Proceeding of the International Conference on Advanced Science, Engineering and Information Technology 2011

Hotel Equatorial Bangi-Putrajaya, Malaysia, 14 - 15 January 2011

ISBN 978-983-42366-4-9

The Effect of Adaptive Gain and Adaptive Momentum in Improving Training Time of Gradient Descent Back Propagation Algorithm on Classification Problems

Norhamreeza Abdul Hamid[#], Nazri Mohd. Nawi[#], Rozaida Ghazali[#]

[#] *Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia*

P. O. Box 101, 86400 Parit Raja, Batu Pahat, Johor, Malaysia

Tel.:+607-4538003, E-mail: gi090007@siswa.uthm.edu.my, nazri@uthm.edu.my, rozaida@uthm.edu.my

***Abstract*— The back propagation algorithm has been successfully applied to wide range of practical problems. Since this algorithm uses a gradient descent method, it has some limitations which are slow learning convergence velocity and easy convergence to local minima. The convergence behaviour of the back propagation algorithm depends on the choice of initial weights and biases, network topology, learning rate, momentum, activation function and value for the gain in the activation function. Previous researchers demonstrated that in 'feed forward' algorithm, the slope of the activation function is directly influenced by a parameter referred to as 'gain'. This research proposed an algorithm for improving the performance of the current working back propagation algorithm which is Gradien Descent Method with Adaptive Gain by changing the momentum coefficient adaptively for each node. The influence of the adaptive momentum together with adaptive gain on the learning ability of a neural network is analysed. Multilayer feed forward neural networks have been assessed. Physical interpretation of the relationship between the momentum value, the learning rate and weight values is given. The efficiency of the proposed algorithm is compared with conventional Gradient Descent Method and current Gradient Descent Method with Adaptive Gain was verified by means of simulation on three benchmark problems. In learning the patterns, the simulations result demonstrate that the proposed algorithm converged faster on Wisconsin breast cancer with an improvement ratio of nearly 1.8, 6.6 on Mushroom problem and 36% better on Soybean data sets. The results clearly show that the proposed algorithm significantly improves the learning speed of the current gradient descent back-propagatin algorithm.**

***Keywords*— back propagation algorithm, gain, activation function, adaptive momentum.**

I. INTRODUCTION

Artificial Neural Networks (ANN) are modelled on the human brain and consists of processing units known as artificial neurons that can be trained to perform complex calculations like human brain. It had been successfully implemented in the real world application which are accounting and finance [1], [2], health and medicine [3], [4], engineering and manufacturing [5], [6], marketing [7], [8] and general applications [9], [10], [11]. Multilayer perceptron is one of the most popular neural network models due to its clear architecture and comparably algorithm [12].

A standard multilayer perceptron consists of input layer, hidden layer and output layer. Each of these layers contains nodes. Each node in a layer is connected to the nodes in the subsequent layer. The most representative learning model for multilayer perceptron is back propagation algorithm. This algorithm has been successfully applied to wide range of practical problems [1], [13] which uses the gradient descent method to correct the network weights formula. A back propagation is a supervised learning technique that uses a gradient descent rule which attempts to minimize the error of the network by moving down the gradient of the error curve [14]. This algorithm is used more than all other combined and used in many different types of applications [15]. Although this algorithm is used successfully, it has some limitations. Since back propagation algorithm uses gradient descent method, the problems include a slow learning convergence and easily get trapped at local minima. Furthermore, the convergence behaviour of the back propagation algorithm depends on the choice of initial weights and biases, network topology, learning rate, momentum coefficient, activation function and value for the

gain in the activation function. Hence, improving the application of back propagation algorithm remains an important research issue.

