ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

Analysis of Neural Network based Approaches for Software effort Estimation and Comparison with Intermediate COCOMO

Manpreet Kaur, Sushil Garg

Abstract: ***Effort estimation consists in predict how many hours of work and how many workers are needed to develop a project. The effort invested in a software project is probably one of the most important and most analyzed variables in recent years in the process of project management. The ability to accurately estimate the time and/or cost taken for a project to come in to its successful conclusion is a serious problem for software engineers. However, in the context of set of resources, planning involves estimation - your attempt to determine how much money, how much effort, how many resources, and how much time it will take to build a specific software-based system or product. In this paper we will study the efficiency of Neural Network based cost estimation model with the traditional cost estimation model like Halstead Model, Bailey-Basili Model, and Doty Model. We conclude our result with the proposal of Neuron based Model basis on Back propagation Technique.***

Index-terms: **Efforts, Intermediate COCOMO, Resilient Back Propagation, Batch Gradient Descent Neural Network.**

I. INTRODUCTION

The original inspiration for the term Artificial Neural Network came from examination of central nervous systems . In an artificial neural network, simple artificial nodes, variously called "neurons", "neurodes", "processing elements" (PEs) or "units", are connected together to form a network of nodes mimicking the biological neural networks — hence the term "artificial neural network". Neural networks are composed of interconnecting artificial neurons (programming constructs that mimic the properties of biological neurons). Artificial neural networks may either be used to gain an understanding of biological neural networks, or for solving artificial intelligence problems without necessarily creating a model of a real biological system. The real, biological nervous system is highly complex: artificial neural network algorithms attempt to abstract this complexity and focus on what may hypothetically matter most from an information processing point of view. A Neural Network (NN) is a computer software (and possibly hardware) that simulates a simple model of neural cells in animals and humans. The purpose of this simulation is to acquire the intelligent features of these cells. In this document, when terms like neuron, neural network, learning, or experience are mentioned, it should be understood that we are using them only in the context of a NN as computer system. NNs have the ability to learn by example, e.g. a NN can be trained to recognize the image of car by showing it many examples of a car.

II. LITRATURE SURVEY

Efficient software project estimation is one of the most demanding tasks in software development. Accurate estimate means better planning and efficient use of project resources such as cost, duration and effort requirements for software projects especially space and military projects [1], [2]. Problem of inaccurate estimate for projects and in many cases inability to set the correct release day for their software correctly lead to inefficient use of project resources. Unfortunately, software industry suffers the problem of incorrect estimate for projects and in many cases inability to set the correct release day for their software correctly. This leads to many losses in their market, e.g. risk due to low quality of the deliverables and penalties for missing the deadlines. Normally, estimation is performed using only human expertise [3], [4], but recently attention has turned to a variety of computer-based learning techniques. In 1995, Standish Group served over 8,000 software projects for the purpose of budget analysis. It was found that 90% of these projects exceeded its initially computed budget. Moreover, 50% of the completed projects lake the original requirements [5]. From these statistics, it can be seen how prevalent the estimation problem is. Evaluation of many software models were presented in [6], [7], [8]. Numerous models were explored to provide better effort estimation [9], [10], [11], [12]. In [4], [13], authors provided a survey on the effort and cost estimation models.
Serious research in the Neural Network area is started in the 1950's and 1960's by researchers like Rosenblatt (Perceptron), Widrow and Hoff (ADALINE). In 1969 Minsky and Papert wrote a book exposing Perceptron limitations. This effectively ended the interest in neural network research. In the late 1980's interest in NN increased with algorithms like Back Propagation, Cognitrons and Kohonen. (Many of them where developed quietly during the 1970s) In the literature of Neural Networks (NNs) the following function is called a sigmoid function.

$$S(x) = 1/ (1 + e^{-a * x})$$

The coefficient *a* is a real number constant. Usually in NN applications *a* is chosen between 0.5 and 2. As a starting point, you could use a=1 and modify it later when you are fine-tuning the network. Note that s(0)= 0.5, s(∞)= 1, s(-∞)=0. (The symbol ∞ means infinity). The Sigmoid function is used on the output of neurons. In a NN context, a neuron is a model of a neural cell in animals and humans. This model is simplistic, but as it turned out, is very practical. In NN the inputs simulate the stimuli/signals that a neuron gets, while the output simulates the response/signal which the neuron generates. The output is calculated by

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

multiplying each input by a different number (called weight), adding them all together, then scaling the total to a number between 0 and 1.The following diagram shows a simple neuron with:

1. Three inputs $[x_1, x_2, x_3]$. The input values are usually scaled to values between 0 and 1.
2. Three input weights $[w_1, w_2, w_3]$. The weights are real numbers that usually are initialized to some random numbers. Do not let the term weight mislead you, it has nothing to do with the physical sense of weight, in a programmer context, think of the weight as a variable of type float/real that you can initialize to a random number between 0 and 1.
3. One output is shown as *z*. A neuron has one (and only one) output. Its value is between 0 and 1. It can be scaled to the full range of actual values.

