Effort Estimation with Neural Network Back Propagation

Puneet Garg[1]
[1]Department of Computer Science &Engineering,
Ganga Institute of Technology and Management, Kablana,
Jhajjar, Haryana, INDIA

Abstract: **Effort Estimation is the process of predicting the effort needed to develop software. Estimating the effort is a very crucial task in software industry. In order to get accurate estimates a lot of algorithmic models have been developed. Estimates done at the initial stage are inaccurate, where requirements are not elaborated but as the project progresses accuracy on estimate increases. So it is necessary to choose a right estimating technique. In this paper Effort estimation is calculated through a non linear approach Neural Network Back Propagation.COCOMONASA dataset has been used to validate and test the effort. It is shown that neural network provides better output than most commonly used COCOMO model.**

Keywords: Software Effort Estimation, Back propagation, RMSE, MMRE,COCOMO Model.

I. INTRODUCTION:

Due to the increase in demand in software industry it has become most rapidly growing field. For a software project to be successful, a lot of evaluation, monitoring of resources, analysis are done. A small deviation from these factors can take the project to a wrong way. Therefore effective management techniques need to be considered. Effort estimates are used to calculate the effort in person-months for the software development. Accurate estimates for development effort and cost are crucial for managing financial matters. There are a lot of estimation models which have been proposed that are estimation by expert, analogy based estimation schemes, algorithmic methods including empirical methods, artificial neural network based approaches, decision tree based methods and fuzzy logic based estimation schemes. Among Software estimation techniques, COCOMO is the most commonly used model because it is simple to use. It uses mathematical formula to estimate the effort.

COCOMO was proposed by Boehm in 1981.Software estimation can be done through three stages: Basic COCOMO, Intermediate COCOMO, and Detailed COCOMO.

1. Basic COCOMO: It gives approximate estimate of project parameters.It uses the expression:

$$Effort=a*(KLOC)^b \text{ PM}$$

Mode	a	b
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

Table I. Coefficients for Basic COCOMO

KLOC is the estimated size of the software in Kilo lines of code, a and b are constants. The values of a and b depends upon the mode of development. Effort is the total effort needed to develop the software expressed in person-months. There are three modes of project:

(a) Organic mode- It is used for the project size up to 2-50 KLOC (thousand lines of code) with experienced developers in a familiar environment.

(b) Semi-Detached mode- It is used for the project size up to 50-300 KLOC with average previous experience on similar projects.

(c) Embedded mode- It is used for the project size over 300 KLOC with developers having very little previous experience.

2. Intermediate COCOMO: It takes into consideration 15 other cost drivers (multipliers) except size. The cost drivers can be classified under:

(a) Product: It includes complexity of the product, Database size, reliability requirements of the product.

(b) Computer: It includes execution speed and storage space required, virtual machine volatility, computer turnaround time.

(c)Personnel: It includes analyst capability, application experience, programmer capability, virtual machine experience, programming language experience.

(d) Development Environment: It includes modern

Programming practices, use of software tools and development schedule.

3. Detailed COCOMO: Two more capabilities were introduced in this model and they are Phase sensitive effort multipliers which help in determining the manpower allocation needed for each phase of the project and three level product hierarchy which are module, subsystem and system levels. The ratings of the cost drivers are done at

appropriate level. This approach reduces the margin of error in the final estimate.

II. ARTIFICIAL NEURAL NETWORK:

A neural network is a interconnected network of neurons which apply some computational model to process the information. Neurons are connected in layers to form a network. A neural network is a non linear approach to deal with complex inputs and outputs. Neural Network works by learning. Learning is the process by which a NN modifies its internal structure in response to the external stimuli. Learning is broadly classified into two types: Supervised and Unsupervised Learning. In supervised learning goal is to minimize the error between desired output and the actual output produced by network. when the learning is over the network structure becomes fixed and becomes operational. In Unsupervised learning the target or the desired output is not given to the network. It follows a self-supervised method and does not use external influences for updating weights or biases. Unsupervised methods are Hebbian Learning, Reinforced Learning.

