International Journal of Academic Research in Computer Engineering
Volume 1, Number 1, Pages: 40-46, September 2016
www.ijarce.org

IJARCE

An Improved Multilayer Perceptron Artificial Neural Network with Genetic Algorithm for Software Cost Estimation

Shin-ichi Oda*, and Osaka Nakazato

Department of Computer and Information Science, Seikei University, JAPAN

Receive Date: 10 June 2016; Accepted Date: 5 August 2016, Published Date: 15 September 2016

*Corresponding Author: shinichi@yahoo.com (S. Oda)

Abstract
Nowadays, Software Cost Estimation (SCE) with high precision has been one of the challenging main complex issues for software companies and their executives in software engineering. In the past several decades, the use of Artificial Neural Network (ANN) models in this field has been more efficient compared to traditional techniques which are based on algorithmic methods. ANN models which work on the basis of the data obtained from the previous projects have the advantage of increasing the accuracy of estimation when faced with similar projects during the steps of software life cycle. In this article, we have used perceptron which is a multilayer perceptron (MLP) ANN model for better estimation; in order to optimize this model, we have utilized a strong method called Genetic Algorithm (GA) and the simulation results indicate that the proposed model is more optimal than Algorithmic COCOMOII and MLP ANN models.

Keywords: Software Cost Estimation, Multilayer Perceptron Artificial Neural Network, Genetic Algorithm.

1. Introduction

Given the fact that software is getting more and more complex day by day, potentially, the importance of research on the estimation of software effort has increased too. Accurate estimation of effort and costs required at the beginning of software life cycle is very vital for software companies [1]. The more accurate and creditable estimating the effort necessary for a project, the more successful will be guiding, planning and completing the project [2]. Accurate cost estimation is important in that a low cost estimation may cause losing or endangering software quality; in short, the software made with partial or inadequate performance will later end in high maintenance cost. On the contrary, if the cost estimation is considered to be high, it will cause unfair allotting of resources [3]. Estimating precocious cost in the initial phases of software life cycle is important because it reduces the likelihood of organizational conflict during the later stages of project management [4].

Algorithmic and mathematical methods are among the first models to be created in order to SCE in the past decades. These models are COCOMO- based. These models were made on the basis of regression analysis to find the relationship between the cost drivers and actual effort value [5]. Cost drivers which include a number of features related to software development such as software size, and project features are considered as input parameters to calculate the overall effort output which is mainly based on month- person [3]. Among them, COCOMO 81 model was one of the first and the best methods for comparing the performance of proposed new procedures [2].

With the increasing advance of technology in recent years, algorithmic models are no longer able to make an accurate estimation for software projects, which is the reason why we have come to

feel the need to have anti-algorithmic models based on ANN methods [6]. They include decision-making trees, ANN and probable networks, fuzzy systems, complementary algorithms, and other methods made to estimate software costs [7]. Among them, ANNs are powerful in estimation because of such advantages as self-learning, community storage, high paralleling strength, high speed and error tolerance against noises which may exist in input parameters, and also their inexpensiveness in reusing existent solutions [8, 9]. The overall aim of this study is to recognize affective factors on the cost of software projects to lessen the complexity of the model using GA, in the case that without missing accuracy, we will ignore some of input parameters and doing so we will increase the accuracy of the estimation made. Moreover, evaluating and investigating various architectures of ANNs including the number of layers and the neurons used exist in its topology.

This paper is organized as follows: In the Section 2, we will review the previous works related to SCE. In the Section 3, we will investigate the selected database and prepare the set of datasets and then implement and simulate the hybrid model of MLP ANN with GA. In the Section 4, we will evaluate the experiments and compare the results of the proposed method with the models used in the previous work along with the simple ANN. And in the Section 5, we will state the conclusions and suggestions for future works.

2. Related Works

In recent decades, many techniques have been made on the basis of artificial intelligence so as to estimate software project costs. Among artificial intelligence techniques, ANNs have attracted a lot of research in the field of the estimation of software projects costs due to their ability to learn and modeling complex non-linear relations [10]. Following, we will express several ANN techniques and other hybrid models which use GA to choose the suitable features of software projects. MLP ANN which is the most common architecture of ANN is mainly utilized for the issue of estimating the software project costs. An adaptive learning method based on MLP ANN technique has been offered to SCE [11]. In this paper, we expressed the impact of the value of parameters and the type of network topology to reach a project cost estimation model with high precision. The proposed model is on the basis of the architecture of COCOMOII model. ANN model has been trained by the projects of two databases called COCOMO and COCOMO NASA63. The results show that the performance of model based on MRE criterion is better than COCOMO model. Another new model based on MLP ANN techniques has been offered in [12]. Network input parameters in this work is in congruence with the factors of middle COCOMO model which have been driven from the completed software projects. This model has also been used to compare his work. The topology of the model is composed of three-layer architecture with 18 neurons in the input layer, 2 neurons in the hidden layer and one neuron in the output layer. His case study is implemented based on C # language and using pre-release algorithm for network training.

