Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 89 (2016) 307 – 312

Procedia Computer Science

Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016)

Enhanced Software Effort Estimation using Multi Layered Feed Forward Artificial Neural Network Technique

Poonam Rijwani* and Sonal Jain

Institute of Engineering and Technology, JK Lakshmipat University, Jaipur, India

Abstract

Software Effort Estimation models are hot topic of study over 3 decades. Several models have been developed in these decades. Providing accurate estimations of software is still very challenging. The major reason for such disappointments in projects are because of inaccurate software development norms; effort estimation is one such practice. Dynamically fluctuating environment of technology in software development industry make effort estimation further perplexing. One of the most commonly used algorithmic model for estimating effort in industry is COCOMO. Capability of machine learning particularly Artificial Neural Networks is to adjust a complex set of bond among the various independent and dependent variables. The paper proposes usage of ANN (Artificial Neural Network) based model technologically advanced using Multi Layered Feed Forward Neural Network which is given training with Back Propagation training method. COCOMO data-set is accustomed to test and train the network. Mean-Square-Error (MSE) and Mean Magnitude of Relative-Error (MMRE) are used as performance measurement indices. The experiment outputs suggest that the suggested model can provide better results and accurately forecast the software development effort.

Peer-review under responsibility of organizing committee of the Organizing Committee of IMCIP-2016

Keywords: Artificial Neural Networks; Back Propagation; COCOMO; Effort Estimation.

1. Introduction

One of the foremost aim of the community dealing with software engineering development is to develop models which are practically relevant. Another aspect that the community considers as a prime objective is to estimate how accurately a model can estimate the effort incorporate in developing a software. Effort estimation is a process by which one can predict the development time and cost for developing a software process or a product. Estimating cost and time accurately is vital for couple of reasons. Over estimation may lead to harm in financial loss in an organization whereas under estimation may result in poor quality of software which eventually leads to failure of the software. Also Effort estimations done at an initial phases of development of a project may turn out to be helpful for project managers[1]. But very less information is available at early stages. There are many already existing algorithmic and non-algorithmic methods for effort estimation. Dynamic environments in software development technology make effort predictions further exciting especially in early stages.

*Corresponding author. Tel: +91-9928080316.
E-mail address: poonamrijwani@pratap.edu.in

Peer-review under responsibility of organizing committee of the Organizing Committee of IMCIP-2016
doi:10.1016/j.procs.2016.06.073

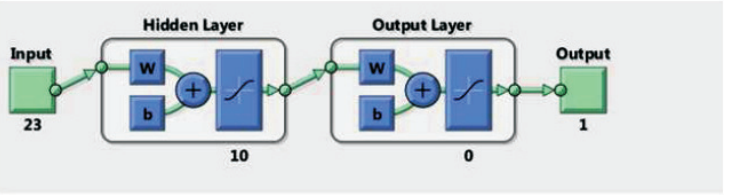

Fig. 1. Architecture of Proposed Neural Network.

Numerous estimation models which have been suggested over decades. They have been categorised on their basic design outlines; prediction by expert[2], analogy based prediction schemes[3], mathematical models which includes empirical based approaches, function point methods[4], artificial neural network based approaches[5,6], Bayesian network approaches[3], decision tree procedures and fuzzy logic prediction schemes[7]. Amongst various software cost estimation approaches, Boehms COCOMO (Constructive Cost Model) is one of the most used empirical cost modeling method in software industry[8] due to its simplicity to use and calculate effort in person-months for a project at various junctures. Many researchers have[5,6] found out the likelihood of implementing artificial neural networks for predicting the effort. In this research an extension of well-known empirical model COCOMO II is developed with multi layered feed forward neural network and is trained with backpropagation learning algorithms.

2. Literature Review

Estimating Software Effort is one of the most crucial responsibility for all the people involved with Software Project Management. These days, a lot of competition is there in the software industry to make quality softwares in stipulated cost and time. Therefore, it is utterly important to accurately estimate effort in key phases of Software development. Regardless of the expanse of investigations over the past two decades, the software industry is still considerably challenging in case of effective resource estimation[9]. Time to time, authors have discussed various methods for the same.