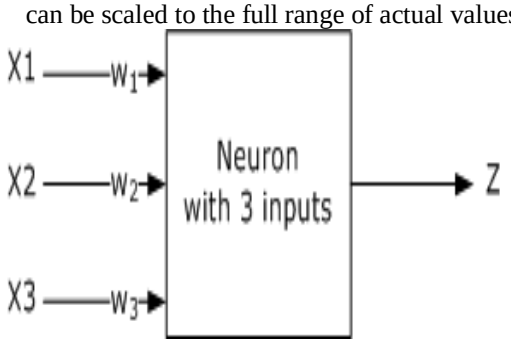

Fig. 1 Neuron Model with 3 inputs

The NN consists of three layers:

- Input layer with three neurons.
- Hidden layer with two neurons.
- Output layer with two neurons.

The output of a neuron in a layer goes to all neurons in the following layer. Each neuron has its own input weights. The weights for the input layer are assumed to be 1 for each input. In other words, input values are not changed and the output of the NN is reached by applying input values to the input layer, passing the output of each neuron to the following layer as input. A suitable training algorithm can be used for updating the weights and thresholds in each iteration to minimize the error. Changing weights and threshold for neurons in the output layer is different from hidden layers. Note that for the input layer, weights remain constant at 1 for each input neuron weight.

III. LIMITATIONS OF PREVIOUS MODELS OF EFFORT ESTIMATION

Inaccurate estimation of software development effort is one of the most important reasons of IT-project failures. While too low effort estimates may lead to project management problems, delayed deliveries, budget overruns and low software quality, too high effort estimates may lead to lost business opportunities and inefficient use of resources. These problems motivated us to conduct research that aims at significant improvements in effort estimation methods.

- Error deviation is high.
- Difficult to calculate cost.
- Models are complex.
- Models like walston-felix, doty, Halsted and bailey-Basili does not include all the parameters for calculating cost.
- Estimates are extremely sensitive to the technology factor
- It is hard to accurately estimate KDSI early on in the project, when most effort estimates are required.
- KDSI, actually, is not a size measure it is a length measure.
- Extremely vulnerable to mis-classification of the development mode
- Success depends largely on tuning the model to the needs of the organization, using historical data which is not always available

IV. JUSTIFICATION OF PROBLEM

Typical main model that is being used as benchmarks for software effort estimation is COCOMO. This model has been derived by studying large number of completed software projects from various organizations and applications to explore how project sizes mapped into project effort. But still this model is not able to predict the Effort Estimation accurately. When one designs with Neural Networks alone, the network is a black box that needs to be defined; this is a highly compute-intensive process. A neural network is good at discovering relationships and pattern in the data, so we have proposed Neuro system to model the effort estimation of the software systems. As Neuro based system is able to approximate the non-linear function with more precision and non of the researcher have explored Neuro approach extensively for the Effort Estimation so there is still scope of exploring more neural Network based modeling approaches. Hence, in this proposed study, it is tried to use Neural Network Based Techniques to build a more accurate model that can improve accuracy estimates of effort required to build a software system. The following are the objectives of the study:

I. Study COCOMO Model of Effort Estimation.
II. Calculate the efforts using intermediate COCOMO effort equation:
Effort=EAF.a (KSLOC)^b
Where EAF= effort adjustment factor which is the product of 15 cost driver attribute.
For organic a=3.2 b=1.05
For semi detached a=3.0 b=1.12
For embedded a=2.8 b=1.20
III. Modeling of the Effort Estimation using Batch Gradient Descent Based Neural Network.
IV. Modeling of the Effort Estimation using Resilient Back propagation
V. Perform comparison between on the basis of MMRE: Following are the steps to calculate MMRE:
Error= Actual efforts – estimated efforts
Relative error (RE) = error/actual efforts
Magnitude of relative error (MRE) = abs (RE)
Mean magnitude of relative error:

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

(MMRE)= (1/T)*(MRE1+MRE2+…..MRET)

Where T is total number of projects.

VI. Deducing conclusions at the end that which one is Better on the basis of MMRE

V. RESULTS AND DISCUSSION

The dataset of NASA is used for the comparison of different algorithms of neural networks with COCOMO. The calculated efforts and errors using different models are shown as below.