The database consisted of 60 NASA projects related to aeronautics. Based on MRE of the experiments, it is indicated that more than 90% of the results, NN techniques were better than COCOMO algorithmic model.

In [13] MLP ANN model has been used with cluster analysis. Lee`s proposed model consists of two phases. The first phase is used for clustering similar projects to facilitate network training. With this technique, similar projects share similar development cost and this similarity is used as a valuable piece of information to increase the effectiveness of network training. And according to the results of tests, the proposed method leads to better performance.

Another study based on MLP ANN has been used according to back propagation training algorithm to update the weights and network connections in [14]. The architecture of MLP ANN considered with three hidden layer and the initial selection of weights randomly and threshold level equals to 0/01. The results of the first experiment show that the performance of MLP ANN model and regression are identical, but in the second test, ANN works better than regression model. The result of MAPE criterion for code line indicator is 601% and 102/74% for the indicator of function points.

Another reseach in [15] based on the architecture of MLP ANN. This architecture that uses back propagation training algorithm is based on suck architecture with sigmoid function in hidden layer neurons. The input layer consisting of 23 nodes corresponding to 7 scale factor parameters, 15 effort coefficient parameters and one parameter related to product size which is regarded as a thousand lines of source code.

The hidden layer has only one layer and the output layer equals one which shows the cost value of software projects. The adaptive model which is called the COCOMOII_ANN is utilized for training and evaluating the two COCOMOI and NASA93 datasets. Assessment criteria for showing

the performance of the proposed network are MMRE and PRED (25), so that these values for the average of MMRE and (PRED (25) are 4579/0 and 45/5%, whereas the two evaluation criteria for algorithmic model of COCOMOII dataset are 0.5025 and 37/5%, respectively.

Furthermore, another work was done in [16] based on MLP ANN for the adaptation of architectural post COCOMOII model. In the meanwhile, he has used expert judgment to reach better performance. Expert judgment has been put forward for ordering network training on the basis of the adjustment of network weights. The select architecture of his proposed method comes from the input layer with 24 nodes, 17 of which are related to effort factors, 5 related to scale factor, and 2 related to bias. Effort factors are pre-processed with the help of logarithm function and the hidden layer is considered with sigmoid activation function. The connections of inter-layer nodes are not completely connected. He has used pre-releasing training algorithm for network training with COCOMOII datasets. And in each network training going on, expert judgment is used for better training before updating each network weight.

3. Proposed Model

The aim of this paper is to create a model based on machine learning for SCE according to Intermediate COCOMO model using MLP ANN (Figure 1). Also, GA is used to lessen the complexity of the model in a way that the selection of appropriate features from among the total dataset features to reduce the costly estimation process is done by only related features. All in all, the process of estimating software project costs has three phases. The first phase is related to collecting pre-processing the datasets; the second phase is related to selecting architecture and network training. And the third phase is related to reducing model costs by choosing appropriate features using GA and comparing performance and reaching the best answer.

Criterion data collections that existed with middle COCOMO model in range of software projects includes three datasets. The first and foremost is COCOMO81 data set which consists of 63 software projects in different programming languages such as FORTRAN, Pascal and C. The nature of the projects includes trade, processing control, human-machine interaction, support, and the system [11]. They are made up of 17 parameters that include the coefficient of effort parameters or cost drivers and the size of the number of program lines (LOC) [13]. In this paper, COCOMO81 dataset is used for training ANNs.

