Volume 4, Issue 12, December 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Comparison and Analysis of Various Artificial Neural Networks for Software Effort Estimation

Poonam Rijwani* CSE Deptt., Pratap University, India

Sonal Jain CSE Deptt., JK Lakshmipat University, India

Dharmesh Santani CE Deptt. Poornima University, India

Abstract— Inspite of the several software effort estimation models developed over the last 30 years, providing accurate estimates of the software project under development is still unachievable goal. Failures in software are mainly due to the erroneous project management practices; one of them is effort estimation. Therefore, many researchers are working on the development of new models and the improvement of the existing ones using artificial intelligence techniques. Dynamic scenarios of software development technology make effort estimation more challenging. Ability of ANN (Artificial Neural Network) to model a complex set of relationship between the dependent variable (effort) and the independent variables (cost drivers) makes it as a budding tool for estimation. This paper presents a performance analysis of different ANNs and compare the results of various ANN models in effort estimation.

Keywords— Effort Estimation, Artificial Neural Network, MMRE, COCOMO, machine learning

I. INTRODUCTION

An important objective of the software engineering community has been to develop useful models that are accurately estimating the software effort. Effort estimation is a process of predicting probable cost and development time to develop a software, process or product. Accurate effort estimation is essential as over estimation may lead to loss of business and under estimation may lead to low quality of software which ultimately leads to software failure. Effort predictions particularly made at an early stage during a project are helpful for project managers. There are lots of existing methods for effort and cost estimation. Continuous changing environment of software development technology make effort estimations more challenging.

The Software effort estimation methods are mainly categorized into algorithmic and non-algorithmic methods. The algorithmic methods are mainly COCOMO [1], Function Points [2] and SLIM [3]. Algorithmic methods are also known as parametric methods as they predict software development effort using a fixed mathematical formula that is parameterized from historical data. However, estimates at the preliminary stages of the project are difficult to obtain because the primary source to estimate the effort comes from the SRS document. Also, they have difficulty in modeling the inherent complex relationships [4]. The limitations of algorithmic methods induce to look towards non-algorithmic methods which are soft computing based. These methods have ability to learn from previous data and are able to model complex relationship between the dependent (effort) and independent variables.

In this paper we have analyzed performance of various artificial neural network models embedded in the COCOMO II to overcome the imprecision and ambiguity of software attributes which results in producing better estimation results.

II. EFFORT ESTIMATION METHODS

Literature Survey reveals that the authors have applied various computational intelligence methods on COCOMO.

COCOMO II

The COCOMO II method was developed using COCOMO-81 model. The model was developed by analyzing the changes in software engineering over the past 20 years reflecting these changes.

COCOMO II provides two main models:

A. Early Design Model

B. Post-architecture Model

A. Early Design Model

In the requirements analysis phase of a software project's lifecycle, the stakeholders must agree upon the requirements. The Early Design Model gets into action when these requirements get agreed upon. The estimated effort required for the software project is calculated using the formula:

$$\text{Effort}_{NS} = A \times (\text{Size})^{E} \times M \qquad (1)$$

Where:

A: Constant (based on the calibration of local conditions and past data of the firm).

Size: Size of the software (expressed in KLOCs).

E: Constant
M: Constant (based on the attributes/cost-drivers of the project.
Effort $_{NS}$: Estimated effort (expressed in units of PMs).

The amount of the development time is calculated using the formula:

$$\text{Time}_{dev} = C \times (\text{Effort})^{F} \quad (2)$$

Where:
C: Constant (based on the calibration of local conditions and past data of the firm).
F: Constant
Effort: Previously calculated estimated effort (expressed in units of PMs).
Time $_{dev}$: Estimated development time (expressed in months).

The constant E used in the calculation of the estimated effort and the constant F used in the calculation of the estimated development time are values that must be derived using formulas.

Differently from the COCOMO-81, in the COCOMO II, the software projects are not categorized as organic, semi-detached or embedded types. Instead, there are *Scaling Factors* (SFs), which are used in determining the constants E and F.

In COCOMO II, there are five different SFs that are listed below. Each of these factors is rated in 6 levels ranging between "very low" to "extra high".