In recent years, a number of research studies have attempted to overcome these problems. These involved the development of heuristic techniques, based on studies of properties of the conventional back propagation algorithm. These techniques include such idea as varying the learning rate, using momentum and gain tuning of activation function. In [16] some convergence results are given where the learning fashion of training examples is batch learning. These results are of global nature in that they are valid for any arbitrarily given initial value of weights. The key for the convergence analysis is monotonicity of the error function during the learning procedure, which is proved under the uniformly boundedness assumption of activation function and its derivatives. However, in order to obtain strong convergence, we assume the error function is equivalently convex, which is little intense. Kamarthi and Pittner [17] presented a universal acceleration technique for the back propagation algorithm based on extrapolation of each individual interconnection weight. This requires the error surface to have a smooth variation along the respective axes, therefore extrapolation is possible. For performing extrapolation, at the end of each epoch, the converge behaviour of each network weight in back propagation algorithm is individually examined. They also focused on the use of standard numerical optimization techniques. Though, this technique often must be tuned to fit a particular application. Møller [18] explained how conjugate gradient algorithm could be used to train multilayer feed forward neural networks. In this algorithm, a search is performed along conjugate directions, which generally leads to faster convergence than steepest gradient descent directions. The error function is guaranteed not to increase consequently of the weights update. However, if it reaches a local minimum, it remains forever, as there is no mechanism for this algorithm to escape. Lera et al. [19] 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.

Using a momentum coefficient is the simplest method to avoid oscillation problems during the search for the minimum value on the error surface [13]. The addition of momentum coefficient can help smooth out the descent path by preventing extreme changes in the gradient due to local anomalies [20]. Consequently, it is liable to suppress any oscillation that result from changes in the slope of the error surface. The momentum coefficient is typically chosen to be constant in the conventional back propagation algorithm with momentum. However, such a momentum with a fixed coefficient seems to speed up learning only when the current downhill gradient of the error function and the last change in weight have a similar direction, while the current negative gradient is in an opposing direction to the previous update, the momentum may cause the weight to be adjusted up the slope of the error surface instead of down the slope as desired [21]. In order to make learning more effective, it is necessary that the momentum should be varied adaptively rather than being fixed throughout the training process.

Nazri et al. [22] demonstrated that changing the 'gain' value adaptively for each node can significantly reduce the training time. Based on [22], this paper proposed a further improvement on the current algorithm that will change the momentum value adaptively which significantly improve the performance of the gradient descent back propagation algorithm. In order to verify the efficiency of the proposed algorithm, the performance of the proposed algorithm will be compare with the conventional gradient descent method (GDM) and gradient descent algorithm with adaptive gain (GDM/AG) proposed by Nazri et al. [22], some simulation experiments was performed on three classification problems including Wisconsin breast cancer [23], mushroom [24] and soybean [25].

The paper is organized as follows. In Section II, effect of the gain parameter on the performance of neural network is reviewed. While in section III presents the proposed algorithm. The performance of the proposed algorithm is tested on classification problems conducted in Section IV. This paper is concluded in the final section.

II. The Effect of The Gain Parameter on The Performance of Back propagation Algorithm

An activation function is a key factor in the artificial neural network structure. It is used for limiting the amplitude of the output of neuron and generates an output value for a node in a predefined range as the closed unit interval $[0,1]$ or alternatively $[-1,1]$. This value is a function of the weighted inputs of the corresponding node. Back propagation algorithm supports a wide range of activation functions such as logistic sigmoid, linear, hyperbolic tangent, step activation function and etc. The most commonly used activation function is the logistic sigmoid activation function. 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*

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

where,

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

where,

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.

α_i momentum coefficient for the i^{th} unit

The value of the gain parameter, c_j, and momentum coefficient α_i directly influence the slope of the activation function. For large gain values $(c \le 1)$, the activation function approaches a 'step function' whereas for small gain values $(0 < c \le 1)$ 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 neural networks tend to agree that neural networks 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, momentum coefficient, hidden nodes, etc. Also, a unit value for gain and momentum coefficient have generally been used for most of the research reported in the literature but a few authors have researched the relationship of the gain parameter and momentum coefficient with other parameters which used in back propagation algorithms. The recent results [27] show that learning rate, momentum coefficient and gain of the activation function have a significant impact on training speed. Unfortunately, higher values of learning rate and/or gain cause instability [28]. Thimm et al. [29] also proved that a relationship between the gain value, a set of initial weight values, and a learning rate value exists. Looney [30] suggested to adjust the gain value in small increments during the early iterations and to keep it fixed somewhere around halfway through the learning. Eom et al. [31] proposed a method for automatic gain tuning using a fuzzy logic system. Nazri et al. [22] proposed a method to change adaptively gain value on other optimisation method such as conjugate gradient.