Back-Propagation: Back-propagation learning (BPL) algorithm was invented in 1969 for learning in multilayer network. The back-propagation algorithm trains a given feed forward multilayer neural network for a given set of input patterns with known classifications. When each entry of the sample set is presented to the network, the network examines its output response to the sample input pattern. The output response is then compared to the known and desired output and the error value is calculated. Based on the error, the connection weights are adjusted. The back propagation algorithm is based on Widrow-Hoff delta learning rule in which the weight adjustment is done through mean square error of the output response to the sample input .The set of these sample patterns are repeatedly presented to the network until the error value is minimized. Back-propagation learning algorithm uses training data to adjust the weights and threshold of neurons so as to minimize the error. It is based on the differences between the actual and the desired output. It works by applying the gradient descent rule to feed-forward network. The algorithm involves two phases, the forward phase that occurs when the inputs (external stimuli) are presented to the neurons of the input layer and are propagated forward to compute the output and the backward phase, when the algorithm performs modifications in the backward direction.

III. EVALUATION 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. It is written as:

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

Where N is total number of projects

Error represents the difference between actual effort and estimated effort.

B. Root Mean Square Error(RMSE)

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

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

Where N is total number of projects

Error represents the difference between actual effort and estimated effort.

IV. EXPERIMENT

Data Preparation

We have used NASA public dataset for this experiment. This dataset consists of 60 projects data. In this dataset 17 attributes are there. The values of 15 attributes were in fuzzy format, so for the experiment we have converted it into numeric format. MATLAB2008 NN tool is used for this experiment. For training the dataset is divided into three divisions-for training 36(60%), for validation 12(20%) and for testing 12(20%). Stopping criteria was set by number of epochs as 1000 and goal as 0.00.

Experimental Parameters:

Parameters used for performing the operation in neural network

Parameters	Back Propagation Algorithm
Network type	Feed-forward back prop
Training Function	Trainlm
Performance Function	MSE
Transfer Function	LOGSIG
No. of Epochs	11

Table II. Parameters for Neural Network learning algorithm

Evaluation of Results

This section analyze the result of the neural network Back Propagation and the COCOMO model. Table-III summarizes the development effort estimates as obtained with different neural networks.

Project No.	Size	Actual Effort	COCOMO Effort	Back Propagation
49.	50	370	145.92	232
50.	101	750	305.31	876
51.	190	420	592.79	460
52.	47.5	252	138.27	164
53.	21	107	58.68	105
54.	423	2300	1373.62	2633
55.	79	400	235.89	471
56	284.7	973	906.39	978
57	282.1	1368	877.04	1387
58	78	571.4	232.76	644
59	11.4	98.8	30.9	166
60	19.3	155	53.7	222

Table III. Comparison of Effort Estimation

Following graph shows that effort calculated using Back Propagation is nearest to the actual effort ac compared to COCOMO Model.

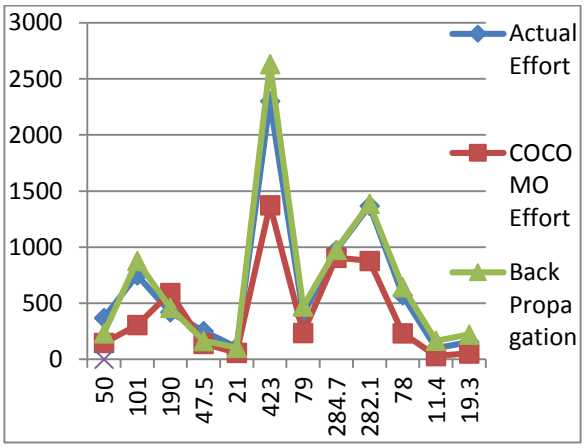

Table-IV presents a performance comparison of neural networks AND COCOMO Model. On the basis of RMSE and MMRE.

Performance criteria	COCOMO model	Back Propagation Algorithm
RMSE	359.29	120.89
MMRE	2.56	0.215

Table IV. Comparison of Result

V.CONCLUSION

Accurate estimation of software development effort is always a challenge in software industry. There are several effort predicting models which have been proposed and applied. We have constructed a estimation model based on Neural Network.BackPropagation algorithm has been used to train the network.NASA Dataset of 60 projects is used and it is observed that Neural Network Back Propagation provides better results than COCOMO Model. So we can conclude that present work provides better prediction software development effort.

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.