC and LR value for conventional BPGD and BPGD-AG for the glass dataset are 0.1 and 0.1 while BPGD-AGAMAL is initialised randomly in range $[0.1,0.3]$ for MC and $[0.1,0.2]$ for LR value.

Table 1. Algorithm performance for Glass Classification Problem [16].

	BPGD	**BPGD-AG**	**BPGD-AGAMAL**
Mean	8613	2057	2052
Total CPU time (s) of converge	572.54	59.57	56.16
CPU time(s)/ Epoch	6.65×10^{-2}	2.9×10^{-2}	2.74×10^{-2}
SD	2.15×10^{3}	2.45×10	3.12×10
Accuracy (%)	79.42	79.98	82.24
Failures	70	0	0

Table 1 shows that the proposed algorithm (BPGD-AGAMAL) exhibit excellent average performance in order to reach the target error. Furthermore, the accuracy of the proposed algorithm is better compared to BPGD and BPGD-AG. Moreover, the proposed algorithm (BPGD-AGAMAL) needs 2052 epochs to converge as opposed to the conventional BPGD at about 8613

epochs, while BPGD-AG needs 2057 epochs to converge. Apart from speed of convergence, the time required for training the classification problem is another important factor when analysing the performance. The graph depicted in Figure 1 clearly show that the proposed algorithm (BPGD-AGAMAL) practically outperformed conventional BPGD with an improvement ratio, 10.2 seconds whilst BPGD-AG, the proposed algorithm outperformed with an improvement ratio nearly 2 seconds for the total time of converged. Besides, the BPGD did not perform well in this dataset since 70% of simulation results failed in learning the patterns.

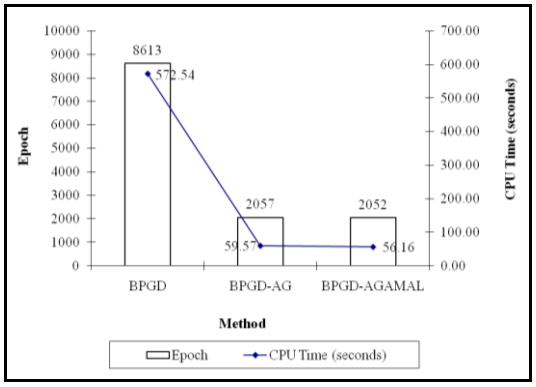

Figure 1. Performance comparison of BPGD-AGAMAL with BPGD-AG and conventional BPGD on Glass Classification Problem.

4.2 Soybean Classification Problem

The soybean data set was constructed to classify 19 different diseases of soybeans. The discrimination is done based on a description of the bean (e.g. whether its size and color are normal or not) and the plant (e.g. the size of spots on the leafs, whether these spots have a halo, whether plant growth is normal whether roots are rotted or not) and also information regarding the history of the plant's life (e.g. whether changes in crop occurred in the last year or last two years, whether seeds were treated or not, the effect of the temperature environment). The selected architecture of the network is 35-5-19 and the target error was set as 0.001. The best MC for conventional BPGD and BPGD-AG is 0.1, meanwhile the best LR value for the soybean dataset is 0.1 and 0.4. The MC value for BPGD-AGAMAL is initialised randomly in range $[0.1,0.2]$ for MC and $[0.3,0.6]$ for LR value.

Table 2. Algorithm performance for Soybean Classification Problem [17].

	BPGD	**BPGD-AG**	**BPGD-AGAMAL**
Mean	3038	1271	1089
Total CPU time (s) of converge	311.47	91.92	78.63
CPU time(s)/ Epoch	1.02×10^{-1}	7.23×10^{-2}	7.22×10^{-2}
SD	3.38×10^{3}	1.92×10^{2}	8.58×10
Accuracy (%)	94.23	91.08	94.82
Failures	8	0	0

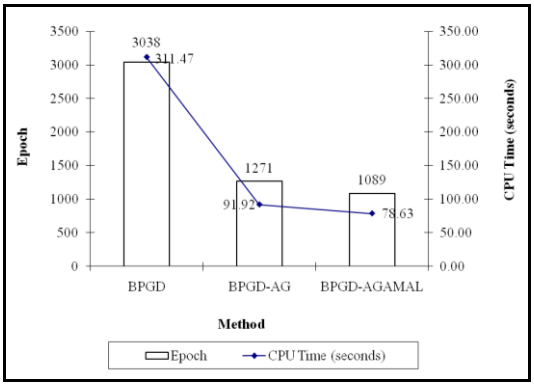

Figure 2. Performance comparison of BPGD-AGAMAL with BPGD-AG and conventional BPGD on Soybean Classification Problem.

