Global Journal of Computer Science and Technology
Neural & Artificial Intelligence
Volume 13 Issue 2 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Bayesian Regularization Based Neural Network Tool for Software Effort Estimation

By Harwinder Kaur & Dalwinder Singh Salaria
Lovely Professional University, India

Abstract - Rapid growth of software industry leads to need of new technologies. Software effort estimation is one of the areas that need more concentration. Exact estimation is always a challenging task. Effort Estimation techniques are broadly classified into algorithmic and non-algorithmic techniques. An algorithmic model provides a mathematical equation for estimation which is based upon the analysis of data gathered from previously developed projects and Non-algorithmic techniques are based on new approaches, such as Soft Computing Techniques. Effective handling of cost is a basic need for any Software Organization. The main tasks for Software development estimation are determining the effort, cost and schedule of developing the project under consideration. Underestimation of project done knowingly just to win contract results into loses and also the poor quality project. So, accurate cost estimation leads to effective control of time and budget during software development. This paper presents the performance analysis of different training algorithms of neural network in effort estimation. For sake of ease, we have developed a tool in MATLAB and at last proved that Bayesian Regularization [20] gives more accurate results than other training algorithms.

Keywords : *effort estimation, levenberg-marquardt (trainlm), back propagation, bayesian regularization (trainbr), gradient descent (traingdx), MATLAB.*

GJCST-D Classification : *I.2.5 , I.2.6*

Strictly as per the compliance and regulations of:

© 2013. Harwinder Kaur & Dalwinder Singh Salaria. This is a research/review paper, distributed under the terms of the Creative Commons Attribution-Noncommercial 3.0 Unported License http://creativecommons.org/licenses/by-nc/3.0/), permitting all non-commercial use, distribution, and reproduction inany medium, provided the original work is properly cited.

Bayesian Regularization Based Neural Network Tool for Software Effort Estimation

Harwinder Kaur [α] & Dalwinder Singh Salaria [σ]

Abstract - Rapid growth of software industry leads to need of new technologies. Software effort estimation is one of the areas that need more concentration. Exact estimation is always a challenging task. Effort Estimation techniques are broadly classified into algorithmic and non-algorithmic techniques. An algorithmic model provides a mathematical equation for estimation which is based upon the analysis of data gathered from previously developed projects and Non-algorithmic techniques are based on new approaches, such as Soft Computing Techniques. Effective handling of cost is a basic need for any Software Organization. The main tasks for Software development estimation are determining the effort, cost and schedule of developing the project under consideration. Underestimation of project done knowingly just to win contract results into loses and also the poor quality project. So, accurate cost estimation leads to effective control of time and budget during software development. This paper presents the performance analysis of different training algorithms of neural network in effort estimation. For sake of ease, we have developed a tool in MATLAB and at last proved that Bayesian Regularization [20] gives more accurate results than other training algorithms.

Keywords : effort estimation, levenberg-marquardt (trainlm), back propagation, bayesian regularization (trainbr), gradient descent (traingdx), MATLAB.

I. Introduction

Software effort estimate is one of the noticeable & mind catching field. But since it was started, it is challenging factor for software industry and Academia to realize the exact estimation of software development. In today's fast changing world, success in managing projects is a critical factor for the success of the entire organization. Estimation that either overestimated or underestimated both is very critical. In case of Overestimating time and effort (or budget), due to a presumed lack of resources or because the projected completion is too late, can convince management not to approve projects that may otherwise contribute to the organization. On the other hand, underestimation may result in approval of projects that will fail to deliver the expected product within the time and budget available. There are many factors that influence the Software estimation, some of them are: uncertainty, level of detail of preparing the project plan, managerial factors, lack of past data, pressure to lower estimation and estimator experience [1]. In spite of the critical role of accuracy, examples of incorrect estimation abound, especially in IT projects, resulting in enormous waste of time and money. Some techniques which were used in the past are not in use during present time, like SLIM [14], checkpoint [2], Seer [2]. In all the way of work time, many of new advance roads have been suggested for effort estimation like Genetic programming [11], Fuzzy logic [10], Neural Network [15], data mining [9], etc.