III. The Proposed Algorithm

In this section, a further improvement on the current working algorithm proposed by Nazri [22] for improving the training efficiency of back propagation is proposed. The following subsection describes the proposed algorithm. The proposed algorithm adaptively changed the gain and momentum value for each node of training. The gradient descent can be implemented in two different ways which are incremental mode and batch mode. In this paper, batch mode was chosen to be implemented for training process. In the batch mode training weights, biases, gains and momentum terms are updated after one complete presentation of the entire training set. An epoch is defined as one complete presentation of the training set. A sum squared error value is calculated after the presentation of the training set and compared with the target error. Training is done on an epoch-by-epoch basis until the sum squared error falls below the desired target value.

A. Algorithm

The following iterative algorithm is proposed for the batch mode of training. The weights, biases, gains and momentum terms are calculated and update 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 value and momentum coefficient.
Step 2.
Use the weight and bias value calculated in ***Step (1)*** *to calculate the new gain value and momentum coefficient.*
Repeat ***Step (1) and Step (2)*** *for each input vector and sum all the weights, biases, momentum and gain updating terms.*
Update the weights, biases, gains and momentum coefficient using the summed updating terms and repeat this procedure on an epoch-by-epoch basis until the error on the entire training data set reduces to a predefined value

The gain and momentum update expression for a gradient descent method are calculated by differentiating the following error term *E* with respect to the corresponding gain parameter. The network error *E* is defined as follows

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

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 momentum values would then be updated with the following equations.

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

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

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

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

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

$$\frac{\partial E}{\partial \alpha_j} = \left[-\sum_k \alpha_k w_{jk} o_k \left(1 - o_k\right)\left(t_k - o_k\right)\right] o_j \left(1 - o_j\right)\left(\left(\sum_j w_{ij} o_i\right) + \theta_j\right) \tag{8}$$

and the momentum update expression for the links connecting hidden nodes is

$$\Delta \alpha_j(n+1) = \eta\left[-\sum_k \alpha_k w_{jk} o_k \left(1 - o_k\right)\left(t_k - o_k\right)\right] o_j \left(1 - o_j\right)\left(\left(\sum_j w_{ij} o_i\right) + \theta_j\right) \tag{9}$$

IV. Results and Discussions

The performance criterion used in this research focuses on the speed of convergence, measured in number of iterations and CPU time. The benchmark problems used to verify our algorithm are taken from the open literature. Three classification problems have been tested including Wisconsin breast cancer [23], mushroom [24] and soybean [25]. The simulations have been carried out on a Pentium IV with 2 GHz HP Workstation, 3.25 GB RAM and using MATLAB version 7.0 (R14).

On each problem, the following three algorithms were analysed and simulated.

- The conventional Gradient Descent with Momentum (GDM)
- The Gradient Descent Method with Adaptive Gain (GDM/AG) [22]
- The proposed Gradient Descent Method with Adaptive Gain and Adaptive Momentum (GDM/AGAM)

To compare the performance of the proposed algorithm with conventional GDM and GDM/AG [22], network parameters such as network size and architecture (number of nodes, hidden layers etc), values for the initial weights and gain parameters were kept the same. For all problems the neural network 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, initialized randomly from range $[0,1]$ and received the input patterns for training in the same sequence.

For all training algorithms, the learning rate is fixed to be 0.3 which is interpreted as the global learning rate of the network. However, as the gain value was modified, the weights and biases were updated using the new value of gain. This resulted in higher values of gain which caused instability [29]. To avoid oscillations during training and to achieve convergence, an upper limit of 2.0 is set for the gain value. The initial value used for the gain parameter is one. The momentum term is randomly generated from range $[0,1]$ by using trial and error method. The best momentum term value is selected. For each run, the numerical data is stored in two files - the results file, and the summary file. The result file lists data about each network. The number of iterations until convergence is accumulated for each algorithm from which the mean, the standard deviation and the number of failures are calculated. The networks that fail to converge are obviously excluded from the calculations of the mean and standard deviation but are reported as failures. For each problem, 100 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 100 runs, the mean of the number of iterations (mean), the standard deviation (SD), and the number of failures are collected. A failure occurs when the network exceeds the maximum iteration limit; Wisconsin breast cancer [23] is run to 5000 iterations, mushroom [24] is run to 1000 iterations and soybean [25] is run to 3000 iterations; otherwise, it is halted and the run is reported as a failure. Convergence is achieved when the outputs of the network conform to the error criterion as compared to the desired outputs.