Table 1: five different scaling factors

SFs	Very Low	Low	Nominal	High	Very High	Extra High
PREC (SF_1)	6.20	4.96	3.72	2.48	1.24	0.00
FLEX (SF_2)	5.07	4.05	3.04	2.03	1.01	0.00
RESL (SF_3)	7.07	5.65	4.24	2.83	1.41	0.00
TEAM (SF_4)	5.48	4.38	3.29	2.19	1.10	0.00
PMAT (SF_5)	7.80	6.24	4.68	3.12	1.56	0.00

The values of these weights are calculated based on statistics of a large number of software projects.
The constants E and F are calculated using the formulas that are shown below:

$$E = B + .01 * \sum_{1}^{5} SFI \quad (3)$$

Where:
B: Constant (varies from 1.1 to 1.24 with respect to the novelty of the project, development flexibility, risk management methods and the process maturity).
SF_i: i^{th} scale factor weight.
E: Constant

$$F = D + 0.2 \times (E - B) \quad (4)$$

Where:
D: Constant (based on the calibration of local conditions and past data of the firm and is taken as 0.28 in the initial calibration).
E: Constant calculated using the previous formula.
B: Constant (varies from 1.1 to 1.24 with respect to the novelty of the project, development flexibility, risk management methods and the process maturity).
F: Constant (used in calculation of the amount of the development time).

The constant M that is used in calculating the estimated effort of a software project must be determined.

$$M = (PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED) \quad (5)$$

Where:
PERS, RCPX, RUSE, PDIF, PREX, FCIL AND SCED: Constant values of the properties of the Early Design Model.

B. Post-architecture Model

In the first phases of a software project's lifecycle, the architecture of the whole lifecycle is developed that defines various aspects about the project in details. The Post architecture Model was introduced to be used after the project lifecycle architecture is developed.

In the Post-architecture Model, the estimated effort for the development of the software project is calculated with the same formula that is used in the Early Design Model. However, the Post-architecture Model uses 17 properties for this calculation instead of 7 properties used by the Early Design Model.

Product Attributes

1. RELY (Required system Reliability)
2. CPLX (Complexity of the System)
3. DOCU(Documentation Required)
4. DATA(Size of database)
5. RUSE (Requirement for reuse)

Computer Attributes

6. TIME (Execution Time constraint)
7. PVOL(Development Platform Volatility)
8. STOR (Memory Constraints)

HR Attributes

9. ACAP (capability of project analyst)
10. PCON (Personnel continuity)
11. PCAP (Programmer capability)
12. PEXP(programmer domain Experience)
13. AEXP (Analyst domain Experience)
14. LTEX (Language and tool experience)

Project Attributes

15. TOOL (Use of Software tools)
16. SCED (Schedule compression)
17. SITE (Extent of multisite working)

Each of these properties has a value associated with it called *Effort Multiplier* (EM).
In this model, the constant M is calculated using the following formula:

$$M = (RELY \times CPLX \times DOCU \times DATA \times RUSE \times TIME \times PVOL \times STOR \times ACAP \times PCON \times PGAP \times PEXP \times AEXP \times LTEX \times TOOL \times SCED \times SITE) \quad (6)$$

COCOMO's well definition offers various advantages:

- It is easy to adapt.
- It is very understandable.
- It provides more objective and repeatable estimations
- It creates the possibility of calibrating the model to reflect any type of software development environment and thus, providing more accurate estimates.

III. MACHINE LEARNING AND EFFORT ESTIMATION

This section illustrates the work of various researchers who have applied ANN on software estimation model.

3.1 Artificial Neural Network

An artificial neural network (ANN) is a network composed of artificial neurons or nodes which imitate the biological neurons [5]. ANN can be trained to be used to approximate a non-linear function, to map an input to an output or to classify outputs. There are several algorithms available to train a neural network but this depends on the type and topology of the neural network.

Each neuron is connected with the other by a connection link. Each connection link is associated with weights which contain information about the input signal. This information is used by the neuron net to solve a particular problem. Each neuron has an internal state of its own. This internal state is called the activation level of neuron, which is the function of the inputs the neuron receives.

The most prominent topology of ANN is the feed-forward networks. In a feed-forward network, the information always flows in one direction (from input to output) and never goes backwards. An ANN is composed of nodes organized into layers and connected through weight elements. At each node, the weighted inputs are aggregated, thresholded and inputted to an activation function to generate an output of that node.

