Proposing Effort Estimation of COCOMO II through Perceptron Learning Rule

Saoud Sarwar
Professor (Department of Computer Sceince)
AFSET, MDU

Monika Gupta
PG Scholar
AFSET, MDU

ABSTRACT
The software cost estimation is now one of centre of attention for computer software industries. As software industry runs many projects simultaneously they have to prioritize different processes based on time, cost, and number of staff, sequentially. With the increasing complexity of software, the cost of the software development is also increasing. So it is required to rely on the effective techniques for estimating software costs. Accurate cost estimation is needed because it can help to prioritize and classify development projects. In this paper, the most popular software cost estimation model, COCOMO II (post architecture model of COCOMO), is discussed. The estimation of COCOMO II is enhanced through neural network. The network is trained trough perceptron learning rule. The company's previous projects dataset of estimation and actual cost can be used to train the network. The cost estimation result of COCOMO II is compared with trained network.

Keywords
Software cost estimation, neural network, COCOMO II, perceptron learning**.**

1. INTRODUCTION
Before starting any project, it is required to estimate the cost of project in terms of time, money and manpower for ensuring its feasibility for both customer and software organization. This estimation will also help the project manager to prioritize the development processes. With more accurate results of estimation, an organization can manage its time, money more efficiently.

There are many software cost estimation techniques **[1, 2, 3]** and models which are classified as algorithmic and non-algorithmic approach **[4]**. The algorithmic approach is based on size of project function point analysis, Linear Models, Multiplicative models and COCOMO.

The COCOMO II **[5, 6]** (Post Architecture Model), based on algorithmic method, is now most popular technique of cost estimation in terms of Person-month for a project because of its simplicity, capability and accuracy.

The drawback of algorithmic techniques led to the introduction of non-algorithmic techniques for cost estimation which includes fuzzy logic, machine learning, neural network, and expert judgement **[7].**

This paper proposed an estimation model that will take the advantage of both algorithmic (COCOMO II) and non-algorithmic approach (neural network). Many researchers are also developing cost estimation model based on non-algorithmic approach **[8, 9, 10, and 11]**.

This paper is organised as follows. Section 1 describes the cost estimation of software and various techniques for estimation. Section 2 describes the COCOMO II model with all its cost drivers. Section 3 describes basic structure of artificial neural network and perceptron learning rule. Sections 4, discusses the related works to the software effort estimation. Section 5, present the new model of estimation that incorporate COCOMO II and neural network. Finally section 6 concludes the paper.

2. COCOMO II - POST ARCHITECTURE MODEL:
The constructive cost model (COCOMO) is most popular method for effort estimation based algorithmic approach. The COCOMO was proposed by Barry Boehm in 1982[12]. The COCOMO 81 was the first model on that time**.** The working environment and the parameters of software are matched with COCOMO to estimate the effort of that project. This output of the COCOMO in terms effort is then used to measure the time, cost, and person needed for the software project development. With changing environment of the software project development in 1990's it became difficult to calculate the effort with COCOMO 81. To overcome the weakness of COCOMO 81, a new version of COCOMO, COCOMO II [13], was introduced and calibrated in 1997 by Boehm.

COCOMO II has three models but the most detailed is Post Architecture Model. This model is used when all the architecture of the project development has been decided. It uses the size of project (in terms of Kilo source line of codes (KSLOC)) as well as 22 cost drivers, which includes five scale factors and 17 effort multipliers, for the estimation of effort. The output of COCOMO II, in terms of person/month, is calculated as

$$\frac{person}{month} = A.(KSLOC)^{B}.\prod_{j=1}^{17} Effort\ Multiplier(j) \quad \text{.....(1)}$$

Where B = 1.01 + 0.01 X $\sum_{j=1}^{5} scale\ factor(j)$

In equation 1, A is called as multiplicative constant.

All the five scale factors and 17 effort multipliers (commonly known as Cost drivers) are described in table 1.

The detailed description of all the cost drivers and effort multipliers is given by Boehm [14], COCOMO II model Definition manual.