Jorgensen presented a elaborated analysis of various researches on the development effort[10]. K. Vinay Kumar *et al.*,[11] applies wavelet neural networks for estimating effort. B. Tirimula Rao *et al.*,[12] given an enhanced FLANN methodology for software effort prediction. G. Witting and G. Finnie[4] practices multilayer perceptron using back propagation as learning algorithm for predicting development cost and time. N. Karunanitthi *et al.*,[13] demonstrated application of ann for estimating reliability of a software which includes different methodologies with feed forward and Jordan networks. N. Tadayon[14] makes use of ann with back propagation training algorithm. Though it was not clearly cited in the literature how the data-set was alienated into training data-set and validation data-set. Other researchers like Reddy and Raju[7] also presented a estimation model using MLFF neural networks. Venkatachalam[15] experimented application of ann to software effort estimation[10]. Presently, none of these simulations are accepted by the entire community since none of them executes reliable-enough in various environments. Machine Learning techniques particularly ANN are the prominent methods for building estimating models. Therefore, more accurate predictive precision models can always be developed using newer machine learning based techniques. In this vision, our work is aimed at refining various dependent variables i.e. cost drivers and five scaling factors of COCOMO II estimation model using artificial neural network techniques.

3. Ann Architecture and Training

Artificial Neural Networks (ANN) are streamlined mathematically measured practices of biological mortal brain. They are network of a large number of processing features called neurons[16]. They can acquire from prior software project facts and practices to deliver fresh information, guidelines, and skills based on inference of learnt data. Indeed, each neuron is corresponding a mathematical function with certain inputs, a scientific computation procedure, and outputs[17–19].

The various estimation dependent variables are KLOC, 17 cost drives, 5 scale factors. These are the 23 inputs and effort as output variable which is used in the proposed architecture.

A simple neural network comprises of a number of inputs that are applied by certain weights that are pooled together to provide an output. We have applied back propagation learning algorithm to give training to the modelled neural network and explain the estimation problem.

3.1 Data description

In the present work, COCOMO II dataset has been used for calculating the effort using ANN model. In algorithmic cost estimation, effort in person months is predicted using mathematical formula. The mathematical relations and constants are derived based on historical data. It was established from the exploration of existing sixty three existing developed software projects. COCOMO 81 dataset is available freely for research purposes on PROMISE repository[20]. COCOMO 81 dataset is converted to COCOMO II dataset using a tool Rosetta stone[21]. Rosetta stone is given by Sunita Chulani at IBM Research to Make COCOMO Estimates functional with COCOMO II model. The output of the model is the Development Effort (DE), which is measured in man-months.

3.2 Our approach

The proposed enhanced Neural Network Model is created using Neural Network toolbox[22] available in MATLAB software. Matlab tools facilitate ease of simulation and modeling. As discussed earlier, size of input and output vector is decided. Trials are first conducted by arbitrarily picking number of processing elements. Neural Network models are created with one hidden layer and varying number of processing elements or neurons. As per Whites theorem, one layer with non-linear activation function is enough to map non-linear functional relationship in a fairly accurate way.

In the present work optimal network architecture has been investigated, using trial and error approach, in an attempt to create more optimum model. Thus to minimize the number of networks that required training and testing, ANNs containing 2 to 20 nodes were considered in order to thin down the search. The learning rate was kept initially to minimum and increased gradually. Thus various permutation and combinations of both these factors were used during the training process. The target error was set to stop during training when the average error i.e. the tolerance level reaches below 0.999999.

Training is the process by which the weights of an ANN are estimated, by using an iterative procedure to minimize a predetermined error, or objective function, such as the MSE. Hence, ANN training is fundamentally a nonlinear least squares problem, which can be solved using standard nonlinear least squares methods. For training the Feed Forward Neural Network architecture[23], backpropagation algorithm is being used. Tangent sigmoid transfer function is used in the hidden layer and linear one in the output layer.

Once the net has been trained to the level where the predicted results are fairly accurate, testing is carried out to assure that predicted results are in close proximity to actual values. For the best developed NN model i.e. 23-10-1, with ten neurons in the hidden layer using Levenberg training algorithm, MSE is used to judge the accuracy of the prediction during training, testing and validation. It was noticed that roughly after 5 epochs the training error continued to decrease even when the performance of testing and validation were somewhat immobile. This can be referred to the effect as overfitting. Therefore, our best choice of network topology at which proposed neural network approach is most accurate is 23-10-1.