There are a number of activation functions that can be applied over net input such as Gaussian, Linear, Sigmoid and Tanh. Figure 1 shows the structure of a basic neural network.

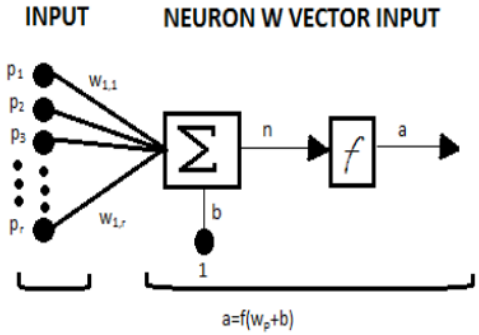

Fig1: Basic Neural Network

The structure of the neural networks help to solve the practical, non linear, decision making problems easily.

3.2 Related work

Accurate and consistent prediction of resource requirements is a crucial component in the effective management of software projects. Despite the amount of researches over the last 20 years, the software community is still significantly challenged when it comes to effective resource prediction [6].

As a main stream, research efforts have focused on the development of quantitatively based techniques, in an effort to remove or reduce subjectivity in the estimation process. Examples of this work include the original parametric and regression-based models: Function Points Analysis (Albrecht, 1979), COCOMO Models (Boehm, 1981; Boehm et al., 2000), and the Ordinal Regression Model (Sentas et al., 2005).

However, other techniques for exploratory data analysis, such as clustering, case-based reasoning and ANN have been effective as means of predicting software project effort. Zhong et al. (2004) describe the use of clustering to predict software quality. A case-based approach called ESTOR was developed for software effort estimation (Vicinanza et al., 1990). They have shown that ESTOR is comparable to a specialist and performs significantly better than COCOMO and Function Points on restricted samples of problems. Some research works have used artificial neural networks to produce more accurate resource estimates Gray and MacDonell, 1997; Witting and Finnie, 1997). In Karunanitthi et al. (1992), neural networks are used to predict software reliability. They conducted experiments with both feed-forward and Jordan networks and the cascade correlation learning algorithm. Witting and Finnie (1994) describe their use of the back propagation learning algorithm on a multilayer perceptron to predict development effort. The work of Samson et al. (1997) uses an Albus multiplayer perceptron in order to predict software effort on the Boehm's COCOMO dataset and compared linear regression with a neural networks approach.

Srinivazan and Fisher (1995) also reported the use of a neural network with back propagation learning algorithm and found that the neural network outperformed other techniques and led to results of MMRE = 70%. However, it is vague how the dataset was divided for the training and the validation purposes.

Khoshgoftaar et al. (2000) presented a case study considering real time software to predict the testability of each module from source code static measures. They consider ANNs as promising techniques to build predictive models, because they are capable of modeling nonlinear relationships.

The interest on the use of ANNs has grown in past years. ANNs have been successfully applied to several problem domains, in areas such as medicine, engineering, geology, and physics, generally to design solutions for estimate, classification, control problems, etc. They can be used as predictive models because they are modeling techniques capable of modeling complex functions. Machine learning algorithms such as Artificial Neural Networks offer a means of addressing the problem of many factors. They are effective when we have a relatively large number of data points as well as a large number of factors.

3.3 A COMPARISON BASED ON SURVEY OF ANN BASED EFFORT PREDICTION TECHNIQUES

We have identified 40 studies in the field of ANN based effort estimation. These papers were published during the time period 2000 to 2013. Based on literature survey, a comparative analysis has been done as shown in table 2. After computing the effort by various techniques, evaluation criteria Magnitude-Relative-Error (MRE) is used to compare the results obtained from various methods which is shown in table 3. The results are then also shown using line chart as given in figure2.

