Predicting for Sustainable Insurance with Adaptive Gradient Methods

Parveen Sehgal[1], **Sangeeta Gupta**[2] and **Dharminder Kumar**[3]

Submitted in January 2015; Accepted in March 2015

Abstract - This paper extends the comparison of gradient based training methods used in the construction of prediction models based upon neural network, for sustainable insurance. Here adaptive gradient based techniques are compared with simple first order gradient based technique and with some second order training techniques for learning of the network. Convergence towards minimum error, for a number of first and second order algorithms are compared while utilizing data taken from live data warehouse of life insurance. Method of back propagation of errors is adopted for training of multilayer feed forward networks, while employing these gradient based algorithms of training error reduction. This paper is extended version of the paper presented in IEEE Conference INDIACom-2014.

Index Terms – Adaptive gradient algorithms, Error back-propagation, Error gradient, Multilayer-perceptron, Neural network, Supervised training, Sustainable insurance.

NOMENCLATURE

SDI – Sustainable Development Indicator
GD – Gradient Descent
CGM – Conjugate Gradient Method
SCGM – Scaled Conjugate Gradient Method
LMA– Levenberg Marquardt Algorithm
GDA – Gradient Descent with Adaptive Learning Rate
GDM – Gradient Descent with Adaptive Momentum

1.0 INTRODUCTION

Insurance industry need to exploit new market opportunities that will come from the shift of economic development to sustainable development. It has the power to inspire and create a greener society and plays a critical role to improve the adverse economic, social and environmental consequences of financial losses arising from fortuitous and accidental events. Sustainable Insurance provides a solution to inflate the innovative risk management and insurance ways that we need to promote saving of natural resources and to shape a safe future for the coming generations. One of the major goals is to offer low premium insurance policies and to support the rural communities in developing countries and to offer retirement planning solutions across the globe especially in rural areas.

[1]*Research Scholar, Dept.of CSE, NIMS University, Jaipur, India, (Corresponding Author), Tel:+91-9896931768.*
[2]*Prannath Parnami Institute for Professional Studies, Hisar, India.*
[3]*Department of CSE, Guru Jambheshwar University of Science & Technology, Hisar, India.*
E-mail: [1]parveensehgal@gmail.com,
[2]sangeet_gju@yahoo.co.in and [3]dr_dk_kumar_02@yahoo.com

Insurance in rural areas is a key sustainable development indicator (SDI) to safeguard the future of people in rural areas, which will otherwise have adverse effects on the usage of Based on the historical data; we have an urgent need to predict the prospective customers in rural areas and to launch new policies to ensure sustainable development. But due to complexity of nonlinear relationships present between input and output variables in the bulk amount of historical data and these types of problems are difficult to solve with older techniques. Various technologies of soft computing like Fuzzy Logic, genetic algorithms, evolutionary algorithms and artificial neural networks can be used to create the prediction models; which is related to field of nonlinear optimization [1][5]. The idea is to find an optimal solution for the complex nonlinear relationship between input and output variable; which generally exists in high dimensional data of real life problems [8].

In this paper, we extend the construction of prediction models based upon neural network and trained with adaptive gradient based techniques to find an approximation for the complex nonlinear relationship present in the insurance data, within the desired limits of accuracy [15]. Neural Networks are employed to solve complex problems of optimization in the areas of software engineering and data mining [17]. The gradient based algorithms used for the training of network optimize the weights present between the layers of the network until a state of minima of error gradient is reached. We descend along a multi-dimensional error gradient surface in a direction towards the minima of gradient, and proceed iteratively in small steps until we reach at the point of minima of the error gradient. We extend the study of gradient based training and compare the convergence and accuracy of adaptive gradient based techniques with first order and second order techniques.

2.0 LITERATURE SURVEY & LEARNING METHODOLOGIES

Taylor approximation of the error energy present during neural network training is written as:

$$E(W_k + z) \approx E(W_k) + E'(W_k)^T z + \tfrac{1}{2} z^T E''(W_k) z$$

When higher order terms are neglected then, first order and second order approximations for E is written as:

$$E_1(W_k + z) \approx E(W_k) + E'(W_k)^T z$$

$$E_2(W_k + z) \approx E_1 + \tfrac{1}{2} z^T E''(W_k) z$$

For minimizing of error function based on the weight vector present in neural network, we have used a number of first and second order approximation techniques.