With this selection of network topology, numbers of layers, processing elements, generalization characteristics are well-kept. It was also noticed that training time is also significantly reduced as there are reduced iterations every time. It is also seen that accuracy of prediction is attained by Neural Network model after successful completion of training criteria i.e. with the value of MSE being within acceptable range as well as agreeable performance measure.

As soon as when the training gets complete, the weights are fixed, network structure is finalized and data to be used for functional requirements of the NN model is converted into useful format, training, testing, and validation of the NN model can be started. Figure 2 gives a snapshot of the GUI developed using MATLAB after freezing various parameters settings for a sample project id 1.

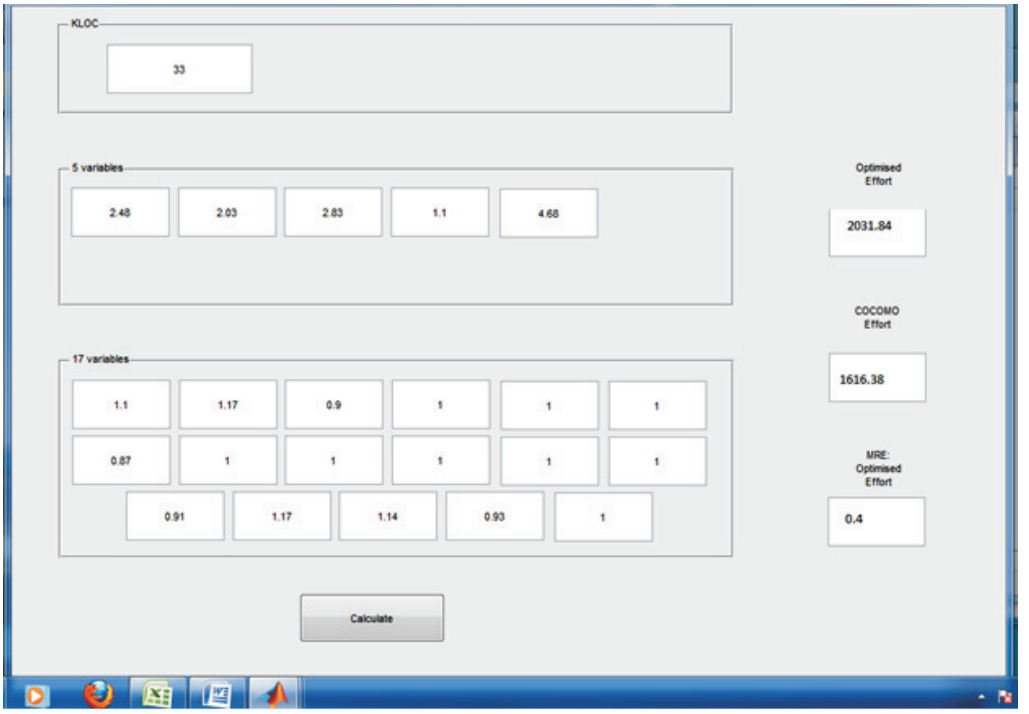

Fig. 2. Interface for Calculating Effort.

Table 1. Effort Computation using Proposed Technique.

Project ID	Concrete Effort Obtained from the Data-Set	Computed Effort with COCOMO	Computed Effort with MLFFN with BP
1	2040	1616.38	2031.84
3	243	233.88	239.039
11	218	189.93	193.802
18	11400	8552.88	11229
20	6400	3603.34	5811.2
26	387	279.93	371.52
27	88	59.002	74.976
50	176	132.162	172.515
51	122	114	119.926
54	20	6.24	12
55	18	7.5	16.902
56	958	537	953.21
60	57	23.91	48.2562

4. Performance Criteria

The capability of the Algorithmic COCOMO II and projected estimate model are inspected and matched using following performance measures.

a. Magnitude of relative error (MRE): MRE is a commonly-used measure which gives the difference between values estimated by suggested model and the values actually estimated[24].

$$\mathrm{MRE}_i = \frac{|\text{Actual Effort} - \text{Predicted Effort}|}{\text{Actual Effort}} \tag{1}$$

Table 2. Performance of Various Techniques MRE.