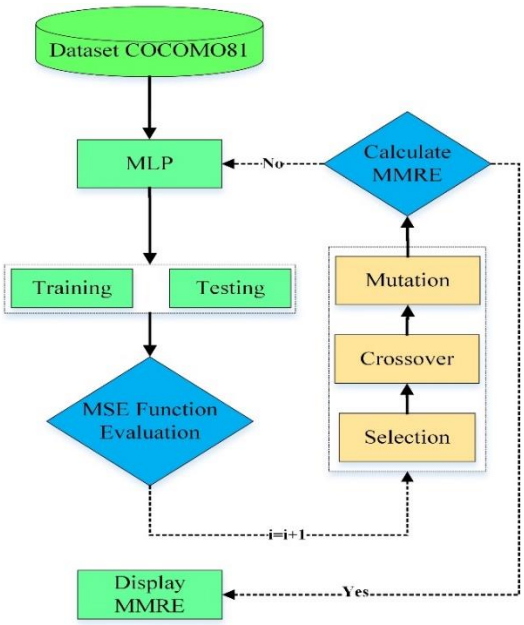

Figure 1. Flowchart of Proposed Model

The most common MLP architecture is feed forward architecture which usually uses back propagation learning algorithm [17]. This architecture is known as MLP ANN, which contains three units, input layer, hidden layer and output layer [18]. The number of the nodes of input layer for the issue of cost estimation is consistent with cost drivers. The number of nodes in the input layer is directly related to the number of cost drivers. [19] The number of nodes in the hidden layer is variable and is chosen intentionally; one node in the output layer shows the estimated effort. In the proposed model, the number of input nodes is equal with the number of the features of COCOMO81 dataset which are the same as cost drivers and program size. The number of nodes in the hidden layer which is regarded on the basis of trial and error as 14 and this value in Table (1) investigates the performance of the network by changing the number of hidden nodes and shows how to reach the optimal value. It is proved that if the number of nodes increases from a given value, it will not only result in cost increase but also to ultra-connection in the network [18]. And the output node is always considered to be 1 for estimation issue.

In addition to the number of nodes, the type of activation function in turn affects network performance so that the best choice of this kind is also made through trial and error done [20]. There are generally two types of linear and nonlinear activation functions; activation function in the hidden groups is non-linear and usually all neurons in a single layer have the same activation function.

The most common activation function in hidden nodes is usually considered to be sigmoid [21] which in turn has two kinds logsig is tansig [22].
Table (1) also shows the performance of MLP ANN in terms of the type of activation function. Thus, on the basis of performance (the total square of training and testing error) obtained in accordance with the training data i.e., COCOMO81, logsig activation function has its highest performance. According to the following table, we consider logsig activation function for the hidden layer neurons and linear identity function for the output layer neurons.

Table 1. Selecting the Most Appropriate Number of Nodes in the Hidden

MSE(tansig)	MSE(logsig)	Number of Node
0.0917	0.0625	2
0.0371	0.0519	4
0.0528	0.0472	6
0.0282	0.0155	8
0.0309	0.0137	10
0.0637	0.0337	12
0.0057	0.0047	14
0.0089	0.0054	16
0.0336	0.0125	18
0.0350	0.0104	20
0.0208	0.0221	22
0.0366	0.0176	24
0.0146	0.0109	26
0.0987	0.0805	28

Training algorithm in an MLP ANN is done based on a set of samples for training network. Each instance consists of the desired inputs and outputs, and these data enter the network through the input layer and are processed by neurons and goes on layer by layer until it makes conclusions with output layer. The difference between actual output and the output considered in an error value is calculated [12]. Based on the error value, weight values or network connections are set gradually [23].
And this work from the hidden layer through the input layer is returned to the output layer until the desired output is produced [24]. Back propagation algorithm is the most widespread algorithm for training feed forward MLP ANN [12]. The algorithm uses gradient descending algorithm technique to adjust the weights and biases in the opposite direction of function tilt [25].
Back propagation algorithm suffers from slowness in convergence. Another weak point of pre-propagation error algorithm is that the process of searching or finding the minimum is carried out only gradient descent [26]. This makes the local minimum fall. While Lonbergomarquart is used to evaluate tilt shifts using second-order derivative method based on Hessain matrix [27].
In the case that, it gets very fast and efficient. Lonbergomarquart which is neuton-gawsy optimization algorithm is utilized to find the most minimum squares between the actual and the predicted output of the network [27]. Estimation issue on the basis of a total dataset of features related to a training dataset is complex and time-consuming. But, if we choose an optimal and related subset among all the existent features, we can remarkably improve estimation issue or better to say machine learning [19]. This optimization not only reduces the complexity but also enhances the performance of machine learning [2]. Since GA is a method for searching efficient, parallel solution based on survival principal [28] we will use it to choose appropriate features and reach better performance of ANN to do training on the basis of all the features. We will choose the most appropriate features from among the 16 features related to COCOMO 81 dataset using GA; we will show experimentally how to improve the performance of Perseptron ANN. The results of implementing MLP reduction model on COCOMO 81 dataset based on MSE criterion are shown according to equation (1) in Figure (2) and Table (2) [4].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \qquad (1)$$

