Software Cost Estimation Model Using Neural Network

Bhavika Tuli[1] Deepali Gupta[2]
[1,2]Geeta Institute of Management and Technology, Kurukshetra

Abstract— Software cost estimation remains an important unsolved practical problem in software engineering. Software cost estimation has failed, in most cases, to accurately predict the actual costs. Different software development projects are going to need different process structures based on the size of the project, the number of people working on development, and the amount of schedule time to complete development. Estimating software development costs with accuracy is very difficult. Part of the desired system could be parameters such as: total system development cost, scheduled delivery date, functionality, and quality. The project manager needs to supervise the software development project so that the desired system is delivered.

Key words: COCOMO, Software Metric, Function Point, Line of Code, Neuron

I. Introduction

There are three requirements for a software cost estimation model that will make accurate predictions of software effort and schedule.

The first requirement is that the estimation model is built on a solid foundation of prior research and empirically tested. The second requirement of a good estimation model is that the development process follows a repeatable process.

The third requirement for a good estimation model is that the model includes relevant factors that vary with project metrics.

Early prediction of completion time is absolutely essential for proper advance planning and aversion of the possible ruin of a project. The benefits of accurate effort estimates, or rather the problems with inaccurate ones are clear. Low effort estimates can lead to delayed deliveries, cost overruns and even low quality software. Too high effort estimates can lead to sub-optimal use of resources or even lost profit by losing software bidding rounds, because a high bid is less likely to get selected by the customer.

A. *The COCOMO model*

The first approach that comes in to existence when talking about software cost estimation is the parametric model, which was known in the 1980's as Barry Boehm's Constructive Cost Model (COCOMO), and Larry Putnam's Software Life Cycle Management (SLIM)[3].

The COCOMO model is a set of three models: basic, intermediate, and detailed. The models depend on the stage of software development and the level of information available. The basic version is used for quick, early, and rough estimates of effort [1]. The intermediate and detailed versions include more information in the form of cost drivers.

COCOMO model takes the following as input:

(1) the estimated size of the software product in thousands of Delivered Source Instructions (KDSI) adjusted for code reuse;
(2) the project development mode given as a constant value B (also called the scaling factor);
(3) 15 cost drivers.

Intermediate model has following attributes:

(1) Product Attributes
  - Required s/w reliability (RELY)
  - Size of application database (DATA)
  - Complexity of the product (CPLX)

(2) Hardware Attributes
  - Run time performance constraints (TIME)
  - Memory constraints (STOR)
  - Virtual machine volatility (VIRT)
  - Turnaround time (TURN)

(3) Personal Attributes
  - Analyst capability (ACAP)
  - Programmer capability (PCAO)
  - Application experience(AEXP)
  - Virtual m/c experience (VEXP)
  - Programming language experience (LEXP)

(4) Project Attributes
  - Modern programming practices (MODP)
  - Use of software tools (TOOL)
  - Required development Schedule (SCED)

Effort and Development Time can be calculated using following:

Where EAF is Effort Adjustment Factor. The product of all effort multipliers results in an effort adjustment factor (EAF). Typical values for EAF range from 0.9 to 1.4.

Project	a_i	b_i	c_i	d_i
Organic	3.2	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	2.8	1.2	2.5	0.32

Table 1.1: Intermediate COCOMO coefficients

Like all empirical models, COCOMO model has some shortcomings. Although it is simple to apply and offers a more adaptable tailored model, COCOMO has some weakness similar to LOC based techniques [4]. First, unlike function points, it is hard to estimate size of software at initial stage of project. Measured number of lines of code is not accurate as function points. Software size is dependent to programming language, because some languages may represent the same functionality with less number of lines of code than another one. According to COCOMO II, effort can be calculated as:

$$\text{Effort} = A \times (\text{Size})^{B+0.01\times\sum_{i=1}^{5}\text{SF}_i} \times \prod_{i=1}^{17}\text{EM}_i$$

where A and B are baseline calibration constants, Size refers to the size of the software project measured in terms of thousands of Source Lines of Code (kSLOC), SF the scale factor, and EM is the effort multiplier. Significant effort has been put into the research of developing software estimation models using neural networks[2].

