The Effect of Adaptive Momentum in Improving the Accuracy of Gradient Descent Back Propagation Algorithm on Classification Problems

M.Z. Rehman and N.M. Nawi

Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia (UTHM).
P.O. Box 101, 86400 Parit Raja, Batu Pahat, Johor Darul Takzim, Malaysia
hi090004@siswa.uthm.edu.my, nazri@uthm.edu.my

Abstract. The traditional Gradient Descent Back-propagation Neural Network Algorithm is widely used in solving many practical applications around the globe. Despite providing successful solutions, it possesses a problem of slow convergence and sometimes getting stuck at local minima. Several modifications are suggested to improve the convergence rate of Gradient Descent Back-propagation algorithm such as careful selection of initial weights and biases, learning rate, momentum, network topology, activation function and 'gain' value in the activation function. In a certain variation, the previous researchers demonstrated that in "feed-forward algorithm", the slope of activation function is directly influenced by 'gain' parameter. This research proposed an algorithm for improving the current working performance of Back-propagation algorithm by adaptively changing the momentum value and at the same time keeping the 'gain' parameter fixed for all nodes in the neural network. The performance of the proposed method known as 'Gradient Descent Method with Adaptive Momentum (GDAM)'is compared with the performances of 'Gradient Descent Method with Adaptive Gain (GDM-AG)' and 'Gradient Descent with Simple Momentum (GDM)'. The learning rate is kept fixed while sigmoid activation function is used throughout the experiments. The efficiency of the proposed method is demonstrated by simulations on three classification problems. Results show that GDAM is far better than previous methods with an accuracy ratio of 1.0 for classification problems and can be used as an alternative approach of BPNN.

Keywords: gradient descent, neural network, adaptive momentum, adaptive gain.

1 Introduction

Artificial Neural Networks (ANN) are modeled to mimic the biological neurons in a human brain. Just like a human brain, ANN consists of processing units known as Artificial Neurons that can be trained to perform complex calculations. Unlike traditional methods in which an output is based on the input it gets, a neuron can be

J.M. Zain et al. (Eds.): ICSECS 2011, Part I, CCIS 179, pp. 380–390, 2011.

trained to store, recognize, estimate, and adapt to new patterns without having the information about the form of function. Since its advent, ANN has shown its mettle in solving many complex real world problems such as predicting future trends based on the huge historical data of an organization. ANN have been successfully implemented in all engineering fields such as biological modeling, decision and control, health and medicine, engineering and manufacturing, marketing, ocean exploration and so on [1-5].

A standard multilayer feed-forward neural network consists of an input layer, hidden layer and an output layer of neurons. Every node in a layer is connected to every other node in the neighboring layer. Back-Propagation Neural Network (BPNN) algorithm is the most popular and the oldest supervised learning multilayer feed-forward neural network algorithm proposed by Rumelhart, Hinton and Williams [6]. The BPNN learns by calculating the errors of the output layer to find the errors in the hidden layers. Due to this ability of Back-Propagating, it is highly suitable for problems in which no relationship is found between the output and inputs. Due to its flexibility and learning capabilities it has been successfully implemented in wide range of applications [7]. Although BPNN has been used successfully it has some limitations. Since it uses gradient descent learning rule which requires careful selection of parameters such as network topology, initial weights and biases, learning rate value, activation function, and value for the gain in the activation function should be selected carefully. An improper choice of these parameters can lead to slow network convergence, network error or failure. Seeing these problems, many variations in gradient descent BPNN algorithm have been proposed by previous researchers to improve the training efficiency. Some of the variations are the use of learning rate and momentum to speed-up the network convergence and avoid getting stuck at local minima. These two parameters are frequently used in the control of weight adjustments along the steepest descent and for controlling oscillations [8].

2 BPNN with Momentum Coefficient (α)

Momentum-coefficient (α) is a modification based on the observation that convergence might be improved if the oscillation in the trajectory is smoothed out, by adding a fraction of the previous weight change [6], [9]. Thus, the addition of momentum-coefficient can help smooth-out the descent path by preventing extreme changes in the gradient due to local anomalies [10]. So it is essential to suppress any oscillations that results from the changes in the error surface [11].

