International Journal of Engineering Sciences Paradigms and Researches (IJESPR)
(Vol. 13, Issue 01) and (Publishing Month: May 2014)
(An Indexed, Referred and Impact Factor Journal)
ISSN (Online): 2319-6564
www.ijesonline.com

Software Cost Estimation with Neural Network Techniques

Shilpa Malhotra[1] and Yogita Gulati[2]

[1]M.Tech. Scholar, CSE Deptt., PIET, Samalkha, Haryana
er.shilpamalhotra25@gmail.com
[2]Astt. Prof., CSE Deptt., PIET, Samalkha, Haryana

Abstract
Software cost Estimation is a process of predicting the effort needed to develop a software system in person-months. It is a very critical process and the estimates made at the starting of the software are not accurate. Change in environment makes cost prediction more difficult. Here most commonly used model that has been successfully applied is compared with neural network techniques. Neural network gives better results than most often used COCOMO model. Here two types of ANN have been simulated.

Keywords*: Effort Estimation, Neural Network, COCOMO Model.*

1. Introduction

In computer system projects most expensive component is the software and the cost of the software development depends heavily on the effort utilized in the project. Most of the cost estimation methods give estimation in terms of person-months. Accurate software cost estimates are critical to both developers and customers. They can be used for generating request for proposals, contract negotiations, scheduling, monitoring and control. Underestimating the costs may get the proposed software approved by management but later on, results in underdeveloped functions and poor quality, and failure to complete on time. Overestimating may result in too many resources committed to the project, or, during contract bidding, result in losing the contract.

Software cost estimation involves the determination of the following estimates:

- effort (usually in person-months)
- project duration (in calendar time)
- cost (in dollars)

Most cost estimation models generate an effort estimate, which is then converted into the project duration and cost. This effort estimate can be converted into cost figure by calculating an average salary and multiplying it by the estimated effort required.

Cost estimation methods can be broadly divided into two methods: algorithmic and non-algorithmic. Non-algorithmic methods are:

a) Analogy costing: This method requires one or more completed project of the same type and the estimation is done through reasoning by analogy using the actual costs of previous projects.

b) Expert judgment**:** This method involves consulting one or more experts. The experts provide estimates using their own methods and experience.

c) Bottom-up: In this approach, each component of the software system is separately estimated and the results aggregated to produce an estimate for the overall system.

d) Top-down: This approach is the opposite of the bottom-up method. An overall cost estimate for the system is derived from global properties, using either algorithmic or non-algorithmic methods. The total cost can then be split up among the various components. This approach is more suitable for cost estimation at the early stage.

Algorithmic models depend upon statistical analysis of historical data follow a computational approach and suggest the use of variables, different attributes

of the software project generally called cost drivers, to produce an estimate. Algorithmic model includes two most popular models used as follows:
a) COCOMO Model
b) Putnam's Model and SLIM

2. Literature Review

2.1COCOMO Model

COCOMO was proposed by Boehm in 1981,it is a linear-least square regression model with Line of Code (LOC) as unit of measure for size. Boehm has proposed three levels of Cocomo namely Basic, Intermediate and Detailed. The general form of equation for estimating Effort, Productivity, and schedule with Basic cocomo is given as:

Effort = a(KLOC)b (1)
Productivity = KLOC/Effort (2)
Schedule (months) = c(Effort)d (3)

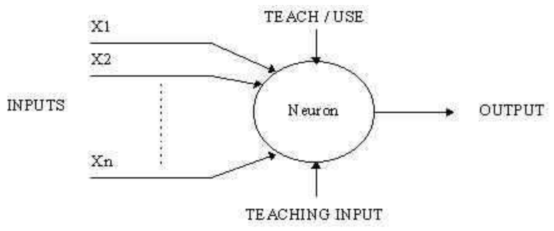
The numeric value for the coefficients *a, b, c* and *d* appeared in these equations represents the three development modes namely Organic, Embedded and Semi-Detached. KLOC is lines of code in thousands. Intermediate Model computes effort as a function of program size and a set of Cost drivers. The equation for estimating software Effort for intermediate model slightly differs from Basic cocomo as:

$$\text{Effort} = a\,(\text{KLOC})\,b \,.\, \prod_{i=1}^{15} \text{EM } i$$

Where EM is effort adjustment factor and it is the product of 15 Effort Multipliers, *a* and *b* are parameters whose values are derived from three modes as discussed above. Detailed model include all characteristics of intermediate model with the difference that the impact of cost drivers is assessed for each and every phase of
software engineering process.