B. Neural Network

Artificial neural networks are purely data driven models which through training iteratively transition from a random state to a final model [6]. While theoretically universal approximations, there are practical problems in neural network model construction and validation when dealing with stochastic relationships, or noisy, sparse or biased data. An artificial neural network [7] is modeled as a massively parallel-interconnected network of elementary processors or neurons. It has been shown that a three-layer feed forward network can generate arbitrary complex decision regions.

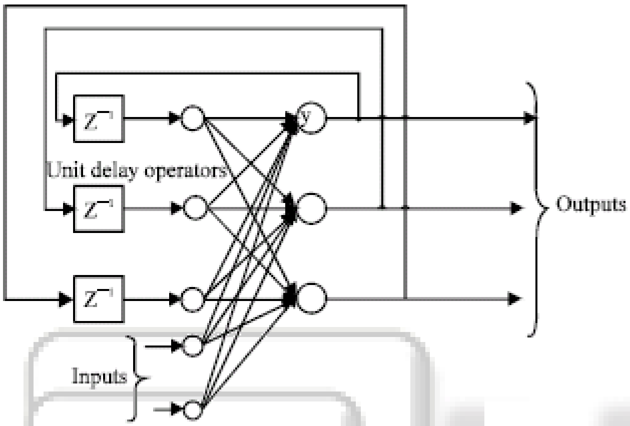

Fig.1.2: The back propagation neural network

An Artificial Neural Network is a network of many very simple processors 'units', each possibly having a small amount of local memory [9]. The units are connected by unidirectional communication channels 'connections', which carry numeric as opposed to symbolic data. The units operate only on their local data and on the inputs they receive via the connections. Artificial neural networks are purely data driven models which through training iteratively transition from a random state to a final model [14]. While theoretically universal approximations, there are practical problems in neural network model construction and validation when dealing with stochastic relationships, or noisy, sparse or biased data. Multiple-layer networks the number of layers determines the superscript on the weight matrices.

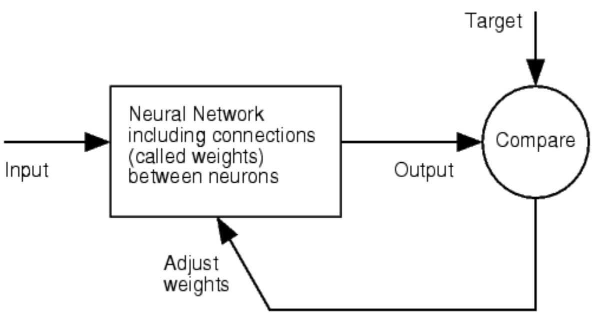

Fig. 1.1: Basic neural network classification

II. PPROBLEM FORMULATION

The goal of this paper is to study the indifference present in the inputs of the algorithmic models like Constructive Cost Model (COCOMO) that yields indifference in the output, resulting in erroneous effort estimation. Neural Network based cost estimation models are suitable to address the automate, vagueness and imprecision in the inputs, to make reliable and accurate estimates of effort. The proposed paper extends the traditional cost estimation model COCOMO by incorporating the concept of neural network into the measurements of size, mode of development for projects and the cost drivers contributing to the overall development effort.

III. OBJECTIVE

(1) COCOMO II which is the most popular tool for estimating software cost and uses lines of code and function points to assess software size. However, these are actually implementation details and difficult to estimate during the early stage of software development [5].
(2) A software cost estimation models will be proposed that fit the different software development environments in the early stages of the software development life cycle [16].
(3) In this research work, a model will be proposed to model the software cost estimation using an artificial neural network approach.
(4) Through this model, initially we will train the network with input values for different projects. Then, we will simulate the values for test data of projects taken from the input itself to check for accuracy of achieved outcomes.