Train, Validation, Test and Best criteria are shown in Figure (2):

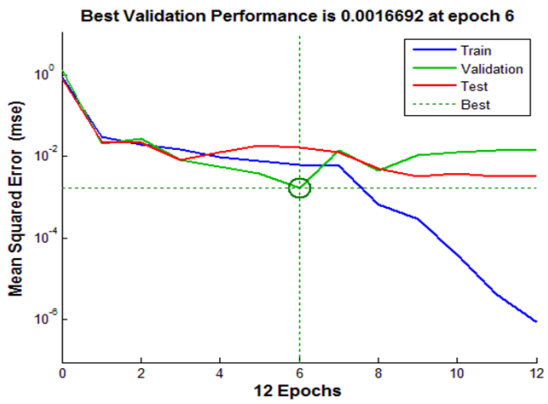

Figure 2. Performance of Improved GA-MLP ANN in the Training Process

Table [2] indicates that GA has reduced the value of MSE error to some extent.

Table 2. The Results of GA Model with MLP ANN

Neuron	35	30	20	18	15	8
MSE (test)	0.0011	2.439 e-04	0.0017	1.9546e 04	1.0210 e-04	0.0347
MSE (train)	0.0010	3.398 e-05	5.709 e-5	7.610e-05	1.015e -05	0.004325

4. Result and Discussion

The evaluation carried out in this paper was based on MATLAB programming language test on COCOMO81 dataset. This evaluation includes training MLP ANN based on Lonbergomark Algorithm and the hybrid model of GA and MLP ANN. First of all, the amount of error resulting from estimating the effort of MLP network model with the effort amount of algorithmic COCOMO model on 63 projects of COCOMO81 dataset is depicted in Figure (3) to show the optimal performance of ANN. In most of the cases, the amount of error in ANN model is much less.

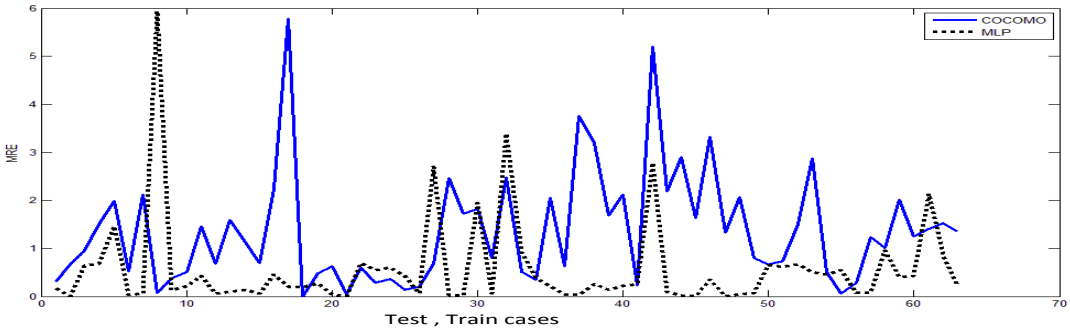

Figure 3. A Comparing the SCE of COCOMO model with MLP ANN

The chart of the MSE error of reduction GA model with MLP network in comparison with the chart of algorithmic model error is given in Figure (4). Furthermore, by comparing the chart of this hybrid model using appropriate features in comparison with the chart of MLP models with the total features of dataset, we will obtain the better performance of hybrid GA model with ANN.