2.2 Neural Network

A *NN* is generally depicted on the basis of learning rules, characteristics adopted by neuron (nodes) and net topology. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output.



A Simple Neuron

There are two types of artificial neural networks: feed forward network and feedback network. Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. Feedback networks can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. There are two types of learning methods: Supervised learning and unsupervised learning. Supervised learning incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning. Unsupervised learning uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian learning and competitive learning.

2.2.1Back Propagation Network:

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (EW). The algorithm computes each EW by first computing the EA, the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EA has been computed for a unit, it is straight forward to compute the EW for each incoming connection of the unit. The EW is the product of the EA and the activity through the incoming connection.

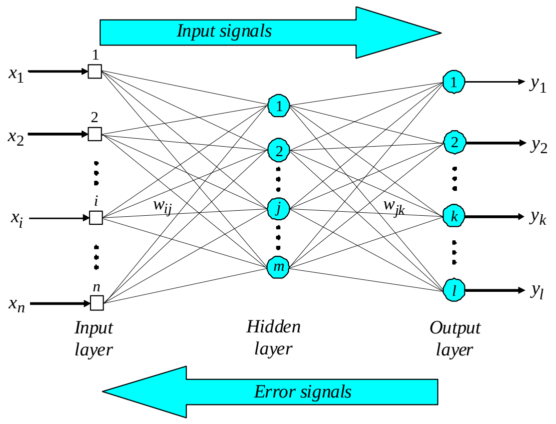


Fig 1.Back Propagation Network

2.2.2Cascade Correlation Learning

Cascade-correlation (CC) is an architecture and generative, feed-forward, supervised learning Algorithm for artificial neural networks. Cascade-Correlation begins with a minimal network, then automatically trains and adds new hidden units one by one creating a multi-layer structure. Cascade-Correlation (CC) combines two ideas: The first is the cascade architecture, in which hidden units are fixed which do not change once added. The second is the learning algorithm, which creates and installs the new hidden units. To install new hidden unit, the algorithm maximize the magnitude of the correlation between the new unit's output and the residual error signal of the network. Steps of the algorithms are following:

Step 1: CC starts with a minimal network consisting only of an input and an output layer. Both layers are fully connected.

Step 2: Train all the connections ending at an output unit with a usual learning algorithm until the error of the net no longer decreases.

Step 3: Generate the so-called candidate units. Every candidate unit is connected with all input units and with all existing hidden units. Between the pool of candidate units and the output units there are no weights.

Step 4: Try to maximize the correlation between the activation of the candidate units and the residual error of the net by training all the links leading to a candidate unit. Learning takes place with an ordinary learning algorithm. The training is stopped when the correlation scores no longer improves.

Step 5: Choose the candidate unit with the maximum correlation, freeze its incoming weights and add it to the net. To change the candidate unit into a hidden unit, generate links between the selected unit and all the output units. Since the weights leading to the new hidden unit are frozen, a new permanent feature detector is obtained. Loop back to step 2.

Step 6: This algorithm is repeated until the overall error of the net falls below a given value.

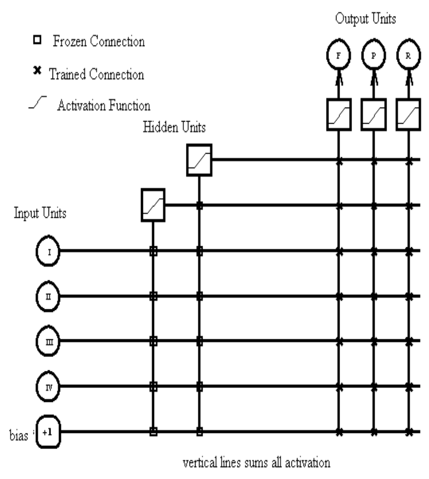


Fig 2.Cascade Correlation Network

3. Experimental Setup

3.1 Data

Results in neural networks will be calculated by taking historical data [36], [37] of 50 projects which is divided into three parts: 30 projects data for training the network, 10 projects for validating the network and 10 projects for testing the network.