A. Breast Cancer Classification Problem

This dataset was created based on the 'Breast Cancer Wisconsin' problem dataset from UCI repository of machine learning databases from Dr. William H. Wolberg [23]. This problem tries to diagnosis of breast cancer by trying to classify a tumor as either benign or malignant based on cell descriptions gathered by microscopic examination. The selected architecture of the Feed-forward Neural Network is 9-5-2. The target error is set as to 0.001. The best momentum term value for conventional GDM and GDM/AG for the Wisconsin breast cancer is 0.4 while GDM/AGAM is initialized randomly from range $[0.1, 0.5]$.

TABLE I
Algorithm Performance for Breast Cancer Problem [23]

	Breast Cancer Problem, Target Error = 0.001		
	GDM	**GDM/AG**	**GDM/AGAM**
Mean	1356	1165	783
Total CPU time(s) of converge	24.9664	22.4136	13.7313
CPU time(s)/Epoch	1.84×10^{-2}	1.92×10^{-2}	1.75×10^{-2}
SD	5.76×10^{2}	8.12×10^{2}	7.09×10^{2}
Failures	0	0	0

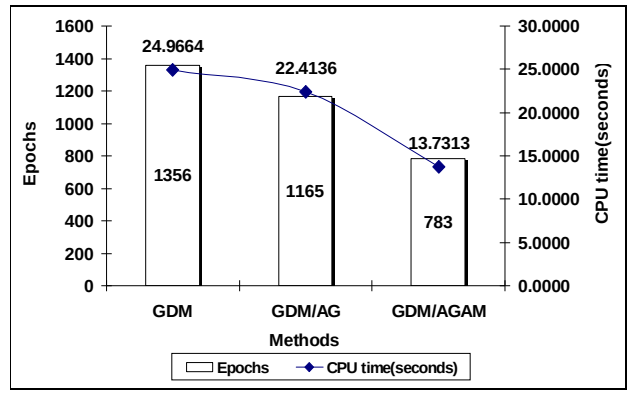

Fig. 1 Performance comparison of GDM/AGAM with GDM/AG and conventional GDM for Breast Cancer Classification Problem

Table I shows that the proposed algorithm (GDM/AGAM) exhibit very good average performance in order to reach target error. The proposed algorithm (GDM/AGAM) needs only 783 epochs to converge as opposed to the conventional GDM at about 1356 epochs while GDM/AG needs 1165 epochs to converge. Apart from speed of convergence, the time required for training the classification problem is another important factor when analyzing the performance. For numerous models, training process may suppose a very important time consuming process. The results in Fig. 1 clearly show that the proposed algorithm (GDM/AGAM) outperform conventional GDM with an improvement ratio, 1.8 seconds for the total time of converge.

B. Mushroom Classification Problem

This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The selected architecture of the Feed-forward Neural Network is 125-5-2. The target error is set as to 0.001. The best momentum term value for conventional GDM and GDM/AG for the Mushroom problem is 0.2 while GDM/AGAM is initialized randomly from range $[0.1, 0.9]$.

TABLE II
ALGORITHM PERFORMANCE FOR MUSHROOM PROBLEM [24]

	Mushroom Problem, Target Error = 0.001		
	GDM	**GDM/AG**	**GDM/AGAM**
Mean	997	414	146
Total CPU time(s) of converge	104.7044	44.4598	15.8311
CPU time(s)/Epoch	1.05×10^{-1}	1.07×10^{-1}	1.08×10^{-1}
SD	3.21×10^{1}	4.09×10^{2}	1.21×10^{2}
Failures	1	32	99