In initial studies, momentum-coefficient was kept fixed but later studies on static momentum-coefficient revealed that Back-propagation with Fixed Momentum (BPFM) shows acceleration results when the current downhill of the error function and the last change in weights are in similar directions, when the current gradient is in an opposing direction to the previous update, BPFM will cause the weight direction to be updated in the upward direction instead of down the slope as desired, so in that case it is necessary that the momentum-coefficient should be adjusted adaptively instead of keeping it fixed [12], [13].

Over the past few years several adaptive-momentum modifications are proposed by researchers. One such modification is Simple Adaptive Momentum (SAM) [14], proposed to further improve the convergence capability of BPNN. Sam works by scaling the momentum-coefficient according to the similarities between the changes in the weights at the current and previous iterations. If the change in the weights is in the same 'direction' then the momentum-coefficient is increased to accelerate convergence to the global minima otherwise momentum-coefficient is decreased. SAM is found to lower computational overheads then the Conjugate Gradient Descent and Conventional BPNN. In 2009, R. J. Mitchell adjusted momentum-coefficient in a different way than SAM [14], the momentum-coefficient was adjusted by considering all the weights in the Multi-layer Perceptrons (MLP). This technique was found much better than the previously proposed SAM [14] and helped improve the convergence to the global minima possible [15].

In 2007, Nazri *et al.* [16] found that by varying the gain value adaptively for each node can significantly improve the training time of the network. Based on Nazri *et al.* [16] research, this paper propose further improvement on the current working algorithm that will change the momentum value adaptively by keeping the gain value fixed.

3 The Proposed Algorithm

There are certain reasons for the slow convergence of the Gradient Descent Back propagation algorithm. Mostly, the magnitude and direction components of the gradient vector play a part in the slow convergence. When the error surface is fairly flat along a weight dimension, the derivative of the weight is small in magnitude. Therefore many steps are required and weights are adjusted by a small value to achieve a significant reduction in error. On the other hand, if the error surface is highly curved along a weight dimension, the derivative of the weight is large in magnitude. Thus, large weight value adjustments may overshoot the minimum of the error surface along that weight dimension. Another reason for the slow rate convergence of the gradient descent method is that the direction of the negative gradient may not point directly toward the minimum error surface [17].

In-order to increase the accuracy in the convergence rate and to make weight adjustments efficient on the current working algorithm proposed by Nazri *et al.* [16], a new Gradient Descent Adaptive Momentum Algorithm (GDAM) is proposed in the following section.

3.1 Gradient Descent Adaptive Momentum (GDAM) Algorithm

The Gradient Descent Back propagation uses two types of training modes which are incremental mode and batch mode. In this paper, batch mode training is used for the training process in which momentum, weights and biases are updated for the complete training set which is presented to the network. The following iterative algorithm known as Gradient Descent Adaptive Momentum Algorithm (GDAM) is proposed which adaptively changes the momentum while it keeps the gain and learning rate fixed for each training node. Mean Square Error (MSE) is calculated after each epoch and compared with the target error. The training continues until the target error is achieved.

```
For each epoch,
      For each input vector,
    Step-1:
      Calculate the weights and biases using the previous mo-
    mentum value
    Step-2:
      Use the weights and biases to calculate new momentum
    value.
      End input vector
IF Gradient is increasing, increase momentum
ELSE decrease momentum
Repeat the above steps until the network reaches the desired
value.
End epoch
```

The gradient descent method is utilized to calculate the weights and adjustments are made to the network to minimize the output error. The output error function at the output neuron is defined as;

$$E = \frac{1}{2}\sum_{k=1}^{n}(t_k - o_k(\alpha_k))^2 \tag{1}$$

An activation function plays a vital role in limiting the amplitude of the output neuron and generates an output value in any predefined range. Back propagation supports a lot of activation functions such as tangent, linear, hyperbolic and log-sigmoid function etc. This research will use log-sigmoid activation function which has a range of [0,1] in finding the output on the jth node;

$$O_j = \frac{1}{1 + e^{-a_{net,j}}} \tag{2}$$

where,

$$a_{net,j} = \left[\sum_{i=1}^{l} w_{ij} O_i\right] + \theta j \tag{3}$$

where,

n : number of output nodes in the output layer.
t_k : desired output of the k^{th} output unit.
o_k : network output of the k^{th} output unit.
O_j : Output of the *jth* unit.
O_i : Output of the *ith* unit.