3. ARTIFICIAL NEURAL NETWORK
Artificial neural network [15, 16] is consisting of large number of interconnected processing elements called neurons. Artificial

neural network is mathematical model that is inspired from central nervous system of human brain.

Table 1: cost drivers of COCOMO II

Symbol	Name
SF1	Precedentedness
SF2	Development Flexibility
SF3	Architecture and Risk Resolution
SF4	Team cohesion
SF5	Process Maturity
EM1	Required Software
EM2	Data Base Size
EM3	Product Complexity
EM4	Required Reusability
EM5	Documentation Match to Life-cycle Needs
EM6	Time Constraint
EM7	Storage Constraint
EM8	Platform Volatility
EM9	Analyst Capability
EM10	Programmer Capability
EM11	Applications Experience
EM12	Platform Experience
EM13	Language and Tool Experience
EM14	Personnel Continuity
EM15	Use of Software Tools
EM16	Multi-Site Development
EM17	Required Development Schedule

As natural neurons receives input signal through synapse and if the strength of received signal is greater than some threshold then the neuron is said to be activated and fires a signal through axon. Similarly in artificial neurons, it consists of inputs and associated weight (strength of signal).

A mathematical function called activation function is used to determine the activation value of artificial neuron. In ANNs either excitatory or inhibitory neural connections are possible. The activation function can be Gaussian, linear, Sigmoid and Tach. Another mathematical function is used to calculate the output of a neuron. Artificial neural network consist of large number of such type of neurons. All the neurons are connected through a connection link and each connection link have a weight associated with it. This weight contains the information about the input signal which is used to solve a particular problem. The first neural network model was proposed by McCulloch and Pitts in 1943.

Artificial neurons are the processing unit of artificial neural network. The basic structure of neuron is given in figure 1.

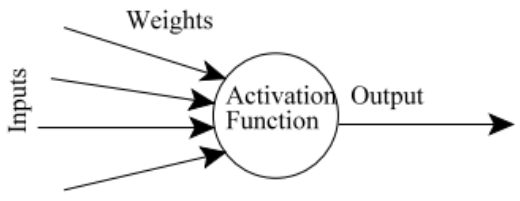

Figure 1: Artificial Neuron

Artificial neural network is of two type's namely feed forward neural network and feed backward or recurrent neural network. In feed forward network, there is no backward loop of current output to input neurons. The flow of information is only in forward direction and not in backward direction. There is no effect of previous results to current results. While in recurrent neural network there is a backward loop to provide the current output to input neurons.

The weights associated with neurons can be adjusted to produce the desired output. There are various algorithms used to adjust the weight of neuron in order to obtain the desired output. The process of adjusting weights is known as training or learning. The learning can be supervised learning or un-supervised learning. In supervised learning target (desired) output is associated with each input. The output of the network is then compared with target output which measures the amount of error. Then the learning rule is used to reduce the amount of error between target and actual output of the network. A training set of input and target output is used to train the network. While in unsupervised leaning, there is no target output. The weights and bias are adjusted with network input only.

The learning rule used in our approach is perceptron learning[17] and the network is perceptron neural network. The output produced by the perceptron network is in True (1) and false (0) value for input presented to it the perceptron learns through the examples provided to it. The weighted sum of inputs is combined to produce the output. If the output is greater than some threshold value then output of perceptron is true or 1 otherwise output is false or 0. When output is false weights are adjusted in order to reduce the amount of error between the target and the actual output.

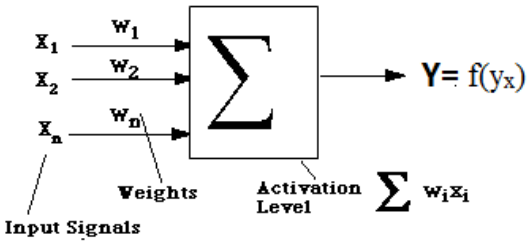

Figure 2: perceptron learning

Where $f(y_x)$ is the activation function and it is defined as:

$$F(x) = \begin{cases} 1 \text{ if } y_x \geq T \\ 0 \text{ if } y_x < T \end{cases} \quad (2)$$

T is some threshold value.

