Neuro-Fuzzy Technique for Software Effort Estimation

Sheikh Fahad Ahmad[1] Anwar Ahmed Sheikh[2] Mohd. Haleem[3] Roshan Jahan[4] Mohd. Atif Kaleem[5]
[1,2,3,4]Assistant Professor [5]Instructor
[1,2,4,5]Department of Computer Science Engineering [3]Department of Computer Application
[1,2,3,4,5]Integral University, India

Abstract— One of the most important tasks of a software development life cycle is to estimate software time and effort. Accuracy in effort estimation is still a great challenge in the software industry. Conventional models like the Constructive Cost Model (COCOMO) are quite helpful but they are unable to manage the unsure nature of the software projects particularly within the early part of the development criterion. There has been considerable work exhausted in this area using neural networks and conjointly some work done using fuzzy logic. These models offer additional exactitude however they're heavily obsessed on the scale of the training set. The aim of this paper is to develop and evaluate a neuro-fuzzy model to estimate software development effort. Our proposed work is compared with neural network approach and fuzzy logic approach based on the value of MMRE (Mean of Magnitude of Relative Error) .The projected Neuro-fuzzy model was found to have lower MMRE value.

Key words: COCOMO, MMRE, neuro-fuzzy

I. INTRODUCTION

There is large number of existing estimation techniques but there is no general technique which is applicable in all the cases. In order to choose the correct technique we have to make a detailed study of the models available and then compare their results to make an accurate estimation [1][21]. Hence our objective would be to combine the neural network approach and the fuzzy logic approach to produce neuro-fuzzy approach which is more precise and more accurate. Neural Network approach requires a large amount of training data without which our estimated cost can never be accurate. Fuzzy Logic can help us make decisions where there is vagueness in the input data. Hence if we combine these two techniques we can use the merits of both [2]. We used MATLAB to evaluate these neural, fuzzy logic and neuro-fuzzy systems.

II. RELATED WORK

To solve software estimation problems, soft computing framework is based on the "divide and conquer" approach. There is availability of subject material in Haykin S, Zadeh, Fuzzy Logic. Ren J, Huang X, Capretz L have given "A Soft Computing Framework for Software Effort Estimation". Ali Idri has published a paper on "Can Neural nets are easily interpreted in Software Cost Estimation". Many software estimation models [20] have been developed [3], [4], [5], [6]. Ting su et al. [4] described a fuzzy logic model for the estimation of software development effort which had the similar capabilities as the previous fuzzy logic model in addition to enhancements in empirical accuracy in terms of MMRE. Xishi Huang et al. [6] also developed neuro-fuzzy Constructive Cost Model (COCOMO) for software cost estimation which uses the desirable features of a neuro-fuzzy approach, such as learning ability and good interpretability, in COCOMO model.

III. FUZZY LOGIC APPROACH

Since fuzzy logic was proposed by Zadeh in 1965, it has found a lot of applications. Some of the major modules are: A stage that transforms the classification tables into a continuous classification, by a process called Fuzzification [10]. Then processing takes place by inference engine in fuzzy domain based on knowledge base (rule base and data base) which is supplied by domain experts [11]. Then process of translating back fuzzy numbers into single "real world" values is called Defuzzification [10]. Here, the development time of forty one modules and for each module, coupling (Dhama), complexity (McCabe), and lines of code metrics were registered. The development time of the forty one modules were registered for five phases: requirements understanding, algorithm design, coding, compiling and testing. The statistics and a brief description related to each module are shown in Table I which is prepared by Lopez-Martin et al. [18].

IV. NEURAL NETWORK MODEL

In recent years, neural networks have been used in various studies in various stages of software development [10]. Due to its ability to learn from previous data artificial neural network are used in estimation. It also has the ability to generalize from the training data set and thus can produce good result for previously unseen data. Using Artificial neural networks complex non-linear relationships can be modelled and we can approximate any measurable function so it is very useful in problems where there is a complex relationship between inputs and outputs [11], [6]. There are many techniques for training a neural network. The main techniques employed by neural networks are supervised and unsupervised learning. Supervised training is same as a human learns new skills, by showing the network a series of examples. Dataset is randomly divided into two parts: some of them are for training and all of them are used for validation. RBF network was created by MATLAB 7.4, data were normalized between 0 and 1, and test data were applied into network. The results of this implementation are gathered in Table II.