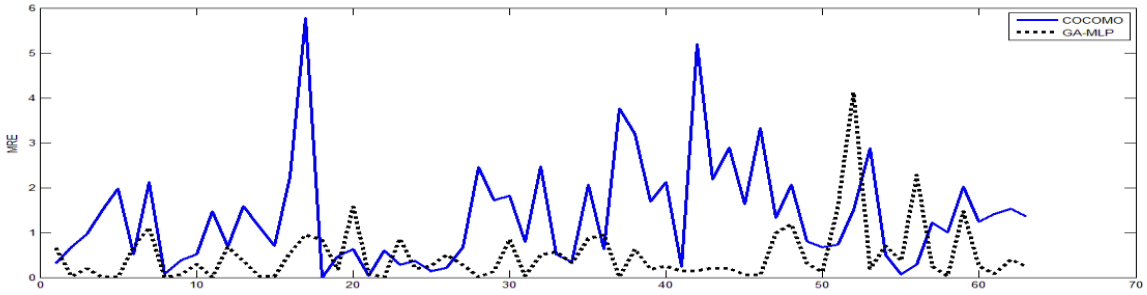

Figure 4. A Comparison of SCE of COCOMO model with GA

The results of training the MLP ANN with a number of features are presented in Figure 4. In addition, it is shown to prove more accurate performance according to several evaluation criteria. The average of square testing error for the hybrid GA-MLP model equals 2.4395 e-4. A comparison of the results of the two models is shown linearly.

Table 3. Total Comparison of the Results of GA-MLP MLP

MdMRE(train)	MdMRE(test)	MMRE(train)	MMRE(test)	MSE(train)	MSE (test)	Models
0.0094	0.0176	1.3325	1.0041	0.0049	0.0047	MLP
0.0088	0.1074	1.1871	0.9477	3.3980e-05	2.4395e-04	GA-MLP

5. Conclusion

In this paper, the SCE is done using hybrid MLP ANN and GA. The comparison and evaluation of ANN with the hybrid of GA showed that the hybrid of ANNs and GA is a strong and fast tool for computer engineers to reach actual effort estimation. The ability of the hybrid model is so much that it has managed to work better than algorithmic COCOMO model and the estimation made is so close to the actual effort value. The results on the basis of GA to choose a group of suitable features from among the total features of middle COCOMO81 dataset for estimation on the basis of ANN has shown that GA-MLP is more accurate in comparison with MLP model. The results also indicate that GA-MLP model has led to more accurate estimation; and this work has caused

to reduce complexity and to save time and money. The results indicated that MMRE value in the training phase for MLP and GA-MLP models is 1.3325 and 1.1871, respectively.

References

[1] F.S. Gharehchopogh, I. Maleki and A. Talebi, Using Hybrid Model of Artificial Bee Colony and Genetic Algorithms in Software Cost Estimation, 9th International Conference on Application of Information and Communication Technologies (AICT), IEEE, Rostov on Don, pp. 102-106, 2015.

[2] F.S. Gharehchopogh, A. Talebi and I. Maleki, Analysis of Use Case Points Models for Software Cost Estimation, International Journal of Academic Research, Part A, Vol.6, No.3, pp. 118-124, 2014.

[3] F.S. Gharehchopogh, L. Ebrahimi, I. Maleki, and S.J. Gourabi, A Novel PSO based Approach with Hybrid of Fuzzy C-Means and Learning Automata in Software Cost Estimation, Indian Journal of Science and Technology, Vol.7, No.6, pp.795-803, 2014.

[4] Z.A. Khalifelu and F.S. Gharehchopogh, Comparison and Evaluation Data Mining Techniques with Algorithmic Models in Software Cost Estimation. Elsevier Press, Procedia- Technology Journal, Vol.1, pp.65-71, 2012.

[5] F.S. Gharehchopogh, I. Maleki and S.R. Khaze, A Novel Particle Swarm Optimization Approach for Software Effort Estimation, International Journal of Academic Re-search, Part A, Vol.6, No.2, pp.69-76, 2014.

[6] E.E. Miandoab and F.S. Gharehchopogh, A Novel Hybrid Algorithm for Software Cost Estimation based on Cuckoo Optimization and K-Nearest Neighbors Algorithms, Engineering, Technology and Applied Science Research, Vol. 6, No. 3, pp. 1018-1022, 2016.

[7] A. Idri, S. Mbarki, and A. Abran, Validating and Understanding Software Cost Estimation Models based on Neural Networks, 2004 International Conference on Information and Communication Technologies: From Theory to Applications, Proceedings, pp. 433-434, 2004.

[8] T.R. Benala, S. Dehuri, R. Mall and K. ChinnaBabu, Software Effort Prediction Using Unsupervised Learning (Clustering) and Functional Link Artificial Neural Networks, 2012 World Congress on Information and Communication Technologies (WICT2012), PP.115-120, IEEE, Trivandrum, 2012.