Figure 2 proved that the proposed algorithm (BPGD-AGAMAL) still outperformed other algorithms in terms of CPU time, number of epochs and accuracy. The proposed algorithm required 1089 epochs in 80.83 seconds CPU times to achieve the target error by 94.82% accurate. Whereas BPGD-AG required 1271 epochs in 91.92 seconds CPU times with 91.08% accurate. At the same time, BPGD needs 3038 epochs in 311.47 seconds CPU times and 94.23% accurate. As we can see in Table 2, the average number of learning iterations for the BPGD-AGAMAL was reduced up to 2.8 and 1.2 faster as compared to BPGD and BPGD-AG.

4.3 Breast Cancer Classification Problem

This dataset was generated from University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [18]. The input attributes are for instance the clump thickness, the uniformity of cell size, the uniformity of cell shape, the amount of marginal adhesion, the single epithelial cell size, frequency of bare nuclei, bland chromatin, normal nucleoli and mitoses. This problem tries to diagnosis of Wisconsin breast cancer by trying to classify a tumor as either benign or malignant based on cell description gathered by microscopic examination. The selected architecture of the network is 9-5-2 with target error 0.001. The best MC for conventional BPGD and BPGD-AG for the breast cancer dataset is 0.1 and LR is 0.4 whilst BPGD-AGAMAL is randomly initialised in range of $[0.3,0.6]$ for MC and $[0.1,0.2]$ for LR value.

Table 3. Algorithm performance for Breast Cancer Classification Problem [18].

	BPGD	**BPGD-AG**	**BPGD-AGAMAL**
Mean	3136	590	526
Total CPU time (s) of converge	128.13	14.43	12.44
CPU time(s)/ Epoch	4.09×10^{-2}	2.45×10^{-2}	2.37×10^{-2}
SD	1.95×10^{3}	2.63×10^{2}	3.12×10
Accuracy (%)	68.29	94.12	95.47
Failures	0	0	0

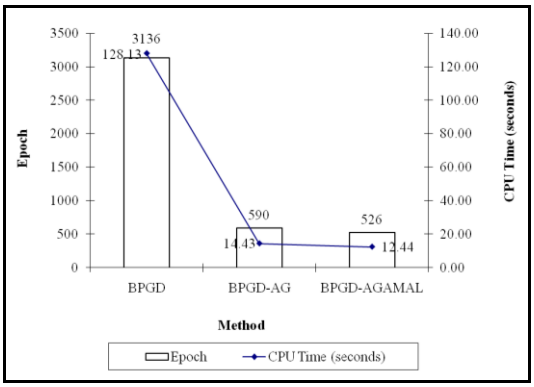

Figure 3. Performance comparison of BPGD-AGAMAL with BPGD-AG and conventional BPGD on Breast Cancer Classification Problem.

From Figure 3, it is worth noticing that the performance of the BPGD-AGAMAL is 83.23% faster than BPGD and almost 10.9% faster than BPGD-AG. Table 3 reveals that BPGD-AGAMAL approximately took 2.37×10^{-2} per epoch to reach target error as well as 95.47% accurate. While, BPGD-AG took 2.45×10^{-2} per epoch to reach target error with 94.12% accurate and BPGD took 4.09×10^{-2} per epoch to reach target error by 68.29% accurate. Still, the proposed algorithm

(BPGD-AG) surpasses the BPGD and BPGD-AG algorithm in terms of total time of converge and accuracy to learn the pattern.