IV. METHODOLOGY/PLANNING OF WORK

A COCOMO intermediate model is implemented with multiple projects for the said problem. This model is made in the form of Neural Network model by taking data in the form of parameter values of COCOMO model from NASA projects to analyze and implement cost estimation model. A set of values of parameters that includes planned effort and actual effort are fed into the Neural Network [12]. An artificial neural network is basically acts as a massively parallel-distributed network of neurons. A three-layer feed forward network can generate arbitrary complex decision regions. The multi-layered neural networks operate in two modes: Training and testing. In the training mode, a set of training data is used to adjust the weights of the network interconnections so that the network responds in a specified manner. In the testing mode, the trained network is evaluated by the test data[10].

A. Network Creation

A feed-forward back-propagation network is created. The function newff creates a feed forward network. It requires three arguments and returns the network object. The first argument is a matrix of sample R-element input vectors. The second argument is a matrix of sample S-element target vectors [15]. The sample inputs and outputs are used to set up network input and output dimensions and parameters. The third argument is an array containing the sizes of each hidden layer. (The output layer size is determined from the targets). More optional arguments can be provided [13]. For instance, the fourth argument is a cell array containing the names of the transfer functions to be used in each layer. The fifth argument contains the name of the training function to be used.

The default training function is trainlm. newff command creates the network object and also initializes the weights and biases of the network; therefore the network is ready for training [17] . The command used in this work is given as under:

```
net=newff(source,target,no_param,{},'trainlm');
```

Before training a feedforward network, initialization of the weights and biases has to be done. The newff command automatically initializes the weights, but you might want to reinitialize them. This can be done with the init command. This function takes a network object as input and returns a network object with all weights and biases initialized. A network can be initialized (or reinitialized) by using:

```
net = init(net);
```

B. *Training*

The first part is the training phase, where we manually identify the correct class. In this research work, a Feed Forward Multi-Layer Perceptron network (MLPN) with one hidden layer has been used. For training, back-propagation algorithm has been implemented

V. RESULTS ANALYSIS

A model is proposed to provide cost in terms of planned effort and development time using COCOMO Model and Neural Network. A comparison between planned effort and actual effort is being done. A model has been proposed that uses a Neural Network with output values obtained through COCOMO model and gives planned effort and Development Time on the basis of values obtained from NASA projects.

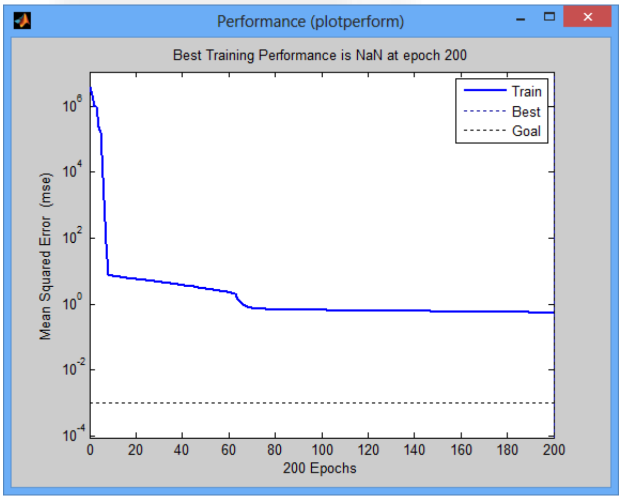

Fig. 1.3: Performance of Neural Network Training

VI. CONCLUSION

Software cost estimation model using COCOMO model Here, have been analyzed and studied by calculating Planned Effort and Development time in this paper. The output in the form of indifference in planned effort and actual effort has been alculated. The proposed model used Neural Network toolbox of MatLab and using output values of COCOMO model that gives planned effort and Development Time on the basis of NASA projects. Software engineers may get benefit of the proposed model for more realistic estimation of the project effort and development time that implies software cost. Thereby, reducing the software cost.