W_{ij} : weight of the link from unit i to unit j.

$a_{net,j}$: net input activation function for the jth unit.

θ_j : bias for the jth unit.

$\frac{\partial E}{\partial \alpha_k}$, needs to be calculated for the output units and $\frac{\partial E}{\partial \alpha_j}$ is also required to be calculated for hidden units, so that the respective momentum value can be updated in the Equation (6):

$$\Delta \alpha_k = \left(- \frac{\partial E}{\partial \alpha_k} \right) \tag{4}$$

$$\Delta \alpha_j = \left(- \frac{\partial E}{\partial \alpha_j} \right) \tag{5}$$

$$\frac{\partial E}{\partial \alpha_k} = (t_{k\_}O_k)O_k(1 - O_k)(\sum w_{jk}O_j + \theta_k) \tag{6}$$

The momentum update expression from input to output nodes becomes;

$$\Delta \alpha_k(n+1) = (t_{k\_}O_k)O_k(1 - O_k)(\sum w_{jk}O_j + \theta_k) \tag{7}$$

$$\frac{\partial E}{\partial \alpha_j} = \left[- \sum_k \alpha_k w_{jk} (t_{k\_}O_k)O_k(1 - O_k) \right] O_j(1 - O_j) \left[\left[\sum_j w_{ij} O_i \right] + \theta_j \right] \tag{8}$$

Therefore, the momentum update expression for the hidden units is:

$$\Delta \alpha_j(n+1) = \left[- \sum_k \alpha_k w_{jk} (t_{k\_}O_k)O_k(1 - O_k) \right] O_j(1 - O_j) \left[\left[\sum_j w_{ij} O_i \right] + \theta_j \right] \tag{9}$$

Weights and biases are calculated in the same way, the weight update expression for the links connecting to the output nodes with a bias is;

$$\Delta w_{jk} = (t_{k\_}O_k)O_k(1 - O_k)\alpha_k O_j \tag{10}$$

Similarly, bias update expression for the output nodes will be;

$$\Delta \theta_k = (t_{k\_}O_k)O_k(1 - O_k)\alpha_k \tag{11}$$

The weight update expression for the input node links would be:

$$\Delta w_{ij} = \left[\sum_k \alpha_k w_{jk} (t_k - O_k) O_k (1 - O_k) \right] \alpha_j O_j (1 - O_j) O_i \tag{12}$$

And, finally the bias update expression for hidden nodes will be like this;

$$\Delta \theta_j = \left[\sum_k \alpha_k w_{jk} (t_k - O_k) O_k (1 - O_k) \right] \alpha_j O_j (1 - O_j) \tag{13}$$

4 Results and Discussions

Basically, the main focus of this research is to improve the Accuracy of the network convergence. Before discussing the simulation test results there are certain things that need be explained such as tools and technologies, network topologies, testing methodology and the classification problems used for the entire experimentation. The discussion is as follows:

4.1 Preliminary Study

The Workstation used for carrying out experimentation comes equipped with a 2.33GHz Core-2 Duo processor, 1-GB of RAM while the operating system used is Microsoft XP (Service Pack 3). The improved version of the proposed algorithms by Nazri [17] is used to carry-out simulations on MATLAB 7.10.0 software. For performing simulations, three classification problems like Breast Cancer (Wisconsin) [18], IRIS [19], and Australian Credit Card Approval [20] are selected. The following three algorithms are analyzed and simulated on the problems:

1) Gradient Descent with Simple Momentum (GDM),
2) Gradient Descent Method with Adaptive Gain (GDM-AG), [18] and
3) Gradient Descent with Adaptive Momentum(GDAM)

Three layer back-propagation neural networks is used for testing of the models, the hidden layer is kept fixed to 5-nodes while output and input layers nodes vary according to the data set given. Global Learning rate of 0.4 is selected for the entire tests and gain is kept fixed. While log-sigmoid activation function is used as the transfer function from input layer to hidden layer and from hidden layer to the output layer. In this research, the momentum term is varied adaptively between the range of [0,1] randomly . For each problem, trial is limited to 3000 epochs. A total of 15 trials are run for each momentum value. The network results are stored in the result file for each trial. Mean, standard deviation (SD) and the number of failures are recorded for each independent trial on Breast Cancer (Wisconsin) [18], IRIS [19], and Australian Credit Card Approval [20].