	Module Description	MC	DC	LOC	DT in
1	Calculates t value	1	0.25	4	13
2	Inserts a new	1	0.25	10	13
3	Calculates a value	1	0.333	4	9
4	Calculates the	2	0.083	10	15
5	Generates range	2	0.111	23	15
6	Determines both	2	0.125	9	15
7	Turns each linked	2	0.125	9	16

8	Copies a list of	2	0.125	14	16
9	Determines parity of	2	0.167	7	16
10	Defines segment	2	0.167	8	18
11	From two lists (X	2	0.167	10	15
12	Calculates a sum	2	0.167	10	15
13	Calculates q values	2	0.167	10	18
14	Generates the sum of	2	0.2	10	13
15	Calculates the sum of	2	0.2	10	14
16	Calculates the	2	0.2	10	15
17	Counts the number	2	0.2	15	13
18	Prints values non	2	0.25	10	12
19	Stores values into a	2	0.25	10	12
20	Generates range	3	0.083	17	22
21	Returns the number	3	0.125	11	19
22	Calculates the sum of	3	0.125	15	18
23	Calculates the sum of	3	0.125	15	19
24	Generates the	3	0.143	13	21
25	Returns the sum of	3	0.143	14	20
26	Prints a matrix	3	0.143	14	21
27	Calculates the sum of	3	0.143	15	19
28	Calculates the sum of	3	0.143	15	20
29	Calculates the	3	0.167	13	15
30	Returns the sum of a	3	0.167	14	13
31	Generates the	3	0.2	18	19
32	Prints a linked list	3	0.25	9	13
33	Calculates gamma	3	0.25	12	12
34	Calculates the	3	0.25	17	12
35	Calculates the range	4	0.077	16	21
36	Calculates beta 1	4	0.077	31	21
37	Returns the product	4	0.111	16	19
38	Counts commented	4	0.2	24	18
39	Reduces final matrix	5	0.143	22	24
40	Reduces a matrix	5	0.143	22	25
41	Counts blank lines	5	0.2	22	18

Table 1: Mc: Mccabe Complexity, Dc: Dhama Coupling, Loc: Lines Of Code, Dt: Development Time (Minutes)

V. NEURO FUZZY SYSTEM

Neural networks and fuzzy logic combination is the basic idea behind the neuro-fuzzy system. Neuro-fuzzy combination is done in two ways [12]: fuzzy neural networks (FNN) and neuro-fuzzy systems (NFS). FNN is a neural network has the capability of handling fuzzy information. NFS is a fuzzy system together with neural networks to enhance some characteristics like flexibility and adaptability [13], [14], [15]. This paper uses the second approach.

There are 3 major components in the intelligent model:

Neuro-Fuzzy Inference System (NFIS): It handles the dependencies among cost drivers. The inputs are the ratings of the cost drivers, and its output is the adjusted ratings of cost drivers.

Neuro-fuzzy subsystem (NFi): The input is the adjusted rating of the ith cost driver; the output is the multiplier value of the ith cost driver that is used for input of the COCOMO.

There are 22 cost drivers in our model. Each cost driver represents one factor of the development effort, such as product complexity, applications experience etc. To evaluate the contribution, we use six qualitative rating levels. When expressed in linguistic terms, these six rating levels are Very Low (VL), Low (L), Nominal (N), High (H), Very High (VH) and Extra High (XH). Each rating level of every cost driver relates to a quantitative value used in the COCOMO model.

Sub-model NFi is used to translate this rating of a cost driver into a quantitative multiplier value and to calibrate these relations by industry project data. In this paper, we define a fuzzy set for each linguistic term of every cost driver, i.e. “very low”, “low”, “nominal”, “high”, “very high”, “extra high”. We let the membership functions be triangular functions or other functions, and choose the universe of discourse to be the interval [1,6]. Roughly speaking, we use fuzzy numbers “about 1”, “about 2”,…, “about 6” to represent linguistic terms “very low”, “low”, “nominal”, “high”, “very high”, “extra high”, respectively.