Table 2: effort computation using various techniques

Project Id	**Actual Effort as per the Dataset**	**Effort Computed**					
		Effort Computed Using COCOMO	**Effort Computed Using SLANN With BP**	**Effort Computed Using SLANN With**	**Effort Computed Using RBNN**	**Effort Computed Using GRNN**	**Effort computed MLFFN With BP**

				RPROP			
1	2040	1616.38	2782.96	2337.41	2048	2031	2031.84
3	243	233.88	502.6	157.22	240	192	239.039
11	218	189.93	229.98	257.03	200	198	193.802
18	11400	8552.88	8892.4	10596.3	11552	10900	11229
20	6400	3603.34	5071.95	4065.97	5827	5800	5811.2
26	387	279.93	584.49	363.66	402	350	371.52
27	88	59.002	113.79	84.92	101	80	74.976
50	176	132.163	182.97	131.98	179	151	172.515
51	122	114	97.6	60.73	124	115	119.926
54	20	6.24	23.05	12	28	14	12
55	18	7.5	12.39	5.33	19	16	16.902
56	958	537	341.41	346.68	960	954	953.21
60	57	23.91	46.17	44.04	68	50	48.2562

Table 3: performance of various techniques using mre

Project Id	**MRE (%)**					
	MRE (%) using COCOMO	**MRE (%) using SLANN With BP**	**MRE (%) using SLANN With RPROP**	**MRE (%) using RBNN**	**MRE (%) using GRNN**	**MRE (%) using MLFFN With BP**
1	20.76	36.41	14.57	0.39	0.441176	0.4
3	3.75	106.83	35.3	1.23	20.98765	1.63
11	12.87	5.49	17.9	8.26	9.174312	11.1
18	24.97	21.99	7.05	1.33	4.385965	1.5
20	43.69	20.75	36.46	8.95	9.375	9.2
26	27.66	51.03	6.03	3.88	9.560724	4
27	32.95	29.3	3.5	14.77	9.090909	14.8
50	24.9	3.96	25.01	1.70	14.20455	1.98
51	72.56	20	50.22	1.64	5.737705	1.7
54	68.78	15.25	40	40.00	30	40
55	73.04	31.11	70.38	5.56	11.11111	6.1
56	77.48	64.36	63.81	0.21	0.417537	0.5
60	58.05	19	22.73	19.30	12.2807	15.34

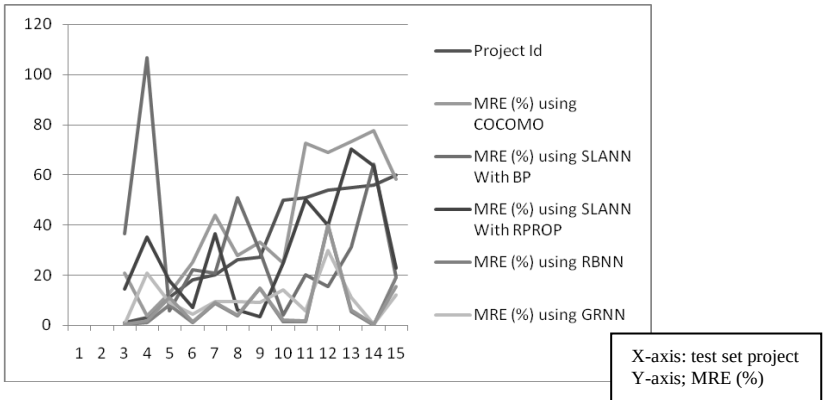

Fig 2: Performance analysis using MRE (%)

IV. CONCLUSION

Estimation is one of the crucial and challenging tasks in software project management. Researchers have developed different models for accurate effort estimation and hence a comparison of various ANN techniques with COCOMO dataset has been carried out. Choosing the right ANN method is important to get accurate estimation. Referring to the chart in fig 2, the results shows that Radial Basis neural network give the best performance, among the various techniques of ANN. Multi layer feed forward ANN gives next best results. The future work is the need to investigate impact of each cost driver and scale factors and accordingly finding the ideal parameter settings for these ANN networks that can help to improve the process of software cost estimation and easy to use.

REFERENCES

[1] Boehm, B.W., "Software Engineering Economics", IEEE Transactions on software Engineering, SE-10, 1, pp. 4-21, January 1984.

[2] Sheta, A., Rine, D., & Ayesh, A. (2008, June). Development of software effort and schedule estimation models using soft computing techniques. In*Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on* (pp. 1283-1289). IEEE.

[3] Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE transactions on Software Engineering*, *4*(4), 345-361.

[4] Clark, B., Chulani, S. and Boehm, B. (1998), "Calibrating the COCOMO II Post Architecture Model," International Conference on Software Engineering, Apr.1998.

[5] Lippmann, Richard P. "An introduction to computing with neural nets." ASSP Magazine, IEEE 4.2 (1987): 4-22.