4.2 Breast Cancer (Wisconsin) Classification Problem

Breast Cancer Dataset was taken from UCI Machine Learning Repository databases. The dataset was created on the information gathered by Dr. William H. Wolberg [18] during the Microscopic study of breast tissue samples selected for the diagnosis of breast cancer. This problem deals with the classification of breast cancer as benign or malignant. The selected feed forward neural network architecture used for this classification problem is 9- 5-2. The target error is set to 0.01. The best momentum values for GDM and GDM-AG is 0.6 and 0.7 respectively. While for GDAM, the best momentum value is found in the interval $[0.3 - 0.8]$.

Table 1. Algorithm Performance for Breast Cancer Problem [19]

	Breast Cancer Problem, Target Error = 0.01		
	GDM	GDM-AG	GDAM
Accuracy (%)	92.84	94.0	94.71
SD	10.75	6.98	0.19
Failures	0	0	0

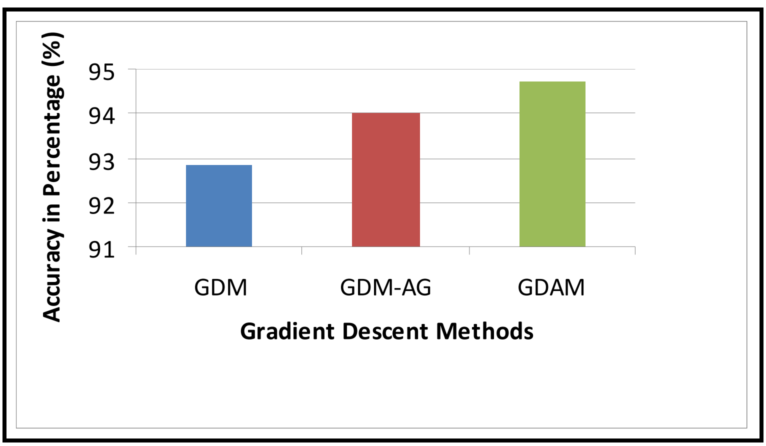

Fig. 1. Performance comparison of GDM, GDM-AG and GDAM for Breast Cancer Classification Problem

Table 1 show that the proposed algorithm (GDAM) shows far better performance in reaching the desired target error value of 0.01 than the previous improvements used for comparison. The proposed algorithm (GDAM) gives 94.71 percent accuracy when the network converges while GDM and GDM-AG is a left behind with 92.84 and 94.0 percentile accuracies respectively. Figure 1 clearly shows that GDAM outperforms GDM with an accuracy improvement ratio of 1.02.

4.3 IRIS Classification Problem

IRIS flower data set classification problem is one of the novel multivariate dataset created by Sir Ronald Aylmer Fisher [19] in 1936. IRIS dataset consists of 150 samples from Iris setosa, Iris virginica and Iris versicolor. Length and width of sepal and petals is measured from each sample of three selected species of Iris flower. The feed forward network is set to 4-5-2. The target error is set to 0.01. For Iris, the best momentum value for GDM and GDM-AG is 0.2. While for GDAM, the best momentum value is found in the interval $[0.6-0.8]$.

Table 2. Algorithm Performance for IRIS problem [20]

	IRIS Problem, Target Error = 0.01		
	GDM	GDM-AG	GDAM
Accuracy (%)	93.85	90.63	94.09
SD	0.23	3.46	1.09
Failures	1	3	0