Table1. Data of Actual cost comparison with others

Project	Actual efforts	Efforts of BGDM	Efforts with RBP	Efforts with cocomo
1	117.6	99.78	115.8	100.8
2	117.6	101.4	113.07	95.2
3	31.2	123.2	82.9	25.9
4	36	122.5	83.6	27.8
5	25.2	120.6	86.1	33.5
6	8.4	130.4	75.1	6.37
7	10.8	128.7	76.8	10.7
8	352.8	50.1	241.9	290.6
9	72	878.7	17.5	45.4
10	72	1087.1	83.2	32.9
11	24	979.2	26.1	10.5
12	360	1114.7	219.4	200.1
13	36	1238.4	8.24	27.9
14	215	1784.2	437.4	442.7
15	48	981.6	31.3	40.5
16	360	55.02	348.4	418.2
17	324	732.4	323.9	491.4
18	60	804.7	19.1	74.7
19	48	980.8	29.3	29.3
20	60	655.2	12.94	140.5

Table2. Comparison On the basis of MMRE

Performance Criteria	Model Used			
	BGDM	RBP	COCOMO	Actual
MMRE	0.214	0.14	0.228	0

V. CONCLUSION

The performance of the Neural Network based effort estimation system and COCOMO is compared for effort dataset available with NASA. The results show that the Neural Network system has the lowest MMRE to actual. Hence, the proposed Neuro based system is able to provide good estimation capabilities. It is suggested to use of Neuro based technique to build suitable generalized type of model that can be used for the software effort estimation of all types of the projects.

VI. FUTURE WORK

The proposed models provided good estimation capability compared to traditional model structures. i.e. COCOMO. In the future, one can follow the following directions:

•Explore the use of neural network technique to build suitable model structure for the software effort estimation.

•Further learning functions can be used to estimate the efforts.

•Further advanced Machine learning techniques can be used

•Rough Set theory and Taguchi Analysis can be used to find the impact of the different attributes towards Effort Prediction.

•Simulated Annealing Technique can be used to improve the performance of the NF system.

REFERENCES

[1] L. C. Briand, K. E. Emam, and I. Wieczorek, “Explaining the cost of European space and military projects,” in ICSE ’99: Proceedings of the 21st international conference on Software engineering, (Los Alamitos, CA, USA), pp. 303–312, IEEE Computer Society Press, 1999.

[2] “Estimating software projects,” SIGSOFT Soft. Eng. Notes, vol. 26, no. 4, pp. 60–67, 2001.

[3] J. W. Park R, W. Goethert, “Software cost and schedule estimating: A process improvement initiative,” tech. report, 1994.

[4] M. Shepper and C. Schofield, “Estimating software project effort using analogies,” IEEE Tran. Software Engineering, vol. 23, pp. 736–743, 1997.

[5] T. S. Group, CHAOS Chronicles. PhD thesis, Standish Group Internet Report, 1995.

[6] M. Boraso, C. Montangero, and H. Sedehi, “Software cost estimation: An experimental study of model performances,” tech. report, 1996.

[7] O. Benediktsson, D. Dalcher, K. Reed, and M. Woodman, “COCOMO based effort estimation for iterative and incremental software development,” Software Quality Journal, vol. 11, pp. 265–281, 2003.

[8] T. Menzies, D. Port, Z. Chen, J. Hihn, and S. Stukes, “Validation methods for calibrating software effort models,” in ICSE ’05: Proceedings of the 27th international conference on Software engineering, (New York, NY, USA), pp. 587–595, ACM Press, 2005.

[9] S. Chulani, B. Boehm, and B. Steece, “Calibrating software cost models using Bayesian analysis,” IEEE Trans. Software Engr., July-August 1999, pp. 573–583, 1999. S. Chulani and B. Boehm, “Modeling software defect introduction and removal: Coqualmo (constructive quality model),” tech. report.

[10] S. Devnani-Chulani, “Modeling software defect introduction,” tech. report.

[11] G. Witting and G. Finnie, “Estimating software development effort with connectionist models,” in Proceedings of the Information and Software Technology Conference, pp. 469–476, 1997.

[12] K. Peters, “Software project estimation,” Methods and Tools, vol. 8, no. 2, 2000.

[13] B. Clark, S. Devnani-Chulani, and B. Boehm, “Calibrating the COCOMO ii post-architecture model,” in ICSE ’98: Proceedings of the 20th international conference on Software engineering, (Washington, DC, USA), pp. 477–480, IEEE Computer Society, 1998.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

AUTHOR'S PROFILE

Manpreet Kaur received her B.Tech. Degree in Computer Science and engineering from Lovely Institute of technology (Punjab Technical University, Jalandhar), Jalandhar, India, in 2009, the M.Tech. degree in CSE from RIMT-IET Mandi Gobindgarh (Punjab Technical University, Jalandhar. She is currently working as a lecturer in Lovely Professional University, Jalandhar from December 2010 onwards. Her research interests include Software testing, Software effort estimation etc.

Prof. Sushil Garg, currently working as a Head of Department (computer science and engineering and IT) in RIMT-IET mandi Gobindgarh. His research interest includes Software engineering.