Steepest descent (simple gradient descent) uses first order information to descend toward the point of local minimum. But this method shows a poor convergence and uses fixed value of learning rate parameter and is not useful for practical applications that require a faster speed for output [2]. A careful selection of the step size is extremely important for faster convergence toward point of minima. With larger values of the step size it will diverge and if taken too small it will take longer times to converge. Also, complex shape of the multi-dimensional error surface usually presents irregularities and makes moving toward global minima problematic. Researchers have suggested modifications to improve the convergence and avoid stagnant learning and oscillations by introducing methods with varying step size. Proposed methods utilize learning rate and momentum parameters to decide for the optimal step size [18, 19].
When error is computed according to second order approximation then first order information is updated and a second order minimization step is performed. We have also applied second order methods like conjugate gradient, scaled conjugate gradient, which have their own positives and negatives in terms of computer memory required, convergence speed and accuracy.
Earlier methods like Newton's method shows a faster convergence toward point of minimum training error than first order methods, because it also utilizes second order information and approximates the second order terms by evaluating the hessian matrix [4]. But computation of hessian becomes very tedious and time consuming when number of input variables goes high and weight vector is large in size[6]. This consumes a large memory and processing time and acts as a bottleneck in reaching towards the point of minima.
An improvement on simple Newton Method is Quasi–Newton technique in which tedious computations of hessian are replaced by an approximation for the hessian [7, 9]. The Levenberg Marquardt Algorithm (LMA) provides a trust region approach to Gauss–Newton Algorithm [12] and interpolates between Gauss–Newton Algorithm and steepest descent algorithm. However, LMA is considered more powerful than Gauss–Newton Algorithm, because of its capability to find a solution in cases, when starting point is very far off the point of minimum error.
Method of conjugate gradient (CGM) bypasses the computation of second order derivatives, and the idea is to restrict the search directions toward paths that are orthogonal to all previous searches [11]. Advantages of CGMare that they consume relatively lesser memory for large size problems and each step is quite fast [4]. It utilizes a simple line search to find the required step size in the search direction and jumps deep inside valley of error gradient. The line search avoids the tedious calculations of the hessian matrix; but needs to calculate the error gradient at a number of points in search direction.CGM saves the time for complex computation of second order derivatives but still line search along conjugate directions every time is very time consuming. The scaled conjugate gradient method (SCGM) avoids the need of time consuming line search [12, 13] and is applicable to larger networks. This method suppresses the instability in computation by combining the trust region approach of LMA with the CGM approach [12]. SCGM regulates the indefiniteness in tedious calculation of hessian with a scalar value and computes the optimal step size, without complex and costly calculations done during line search by the standard CGM.

3.0 FIRST & SECOND ORDER ALGORITHMS EMPLOYED FOR TRAINING OF NETWORK

We have applied following techniques for training the predictive neural networks.

3.1 Gradient Descent (GD) Algorithm

We move towards the point of minimum of error gradient along error surface in very small steps and descend in opposite direction of the error gradient [14]. New updated weight vector is computed as:

$$w_{ji_{next}} = w_{ji_{prev}} + \Delta w_{ji} \tag{5}$$

A control parameter η is introduced as shown in Eq. (5) to have an extra control over the speed of training which can regulate the amount of overall corrective adjustment applied to weight vector.

$$w_{ji_{next}} = w_{ji_{prev}} + \eta(T_j - O_j)I_i \tag{6}$$

But here learning rate parameter η is kept constant leading to a slow convergence towards minima. In areas where the error surface is flat and error derivative is small in magnitude and here a small value of step size is required to find a significant decrease in error. On the contrary, if the error surface is highly curved and the derivative is large in magnitude and here small values will reduce the speed of convergence. Some variations are suggested to improve in these kinds of situations [20, 23, 24, 25].