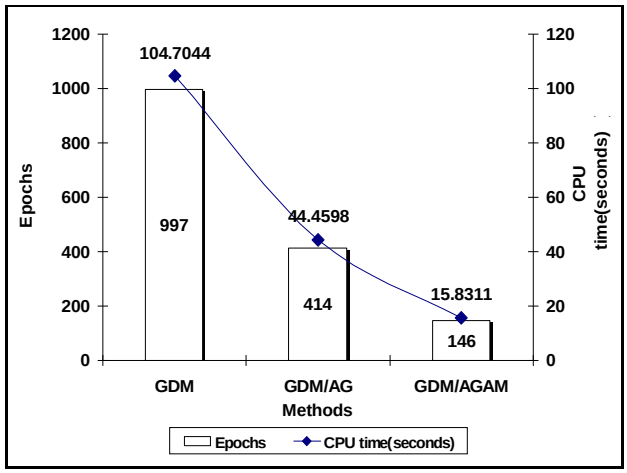

Fig. 2 Performance comparison of GDM/AGAM with GDM/AG and conventional GDM for Mushroom Classification Problem

Fig. 2 shows that the proposed algorithm (GDM/AGAM) still outperforms with other algorithms in terms of CPU time and number of epochs. The proposed algorithm (GDM/AGAM) only required 146 epochs in 15.8311 seconds CPU times to achieve the target error, whereas GDM/AG required 414 epochs in 44.4598 seconds CPU times and GDM required 997 epochs in 104.7044 seconds to achieve the target error. As we can see in the Table II, the number of success rate for the proposed algorithm (GDM/AGAM) was 99% as compared to GDM in learning the patterns. Besides, the conventional GDM did not perform well in this dataset since 99% of the simulation results failed in learning the patterns. While the average number of learning iterations for the proposed algorithm (GDM/AGAM) was reduced up to 6.6 times faster as compared to GDM. The result shown that the GDM/AGAM perform better as compared to GDM and GDM/AG.

C. Soybean Classification Problem

Soybean is a well known propositional data set. This dataset contains 82 inputs, 19 outputs, and 683 examples [26]. The selected architecture of the Feed-forward Neural Network is 82-5-19. The target error is set as to 0.001. The best momentum term value for conventional GDM and GDM/AG for the Soybean problem is 0.3 while GDM/AGAM is initialized randomly from range $[0.0, 0.8]$.

TABLE III
ALGORITHM PERFORMANCE FOR SOYBEAN PROBLEM [25]

	Soybean Problem, Target Error = 0.001		
	GDM	**GDM/AG**	**GDM/AGAM**
Mean	1837	1381	1081
Total CPU time(s) of converge	65.57456	56.6430	42.40813
CPU time(s)/Epoch	3.57×10^{-2}	4.10×10^{-2}	3.92×10^{-2}
SD	1.10×10^{3}	1.32×10^{3}	9.84×10^{2}
Failures	0	2	35

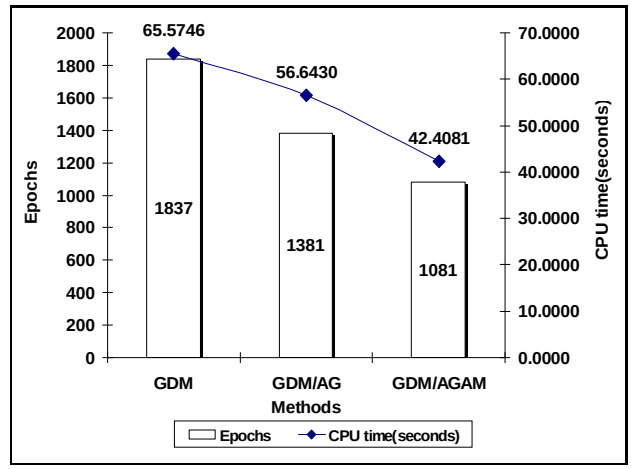

Fig. 3 Performance comparison of GDM/AGAM with GDM/AG and conventional GDM for Soybean Classification Problem

Table III reveal that GDM needs 65.5746 seconds with 1837 epochs to converge. Whereas GDM/AG needs 56.6430 seconds and 1381 epochs to converge. Conversely, the proposed algorithm (GDM/AGAM) performed significantly better with only needs 42.4081 seconds with 1081 epochs to converge. From Fig. 3, it is worth noticing that the performance of the GDM/AGAM is almost 1.55 faster than GDM. Still the proposed algorithm (GDM/AGAM) performs better as compared to GDM and GDM/AGM.

