ENHANCEMENT OF PREDICTION ACCURACY IN COCOMO MODEL FOR SOFTWARE PROJECT USING NEURAL NETWORK

M.MADHESWARAN[1]
madheswaran.dr@gmail.com

D.SIVAKUMAR[2]
dskumarcse@yahoo.co.in

[1]Department of Electronics and Communication Engineering
[2]Department of Information Technology
Mahendra Engineering College, Namakkal- 637503 , Tamilnadu, India

Abstract

In this research, it is concerned with constructing software effort estimation model based on artificial neural networks. The model is designed accordingly to improve the performance of the network that suits to the COCOMO model. In this paper, it is proposed to use multi layer feed forward neural network to accommodate the model and its parameters to estimate software development effort. The network is trained with back propagation learning algorithm by iteratively processing a set of training samples and comparing the network's prediction with the actual effort. COCOMO dataset is used to train and to test the network and it was observed that proposed neural network model improves the estimation accuracy of the model. The test results from the trained neural network are compared with that of the COCOMO model. The preliminary results obtained suggest that the proposed architecture can be replicated for accurately forecasting the software development effort. The aim of this study is to enhance the estimation accuracy of COCOMO model, so that the estimated effort is more close to the actual effort.

Index Terms -- back propagation, COCOMO, feed forward neural networks, software cost estimation, software effort estimation.

1. INTRODUCTION

Estimating software development effort remains a complex problem, and one which continues to attract considerable research attention. Accurate cost estimation of a software development effort is critical for good management decision making. The precision and reliability of the effort estimation is very important for software industry because both overestimates and underestimates of the software effort are harmful to software companies. Thus, from an organizational perspective, an early and accurate cost estimate will reduce the possibility of organizational conflict during the later stages. Predicting software development effort with high precision is still a great challenge for project managers. An important objective of the software engineering community has been to develop useful models that are accurately estimating the software effort [1].

Among the software cost estimation techniques, COCOMO (**Co**nstructive **Co**st **Mo**del) is the mostly used algorithmic cost modeling technique because of its simplicity for estimating the effort in person-months for a project at different stages. COCOMO uses the mathematical formulae to predict project cost estimation. Many researchers have explored the possibility of using neural networks for estimating the effort. Artificial neural networks can model complex nonlinear relationships and approximate any measurable function [2]. They are particularly useful in problems where there is a complex relationship between an input and output.

The most commonly adopted architecture for estimating software effort is feed forward multilayer perceptron with back propagation learning algorithm and the sigmoid activation function. One of the major drawbacks in back propagation learning is its slow convergence. The main reason for slow convergence is the sigmoidal activation function used in its hidden and output layers. The network with sigmoid function will be more sensitive to the loss of parameters. Besides that inappropriate selection of network patterns and learning rules may cause serious

difficulties in network performance and training. So the number of layers and nodes should be minimized to amplify the performance. Hence to overcome the above limitations and to improve the performance of the network that suits to the COCOMO Model, in this research, it is proposed concise neural network architecture with linear activation function.

In this paper, it is concerned with constructing cost estimation model based on artificial neural networks, particularly multi layer feed forward neural networks. The network model is designed accordingly to accommodate the COCOMO model and its parameters [3], and back propagation learning algorithm is used to train the network by iteratively processing a set of training samples and comparing the network's prediction with the actual. The difference in the estimation is propagated to the input for adjusting the coefficients. This process consists of repeatedly feeding input and output data from empirical observations, propagating the error values, and adjusting the connection weights until the error values fall below a user-specified tolerance level [4].

2. RELEVANT WORK

Software cost estimation is one of the most critical and complex, but an inevitable activity in the software development process. The algorithmic as well as the non algorithmic cost estimation models are not so accurate.

Samson [5] applied a neural network model, Cerebellar Model Arithmetic Computer (CMAC) to prediction of effort from software code size. CMAC is a perception and function approximator developed by Albus.This neural network was trained on Boehm's COCOMO data set in order to predict effort from size, in the same manner regression techniques are used to fit a line to a data set for prediction purposes.

G.E. Witting examined the performance of back propagation artificial neural networks in estimating software development effort. Two different experiments were carried out. In this research the size of software was measured in function points and project effort was measured in development hours [6].

A recent study by Jorgensen [7] provides a detailed review of different studies on the software development effort. Neural networks have learning ability and are good at modeling complex nonlinear relationship, provides more flexibility to integrate expert knowledge into the model. Although these efforts showed a promising research direction in software cost estimation, the approaches based on neural network are far from mature.

Some primary work in the use of neural network in estimating software cost by Karunanithi [8] produced accurate results, but the major set back is that the result heavily relied on training set. Srinivasan and Fisher [9] report the use of neural network with a back propagation learning algorithm. They found that the neural network outperformed other techniques.

COCOMO however has some limitations. It cannot effectively deal with imprecise information and calibration of COCOMO is one of the most important tasks that need to be done in order to get accurate estimations. So there is always scope for tuning the cost estimation models [14] with better predictive accuracy.