One cannot state that one model give better accuracy above all. Each and every give different level of accuracy in different Environment. But in recent days, Neural Network gains main attention due to many flavor of algorithm available for it. The main focus of this paper is to investigate the accuracy of estimation using neural network approach based on three different training algorithms: Levenberg-Marquardt (trainlm) [20], Back propagation [20], Bayesian Regularization (trainbr) [20] and this has been done with the help of tool generated by us in MATLAB.

This paper comprises as follow: section II describes the some former effort estimation models and review of related work to Neural Network, section III includes introduction of Neural Network and training algorithms used for this paper, in section IV problem is stated, section V describes methodology used, section VI includes experimental results and comparisons. In last conclusion and future scope is given.

II. Review of Literature

The period of Effort Estimation was started from the expert judgments, which is based on the experiences of experts. But it is only proceed as pillar when current project & pertinent Past projects are similar. Choices of effort estimation techniques footstep from COCOMO [14] to AI approaches [2]. In 1970, Larry Putnam developed the method *SLIM* [14], based on the Rayleigh function and the influence used to Rayleigh curve was Manpower Buildup Index (MBI and Productivity Factor (PF) [2]. Linear programming was key work to drive effort estimation in SLIM [14] and depend upon the source line of code.

In 1981 developer Barry Boehm developed *COCOMO* as constructive cost model [4]. Which is one

Author α : Student, Dept. of CSE Lovely Professional University Phagwara , Punjab (India)-144411.
E-mail : Final destiny 210@gmail.com
Author σ : Assistant Professor, Dept. of CSE Lovely Professional University Phagwara, Punjab (India)-144411.
E-mail : Ds_salaria@yahoo.com

© 2013 Global Journals Inc. (US)

of an easy going & understandable model, could call the effort & time period of project. Due to some problems and some misses found in COCOMO, later on Barry Bohem developed the advance road of this model i.e. COCOMO 2.0 [7]. As growth of software industry rising tremendously and previous version was not up to need.

After that, Howard Rubin proposed the ESTIMAC model to estimate effort at conception stage [4]. Equations used in this model are not available, because it was a proprietary model. ESTIMAC is high level model but doesn't provide accurate solution [3]. Six critical estimation dimensions identified by Rubin for this model are: effort hour, staff size, cost, hardware resource requirement, risk, portfolio impact [2]. But these methods (COCOMO, SLIM, ESTIMAC) are based on Line of code (LOC). The main problems in Line of Code methods are: lack of universally accepted definition for exactly what line code really is? Other side line of code is language dependence.

So, in 1979 at IBM, developer Allan Albrecht developed measurement method called Function point [3] in order to reduce the issues related with LOC methods. Function point defines the complexity of software system in terms of functions that system delivers to user. It comprise linear combination of five basic software components (input, output, master files, interfaces, inquiries) consider to be low, average, high [3]. In 1990, GC Low and DR. Jeffery also concluded in their paper that Function point method is more consistent then the line of code measure [6]. But on the other side, function point method is unable to deal with Uncertain, imprecise and incomplete data.

Many researcher's use different Neural Network with different datasets in order to generate more accurate result for effort estimation. The main advantage of neural network is its ability to handle non-linear data and confidence in decision making. In 1995, Krishna moorthy Srinivasan and Douglas Fisher applied the machine learning approach for Software Effort Estimation [16]. They applied the Back propagation algorithm on COCOMO dataset, along with configuration of 33 neuron of input layer, 10 neurons for hidden layer and 1 output neuron. Actually they had done three experiments on different datasets. They concluded that Back propagation competitive again traditional approaches but quite sensitive.

