Using Bayesian regulation back propagation algorithm for Software effort estimation and comparing with COCOMO

Manpreet Kaur
M. Tech Computer science & engineering.,
RIMT-IET Mandi Gobindarh.
Punjab,India.

Sushil Garg
Prof. Computer science & engineering,
RIMT-IET Mandi Gobindarh.
Punjab,India.

Abstract: **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. 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. In this paper we will study the efficiency of Bayesian regulation back propagation 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.**

Keywords:: **efforts, intermediate COCOMO, resilient back propagation, batch gradient descent bayesian regulation back propagation.**

I. INTRODUCTION

Bayesian regulation back propagations are composed of interconnecting artificial neurons (programming constructs that mimic the properties of biological neurons). Artificial bayesian regulation back propagations may either be used to gain an understanding of biological bayesian regulation back propagations, 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 bayesian regulation back propagation algorithms attempt to abstract this complexity and focus on what may hypothetically matter most from an information processing point of view. The original inspiration for the term *Artificial Bayesian regulation back propagation* came from examination of central nervous systems . In an artificial bayesian regulation back propagation, simple artificial nodes, variously called "neurons", "neurodes", "processing elements" (PEs) or "units", are connected together to form a network of nodes mimicking the biological bayesian regulation back propagations — hence the term "artificial bayesian regulation back propagation". A Bayesian regulation back propagation (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, bayesian regulation back propagation, learning, or experience are mentioned, it should be understood that we are using them only in the context of a NN as computer system.

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.

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. Good performance (e.g. as measured by good predictive ability, low generalization error), or performance mimicking animal or human error patterns, can then be used as one source of evidence towards supporting the hypothesis that the abstraction really captured something important from the point of view of information processing in the brain. Another incentive for these abstractions is to reduce the amount of computation required to simulate artificial neural networks, so as to allow one to experiment with larger networks and train them on larger data sets.

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.

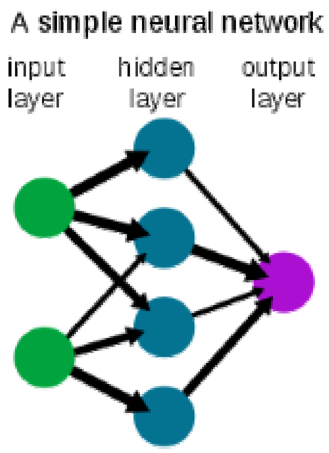
Serious research in the Bayesian regulation back propagation area is started in the 1950's and 1960's by researchers like Rosenblatt (Perceptron), Widrow and Hoff (ADALINE). In 1969 Minsky and Papert wrote a book exposing Perceptron limitations. This effectively ended the interest in bayesian regulation back propagation 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 Bayesian regulation back propagations (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 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 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.

Major problems were:

I. Error deviation is high.
II. Difficult to calculate.
III. Models are complex.
IV. Models like walston-felix, Doty, Halsted and bailey-Basili does not include all the parameters for calculating efforts.
V. Estimates are extremely sensitive to the technology factor.
VI. It is hard to accurately estimate KDSI early on in the project, when most effort estimates are required.
VII. KDSI, actually, is not a size measure it is a length measure.
VIII. Extremely vulnerable to mis-classification of the development mode
IX. Success depends largely on tuning the model to the needs of the organization, using historical data which is not always available.
X. This approach is not repeatable and the means of deriving an estimate are not explicit.
XI. It is difficult to find highly experienced estimators for every new project.
XII. The relationship between cost and system size is not linear. Cost tends to increase exponentially with size. The expert judgment method is appropriate only when the sizes of the current project and past projects are similar.

IV. **PROBLEM FORMULATION**

So following problem is formulated to overcome these limitations. First,

- Study COCOMO Model of Effort Estimation. Then,
- Calculate the efforts using Intermediate COCOMO cost estimation equation".
- Modeling of the effort estimation using Bayesian regulation BP.
- Perform comparison between on the basis of MMRE.

- Deducing conclusions at the end, which one is better 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 RE (MRE) = abs(RE)
  (MMRE)= (1/T)*(MRE1+MRE2+…..MRET)

where T is total number of projects.