VII. REFERENCES

[1] Alaa F. Sheta, “Estimation of the COCOMO Model Parameters Using Genetic Algorithms for NASA Software Projects”, Journal of Computer Science, Vol. 2, No. 2, pp. 118-123, 2006.

[2] Kaushik, A.K. Soni, Rachna Soni, “A Simple Neural Network Approach to Software Cost Estimation”, Global Journals of Computer Science & Technology, Vol. 13, Issue 1, Version 1, pp. 23-30, 2013.

[3] Abeer Hamdy, “Fuzzy Logic for Enhancing the Sensitivity of COCOMO Cost Model”, Journal of Emerging Trends in Computing and Information Sciences, Vol. 3, No. 9, pp. 1292-1297, Sep. 2012.

[4] Anupama Kaushik, A. K. Soni, Rachna Soni, “A Comparative Study on Fuzzy Approaches for COCOMO’s Effort Estimation”, International Journal of Computer Theory and Engineering, Vol. 4, No. 6, pp. 990-993, Dec. 2012.

[5] Boehm, A.W. Brown, R. Madachy, Ye Yang, “A software product line life cycle cost estimation model”, IEEE Proc. of International Symposium on Empirical Software Engineering, pp. 156 - 164, 19-20 Aug. 2004.

[6] Ch. Satyananda Reddy, P. Sankara Rao, KVSVN Raju, V. Valli Kumari, “A New Approach For Estimating Software Effort Using RBFN Network”, International Journal of Computer Science and Network Security, Vol. 8, No.7, pp. 237-241, July 2008.

[7] Harish Mittal, Pradeep Bhatia, "A comparative study of conventional effort estimation and fuzzy effort estimation based on Triangular Fuzzy Numbers", International Journal of Computer Science and Security, Volume 1, Issue 4, pp. 36-47, 2007.

[8] Harish Mittal, Pradeep Bhatia, "Optimization Criteria for Effort Estimation using Fuzzy Technique", CLEI Electronic Journal , Vol. 10, No. 1, Paper 2, pp. 1-11, June 2007.

[9] I. Attarzadeh, Siew Hock Ow, “A novel soft computing model to increase the accuracy of software development cost estimation”, IEEE 2nd International Conference on Computer and Automation Engineering (ICCAE), Vol. 3, pp. 603 - 607, 26-28 Feb. 2010.

[10] Iman Attarzadeh, Siew Hock Ow, “A Novel Algorithmic Cost Estimation Model Based on Soft Computing Technique”, Journal of Computer Science, Vol. 6, No. 2, pp. 117-125, 2010.

[11] J.N.V.R Swarup Kumar, Aravind Mandala, M. Vishnu Chaitanya, G.V.S.N.R.V Prasad, “Fuzzy logic for Software Effort Estimation Using Polynomial Regression as Firing Interval”, Int. J. Comp. Tech. Appl., Vol. 2, No. 6,pp. 1843-1847, Dec. 2011.

[12] J. S. Pahariya, V. Ravi, M. Carr, "Software Cost Estimation using Computational Intelligence Techniques", IEEE World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), pp. 849-854, 2009.
[13] K. Pillai, V. S. Sukumaran Nair, "A model for software development effort and cost estimation", IEEE Transactions on Software Engineering, Vol. 23, Issue 8, pp. 485-497, Aug. 1997.
[14] K. Hamdan, H. El Khatib, J. Moses, P. Smith, "A Software Cost Ontology System for Assisting Estimation of Software Project Effort for Use with Case-Based Reasoning", IEEE Innovations in Information Technology, pp. 1-5, Nov. 2006.
[15] Madhu S. Nair and Jaya vijayan, "Simplified Neural Model for the Software Development Team Optmization", International Arab Journal of Information Technology, Vol. 5, No. 2, April 2008.
[16] MatLab R2010 Neural Network Tool Box Product Help
[17] Mitat Uysal, “Estimation of the Effort Component of the Software Projects Using Simulated Annealing Algorithm”, World Academy of Science, Engineering and Technology, Vol. 17, pp. 234-237, 2008.