The simulation results from three classification problems allow to compares the proposed algorithm (GDM/AGAM) with conventional GDM and algorithm which proposed by Nazri et al. [22] (GDM/AG) in terms of CPU time, speed of convergence and measured in number of iterations. Consequently, we can claim that, the proposed algorithm

(GDM/AGAM) presents better performance than other algorithm which are conventional GDM and GDM/AG. This conclusion enforces the usage of proposed algorithm as alternative training algorithm of gradient descent back propagation algorithm.

V. Conclusions

Although back propagation algorithm is widely implemented in the most practical neural networks 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 [22]. The proposed algorithm adaptively change the gain parameter of the activation function together with momentum coefficient to improve the learning speed. The effectiveness of the proposed algorithm has been compared with the conventional Gradient Descent Method (GDM) and Gradient Descent Method with Adaptive Gain (GDM/AG) [22], verified by means of simulation on three classification problems including Wisconsin breast cancer with an improvement ratio nearly 1.8 for the total time of converge, mushroom almost 6.6 faster respectively and soybean took almost 36% less time to converge by using batch mode training. The result shows that the proposed algorithm (GDM/AGAM) has a better convergence rate and learning efficiency as compared to conventional Gradient Descent Method (GDM) and Gradient Descent Method with Adaptive Gain (GDM/AG) [22].

Acknowledgment

The authors would like to thank **Universiti Tun Hussein Onn Malaysia** for supporting this research under the Postgraduate Incentive Research Grant (Vote 0682).

References

[1] 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*, vol. 29, no. 1, pp.1–16, 2005.

[2] Landajo, M., Andres, J. D., and Lorca, P., "Robust Neural Modeling for the Cross-Sectional Analysis of Accounting Information," *European Journal of Operational Research*, vol. 177. no. 2, pp. 1232–1252, 2007.

[3] Razi, M. A., and Athappily, K., "A Comparative Predictive Analysis of Neural Networks (NNs), Nonlinear Regression and Classification and Regression Tree (CART) Models," *Expert Systems with Applications*, vol. 2, no. 1, pp. 65–74, 2005.

[4] Behrman, M., Linder, R., Assadi, A. H., Stacey, B. R., and Backonja, M. M., "Classification of Patients with Pain Based on Neuropathic Pain Symptoms: Comparison of an Artificial Neural Network against an Established Scoring System," *European Journal of Pain*, vol. 11, no. 4, pp. 370–376, 2007.

[5] Yesilnacar, E., and Topal, T., "Landslide Susceptibility Mapping: A Comparison of Logistic Regression and Neural Networks Methods in a Medium Scale Study, Hendek region (Turkey)," *Engineering Geology*, vol. 79 no. 3–4, pp. 251–266, 2005.

[6] Dvir, D., Ben-Davidb, A., Sadehb, A., and Shenhar, A. J., "Critical Managerial Factors Affecting Defense Projects Success: A Comparison between Neural Network and Regression Analysis," *Engineering Applications of Artificial Intelligence*, vol. 19, pp. 535–543, 2006.

[7] Gan, C., Limsombunchai, V., Clemes, M., and Weng, A., "Consumer Choice Prediction: Artificial Neural Networks versus Logistic Models," *Journal of Social Sciences*, vol. 1, no. 4, pp. 211–219, 2005.

[8] Chiang, W. K., Zhang, D., and Zhou, L., "Predicting and Explaining Patronage Behavior toward Web and Traditional Stores Using Neural Networks: A Comparative Analysis with Logistic Regression," *Decision Support Systems*, vol. 41, pp. 514–53, 2006.

[9] Chang, L. Y., "Analysis of Freeway Accident Frequencies: Negative Binomial Regression versus Artificial Neural Network. Safety Science," vol. 43, pp. 541–557, 2005.

[10] Sharda, R., and Delen, D., "Predicting box-office success of motion pictures with neural networks," *Expert Systems with Applications*, vol. 30, pp. 243–254, 2006.

[11] Nikolopoulos, K., Goodwin, P., Patelis, A., and Assimakopoulos, V., "Forecasting with cue information: A comparison of multiple regression with alternative forecasting approaches," *European Journal of Operational Research*, vol. 180, no. 1, pp. 354–368, 2007.

[12] H. Yan, Y. Jiang, J. Zheng, C. Peng and Q. Li, "A Multilayer Perceptron-Based Medical Decision Support System for Heart Disease Diagnosis, " *Expert Systems with Applications*, vol. 30, no. 2, pp. 272–281, 2006.