4. RELATED WORK

Many researchers are have worked upon the cost estimation of software project using Artificial Neural Network that incorporate COCOMO [18, 19, 20, 21, 22 and 23]. Ch.Satyananda Reddy [18], adopted feed forward multilayer

perceptron with linear activation function to avoid slow convergence problem Jorgensen [19] provides a detailed review of different studies on the software development effort. Another study by Samson et al. [20] used an albus multilayer perceptron to predict software effort. They have used Boehm's COCOMO dataset. Srinivasan and Fisher [22] discussed the use of a neural network with a back propagation learning algorithm for cost estimation. They found that the neural network performed well than other techniques. K. Vinay Kumar, V. Ravi, Mahil Carr, and N. Raj Kiran [23] have used the wavelet neural network for estimating the software project development cost. N. Tadayon [8] also discussed the neural network with a back propagation learning algorithm.

5. PROPOSED ESTIMATION MODEL INCORPORATES ANN

The proposed structure of neural network is organised to incorporate the COCOMO II-post architecture mode and trained through perceptron learning rule. There are three main entities in the neural network namely the neurons (nodes), the interconnection structure and the training algorithm. The performance of an artificial neural network depends on number of layers, number of neuron in each layer and the training method. The COCOMO II includes 22 cost drivers, five scale factor and 17 effort multipliers. Each of these cost drivers represents the different attribute of a project like staff attributes, software and hardware attributes, environment attributes. The value of each cost driver can change with change in project attribute. The effort in terms of person per month is calculated through equation 1 which will act as target input for the calculated output.

The proposed structure of neural network consist of three layers namely: input layer, hidden layer and output layer. In input layer, there are 24 neurons which are 5 scale factor (SF), 17 effort multipliers (EM) and 2 bias inputs. All of these inputs have some initial weights associated with them. In hidden layer there are two neurons and at the output layer there is only one neuron as shown in figure 3.

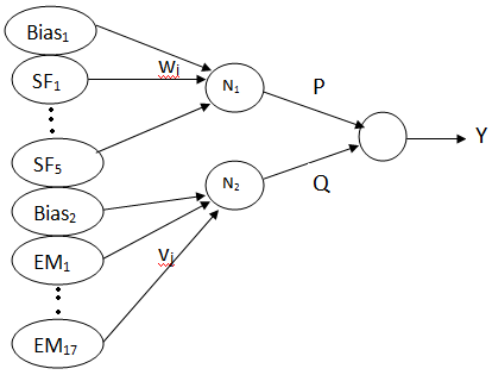

Figure 3: architecture of neural network.

The processing at neuron N_1 & N_2 is done as:

$$N_1 = \text{Bias1} + \sum_{j=1}^{5}[V_j + \log(\text{size})]\, SF_j \qquad (3)$$

$$N_2 = \text{Bias2} + \sum_{i=1}^{17} w_i \log EM_i \qquad (4)$$

The value of each cost driver is provided to the network through the input layer. The summations of inputs (that are multiplied with the associated weights) are processed as given in equation 3 & 4 for scale factors and effort multipliers. As the COCOMO uses the multiplication of inputs and network uses summation, so to neutralize the input, a log function is used.

The activation function given in equation (2) is used to calculate the output of hidden layer neurons as $f(N_{1j})$ and $f(N_{2i})$. Depending on the output signal produced by the neurons of hidden layer, the output of the neuron at the output layer is either true or false. If the output is true then no weights are need to be adjusted but if the output is false then weights are adjusted.

The value at output layer neuron is calculated as given in equation 1 by taking the value of Bias1 equals to 1.01 and the value of Bias2 equal to Log (A). The value of weights at input layer are initialised as $w_i = 1$ for $1 < i <= 5$ and $v_j = 1$ for $1<j<=17$. If the value is greater or equal to activation function then true signal is produced by the network which means estimation of cost drivers are correct else weights are need to be modified. The weights are modified as

$$wi(new) = wi(old) + \alpha \times input(i)$$

6. CONCLUSION

The accurate and reliable estimation of effort is very important for software project development. In this paper, we have constructed an enhanced model of COCOMO II through neural network. The model has been trained through perceptron learning rule. The model will take the advantage of COCOMO II for effort estimation and neural network for learning capability. The model will check whether the estimation of COCOMO II is correct or not. If the estimation is correct then model will output as TRUE otherwise FLASE. In case of FALSE result, modifications needs be done to correct the estimation.