4. Performance Criteria

A. Mean Magnitude Relative Error (MMRE)

MMRE is frequently used to evaluate the performance of any estimation technique. It measures the percentage of the absolute values of the relative errors, averaged over the N items in the "Test" set and can be written as

$$MMRE = ((1/N) (\sum_{i=1}^{N} |(E_{rror})_i| / E_i))$$

Where N is total number of projects

B. Root Mean Square Error (RMSE)

RMSE is another frequently used performance criteria which measures the difference between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated. It is just the square root of the mean square error, as shown in equation given below:

$$RMSE = (\sqrt{((1/N) (\sum_{i=1}^{N}((E_{rror})_i)^2))})$$

Where N is total number of projects.

5. Comparison of Different Cost Prediction Techniques

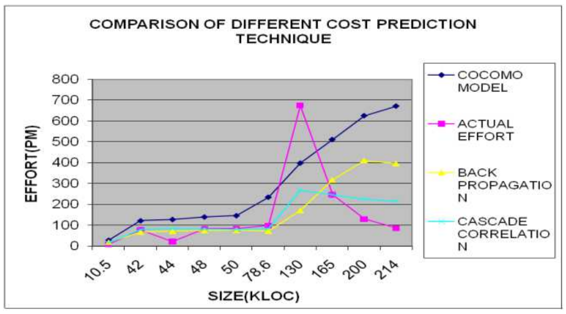


Fig 3.Comparison of COCOMO Model, Back-propagtion and Cascade Correlation learning algorithms

5.1Error Prediction of Different Cost Prediction Techniques

Here, error will be calculated of different Cost Prediction Techniques.
To calculate the error following formule will be used:

$E_{rror} = | E – E' |$
Where
E_{rror} is the output error,
E is the actual effort,
E' is the estimated effort
In this section
$E_1 = | E - E'_{coc} |$
$E_2 = | E - E'_{back} |$
$E_3 = | E - E'_{casc} |$

Where
E is the actual effort,
E_1 error of COCOMO Model
E'_{coc} is the estimated effort using COCOMO Model
E_2 error of Back-propagation learning algorithm
E'_{back} is the estimated effort using Back-propagation learning
E_3 error of Cascade Correlation learning algorithm
E'_{casc} is the estimated effort using Cascade Correlation learning.

Table1

Performance Criteria	COCOMO Model	Back-propagation	Cascade Correlation
RMSE	277.74	208.71	139.15
MMRE	2.1557	1.1072	0.6228

5.2Perforamnce Evaluation of COCOMO Model,Back-propagation and Cascade Correlation learning algorithms

Following graphs shows the comparison using RMSE(Root Mean Square Error) of different cost prediction techniques.

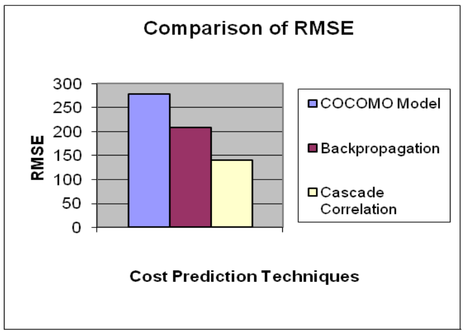


Figure 4 Comparison using RMSE

Following graphs shows the comparison using MMRE(Mean Magnitude Relative Error) of different cost prediction techniques.