4.4 Card Classification Problem

This dataset contained all the details on the subject of credit card applications. It predicted the approval or non-approval of a credit card to a customer [19]. Descriptions of each attribute name and values were not enclosed for confidentiality reason. There were 690 instances, 51 inputs, and 2 outputs in this dataset. The dataset classified whether the bank granted the credit card or not.

The selected architecture of NN is 51-5-2 while the target error and maximum epoch were set as 0.001 and 10 000 respectively. The best momentum coefficient value for conventional BPGD and BPGD-AG is 0.4 meanwhile the best learning rate value is 0.6. The best momentum coefficient value for BPGD-AGAMAL is found in the range $[0.1,0.4]$ and $[0.4,0.8]$ for learning rate value.

Table 4. Algorithm performance for Card Classification Problem [19].

	BPGD	BPGD-AG	BPGD-AGAMAL
Mean	8645	1803	1328
Total CPU time (s) of converge	547.1	47.19	22
CPU time(s)/ Epoch	6.33×10^{-2}	2.61×10^{-2}	1.66×10^{-2}
SD	2.76×10^{-3}	6.55×10^{-1}	6.75×10^{-2}
Accuracy (%)	83.45	82.33	83.9
Failures	82	0	0

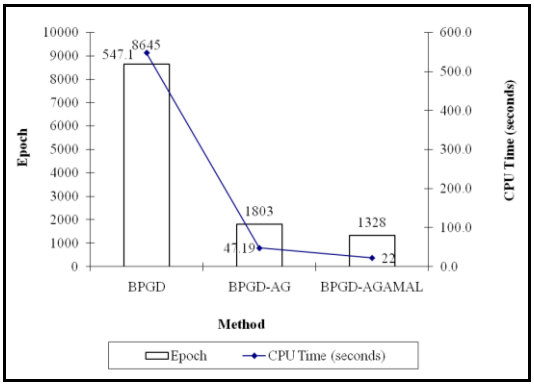

Figure 4. Performance comparison of BPGD-AGAMAL with BPGD-AG and conventional BPGD on Card Classification Problem.

Table 4 reveals that BPGD needs 547 seconds with 8645 epochs to converge, whereas BPGD-AG needs 47.2 seconds with 1803 epochs to converge. Conversely, the proposed algorithm (BPGD-AGAMAL) performed significantly better with only 41.1 seconds and required 1328 epochs to converge. Figure 4, demonstrates that the performance of the BPGD-AGAMAL is almost 96% faster than BPGD and 53.4% faster than BPGD-AG.

4.5 Iris Classification Problem

This dataset was a classical classification dataset made famous by Fisher, who used it to illustrate principles of discriminant analysis [20]. There were 75 instances, 4 inputs, and 3 outputs in this dataset. The classification of Iris dataset involves classifying the data of petal width, petal length, sepal width, and sepal length into three classes of species, which are Iris Sentosa, Iris Versicolor, and Iris Verginica.

The selected network topology for Iris classification problem is 4-5-3, which is 4 input nodes, 5 hidden nodes and 3 output nodes. 50 instances were represented as training dataset and the rest as testing dataset. The target error was set as 0.001 and the maximum epochs to 10 000. The best momentum term and learning rate value for conventional BPGD and BPGD-AG for the Iris dataset is 0.4 and 0.6 respectively while BPGD-AGAMAL is found in the interval $[0.1,0.4]$ for the best momentum coefficient value and $[0.4,0.8]$ for learning rate value.

Table 5. Algorithm performance for Iris Classification Problem [20].