3.2 Gradient descent with adaptive learning rate (GDA)

GDA attempts to keep the learning step size large for the speedy convergence and also keeping the training stable. Learning rate is varied according to the complexity of the error surface. If the new error surpasses the old error by a predefined value, the new weight vector is rejected. Also, the rate of learning parameter is decreased by multiplying with fractional value which is less than 1. Otherwise, the new weight vector is retained. If the new error is lesser than previous error, the learning rate is further increased by multiplying with a factor slightly greater than 1 [21, 23].
Silva and Almeida implemented the same idea in a simpler way and suggested the changes to be done in constant fractional values. Rate of learning in the training algorithm is enhanced by multiplying with a fixed fractional value slightly more than 1, generally 1.20 and reduced by multiplying with the reciprocal of fixed fractional value (i.e. 1/1.20), just to increase

or decrease by a small fraction. Momentum term can also participate in the algorithm; but is kept at fixed value and is non-adaptive in this method [21]. A backtracking process is utilized to keep an eye on the continuous increase in the error values and to revert back after a number of iterations to the historical points of lesser error values as accomplished in the preceding iterations.

3.3 Gradient descent with adaptive momentum (GDM)

GDM allows the network to be sensitive not only to the error gradient, but also respond to the latest trends in the error surface. Momentum term allows the network to ignore small irregularities on the error surface. Without momentum term a network can be caught in a small narrow local minimum; and momentum helps to slide through such a minimum. The parameter momentum constant α determines the amount of influence of previous iterations on the on-going iterations and hence can be called to approximate the second order algorithms [14, 22].

Here, delta rule now becomes:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} + \alpha \Delta w_{ji}\ (n-1)$$

The reduced learning due to momentum parameter tries to slow down the search in irregular areas on the error surface and increased learning rate enhances the search in smooth areas on the error surface [21].

In initial studies, momentum parameter was kept fixed but later studies discovered that this technique suffers from unnecessary acceleration, when the current error gradient is in opposite direction to the previous searches. This results to move in the upward direction instead of going down the slope as required. Therefore, it is necessary that the momentum to be adjusted adaptively instead of setting it to a constant value [19, 21].

3.4 Conjugate Gradient Method (CGM)

Early methods involve the calculation and storing of second order derivatives of error energy, the hessian and its inverse, which are very tedious and memory consuming, when the number of prediction variables is very large and overall weight matrix is large in size. Hence, in these situations it is better to employ algorithms like CGM; which avoids the computation of second order derivatives but still achieves quadratic termination. To avoid the need of second order derivatives, while descending across the error surface, we restrict the search direction to the paths that are orthogonal to all previous searches. An enhancement over the steepest descent method, the conjugate gradient method comprises of these computational steps:

1. Choose the initial search direction opposite to error gradient i.e. $d_0 = -g_0$.
2. Compute an optimal value for the training parameter α_k for minimum value of the gradient function with help of line search method.
$$x_{k+1} = x_k + \alpha_k d_k$$
3. Find out a new search direction orthogonal to all previous searches and compute the parameter β_k such that:
$$d_{k+1} = -g_{k+1} + \beta_k d_k$$
4. If convergence to point of minima is not reached or the stopping conditions set for training are not met, then proceed to second step of algorithm.

Researchers have developed a number of variations for calculation of parameter β_k such as Hestenes and Stiefel variation, Fletcher and Reeves variation, Polak and Ribiere variation are widely used for practical purposes. The main advantage of conjugate gradient algorithm is that it uses relatively less memory when applied to bigger networks and bypass computations for second order derivatives. But on the negative side the line search method may require more computation for an optimal value of the parameter α_k [11].