In one paper written by Ali Idri, et al. [17] in 2002, in which he uses COCOMO-81 dataset and three layered back-propagation ANN, concluded that accuracy provide by back propagation is acceptable.

In 2005, N Tadayon compares the three models COCOMO II, Neural Network and expert judgments to state the strength of different estimation techniques [13]. In 2006, according to Barcelos Tronto et.al Neural Network approach provides better tune result than the linear regression [18]. In his methodology, he used the Back propagation as training algorithm on COCOMO dataset.

In 2010, Iman Attarzadeh, proposed new model of COCOMO II using neural network, and comcluded that neural network approach gives best accuracy than COCOMO II.

Mrinal Kantri, et al. [19] implemented a back-propagation ANN of 3-3-1 architecture on Lopez Martin dataset consist of 41 projects.

Table I : Summary of Datasets and Neural Network used

Author's Name	Year	Dataset	Project	Training Algorithm	ANN Layers	Conference/ Journal
Krishnamoorthy et.al	1995	COCOMO	63	BPA	33-10-1	IEEE
Ali Idri, et al.	2002	COCOMO	63	BPA	13-13-1	IEEE
N Tadayon	2005	-	-	BPA	-	IEEE
Barcelos Tronto et.al	2006	COCOMO	68	BPA	1-9-4-1	IEEE
Attarzadeh	2010	COCOMO, Artifical	100	BPA	24 input neuron	IEEE
Mrinal Kantri	2011	Lopez martin	41	BPA	3-3-1	IEEE

There are many other techniques such as ordinary least square (OLS) [2], Case based reasoning [12], Date mining [9], Bayesian COCOMO II [2], Genetic Programming [5] etc. also used for the effort estimation but not discussed in this paper.

III. Introduction to Neural Network and Training Algorithms

A Neural Network is massively distributed processor made up of simple processing elements called neuron, which model some functionality like human brain [15]. The use of Neural Network offers the

© 2013 Global Journals Inc. (US)

some useful properties and capabilities: - Nonlinearity, Adaptivity, Evidential Response, Confidence in decision made. A primary advantage of learning systems is that they are nonparametric; predictive models can be tailored to, the data at a particular site [8].

Figure 1 : Model of Neural Network

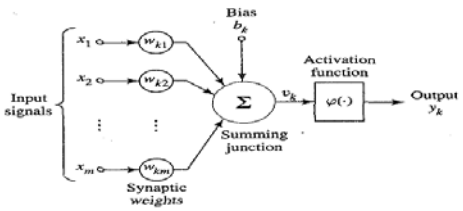

Figure 1 represents the model of neural network. A set of synapses or connecting links is characterized by a weight or strength of its own. A signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}. An adder for summing the input signals. An activation function for limiting the amplitude of output of a neuron.

By adjusting of number of neurons in hidden layer, input layer and choosing of better algorithm for training data and testing phase will give result with high level of accuracy. Training a neural network model refers to choosing one model from the set of permitted models that reduces the cost criterion. Basically, training is of two types Supervised and Unsupervised [15]. In case of Supervised, desired input and output is given to network but in case of unsupervised only desired inputs are given. The network itself makes the decision of output. There are plenty of the training algorithms available in neural network. In this paper three algorithms are used: Leven berg-Marquardt (trainlm) [20], Back propagation algorithm [20], and Bayesian Regularization [20].

Leven berg-Marquardt (trainlm) [20] is a network training function that updates weight and bias values according to Leven berg-Marquardt optimization [20]. This is a simple method for approximating a function. trainlm is highly recommended as a first-choice supervised algorithm. One of the main drawbacks of the Leven berg-Marquardt algorithm is that, for certain problems it needs the large storage of some matrices.