- COCOMO II Model: $effort = A \times (Size)^{B+\sum_{i=1}^{5} SF_i} \times \prod_{i=1}^{17} EM_i$

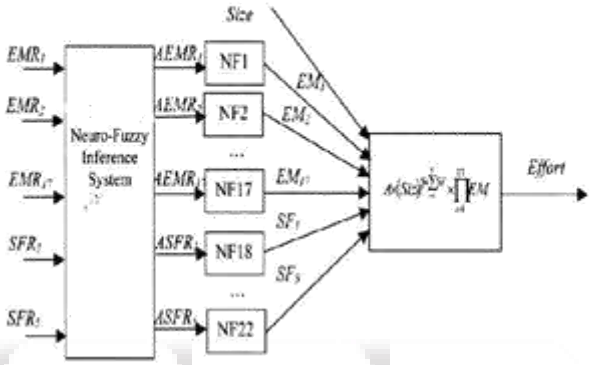

Fig. 1: Effort estimation in COCOMO II Model

There are 22 cost drivers in our model. Each cost driver represents one factor of the development effort, such as product complexity, applications experience etc. To evaluate the contribution, we use six qualitative rating levels. When expressed in linguistic terms, these six rating levels are Very Low (VL), Low (L), Nominal (N), High (H), Very High (VH) and Extra High (XH). Each rating level of every cost driver relates to a quantitative value used in the COCOMO model.

Sub-model NFi is used to translate this rating of a cost driver into a quantitative multiplier value and to calibrate these relations by industry project data. In this paper, we define a fuzzy set for each linguistic term of every cost driver, i.e. “very low”, “low”, “nominal”, “high”, “very high”, “extra high”. We let the membership functions be triangular functions or other functions, and choose the universe of discourse to be the interval [1,6]. Roughly speaking, we use fuzzy numbers “about 1”, “about 2”,…, “about 6” to represent linguistic terms “very low”, “low”, “nominal”, “high”, “very high”, “extra high”, respectively.

ANFIS was used here for implement neuro-fuzzy model and its’ architecture is very similar to Fig. 1.It is the first integrated hybrid neuro-fuzzy model is ANFIS, it has lowest Root Mean Square Error (RMSE) among the other Neuro-Fuzzy models. In ANFIS, the adaptation (learning) process is only concerned with parameter level adaptation within fixed structures [16]. The objective of the parameter learning phase is to adjust parameters of the fuzzy inference system (FIS) such that the error function during training dataset, reaches minimum or is less than a given threshold [17]. When Gaussian membership [22] functions were used, operationally ANFIS can be compared with a radial basis

function network. Our model was just trained at 20 epochs, also the previous training and testing data were used. Here, the development time of forty one modules and for each module, coupling (Dhama), complexity (McCabe), and lines of code metrics were registered, all programs were written in Pascal, hence, module categories belong to Estimating Development Time of Software Projects Using a Neuro Fuzzy Approach Venus Marza, Amin Seyyedi, and Luiz Fernando Capretz M procedures or functions. The development time of each of the forty-one modules were registered including five phases: requirements understanding, algorithm design, coding, compiling and testing. The statistics and a brief description related to each module are depicted in Table I which is prepared by Lopez-Martin et al. [18]. By MATLAB, the ANFIS structure with type: 'sugeno', and method: 'prod', or method: 'max', impMethod: 'prod' and aggMethod: 'max' was implemented and the results are given at Table II.

The Validation results of our experiments are assessed by Mean Magnitude Relative Error (MMRE) as estimation accuracy. MMRE is defined as [19]:

$$MMRE = \frac{1}{n}\sum_{i=1}^{i=n}\left(\frac{|T_i - \overline{T}_i|}{T_i}\right) = \frac{1}{n}\sum_{i=1}^{i=n} MRE_i$$

Module	Annual DT	Fuzzy Logic	Neural Network	Neuro-Fuzzy
		MRE	MRE	MRE
1	13	0.0000	0.1010	0.0000
2	13	0.0000	0.0000	0.0000
3	9	0.0167	0.0764	0.0000
4	15	0.1200	0.0616	0.0000
5	15	0.1867	0.0112	0.0714
6	15	0.0867	0.1118	0.0000
7	16	0.0188	0.1098	0.0000
8	16	0.0813	0.1363	0.1667
9	16	0.0250	0.2425	0.2222
10	18	0.1389	0.4339	0.2500
11	15	0.0400	0.0440	0.1667
12	15	0.0400	0.2247	0.0000
13	18	0.1333	0.0475	0.0000
14	13	0.0769	0.0096	0.0000
15	14	0.0000	0.1896	0.0000
16	15	0.0667	0.1037	0.0500
17	13	0.1615	0.1008	0.0455
18	12	0.0000	0.0454	0.0500
19	12	0.0000	0.3910	0.0000
20	22	0.2000	0.2221	0.0000
21	19	0.0737	0.1310	0.0000
22	18	0.0222	0.0374	0.0000
23	19	0.0737	0.1268	0.0000
24	21	0.1762	0.0442	0.0333
25	20	0.1350	0.0760	0.0000
26	21	0.1762	0.6917	0.0000
27	19	0.0947	0.1536	0.0833
28	20	0.1350	0.2796	0.0001
29	15	0.1133	0.1363	0.1667
30	13	0.2816	0.1471	0.0000
31	19	0.2000	0.0475	0.0000
32	13	0.0000	0.2263	0.0556
33	12	0.0833	0.2279	0.0000
34	12	0.0833	0.1758	0.0417
35	21	0.1905	0.0497	0.0455
36	21	0.1048	0.2159	0.0000
37	19	0.0947	0.5766	0.0000
38	18	0.1556	0.0879	0.0000
39	24	0.2792	0.8469	0.0000
40	25	0.3080	0.2495	0.0000
41	18	0.1556	0.1039	0.0313
MMRE		0.1057	0.202305	0.036098