	BPGD	BPGD-AG	BPGD-AGAMAL
Mean	1081	721	533
Total CPU time (s) of converge	12.29	5.89	4.26
CPU time(s)/ Epoch	1.14×10^{-2}	8.17×10^{-3}	7.99×10^{-3}
SD	1.4×10^{2}	4.09×10^{2}	2.45×10^{2}
Accuracy (%)	91.9	90.3	93.1
Failures	2	0	0

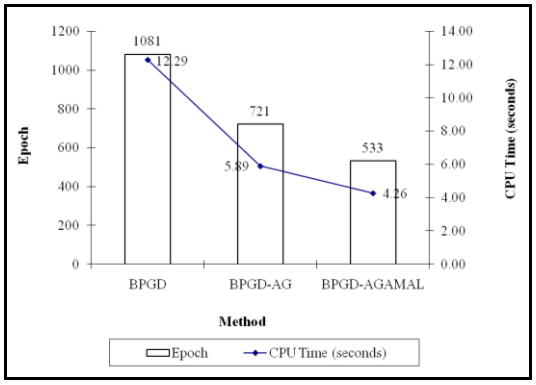

Figure 5. Performance comparison of BPGD-AGAMAL with BPGD-AG and conventional BPGD on Iris Classification Problem.

Table 5 shows that the proposed algorithm (BPGD-AGAMAL) still outperforms other algorithms in terms of CPU time and number of epochs. The proposed algorithm (BPGD-AGAMAL) needs only 533 epochs to converge as opposed to the conventional BPGD at about 1081 epochs while BPGD-AG needs 721 epochs to converge. Apart from speed of convergence, the time required for training the classification problem is another important factor when analysing the performance. The results in Figure 5 clearly show that the proposed algorithm (BPGD-AGAMAL) outperforms conventional BPGD with an improvement ratio, nearly 2.9 seconds while BPGD-AG, the proposed algorithm outperformed 1.38 seconds for the total time of converge. Furthermore, the accuracy of BPGD-AGAMAL is much better than BPGD and BPGD-AG algorithm.

The results show that the BPGD-AGAMAL perform considerably better as compared to BPGD and BPGD-AG. Moreover, when comparing the proposed algorithm with BPGD and BPGD-AG, it has been empirically demonstrated that the proposed algorithm (BPGD-AGAMAL) performed highest accuracy than BPGD and BPGD-AG algorithm. This conclusion enforces the usage of the proposed algorithm as an alternative training algorithm of BP algorithm.

5 CONCLUSIONS

Although BP algorithm is widely implemented in the most practical NN applications and performed relatively well, this algorithm still needs some improvements. We have proposed a further improvement on the current working algorithm proposed by Nazri *et al.* [15]. The proposed algorithm adaptively changes the gain parameter of the activation function together with MC and LR to hasten up the convergence behaviour as well as slide the network through shallow local minima. The effectiveness of the proposed algorithm has been compared with the conventional Back Propagation Gradient Descent (BPGD) and Back Propagation Gradient Descent with Adaptive Gain (BPGD-AG) [15]. The three algorithms had been verified by means of simulation on five classification problems including glass dataset with an improvement ratio 10.2 seconds for the BPGD and nearly 2 seconds better for the BPGD-AG in terms of total time to converge. Meanwhile, for soybean dataset, BPGD-AGAMAL was reduced up to 2.8 and 1.2 faster as compared to BPGD and BPGD-AG. While breast cancer dataset indicates that BPGD-AGAMAL is 83.23% faster than BPGD and almost 10.9% faster than BPGD-AG respectively. Whereas card almost 96% and 53.4% faster than BPGD and BPGD-AG respectively. Whilst Iris improved nearly 2.9 seconds than BPGD and improved 1.38 seconds than BPGD-AG for the total time of converged. The results show that the proposed algorithm (BPGD-AGAMAL) has a better convergence rate and learning efficiency as compared to conventional BPGD and BPGD-AG.