3.5 Scaled Conjugate Gradient Method (SCGM)

Similar to CGM, while computing the error gradient, SCGM bypasses the tedious and memory consuming computation of complex hessian matrix and computes an approximation which is close to second order derivatives. A new search direction d_k ; but a new step size α_k are calculated every time during k^{th} iteration in order to update the weight vector for training of network, such that:

$$E(W_k + \alpha_k d_k) < E(W_k)$$

The quadratic approximation on the error surface $E(W_k)$ in a neighboring point of the weight vector W_k is given by the Taylor's approximation as [16]:

$$E(W_k + w) \approx E(W_k) + E'(W_k)^T w + \tfrac{1}{2} w^T E''(W_k) w$$

But it is very time and memory consuming to exactly calculate the hessian $E''(W_k)$ and particularly when weight vector is large in size, therefore, second order information S_k is approximated in terms of first order derivatives as:

$$S_k = E''(W_k) d_k \approx \frac{E'(W_k + \sigma_k d_k) - E'(W_k)}{\sigma_k}$$

for $0 < \sigma_k \leq 1$

Where, $E(W_k)$ denotes the multidimensional error surface and W_k is the weight matrix for k^{th} iteration, $E'(W_k)$ is the error gradient. And parameter σ_k denotes the incremental change in the network weights for this second order method. In this algorithm, the trust region approach of Levenberg Marquardt algorithm is used with the conjugate gradient approach to calculate the next step size [12]. Variable c_k is used to control the indefiniteness of $E''(W_k)$. This is done by computing the second order information as:

$$S_k = \frac{E'(W_k + \sigma_k d_k) - E'(W_k)}{\sigma_k} + c_k d_k$$

And at every step, we keep on computing value $\delta_k = d_k{}^T S_k$ for checking the indefiniteness of $E''(W_k)$. We keep on adjusting c_k and keep looking at the sign of δ_k in each iteration, which checks that hessian $E''(W_k)$ is not positive definite. When $\delta_k \le 0$; c_k is increased and S_k is calculated again. The new weight vector is now calculated as:

$$W_{k+1} = W_k + \alpha_k d_k$$

, till we reach the point of minima or stopping conditions for the training are met.

4.0 EXPERIMENTAL OBSERVATIONS AND RESULTS

Training functions of first and second order (traingd, traingda, traingdm, traincgp, trainscg) available in MATLAB Neural toolbox package were employed to train the neural networks. Large data sets taken from live data warehouse are employed for experimentation and model development. Training performance based on Mean Squared Error (MSE), gradient plot for algorithm convergence are observed to check for the behavior and efficiency of gradient algorithms under consideration. A variety of neural networks architectures are tested for development of desired prediction models. But two layered architecture with 15 neurons is observed as the optimal configuration with data sets employed under similar hardware/software configurations.

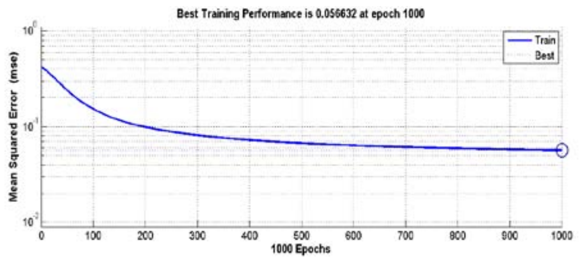

Figure 1: Performance plot when employing simple gradient descent learning

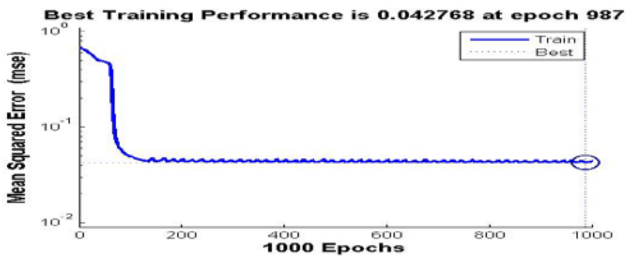

Figure 2: Performance plot when employing gradient descent with adaptive learning

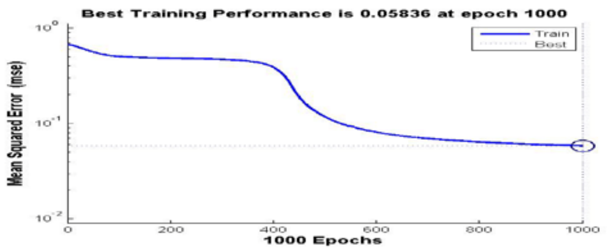

Figure 3: Performance plot when employing gradient descent with adaptive momentum