3. PROBLEM STATEMENT

Calculation based on historical data are difficult due to inherent complex relationships between the related attributes, are unable to handle categorical data as well as lack of reasoning capabilities. Besides, attributes and relationships used to estimate software development effort could change over time and differ for software development environments. In order to address and overcome to these problems, a new model with accurate estimation will be considerable.

4. COCOMO

The COnstructive COst MOdel (COCOMO) was introduced by Barry Boehm in 1981 as a model for estimating effort, cost, and schedule for software projects in his classic text, *Software Engineering Economics*. The model was defined based on the analysis of 63 completed projects from diverse domains. COCOMO 81 consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO, is good for quick, early,

rough order of magnitude estimates of software costs. Intermediate COCOMO takes 15 different cost drivers or effort multipliers into account. Detailed COCOMO additionally accounts for the influence of individual project phases. The COCOMO 81 model has been extensively used in industry as well as academia. In 1997, COCOMO 81 was extended to COCOMO II [10] to include the newer paradigms in software development and finally published in a book in 2000. In both the COCOMO models, the effort required to develop a project can be written in the following general form:

$$PM = a*(KLOC)^{b}*\left(\prod_{j} EM_{j}\right) \quad \ldots\ldots\ldots (1)$$

where,

PM = effort estimate in person months
a = productivity coefficient
b = economies (or diseconomies) of scale coefficient.
KLOC = kilo lines of code
EM = effort multipliers or cost drivers

5. DATASETS

We used the COCOMO I dataset for our experimentation because it is an open model with published data. The details are published in the text *Software Engineering Economics*. Besides, the datasets are publicly available online in the PROMISE [11] repository. The datasets are as follows.

- COCOMO

There are 63 projects in this data set [12]. They are collected from a variety of domains including science, engineering and finance.

6. TUNING THE COCOMO

Although many algorithmic models are usually not very accurate, they reflect a lot of useful expert knowledge in this field. In order to make the best use of this kind of expert knowledge, it is worthwhile to integrate the algorithmic model into the present neural network model [13]. Calibrating the parameters of algorithmic models is usually is very challenging task. In the current neural network model, parameters of the algorithmic model are adjusted through learning [15]. The initial definition of the COCOMO II model and its results are provided in the initial COCOMO II model had the following form:

$$Effort = A\times[Size]^{1.01+\sum_{i=0}^{5} SF_i} \times\prod_{i=1}^{15} EM_i \quad \ldots\ldots\ldots (2)$$

where, A is the multiplicative constant, Size of the software project measured in terms of KSLOC, SF is the Scale Factors and EM is the Effort Multipliers. The COCOMO Model shown in "(2)," is a non linear model. To solve this problem we transform the non-linear model of "(2)," into a linear model using the logarithms to the base e, is shown as follows in "(3)".

$$\ln(PM) = \ln A + \ln(EM_1) + \ln(EM_2) + \ldots\ldots + \ln(EM_{15}) + [1.01 + SF_1 + \ldots\ldots + SF_5]*\ln(Size) \quad \ldots\ldots (3)$$

The above equation has the form of the model below:

$$O_{est} = [b_1 + u_1 * I_1 + u_2 * I_2 + \ldots\ldots + u_{15} * I_{15}] + [b_2 + I_{16} + \ldots\ldots\ldots + I_{20}]*[v_i + \ln(size)] \quad \ldots. (4)$$

Where,

$O_{est} = \ln(PM)$;
$I_1 = \ln(EM_1)$;……; $I_{15} = \ln(EM_{15})$;
$I_{16} = SF_1$; ….. ; $I_{20} = SF_5$;

b_1 and b_2 are the biases and the coefficients u_i, v_i are the additional terms introduced in the model, which are to be estimated as follows. Initially we assume that the values of the coefficients are to be 1 for u_i and 0 for v_i to estimate the output O_{est}. Then this estimated effort is compared with that of the actual observed effort in the log space. The difference is the error in the estimation, which should be minimized. All the model's parameters are assumed to be responsible for the output error.

The difference in this estimation is propagated to the inputs for adjusting the coefficients, so that this knowledge can be incorporated in to the model in the form of coefficients. Thus these coefficients of the model can be estimated by the proposed neural network as discussed below. Note that part 1 of "(4)," deals with effort multipliers, which represents the upper section of the neural network under study and part 2 deals with scale factors that represents the lower section.

7. EVALUATION CRITERIA

This section presents the methodology used in conducting experiments and discusses the results obtain when applying our proposed neural network to the COCOMO dataset. The analysis undertaken in this study and the dataset used in this work are from COCOMO database, a dataset publicly available which consists of 63 projects. Based on this process, we have divided the entire dataset into two sets, training set and validation set in the ratio of 80%: 20%. Training set consists of 50projects selected randomly and validation set consists of remaining 13 projects.

All the data is normalized to fit in the natural logarithmic space. The calculations were made using a software prototype developed in java language. The proposed neural network is implemented with 20 input nodes, two hidden layers and one output node and the network is trained with training data and tested with the validation data. The evaluation consists in comparing the accuracy of the estimated effort with the actual effort.

Table 1. Comparison of Actual and Estimated Effort.