6 REFERENCES

1. Haykin, S. S., Neural Networks and Learning Machines. New Jersey: Prentice Hall. (2009).
2. Yu, L., Wang, S. and Lai, K., An Adaptive BP Algorithm with Optimal Learning Rates and Directional Error Correction for Foreign Exchange Market Trend Prediction, in Advances in Neural Networks - ISNN 2006. Springer Berlin / Heidelberg. p. 498-503. (2006).
3. Nazri, M. N., Ransing, R. S., Najib, M. S. M., Rozaida, G. and Norhamreeza, A. H., An Improved Back Propagation Neural Network Algorithm on Classification Problems, in Database Theory and Application, Bio-Science and Bio-Technology, Zhang, Y., Cuzzocrea, A., Ma, J., Chung, K.-i., Arslan, T. and Song, X., Editors. Springer Berlin Heidelberg. p. 177-188. (2010).
4. Nazri, M. N., Najib, M. S. M. and Rozaida, G., The Development of Improved Back-Propagation Neural Networks Algorithm for Predicting Patients with Heart Disease, in Information Computing and Applications, Zhu, R., Zhang, Y., Liu, B. and Liu, C., Editors. Springer Berlin / Heidelberg. p. 317-324. (2010).
5. Sabeti, V., Samavi, S., Mahdavi, M. and Shirani, S., Steganalysis and payload estimation of embedding in pixel differences using neural networks. Pattern Recogn. 43(1): p. 405-415. (2010).
6. Mandal, S., Sivaprasad, P. V., Venugopal, S. and Murthy, K. P. N., Artificial neural network modeling to evaluate and predict the deformation behavior of stainless steel type AISI 304L during hot torsion. Applied Soft Computing. 9(1): p. 237-244. (2009).
7. Subudhi, B. and Morris, A. S., Soft computing methods applied to the control of a flexible robot manipulator. Applied Soft Computing. 9(1): p. 149-158. (2009).
8. Lee, K., Booth, D. and Alam, P., A comparison of supervised and unsupervised neural networks in predicting bankruptcy of Korean firms. Expert Systems with Applications. 29(1): p. 1-16. (2005).
9. Sharda, R. and Delen, D., Predicting box-office success of motion pictures with neural networks. Expert Systems with Applications. 30(2): p. 243-254. (2006).
10. Lera, G. and Pinzolas, M., Neighborhood based Levenberg-Marquardt algorithm for neural network training. IEEE Transaction on Neural Networks. 13: p. 1200-1203. (2002).
11. Ng, W. W. Y., Yeung, D. S. and Tsang, E. C. C. Pilot Study on the Localized Generalization Error Model for Single Layer Perceptron Neural Network. in Machine Learning and Cybernetics, 2006 International Conference on. (2006).
12. Wang, X. G., Tang, Z., Tamura, H., Ishii, M. and Sun, W. D., An improved backpropagation algorithm to avoid the local minima problem. Neurocomputing. 56: p. 455-460. (2004).

13. Otair, M. A. and Salameh, W. A. Speeding Up Back-Propagation Neural Networks. in Proceeding of the 2005 Informing Science and IT Education Joint Conference. Flagstaff, Arizona, USA. (2005).
14. Ji, L., Wang, X., Yang, X., Liu, S. and Wang, L., Back-propagation network improved by conjugate gradient based on genetic algorithm in QSAR study on endocrine disrupting chemicals. Chinese Science Bulletin. 53(1): p. 33-39. (2008).
15. Nazri, M. N., Ransing, R. S. and Ransing, M. S., An Improved Conjugate Gradient Based Learning Algorithm for Back Propagation Neural Networks. International Journal of Information and Mathematical Sciences. 4(1): p. 46-55. (2008).
16. Evett, I. W. and Spiehler, E. J., Rule induction in forensic science, in Knowledge Based Systems. Halsted Press. p. 152-160. (1988).
17. Michalski, R. S. and Chilausky, R. L., Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis. International Journal of Policy Analysis and Information Systems. 4:2. (1980).
18. Mangasarian, O. L. and Wolberg, W. H., Cancer diagnosis via linear programming. SIAM News. 23(5): p. 1-18. (1990).
19. Quinlan, J. R., C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning). Morgan Kaufmann. (1993).
20. Fisher, R. A., The use of multiple measurements in taxonomic problems. Annals of Eugenics. 7(2): p. 179-188. (1936).
21. Ye, Y. C., Application and Practice of the Neural Networks. Taiwan: Scholars Publication. (2001).
22. Maier, H. R. and Dandy, G. C., The effect of internal parameters and geometry on the performance of back-propagation neural networks: an empirical study. Environmental Modelling and Software. 13(2): p. 193-209. (1998).