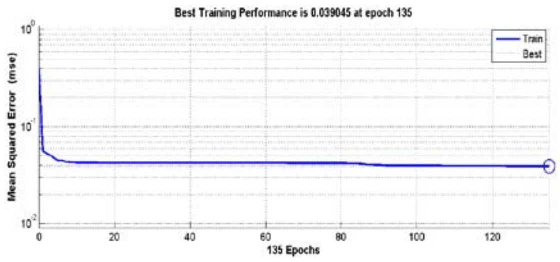

Figure 4: Performance plot when employing conjugate gradient learning

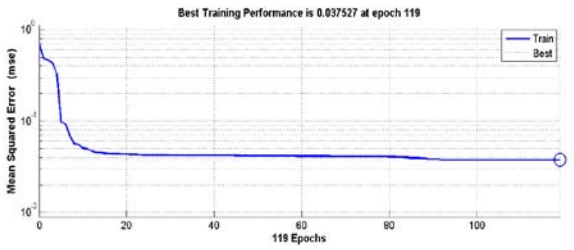

Figure 5: Performance plot when employing scaled conjugate gradient learning

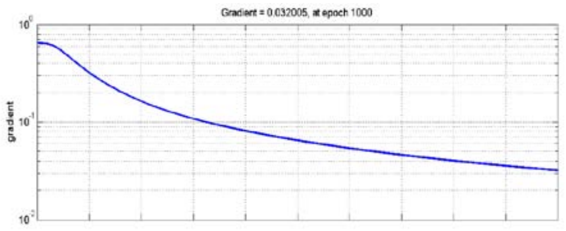

Figure 6: Error gradient plot when employing simple gradient descent learning

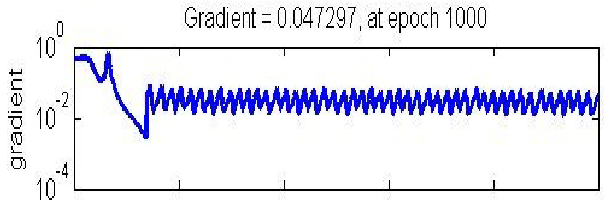

Figure 7: Error gradient plot when employing gradient descent with adaptive learning

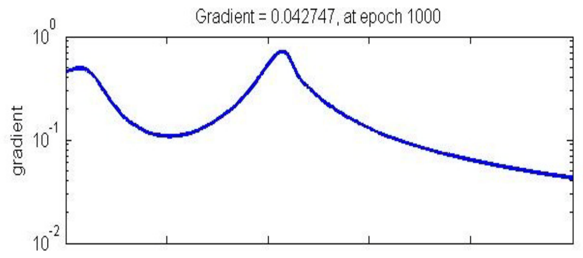

Figure 8: Error gradient plot when employing gradient descent with adaptive momentum

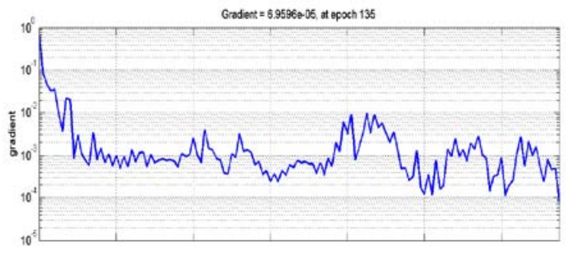

Figure 9: Error gradient plot when employing conjugate gradient learning

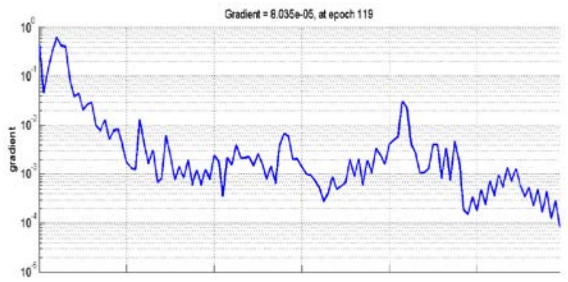

Figure 10: Error gradient plot when employing scaled conjugate gradient learning

Figures 1 to 5 show changes in mean square error (MSE) verses numbers of epochs achieved in the best cases while applying the simple, adaptive(learning, momentum) and second order gradient methods. Error gradient plots, while employing different training methods are shown in Figures 6 to 10. It has been observed that gradient decent, adaptive learning and adaptive momentum methods were unable to accomplish the error gradient of 10^{-4} even in the 1000th epoch but second order methods like CGM converged in 135 epochs and SCGM in 119 epochs.