[6] Molokken K.; Jorgensen M., 2003. A review of software surveys on software effort estimation, Proceedings of IEEE International Symposium on Empirical Software Engineering ISESE, 223-230.

[7] Simon Haykin, "Neural Networks: A Comprehensive Foundation", Second Edition, Prentice Hall, 1998.

[8] Ali Idri and Taghi M. Khoshgoftaar& Alain Abran,"Can Neural Networks be easily Interpreted in Software Cost Estimation", IEEE Transaction, 2002, page:1162-1167.

[9] Attarzadeh and Siew Hock Ow, "Software Development Effort Estimation Based on a New Fuzzy Logic Model," International Journal of Computer Theory and Engineering, Vol. 1, No. 4, pp.1793-8201, October 2009.

[10] G.E. Wittig and G.R Finnic, "Using Artificial Neural Networks and Function Points to estimate 4GL software development effort", AJIS,1994, page:87-94.

[11] 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.

[12] JaswinderKaur, Satwinder Singh, Dr. Karanjeet Singh Kahlon, 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.

[13] 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.

[14] K.K. Aggarwal, Yogesh Singh, Pravin Chandra and Manimala Puri, "Evaluation of various training algorithms in a neural network model for software engineering applications" , ACM SIGSOFT Software Engineering , July 2005, Volume 30 Number 4 , page: 1-4.

[15] Mrinal Kanti Ghose, Roheet Bhatnagar and Vandana Bhattacharjee , "Comparing Some Neural Network Models for Software Development Effort Prediction" , IEEE, 2011.

[16] Satish Kumar, "Neural Networks: A Classroom Approach", Tata McGraw-Hill, 2004.

[17] Howard Demuth and Mark Beale, "Neural Network Toolbox-For use with MATLAB",User's Guide,Version-4,Page-5.28.

[18] N.K.Bose and P.Liang, "Neural Network Fundamentals with Graphs, Algorithms and Applications", Tata McGraw Hill Edition,1998.

[19] B. Yegnanarayana, "Artificial Neural Networks", Prentice Hall of India, 2003.

[20] K. Vinay Kumar, V. Ravi and Mahil Carr, "Software Cost Estimation using Soft Computing Approaches," Handbook on Machine Learning Applications and Trends: Algorithms, Methods and Techniques, Eds. E. Soria, J.D. Martin, R. Magdalena, M.Martinez, A.J. Serrano, IGI Global, USA, 2009.

[21] Idri, A., Khoshgoftaar, T. M. and Abran, A. 2002. Can neural networks be easily interpreted in software cost estimation, Proceedings of the IEEE International Conference on Fuzzy Systems, 1162-1167.

[22] www.mathworks.com

[23] Rao, B. S. and Sarda, N. L. 2003. Effort drivers in maintenance outsourcing - an experiment using Taguchi's methodology, Proceedings of Seventh IEEE European Conference on Software Maintenance and Reengineering, 1- 10. 112.

[24] Reddy C.S.; Raju KVSN, (2009). An Improved Fuzzy Approach for COCOMO's Effort Estimation using Gaussian Membership Function. Journal of software 4(5), 452-459.

[25] Venkatachalam A.R., (1993). Software Cost Estimation using artificial neural networks. Proceedings of the International Joint Conference on Neural Networks, 987-990.

[26] Sivanandam S.N.; Deepa S.N., (2007). Principles of Soft Computing, Wiley, India.

[27] 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.

[28] Huang S.; Chiu N., (2009). Applying fuzzy neural network to estimate software development effort, Proceedings of Applied Intelligence Journal 30(2), 73-83.

[29] Idri A.; Zakrani A.; Zahi A., (2010), Design of radial basis function neural networks for software effort estimation, IJCSI International Journal of Computer Science 7(4), 11-17.

[30] Idri A.; Zahi A.; Mendes E.; Zakrani A., (2007), Software Cost Estimation Models using Radial Basis Function Neural Networks, International Conference on Software process and product measurements, 21-31.

[31] Prasad Reddy P.V.G.D; Sudha K.R; Rama Sree P; Ramesh S.N.S.V.S.C, (2010) Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks, Journal of computing 2(5), 87-92.

[32] Nasser Tadayon, "Neural Network Approach for Software Cost Estimation", IEEE proceedings of the International Conference on Information Technology: Coding and Computing(ITCC '2005)

[33] 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)