Back propagation [20] learning updates the network weights and biases in the direction in which the performance function decline most quickly, the negative of the gradient [20]. There are too many flavors of Back propagation. For this study, Gradient descent with momentum and adaptive learning rate back propagation (traingdx) is used. The function traingdx combines adaptive learning rate with momentum training. It is a simple method with no specialization needed. But due to low prediction capability, results are not accurate. This has been shown in Experiment section.

One of the problems that occur during above neural network training algorithms is over fitting. Due to this, error in early stage is very small, but, when new data is presented to the network the error is large. The solution to this problem is Bayesian regularization (trainbr) [20]. trainbr updates the weight and bias values according to Levenberg-Marquardt [20] optimization. It minimizes a grouping of squared errors and weights, and generates a network that generalizes well. The process is called Bayesian regularization. It is suitable method for estimation when large number of inputs is used for best output. Till now, Levenberg-Marquardt and Back-propagation algorithm used by many researchers for training phase.

IV. Problem Statement

The main aim of any software development organizations is to finish the project within acceptable or customary schedule and budget. Budget is mainly driven by labor cost and time and together they form a measure called effort. From quality point of view estimating effort is one of the major important factors. Because estimation either it be over estimate or under estimate, produces worst results. In case of over estimation of time and effort project completion is too late due to lack of resources, which refuses the management to approve that favored project. On the other hand, under estimation may result in approval of projects that will fail to deliver the expected product within the time and budget available [1]. So, there is a need of accurate estimation effort technique at early stages of software development. In this research, the main aim is to improve software effort estimation by using different training algorithms of Neural network.

The main reason for using such a learning system for this problem is to keep the estimation process up-to-date by incorporating up-to-date project data. At last Comparison is drawn between training algorithms used in this research to state that Bayesian Regularization gives much accurate estimation. One algorithmic approach, COCOMO is also compared with all three algorithms.

V. Proposed Methodology

Following are the steps used for Effort Estimation:

a) Data Collection

The dataset used in this work is NASA93 (http://promisedata.googlecode.com) a public available data set consisting of a total of 93 projects at the time of this study.

b) Division of Data

Data set is divided into two parts: Training and Testing. For our work we divide the data into 85-15% ratio i.e. 80 rows for training and 13 for testing. These 13 rows are randomly choose by formula (ceil (1+(93-1)*rand(13,1))), available in MATLAB. From this, for

© 2013 Global Journals Inc. (US)

testing row number 15,40,92,74,91,94,5,80,59,64,71,63, 38 are chosen.

c) Cost Drivers

Cost drivers for this work choose from the cost drivers designed for COCOMO II. Table II represents, Cost drivers for COCOMO.

Table II : Cost-drivers of COCOMO model

Attribute	Type	Description
RELY	Product	Required system reliability
CPLX	Product	Complexity of system modules
DOCU	Product	Extent of documentation required
DATA	Product	Size of database used
RUSE	Product	Required percentage of reusable components
TIME	Computer	Execution time constraint
PVOL	Computer	Volatility of development platform
STOR	Computer	Memory constraints
ACAP	Personnel	Capability of project analysts
PCON	Personnel	Personnel continuity
PCAP	Personnel	Programmer capability
PEXP	Personnel	Programmer experience in project domain
AEXP	Personnel	Analyst experience in project domain
LTEX	Personnel	Language and tool experience
TOOL	Project	Use of software tools
SCED	Project	Development schedule compression
SITE	Project	Extent of multisite working and quality of inter-site communications

d) Tool Generation

For the sake of ease, tool is generated with the help of MATLAB. This has been shown in Figure II.

e) Preparation of Neural Network

Depending upon the architecture the Neural Network is of two types: feed-forward and Feed-back. A feed-forward, is the architecture in which the network has no loops. But feed-back neural network is architecture in which loops occurs in the network.