Project Id	MRE (%) with COCOMO	MRE(%) with MLFFN with BP
1	20.76	0.4
3	3.75	1.63
11	12.87	11.1
18	24.87	81.5
20	43.69	9.2
26	27.66	4
27	32.95	14.8
50	24.9	1.98
51	72.56	1.7
54	68.78	40
55	73.04	6.1
56	77.48	0.5
60	58.05	15.34

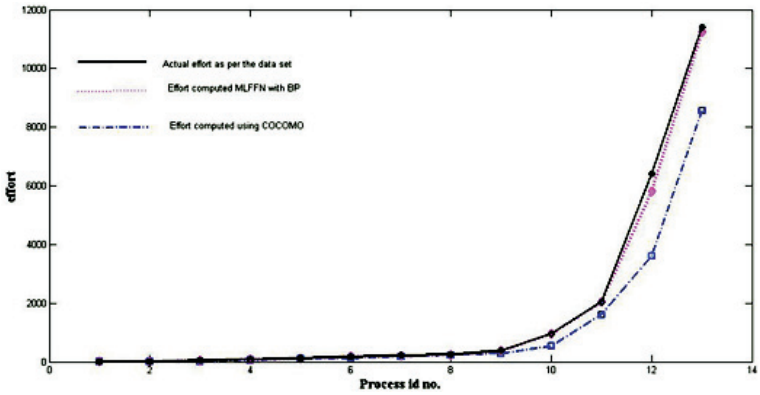

Fig. 3. Comparative Chart Showing Effort Computed using Algorithmic and Proposed Model.

b. Mean magnitude of relative error (MMRE): MMRE, is the mean measurement of the absolute values of the relative errors from complete data set[24].

$$\mathrm{MMRE} = \frac{1}{N}\sum_{1}^{N}\mathrm{MRE}_i \tag{2}$$

where N signifies total no. of estimates.

5. Results and Discussions

Table 2 displays the relative chart of estimated and predicted effort for randomly selected 13 project values using COCOMO and proposed ANN methodology. Further Table 3 tabulates the Magnitude of Relative Error (MRE) values for both the COCOMO and proposed ANN model. The chart shown in Fig. 3 clearly says that there is a significant reduction in the relative error, and that is why the proposed model is more appropriate for estimating effort. The primary results obtained can be simulated for anticipating the accurate software development effort.

6. Conclusions and Future Work

Our proposed model maps existing COCOMOII to a ann with minimum number of layers and nodes which enhances the efficiency of the network. The proposed ann that has been used to predict the development effort is the

multi-layer feed forward neural network with 23 inputs and a hidden layer using back propagation training algorithm. The COCOMOII dataset has been used to train, test and to validate the network and it was observed that neural network model provided significantly better effort estimations than the estimation done using popular algorithmic model COCOMOII. Another great improvement of using neural network model is that we can include expert knowledge, and the traditional mathematical models in a common architecture that can have extensive applicability in software cost estimation. In future, this model will be extended by integrating the proposed approach with genetic algorithm techniques to effectively deal with vague and uncertain facts accompanying with COCOMOII values. Hence, a likely track of future work is to outspread it to the hybrid method.

References

[1] Anupama Kaushik, A. K. Soni and Rachna Soni, A Simple Neural Network Approach to Software Cost Estimation, *Global Journal of Computer Science and Technology*, vol. 13(1), (1969).

[2] Magne Jørgensen and Dag I. K. Sjøberg, The Impact of Customer Expectation on Software Development Effort Estimates, *International Journal of Project Management*, vol. 22(4), pp. 317–325, (2004).

[3] Nan-Hsing Chiu and Sun-Jen Huang, The Adjusted Analogy-Based Software Effort Estimation Based on Similarity Distances, *Journal of Systems and Software*, vol. 80(4), pp. 628–640, (2007).

[4] Gerhard E. Wittig and G. R. Finnic, Using Artificial Neural Networks and Function Points to Estimate 4gl Software Development Effort, *Australasian Journal of Information Systems*, vol. 1(2), (1994).

[5] Krishnamoorthy Srinivasan and Douglas Fisher, Machine Learning Approaches to Estimating Software Development Effort, *IEEE Transactions on Software Engineering*, vol. 21(2), pp. 126–137, (1995).

[6] Ali Idri, Taghi M. Khoshgoftaar and Alain Abran, Can Neural Networks be Easily Interpreted in Software Cost Estimation?, In *Proceedings of the 2002 IEEE International Conference on IEEE Fuzzy Systems, 2002, FUZZ-IEEE'02*, vol. 2, pp. 1162–1167, (2002).