Table 1: Experimental results of employing different gradient based learning algorithms of first and second order

Training Method	**Steepest (Gradient) Descent**	**Gradient descent with adaptive learning rate**	**Gradient descent with momentum**	**Conjugate Gradient (Pollok Variation)**	**Scaled Conjugate Gradient**
Training Function	traingd	traingda	traingdm	traincgp	trainscg
Hidden Layer Neurons	15	15	15	15	15
MinimumGradient	0.0001	0.0001	0.0001	0.0001	0.0001
TransferFunction	TANSIG	TANSIG	TANSIG	TANSIG	TANSIG
Final No. of Epochs	10^3	10^3	10^3	135	119
TrainingTime	0:16:14	0:14:39	0:15:27	0:07:24	0:04:41
TrainingPerformance	0.0566	0.0427	0.05836	0.0390	0.0375
InitialGradient Value	0.6460	0.4470	0.4470	0.6760	0.4470
FinalGradient Value	3.20E-02	0.0473	0.0427	6.96E-05	8.04E-05

5.0 CONCLUSION

The main focus of the research is to improve the speed and accuracy of convergence in network training. Convergence of adaptive methods is compared with simple gradient descend and second order methods. It is concluded that adaptive gradient based method are slightly better than simple gradient but still there performance is not comparable to second order techniques. Adaptive methods took lesser training time than simple gradient but they were not able to converge even in 1000 epochs toward error gradient of the order of 10^{-4}. While on the other hand, second order techniques were able to reach an accuracy level of 10^{-4} and 10^{-5} and prove far better in terms of training time and accuracy. Simple gradient (GD) with constant learning rate has shown the poor convergence, conjugate and scaled conjugate gradient methods (CGM, SCGM) show the fastest convergence and adaptive methods (GDA, GDM) fall in between, in terms of convergence towards the minimum error gradient.

REFERENCES

[1]. S. V. Chande and M. Sinha, "Genetic Algorithm: A Versatile Optimization Tool", *BIJIT - BVICAM's International Journal of Information Technology*, Vol. 1, No. 1, pp. 7-12, Jan.-Jun. 2009.

[2]. J. C. Meza, "Steepest descent", *Wiley Interdisciplinary Reviews: Computational Statistics*, Vol. 2, No. 6, pp. 719-722, Dec. 2010.

[3]. S. Osowski, P. Bojarczak, and M. Stodolski, "Fast second order learning algorithms for feed forward multilayer neural networks and its applications", *Neural Networks, Elsevier*, Vol. 9, No. 9, pp. 1583–1596, Dec. 1996.

[4]. R. Battiti, "First & second order methods for learning between steepest-descent & Newton's method", MIT Press, *Neural Computation*,vol. 4(2), pp. 141-166, Mar. 1992.

[5]. A. K. Verma, R. Anil and Dr. O. P. Jain, "Fuzzy Logic Based Revised Defect Rating for Software Lifecycle Performance Prediction Using GMR", *BIJIT - BVICAM's International Journal of Information Technology*, Vol. 1, No. 1, pp. 1-6, Jan.-Jun. 2009.

[6]. R. Rojas, *Neural Networks - A Systematic Introduction*, Berlin, Springer, 1996.

[7]. A. Likas, and A. Stafylopatis, "Training the random neural network using Quasi–Newton methods", *Euro. Jour. of Operational Research, Elsevier*, Vol. 126, Issue 2, pp. 331-339, Oct. 2000.