Figure II : Tool developed with the help of MATLAB

© 2013 Global Journals Inc. (US)

For our work, we use feed-forward network with three different training algorithms: LM, BPA, BR. The Neural Network is implemented using 12 neurons for input layer, 12 for hidden layer and 1 for output layer.

f) Performance Criteria

Mean Magnitude Relative Error: MMRE is frequently used to evaluate the performance of any estimation technique. It seems obvious that the purpose of MMRE is to assist us to select the best estimation approach. 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 [18]:

MMRE = {actual effort} - {estimated effort}|}\ {actual effort}

VI. Experimental Results and Comparison

Neural Network trained by three different training algorithms, with same dataset i.e. NASA93. Table III summarizes the result obtained by COCOMO model and three different training algorithms.

Table III : Effort Estimation by using different training algorithms in Neural Network and COCOMO model

Row No.	Expected	COCOMO	LM	BPA	BR
15	48	85.9557	53.7929	1737.61	61.9294
40	114	66.9477	186.747	1702.08	121.206
92	240	85.9557	117.681	1694.9	85.847
74	4178.2	1649.24	1730.38	1843.92	4058.46
91	1772.5	539.26	1400.97	1829.47	2902.12
94	1924.5	393.61	2524.9	1830.02	1201.62
5	25.2	38.2213	260.445	1731.69	83.0016
80	703	904.279	367.178	1836.86	562.929
59	4560	6718.84	1347.35	1945.73	4471.23
64	150	115.445	270.15	1048.19	61.3017
71	72	155.732	85.29	1759.78	106.606
63	160	270.499	294.428	1056.21	61.7749
38	444	463.311	150.555	1706.99	338.233

Figure III : Column chart for effort estimation

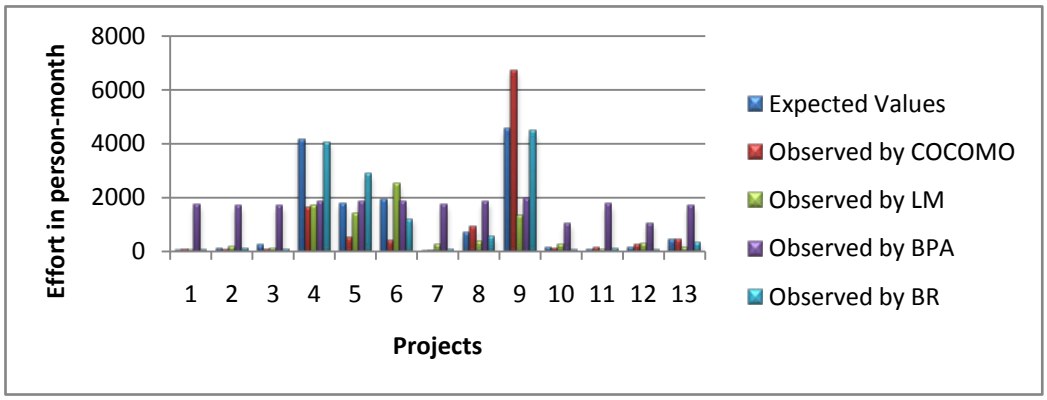

Table IV : Comparison between different training algorithms

Performance Criteria	COCOMO	LM	BPA	BR
MMRE	0.52	1.23	12.18	0.48

In the testing phase the calculated efforts and errors using different training algorithms and COCOMO is shown in table III and table IV respectively. Figure III clearly present Bayesian Regularization is more accurate than others. As evident from the table III, the predicted values of the Bayesian Regularization efforts is very close to the expected or actual values as compare to LM, Back propagation and COCOMO.

VII. Conclusion

Effort Estimation is one of the crucial tasks in software project management. This simulation with NASA93 dataset has been carried out using tool created with the help of MATLAB. Neural Network is trained using “trainlm”, “traingdx” and “trainbr” algorithm. The result from our simulation shows that Bayesian

© 2013 Global Journals Inc. (US)

Regularization gives the best performance, among the other training algorithms. We have experimented with 15 attributes of the COCOMO and further investigation can be done with other attributes and also concentration needed for process maturity.