- Study COCOMO Model of Effort Estimation.
- Calculate the efforts using intermediate COCOMO effort equation:

$$\textbf{Effort=EAF.a(KSLOC)}^{b}$$

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

V. OBJECTIVE OF THE STUDY

The following are the objectives of the study:

I. Study the existing Models of Effort Estimation.

II. Modeling of the Effort Estimation using bayesian regulation back propagation techniques.

III. Designing of the more accurate effort estimation Model using Soft Computing Techniques.

IV. Testing of the developed Model and Comparison of the proposed Model Results with the existing known Models.

The goal of the research is to improve software cost estimates through better training, practices and tools. Results from the research aims at better control of software projects and more efficient use of IT resources and investments.

VI. RESULTS AND DISCUSSION

The dataset of NASA is used for the comparison of bayesian regulation back propagation with COCOMO. The calculated efforts and errors using different models is shown as below.

Table1. comparison On the basis of MMRE

Performance Criteria	Model Used			
	Doty	BRBP	COCOMO	Actual
MMRE	1.23	0.17	0.274	0

VII. CONCLUSION

The performance of the Bayesian regulation back propagation based effort estimation system and COCOMO is compared for effort dataset available with NASA. The results show that the Bayesian regulation back propagation 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.

VIII. 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 bayesian regulation back propagation 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 Softw. 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 developemnt 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.
14. R. Nelson, Management HandBook for the Estimation of Computer Programming Costs, AD-A648750, Systems Development Corp., 1966, pp. 20-34.
15. R.K.D. Black, R. P. Curnow, R. Katz and M. D. Gray, BCS Software Production Data, Final Technical Report, RADC-TR-77-116, Boeing Computer Services, Inc., March 1977, pp. 5-8.
16. R. E. Park, "PRICE S: The calculation within and why", Proceedings of ISPA Tenth Annual Conference, Brighton, England, July 1988, pp. 231-240.
17. R. Tausworthe, Deep Space Network Software Cost Estimation Model, Jet Propulsion Laboratory Publication 81-7, 1981, pp. 67-78.
18. D. St-Pierre, M Maya, A. Abran, J. Desharnais and P. Bourque, Full Function Points: Counting Practice Manual, Technical Report 1997-04, University of Quebec at Montreal, 1997, pp. 40-45.
19. Abraham A. (2001), "Neuro-fuzzy systems: state-of-the-art modeling techniques, connectionist models of neurons, learning processes, and artificial intelligence", In Mira Jose, Prieto Alberto, editors Lecture notes in computer science, vol. 2084, 2001, pp. 269-276.
20. Abraham, A. (2005), "Adaptation of Fuzzy Inference System Using Neural Learning", Springer Berlin, ISSN: 1434-9922 (Print) 1860-0808 (Online), vol. 181, 2005.
21. Abraham, A. and Khan M.R. (2003), "Neuro-Fuzzy Paradigms for Intelligent Energy Management, Innovations in Intelligent Systems: Design, Management and Applications", Abraham A., Jain L. and Jan van der Zwaag B. (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, Chapter 12, 2003, pp. 285-314.
22. Bherenji, H. R. and Khedkar, P (1992), "Learning and Tuning Fuzzy Logic Controllers through Reinforcements", IEEE Transactions on Neural Networks, vol. 3, 1992, pp. 724-740.
23. Czogala, E., Leski, J., (2000), "Fuzzy and Neuro-Fuzzy Intelligent Systems", Physica-Verlag Heidelberg, New York
24. Feng J.C. and Teng L.C. (1998), "An Online Self Constructing Neural Fuzzy Inference Network and its Applications", IEEE Transactions on Fuzzy Systems, vol. 6, no.1, pp.12-32, 1998.
25. Jang, J.-S.R. and Sun, C.-T. (1995), "Neuro-Fuzzy Modeling and Control", Proceedings of the IEEE, vol. 83, 3, March 1995. pp 378-406.
26. Jang, J.-S.R., Sun, C.-T. and Mizutani, E. (1997), "Neuro-Fuzzy and Soft Computing – A computational approach to learning and machine intelligence", Pearson Education, Singapore, Indian edition, Delhi.
27. Jang, R. (1992), "Neuro-Fuzzy Modeling: Architectures, Analyses and Applications", Ph.D. Thesis, University of California, Berkeley, 1992.

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.