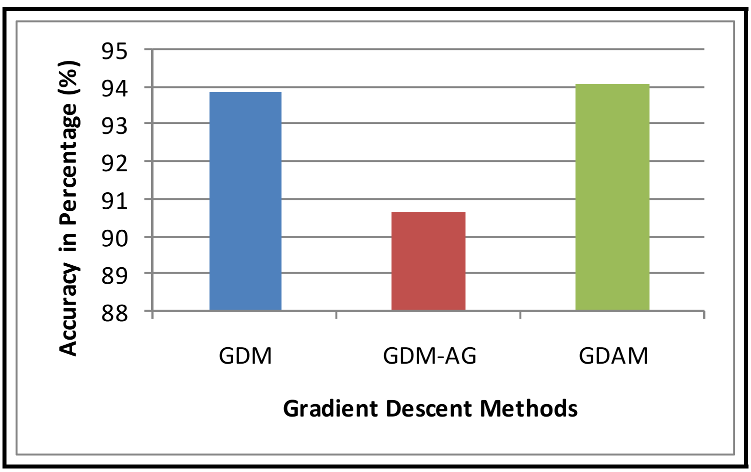

Fig. 2. Performance comparison of GDM, GDM-AG and GDAM for IRIS Classification Problem

From the Table 2, it is easily seen that the proposed algorithm (GDAM) is superior in performing convergence with a percentile accuracy of 94.09 which is better than the accuracies shown by GDM and GDM-AG. Also, it can be noted that GDM and GDM-AG have a failure rate of 1 and 3 respectively. In the Figure 2, GDAM can be seen outperforming GDM with an accuracy ratio of 1.0.

4.4 Australian Credit Card Approval Classification Problem

Australian Credit Approval dataset is taken from the UCI Machine learning repository. With a mix of 690 samples, 51 inputs and 2 outputs, this dataset contains all the details about credit card applications. Each sample in this data set represents a real credit card application and the output describes whether the bank (or similar institution) will grant the credit card or not. It was first submitted by Quinlan in 1987 [20]. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. The feed forward network architecture for this classification problem is set to 51-5-2. The target error is again set to 0.01. For Australian Credit Card Approval, GDM and GDM-AG both have the same best momentum value of 0.4. GDAM works best on the momentum value interval of $[0.7-0.8]$.

Table 3. Algorithm Performance for Australian Credit Card Approval [21]

	Echocardiogram Problem, Target Error = 0.01		
	GDM	GDM-AG	GDAM
Accuracy (%)	94.28	91.05	96.60
SD	1.15	11.87	0.53
Failures	1	1	0

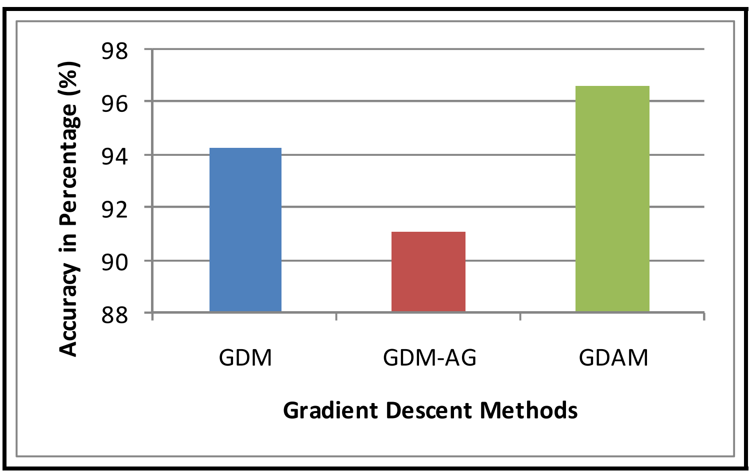

Fig. 3. Performance comparison of GDM, GDM-AG and GDAM for Australian Credit Card Approval Classification Problem

It is apparent from the Table 3, that GDAM is giving a percentile accuracy of 96.60 during convergence. Here also GDM and GDM-AG show 1 failed trial while there is no failure with GDAM even once on this classification problem. From the Figure 3, GDAM clearly shows an accuracy ratio of 1.02 on GDM.

5 Conclusions

The Back-propagation Neural Network (BPNN) is one of the most capable supervised learning algorithms deployed successfully in all engineering fields. Regardless of its high success rate at providing many practical solutions, it has a problem of slow convergence and network stagnancy which still needs to be answered. This paper proposed a further improvement on the current working algorithm by Nazri [17]. The proposed 'Gradient Descent Method with Adaptive Momentum (GDAM)' works by adaptively changing the momentum and keeping the 'gain' parameter fixed for all nodes in the neural network. The performance of the proposed GDAM is compared with 'Gradient Descent Method with Adaptive Gain (GDM-AG)' and 'Gradient Descent with Simple Momentum (GDM)'. The performance of GDAM is verified by means of simulation on the three classification problems like Breast Cancer (Wisconsin), IRIS, and Australian Credit-Card Approval respectively. The final results show that GDAM is far better than previous methods with an accuracy ratio of 1.0 for all classification problems.