References Références Referencias

1. Morgenshtern, Ofer, Tzvi Raz, and Dov Dvir. "Factors affecting duration and effort estimation errors in software development projects." Information and Software Technology 49, no. 8 (2007): 827-837.
2. Boehm, Barry, Chris Abts, and Sunita Chulani. "Software development cost estimation approaches—A survey." Annals of Software Engineering 10, no. 1 (2000): 177-205.
3. Matson, Jack E., Bruce E. Barrett, and Joseph M. Mellichamp. "Software development cost estimation using function points." Software Engineering, IEEE Transactions on 20, no. 4 (1994): 275-287.
4. Kemerer, Chris F. "An empirical validation of software cost estimation models."Communications of the ACM 30, no. 5 (1987): 416-429.
5. Burgess, Colin J., and Martin Lefley. "Can genetic programming improve software effort estimation? A comparative evaluation." Information and Software Technology 43, no. 14 (2001): 863-873.
6. Low, Graham C., and D. Ross Jeffery. "Function points in the estimation and evaluation of the software process." Software Engineering, IEEE Transactions on 16, no. 1 (1990): 64-71.
7. Boehm, Barry, Bradford Clark, Ellis Horowitz, Chris Westland, Ray Madachy, and Richard Selby. "Cost models for future software life cycle processes: COCOMO 2.0." Annals of software engineering 1, no. 1 (1995): 57-94.
8. Srinivasan, Krishnamoorthy, and Douglas Fisher. "Machine learning approaches to estimating software development effort." Software Engineering, IEEE Transactions on 21, no. 2 (1995): 126-137.
9. Dejaeger, Karel, Wouter Verbeke, David Martens, and Bart Baesens. "Data mining techniques for software effort estimation: a comparative study." Software Engineering, IEEE Transactions on 38, no. 2 (2012): 375-397.
10. Nisar, M. W., Wang, Y. J., & Elahi, M. (2008, October). Software development effort estimation using fuzzy logic-A survey. In Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on (Vol. 1, pp. 421-427). IEEE.
11. Shan, Y., McKay, R. I., Lokan, C. J., & Essam, D. L. (2002, July). Software project effort estimation using genetic programming. In Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on (Vol. 2, pp. 1108-1112). IEEE.
12. Shepperd, M., Schofield, C., & Kitchenham, B. (1996, May). Effort estimation using analogy. In Proceedings of the 18th international conference on Software engineering (pp. 170-178). IEEE Computer Society.
13. Tadayon, N. (2005, April). Neural network approach for software cost estimation. In Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on (Vol. 2, pp. 815-818). IEEE.
14. Boehm, B.W. Software Engineering Economics. Prentice-Hall, Englewood Cliffs, N.J. 1981.
15. Simon Haykin, "Neural Networks: A Comprehensive Foundation", Second Edition, Prentice Hall, 1998.
16. Srinivasan, K., & Fisher, D. (1995). Machine learning approaches to estimating software development effort. Software Engineering, IEEE Transactions on, 21(2), 126-137.
17. Ali Idri and Taghi M. Khoshgoftaar& Alain Abran,"Can Neural Networks be easily Interpreted in Software Cost Estimation", IEEE Transaction, 2002, page: 1162-1167.
18. I.F. Barcelos Tronto, J.D. Simoes da Silva, N. Sant. Anna, "Comparison of Artificial Neural Network and Regression Models in Software Effort Estimation", INPE ePrint, Vol.1, 2006.
19. Mrinal Kanti Ghose, Roheet Bhatnagar and Vandana Bhattacharjee, "Comparing Some Neural Network Models for Software Development Effort Prediction", IEEE, 2011.
20. http://automatika.etf.bg.ac.rs/files/predmeti/os4nm/Materijali/03_BackPropagation/MATLAB_nnet_BackPropagation.pdf

© 2013 Global Journals Inc. (US)