S.No	Project ID	ACTUAL EFFORT	Estimated EFFORT using	
			COCOMO	NN Model
1	1	2040	2218	2050
2	5	33	39	32
3	9	423	397	407
4	29	7.3	7	6.5
5	34	230	201	203
6	42	45	46	46
7	47	36	33	40
8	50	176	193	162
9	51	122	144	117
10	52	41	55	43
11	55	18	7.5	17
12	56	958	537	951
13	58	130	145	126

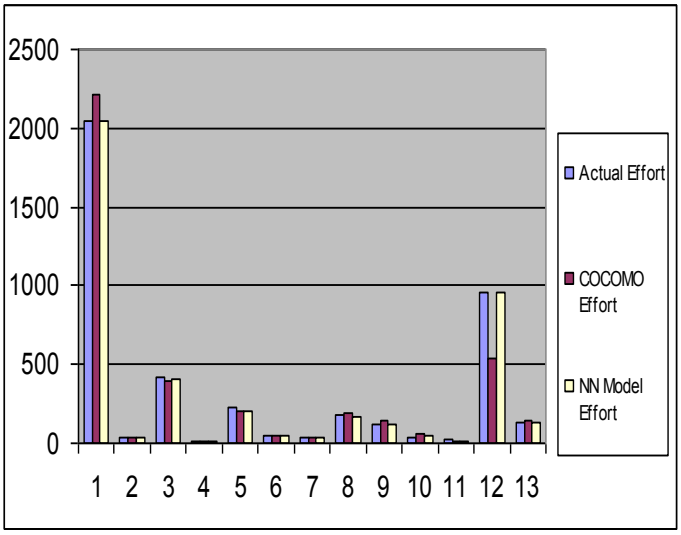

Figure: 1 Chart showing the Actual and Estimated Effort

8. CONCLUSIONS

A reliable and accurate estimate of software development effort has always been a challenge for both the software industrial and academic communities. There are several software effort forecasting models that can be used in forecasting future software development effort. We have constructed a cost estimation model based on artificial neural networks. Our idea consists in the use of a model that maps COCOMO model to a neural network with minimal number of layers and nodes to increase the performance of the network.

The neural network that we have used to predict the software development effort is the multi layer feed forward neural network with the identity function at input, hidden and output units. The back propagation algorithm is used to train the network. We have used full COCOMO dataset to train and to test the network and it was observed that our neural network model provided significantly better cost estimations than the estimation done using COCOMO model.

9. REFERENCES

1. Ch.Satyananda., KVSVN Raju., "An Improved Fuzzy Approach for COCOMO's Effort Estimation Using Gaussian Membership Function", Journal of Software, vol 4, pp 452-459, 2009.

2. Idri, A. Khoshgoftaar, T.M. Abran, A., "Can neural networks be easily interpreted in software cost estimation?" Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'02, Vol.: 2, 1162-1167, 2002.

3. J., Hale, A. Parrish, B. Dixon, and R.K. Smith, "Enhancing the Cocomo estimation models", *IEEE Software*, 17 (6), 2000, pp. 45 – 49.

4. J.E. Helm, "The viability of using COCOMO in the special application software bidding and estimating process", *IEEE Transactions on Engineering Management*, 39 (1), 1992, pp. 42 - 58.

5. Samson, B., David Ellison., Pat Dugard., "Software cost estimation using an albus perceptron", journal of Info & Softw., vol.39, pp.55-60, 1997.

6. G.E Witting., G.R. Finnic., "Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort", Australian Journal of Info.System, 1994.

7. Jørgensen. M., "A Review of Studies on Expert Estimation of Software Development Effort," Journal of Systems and Software, Volume 70, pp. 37-60, 2004.

8. Karunanithi, N., D.Whitley, Yashwant K. Malaiya., "Using neural networks in reliability prediction," IEEE Software, pp. 53-59, 1992.

9. Srinivasan, K., Fisher, D., "Machine learning approaches to estimating software development effort," IEEE Transactions on Software Engineering, Volume 21 (2), 126–137, 1995.

10. Boehm, B.W., E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W. Brown, S. Chulani, and C. Abts, *Software cost estimation with COCOMO II*, Prentice Hall, 2000.

11. J. S. Shirabad, and T. J. Menzies, "The PROMISE repository of software engineering databases", School of Information Technology and Engineering, University of Ottawa, Canada, 2005.

12. http://promise.site.uottawa.ca/SERepository.

13. Tad Gonsalves, Kei Yamagishi and Kiyoshi Itoh, "Swarm Intelligence in the fine-tuning of Software Development Cost Estimation Models," Annual IEEE International Computer Software and Applications Conference, pp. 593–598, 2009.

14. Abedallah Zaid, Mohd Hasan Selamat, Abdual Azim Abd Ghani, "Issues in Software Cost Estimation", IJCSNS International journal of Computer Science and Network Security, vol. 08, no. 11, pp. 350–356, 2008.

15. Guyon and A. Elisseeff, "An introduction to variable and feature selection", *Journal of Machine Learning Research*, vol.3, 2003, pp.1157–1182.