[9] A.A. Suratgar, M.B. Tavakoli and A. Hoseinabadi, Modified Levenberg-Marquardt method for neural networks training, Proc. World Academy of Science, Engineering and Technology, pp.46-48, 2005.

[10] A.B. Nassif, D.H. Ho and L.F. Capretz, Towards an Early Software Estimation Using Log-Linear Regression and A Multilayer Perceptron Model, Journal of Systems and Software, Vol. 86, No.1, pp.144-160, 2013.

[11] A. Kaushik, A.K. Soni, and R. Soni, An Adaptive Learning Approach to Software Cost Estimation, National Conference on Computing and Communication Systems (NCCCS), India, pp.1-6, 21-22 Nov 2012.

[12] F.S. Gharehchopogh, Neural Networks Application in Software Cost Estimation: A Case Study, 2011 International Symposium on Innovations in Intelligent Systems and Applications (INISTA 2011), IEEE, Istanbul, Turkey, pp. 69-73, 2011.

[13] A.Lee, C.H. Cheng, and J. Balakrishnan, Software Development Cost Estimation: Integrating Neural Network With Cluster Analysis, Information and Management, Vol.34, No.1, pp.1-9, 1998.

[14] A.Heiat¸ Comparison of Artificial Neural Network and Regression Models for Estimating Software Development Effort, Information and Software Technology, Vol.44, No.15, pp. 911-922, 2002.

[15] I. Attarzadeh, A. Mehranzadeh,, and A Barati, Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation, 2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks, IEEE, Phuket, pp.167-172, 2012.

[16] N. Tadayon, Neural Network Approach for Software Cost Estimation, IEEE, Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05), Vol. 2, pp. 815-818, 2005.

[17] J. Kamruzzaman, and S.M. Aziz, A Note on Activation Function in Multilayer Feedforward Learning, International Joint Conference on Neural Networks (IJCNN), Vol.1, pp.519-523, Honolulu, Hawaii, 2002.

[18] D. Hunter, A.Yu, M.S. Pukish, and J. Kolbusz, Selection of Proper Neural Network Sizes and Architectures—A Comparative Study ",IEEE Transactions on Industrial Informatics, Vol.8, No.2, pp.228-240.

[19] Z. Li, Intelligently Predict Project Effort by Reduced Models, International Conference on E-Business and E-Government (ICEE), Guangzhou pp.1536-1542, 2010.

[20] F.S. Gharehchopogh, M. Molany and F.D. Mokri, Using Artificial Neural Network In Diagnosis Of Thyroid Disease: A Case Study. International Journal on Computational Sciences and Applications (IJCSA), Vol.3, No.4, pp.49-61., 2013.

[21] A. Slowik, Application of an Adaptive Differential Evolution Algorithm with Multiple Trial Vectors To Artificial Neural Network Training", IEEE Trans.Ind.Electron, vol.58,no.8,pp.3160-3167, 2011.

[22] A.R. Venkatachalam, Software Cost Estimation using Artificial Neural Networks, International Joint Conference on Neural Networks(IJCCN), Vol.1, pp.987-990, IEEE, Nagoya, Japan, 1993.

[23] B.M. Wilamowski, and H. Yu, Improved Computation for Luxenberg Marquardt Training, IEEE Trans. Neural Netw., Vol. 21, No. 6, pp. 930-937, 2010.

[24] P. Jodpimai, P. Sophatsathit, and C. Lursinsap, Estimating Software Effort with Minimum Features Using Neural Functional Approximation, International Conference on Computational Science and Its Applications (ICCSA), pp.266-273, Fukuoka, 2010.

[25] X. Zeng, S. Wu and L. Han, Sensitivity-Based Adaptive Learning Rules for Binary Feedforward Neural Networks, IEEE Transactions on Neural Networks and Learning Systems, Vol.23, No.3, pp.480-491, 2012.

[26] B.M. Wilamowski and H. Yu, Improved Computation for Levenberg Marquardt Training, IEEE Trans. Neural Netw., Vol. 21, No.6, pp. 930–937, 2010.

[27] A.A. Suratgar, M.B Tavakoli, and A. Hoseinabadi, Modified Levenberg-Marquardt Method for Neural Networks Training, World Academy of Science, Engineering and Technology, pp.46-48, 2005.

[28] J. Lin, C. Chang, Sheng-Yu Huang, Research on Software Effort Estimation Combined with Genetic Algorithm and Support Vector Regression, International Symposium on Computer Science and Society (ISCCS), pp.349-352, 2011.