Acknowledgments. The Authors would like to thank Universiti Tun Hussein Onn Malaysia (UTHM) for supporting this research under the Post Graduate Research Incentive Grant Vote No. 0737.

References

1. Kosko, B.: Neural Network and Fuzzy Systems, 1st edn. Prentice Hall of India, Englewood Cliffs (1994)
2. Krasnopolsky, V.M., Chevallier, F.: Some Neural Network application in environ-mental sciences. Part II: Advancing Computational Efficiency of environmental numerical models. Neural Networks 16(3-4), 335–348 (2003)
3. Coppin, B.: Artificial Intelligence Illuminated, USA. Jones and Bartlet Illuminated Series, ch.11, pp. 291–324 (2004)
4. Basheer, I.A., Hajmeer, M.: Artificial neural networks: fundamentals, computing, design, and application. J. of Microbiological Methods 43(1), 3–31 (2000)
5. Zheng, H., Meng, W., Gong, B.: Neural Network and its Application on Machine fault Diagnosis. In: ICSYSE 1992, September 17-19, pp. 576–579 (1992)
6. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representations by error Propagation. J. Parallel Distributed Processing: Explorations in the Microstructure of Cognition (1986)
7. Lee, K., Booth, D., Alam, P.A.: Comparison of Supervised and Unsupervised Neural Networks in Predicting Bankruptcy of Korean Firms. J. Expert Systems with Applications 16, 1–16 (2005)
8. Zweiri, Y.H., Seneviratne, L.D., Althoefer, K.: Stability Analysis of a Three-term Back-propagation Algorithm. J. Neural Networks 18, 1341–1347 (2005)
9. Fkirin, M.A., Badwai, S.M., Mohamed, S.A.: Change Detection Using Neural Network in Toshka Area. In: NSRC, 2009, Cairo, Egypt, pp. 1–10 (2009)
10. Sun, Y.J., Zhang, S., Miao, C.X., Li, J.M.: Improved BP Neural Network for Trans-former Fault Diagnosis. J. China University of Mining Technology. 17, 138–142 (2007)

11. Hamreeza, N., Nawi, N.M., Ghazali, R.: The effect of Adaptive Gain and adaptive Momentum in improving Training Time of Gradient Descent Back Propagation Algorithm on Classification problems. In: 2nd International Conference on Science Engineering and Technology, pp. 178–184 (2011)
12. Shao, H., Zheng, H.: A New BP Algorithm with Adaptive Momentum for FNNs Training. In: GCIS 2009, Xiamen, China, pp. 16–20 (2009)
13. Rehman, M.Z., Nawi, N.M., Ghazali, M.I.: Noise-Induced Hearing Loss (NIHL) Prediction in Humans Using a Modified Back Propagation Neural Network. In: 2nd International Conference on Science Engineering and Technology, pp. 185–189 (2011)
14. Swanston, D.J., Bishop, J.M., Mitchell, R.J.: Simple adaptive momentum: New algorithm for training multilayer Perceptrons. J. Electronic Letters 30, 1498–1500 (1994)
15. Mitchell, R.J.: On Simple Adaptive Momentum. In: CIS 2008, London, United Kingdom, pp. 1–6 (2008)
16. Nawi, N.M., Ransing, M.R., Ransing, R.S.: An improved Conjugate Gradient based learning algorithm for back propagation neural networks. J. Computational Intelligence. 4, 46–55 (2007)
17. Nawi, N. M.: Computational Issues in Process Optimization using historical data: PhD Eng. Thesis.Swansea University, United Kingdom (2007)
18. Wolberg, W.H., Mangasarian, O.L.: Multisurface method of pattern separation for medical diagnosis applied to breast cytology. National Academy of Sciences 87, 9193–9196 (1990)
19. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annual Eugenics 7, 179–188 (1936)
20. Quinlan, J.R.: Simplifying Decision Trees. J. Man-Machine Studies 27, 221–234 (1987)