7. REFERENCES

[1] J.P. Lewis, "Large Limits to software estimation", Software Engineering Notes, Vol. 26, No. 4, July 2001.

[2] Stamelos, etal. "Estimating The Development cost of custom software", Information and Management, v.40 n.8, pp.729-741, 2003

[3] R. W. Jensen, "Extreme Software Cost Estimating", CrossTalk, Journal of defense software Engg., Jan 2004.

[4] Vahid Khatibi "Software Cost Estimation Methods: A Review", Journal of Emerging Trends in Computing and Information Sciences, Volume 2 No. 1 on 2010.

[5] Boehm, B., B. Clark, E. Horowitz, C. Westland, R. Madachy, R. Selby, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," Annals of Software Engineering Special Volume on Software Process and Product Measurement, J.D. Arthur and S.M. Henry (Eds.),

J.C. Baltzer AG, Science Publishers, Amsterdam, The Netherlands, Vol 1, 1995, pp. 45 – 60

[6] Center for Software Engineering , "COCOMO II Model Definition Manual," Computer Science Department, University of Southern California, Los Angeles, Ca. 90089, http://sunset.usc.edu/Cocomo.html, 1997.

[7] R. T. Hughes, "Expert judgment as an estimating method", Information & Soft. Technology, pp. 67-75, 1996.

[8] Nasser Tadayon., "Neural network approach for software cost estimation," In proceedings of the IEEE International Conference on Information Technology: coding and computing Computing(ITCC'05), Vol. 2, pp. 815-818, 2005.

[9] 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.

[10] Finnie, G.R. and Wittig, G.E., "AI tools for software development effort estimation," In proceedings of the IEEE International Conference on Software Engineering: Education and Practice, Washington DC, pp 346-353, 1996.

[11] B. Tirimula Rao, B. Sameet, G. Kiran Swathi, K. Vikram Gupta, Ch. Ravi Teja, 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 Society, Vol.9 No.6, June 2009.

[12] B. Boehm, C. Abts, and S. Chulani, "Software Development Cost Estimation Approaches – A Survey," University of Southern California Center for Software Engineering, Technical Reports, USC-CSE-2000- 505, 2000.

[13] Bradford Clark, Sunita Devnani-Chulani, Barry Boehm., "Calibrating the COCOMO II Post-Architecture Model".

[14] Boehm, "Cost Models for Future Software Live Cycle Processes: COCOMO 2.0", Annl Soft. Eng. pp. 45-60,1995.

[15] N. Kassabov, "Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering", MIT press, 1996.

[16] D. E. Neumann, "An Enhanced Neural Network Technique for Software Risk Analysis", IEEE Trans. Soft. Eng., pp 904-912, vol. 28, 2002

[17] www.msu.edu/course/lin/463/ss04/learning.pdf

[18] Ch. Satyananda Reddy and KVSVN Raju, " An Optimal Neural Network Model for Software Effort Estimation", Int.J. of Software Engineering, IJSE Vol.3 No.1 January 2010

[19] Jorgerson, M., "Experience with accuracy of software maintenance task effort prediction models," IEEE Transactions on Software Engineering, Volume 21 (8), 674–681, 1995.

[20] Samson, B., Ellison, D., Dugard, P., "Software cost estimation using an Albus perceptron (CMAC)," Journal of Information and Software Technology, Volume 39 (1), 55–60, 1997.

[21] Seluca, C., "An investigation into software effort estimation using a back propagation neural network," M.Sc.Thesis, Bournemouth University, UK, 1995.

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

[23] K. Vinay Kumar, V. Ravi, Mahil Carr, N. Raj Kiran, "Software development cost estimation using wavelet neural networks", The journal of Systems and Software 81(2008) 1853-1867.