[7] ChSatyananda Reddy and K. V. S. V. N. Raju, An Improved Fuzzy Approach for Cocomos Effort Estimation using Gaussian Membership Function, *Journal of Software*, vol. 4(5), pp. 452–459, (2009).

[8] M. Madheswaran and D. Sivakumar, Enhancement of Prediction Accuracy in Cocomo Model for Software Project using Neural Network, In *Proceedings of International Conference on IEEE Computing, Communication and Networking Technologies (ICCCNT), 2014*, pp. 1–5, (2014).

[9] Kjetil Moløkken and Magne Jørgensen, A Review of Software Surveys on Software Effort Estimation, In *Proceedings of International Symposium on IEEE Empirical Software Engineering, ISESE, 2003*, pp. 223–230, (2003).

[10] Anupama Kaushik, Ashish Chauhan, Deepak Mittal and Sachin Gupta, Cocomo Estimates using Neural Networks, *International Journal of Intelligent Systems and Applications*, vol. 4(9), pp. 22, (2012).

[11] K. Vinay Kumar, V. Ravi, Mahil Carr and N. Raj Kiran, Software Development Cost Estimation using Wavelet Neural Networks, *Journal of Systems and Software*, vol. 81(11), pp. 1853–1867, (2008).

[12] B. Tirimula Rao, B. Sameet, G. Kiran Swathi, K. Vikram Gupta, Ch. Ravi Teja and S. Sumana, A Novel Neural Network Approach for Software Cost Estimation using Functional Link Artificial Neural Network (FLANN), *International Journal of Computer Science and Network Security*, vol. 9(6), pp. 126–131, (2009).

[13] Nachimuthu Karunanithi, Darrell Whitley and Yashwant K. Malaiya, Using Neural Networks in Reliability Prediction, *Software, IEEE*, vol. 9(4), pp. 53–59, (1992).

[14] Nasser Tadayon, Neural Network Approach for Software Cost Estimation, In *International Conference on IEEE Information Technology: Coding and Computing, 2005, ITCC 2005*, vol. 2, pp. 815–818, (2005).

[15] A. R. Venkatachalam, Software Cost Estimation using Artificial Neural Networks, In *Proceedings of International Joint Conference on IEEE Neural Networks, IJCNN'93-Nagoya*, vol. 1, pp. 987–990, (1993).

[16] Sanguthevar Rajasekaran and G. A. Vijayalakshmi Pai, Neural Networks, Fuzzy Logic and Genetic Algorithm: Synthesis and Applications (with cd), PHI Learning Pvt. Ltd., (2003).

[17] T. Nakano, T. Suda, T. Koujin, H. Tokuko and Y. Hiraoka, Bio-Inspired Models of Network, Information and Computing systems, *Molecular Communication through Gap Junction Channels: System Design, Experiments and Modeling*, vol. 2, pp. 139–146, (2007).

[18] Richard P. Lippmann, An Introduction to Computing with Neural Nets, *ACM SIGARCH Computer Architecture News*, vol. 16(1), pp. 7–25, (1988).

[19] Anil K. Jain, Jianchang Mao and K. M. Mohiuddin, Artificial Neural Networks: A Tutorial, *Computer*, vol. (3), pp. 31–44, (1996).

[20] J. Sayyad Shirabad and T. J. Menzies, The PROMISE Repository of Software Engineering Databases, *School of Information Technology and Engineering*, University of Ottawa, Canada, (2005).

[21] B. Clark and D. Reifer, The Rosetta Stone: Making your Cocomo Estimates Work with Cocomo ii, In *Software Technology Conference, Salt Lake City, UT*, (1998).

[22] Howard Demuth, Mark Beale and Martin Hagan, Neural Network Toolbox 6, *Users Guide*, (2008).

[23] S. Ajitha, T. V. Suresh Kumar, Evangelin D. Geetha and K. Rajani Kanth, Neural Network Model for Software size Estimation using use Case Point Approach, In *International Conference on IEEE Industrial and Information Systems (ICIIS)*, pp. 372–376, (2010).

[24] Daniel Ryan Baker, A Hybrid Approach to Expert and Model based Effort Estimation, ProQuest, (2007).