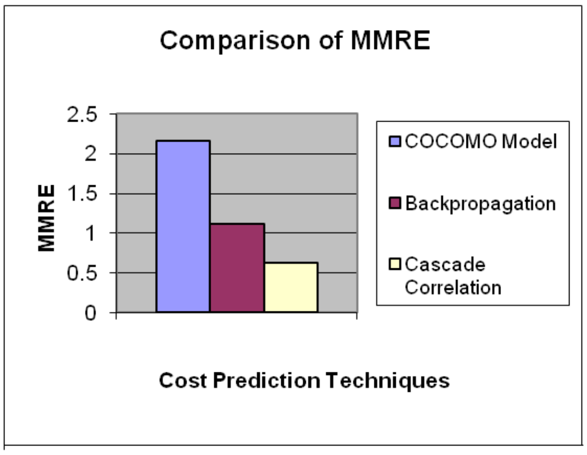


Figure 5 Comparison using MMRE

Comparison using RMSE and MMRE shows that Cascade Correlation is better than Back-propagtion and COCOMO Model. Accuracy is high in the Cascade Correlation.

6. Conclusion

A reliable and accurate estimate of software development effort has always been a challenge for both the software industrial and academic communities. Here three most popular approaches were suggested to predict the software cost estimation. First COCOMO which has been already proven and successfully applied in the software cost estimation field and in second the Back-propagation learning algorithm and the third one is cascade-correlation learning algorithm in Neural Network. After testing the network it is concluded that learning algorithms of neural network perform better then the COCOMO model and from learning algorithms Cascade correlation performs better then the Back-propagation learning algorithm. It has less error values, so accuracy is high in Cascade Correlation.

References

[1] Clark, B., Chulani, S. and Boehm, B. (1998), “Calibrating the COCOMO II Post Architecture Model,” International Conference on Software Engineering, Apr. 1998.
[2] Christos Stergiou and Dimitrios Siganos, “Neural Networks”
[3] Christopher M. Fraser (2000), “Neural Networks: A Review from a Statistical Perspective”, Hayward Statistics
[4] Nasser Tadayon, “Neural Network Approach for Software Cost Estimation”, IEEE proceedings of the International Conference on Information Technology: Coding and Computing (ITCC ’ 2005)
[5] S. Kanmani, J. Kathiravan, S. Senthil Kumar and M. Shanmugam, “Neural Networks Based Effort Estimation using Class Points for OO Systems”,IEEE proceedings of the International Conference on Computing: Theory and Applications(ICCTA’2007)
[6] Ch. Satyananda Reddy, P. Sankara Rao, KVSVN Raju, V. Valli Kumari, “A New Approach For Estimating Software Effort Using RBFN Network” , IJCSNS International Journal of Computer Science and Network Security, VOL.8 No. 7, July 2008

[7] Kiyoshi Kawaguchi," Back propagation Learning Algorithm", Wikipedia.org, June 2000.

[8] Cascade Correlation Architecture and Learning Algorithms for Neural Networks", Wikipedia.org, Nov. 1995

[9] Konstantinos Adamopoulos," Application of Back.

[10] P.V.G.D. Prasad Reddy and CH.V.M.K. Hari, A Fine parameter tuning for COCOMO 81 software effort estimation using Particle swarm optimization, A. Iman and H.O. Siew, Soft Computing Approach for Software Cost Estimation, Int.J. of Software Engineering, IJSE Vol.3 No.1, pp.1-10, January 2010.

[11]Srinivasa Rao et al, Predictive and Stochastic Approach for Software Effort Estimation, Int. J. of Software Engineering, IJSE Vol. 6 No. 1 January 2013.

[12] Peram Subba Rao, Dr.K.Venkata Rao and Dr.P.Suresh Varma, "A Novel Software Interval Type - 2 Fuzzy Effort Estimation Model using S-Fuzzy Controller With Mean and Standard Deviation", International Journal of Computer Engineering & Technology (IJCET), Volume 4, Issue 3, 2013, pp. 477 - 490, ISSN Print: 0976 – 6367, ISSN Online: 0976 – 6375.

[13]PourushBassi, "Neural Network-A Novel Technique for Software Effort Estimation", International Journal of Computer Theory and Engineering, Vol. 2, No. 1 February, 2010, page:17 19.

[14] Roheet Bhatnagar, Vandana Bhattacharjee and Mrinal Kanti Ghose, "Software Development Effort Estimation –Neural Network Vs. Regression Modeling Approach", International Journal of Engineering Science and Technology, Vol. 2(7), 2010, page: 2950-2956.