[13] Zou, H., Xia, G., Yang, F. and Yang, H., "A Neural Network Model Based On The Multi-Stage Optimization Approach For Short-Term Food Price Forecasting In China." *Expert Systems with Applications*, vol. 33, no. 2, pp. 347-356, 2007.

[14] Mutasem, K. S. A., Khairuddin, O. and Shahrul, A. N., "Back Propagation Algorithm: The Best Algorithm among the Multi-layer Perceptron Algorithm," *International Journal of Computer Science and Network Security*, vol. 9, no. 4, pp. 378 – 383, 2009.

[15] A. Majdi and M. Beiki, "Evolving Neural Network Using Genetic Algorithm for Predicting the Deformation Modulus of Rock Masses," I*nternational Journal of Rock Mechanics and Mining Science*, vol. 47, no. 2, pp. 246–253, 2010.

[16] N. M. Zhang, W, Wu, G. F. Zheng, "Deterministic Convergence of Gradient Method with Momentum for Two-layer Feed Forward Neural Networks,"*IEEE Transactions on Neural Networks*, vol. 17, no. 2, pp. 522−525, 2006.

[17] Kamarthi S. V., Pittner S., "Accelerating Neural Network Training using Weight Extrapolations," *Neural Networks*, vol. 12, pp. 1285−1299, 1999.

[18] Møller M. F., "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, vol. 6, no. 4, pp. 525−533, 1993.

[19] Lera G., Pinzolas M., "Neighborhood based Levenberg-Marquardt Algorithm for Neural Network Training," *IEEE Transaction on Neural Networks*, vol. 13, no. 5, pp. 1200−1203, 2002.

[20] Sun, Y. J, Zhang, S. Miao, C.X. and Li, J. M., "Improved BP Neural Network for Transformer Fault Diagnosis," *Journal of China University of Mning Technology*, vol. 17, no. 1, pp. 138−142, 2007.

[21] H. Shao, G. Zheng, "A New BP Algorithm with Adaptive Momentum for FNNs Training," Intelligent Systems, WRI Global Congress on, pp. 16–20, 2009 WRI Global Congress on Intelligent Systems, 2009.

[22] Nazri Mohd Nawi, M. R. Ransing, R. S. Ransing, "An Improved Conjugate Gradient Based Learning Algorithm for Back Propagation Neural Networks. International Journal of Computational Intelligence," vol. 4, no. 1, pp. 46−55, 2007.

[23] Mangasarian O. L., Wolberg W. H., "Cancer Diagnosis via Linear Programming," *SIAM News*, vol. 23(5), pp. 1−18, 1990.

[24] UCI Machine Learning Repository [Online]. Available: ftp://ftp.ics.uci.edu/pub/ machine-learning-databases/mushroom/.

[25] R.S. Michalski and R.L. Chilausky , "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*, vol. 4, No. 2, pp. 125-160, 1980.

[26] D.B. Leake and D.C. Wilson, *Remembering why to remember: performance-guided case-base maintenance*, ser. Lecture Notes in Computer Science, Berlin, Germany: Springer, 2000, vol. 1898.

[27] Holger R. M., Graeme C. D., "The Effect of Internal Parameters and Geometry on the Performance of Back propagation Neural Networks," *Environmental Modeling and Software*, vol. 13, no. 1, pp. 193−209, 1998.

[28] Hollis P. W., Harper J. S., Paulos J. J., "The Effects of Precision Constraints in a Backpropagation Learning Network," *Neural Computation*, vol. 2, no. 3, pp. 363−373, 1990.

[29] Thimm G., Moerland F., Fiesler E., "The Interchangeability of Learning Rate and Gain in Backpropagation Neural Networks," *Neural Computation*, vol. 8, no. 2, pp. 451−460, 1996.

[30] Looney C. G., “Stabilization and Speedup of Convergence in Training Feed Forward Neural Networks,” *Neurocomputing*, vol. 10, no. 1, pp. 7–31, 1996.
[31] Eom K., Jung K., Sirisena H., “Performance Improvement of Backpropagation Algorithm by Automatic Activation Function Gain Tuning Using Fuzzy Logic,” *Neurocomputing*, vol. 50, pp. 439–460, 2003.