[8]. R. Rastogi, S. Agarwal, P. Sharma, U. Kaul and S. Jain, "Business Analysis and Decision Making Through Unsupervised Classification of Mixed Data Type of Attributes Through Genetic Algorithm", *BIJIT - BVICAM's International Journal of Information Technology*, Vol. 6, No. 1, pp. 683-689, Jan.-Jun. 2014.

[9]. S. M. A. Burney, T. A. Jilani, and C. Ardil, "A Comparison of first and second order training algorithms for artificial neural networks", *International Journal of Computational Intelligence*, Vol. 1, No. 3, pp. 176-182, 2005.

[10]. M. T. Hagan, and M. B. Menhaj, "Training feed forward networks with Marquardt algorithm", *IEEE Tran. on Neural Networks*, Vol. 5, No. 6, pp. 989-993, Nov. 1994.

[11]. E. K. P. Chong, and S. H. Zak, *An Introduction to Optimization*, 2ndEdition, John Wiley and Sons, 2001.

[12]. M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning", *Neural Networks, Elsevier*, Vol. 6, Issue 4, pp. 525–533, 1993.

[13]. J. Lunden, and V. Koivunen, "Scaled conjugate gradient method for radar pulse modulation estimation", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Proceedings ICASSP'07, Vol. 2, pp. 297–300, Apr.2007.

[14]. M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*, Chapter 12, PWS Publishing, Boston, 1996.

[15]. S. Goel, J. B. Singh and A. K. Sinha, "Traffic Generation Model For Delhi Urban Area Using Artificial Neural Network", *BIJIT - BVICAM's International Journal of Information Technology*, Vol. 2, No. 2, pp. 239-244, Jul.-Dec. 2010.

[16]. Swanston D. J., Bishop J. M., and Mitchell R. J., "Simple Adaptive Momentum New Algorithm for training Multilayer Perceptron, *J. Engineering Letters*, Vol. 30 (18), pp. 1498-1500, 1994.

[17]. G. Kumar and P. K. Bhatia, "Optimization of Component Based Software Engineering Model Using Neural Network", *BIJIT - BVICAM's International Journal of Information Technology*, Vol. 6, No. 2, pp. 732-742, Jul.-Dec. 2014.

[18]. Y. H. Zweiri, L. D. Seneviratne, and K. Althoefer, "Stability Analysis of a Threeterm Back Propagation Algorithm", *Neural Networks,Elsevier*,vol. 18, no. 10, 2005.

[19]. M. Z. Rehman, and N. M. Nawi, "Studying the Effect of Adaptive Momentum in Improving the Accuracy of Gradient Descent Back Propagation Algorithm on Classification Problems", *International Journal of Modern Physics, World Scientific,*Vol. 1(1),pp. 1–5, 2010.

[20]. R. A. Jacobs, "Increased Rate of Convergence through Learning-Rate Adaptation", *Elsevier*, *Neural Networks*; 1:295–307, 1988.

[21]. M. Moreira, and E. Fiesler, *Technical-Report IDIAP, Neural Networks with Adaptive Learning Rate & Momentum Terms*, No.95-04, Oct.1995.

[22]. Y. Bai, H. Zhang , and Y. Hao, "The performance of the back propagation algorithm with varying slope of the activation function", *Chaos, Solitons and Fractals, Elsevier*, 40, pp.69–77, 2009.

[23]. Saduf, M. A. Wani, "Comparative Study of Back Propagation Learning Algorithms for Neural Networks", International *Journal of Advanced Research in*

Computer Science and Software Engineering, Volume 3(12), pp. 1151-1156, December 2013.

[24]. M. Z. Rehman, and N. M. Nawi, “Improving the Accuracy of Gradient Descent Back Propagation Algorithm on Classification Problems”, *Int. Journal on New Computer Architectures and Their Applications*,1(4): pp. 838-847, 2011.

[25]. G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis, “Improving Convergence of the Back-Propagation Algorithm Using Learning-Rate Adaptation Methods”, MIT Press, *Neural Computation*,vol. 11, No. 7,pp. 1769–1796 Oct. 1999.