Table 2: The Mre And Mmre Comparison Between Estimation Models

VI. Conclusion and Future Research

The paper suggests a new approach for estimating of software projects development time. The major difference between our work and previous works is that neuro-fuzzy technique is used for software development time estimation and then it's validated with gathered data. Advantages of neural network and fuzzy logic are combined and learning ability and good generalization are obtained. The main benefit of this model is its good interpretability by using the fuzzy rules and another great advantage of this research is that it can put together expert knowledge (fuzzy rules) project data and the learning ability of neural network model into one general framework that may have a wide range of applicability in software estimation. The results showed that neuro-fuzzy system is much better than two other mentioned methods (fuzzy logic and neural network).

In order to achieve more accurate estimation, voting the estimated values of several techniques and combine their results maybe be useful.

VII. Acknowledgment

First and foremost we would thank Almighty Allah, without His help this work would never have been completed. We wish to express our most sincere and profound gratitude to our mentor Prof. (Dr.) Mohd. Rizwan Beg. Last but not the least, we thank our family and friends for their support, understanding and help, without them this work would not have been possible.

References

[1] C. Lopez-Martin, C.Yanez-Marquez, A.Gutierrez-Tornes, "Predictive accuracy comparison of fuzzy models for software development effort of small programs, The journal of sys-tems and software", Vol. 81, Issue 6, 2008, pp. 949-960

[2] W. Xia, L.F. Capretz, D. Ho, F.Ahmed, "A new calibration for function point complexity weights", Information and Software Technology, Vol.50, Issue 7-8, 2007, pp.670-683.

[3] C.L. Martin, J.L. Pasquier, M.C. Yanez, T.A. Gutierrez, "Software Development Effort Estimation Using Fuzzy Logic: A Case Study", IEEE Proceedings of the Sixth Mexican Inter-national Conference on Computer Science (ENC'05), 2005, pp. 113-120.

[4] M.T. Su, T.C.Ling, K.K.Phang, C.S.Liew, P.Y.Man, "Enhanced Software Development Effort and Cost Estimation Using Fuzzy Logic Model", Malaysian Journal of Computer Science, Vol. 20

[5] A. Heiat, "Comparison of neural network and regression models for estimating software development effort", Information and Technology, Vol. 44, Issue 15, 2002, pp. 911-922
[6] X. Huang, Danny Ho, J. Ren, L.F. Capretz, "Improving the COCOMO model using a neuro-fuzzy approach", Applied Soft Computing , Vol.7, Issue 1, 2007, pp. 29-40.
[7] A. Idri, A.Abran, "A Fuzzy Logic Based Set of Measures for Software Project Similarity: Validation and Possible Improvements", Proceedings of the seventh international software metrics symposium (METRICS '01), 2001, pp.85-96.
[8] S.N. Sivanandam, S. Sumathi, S.N. Deepa, "Introduction to fuzzy logic using MATLAB", Springer, 2007.
[9] A. Lotfi Zadeh, "From Computing with Numbers to Computing with Words – From Manipulation of Measurements to Manipulation of Perceptions", IEEE Transactions on Circuits and Systems, Fundamental Theory and Applications, Vol. 45, No 1, 1999, pp 105-119.
[10] M.R.Braz & S.R.Vergilio, "Using Fuzzy Theory for Effort Estimation of Object-Oriented Software", Proceedings of the 16 th IEEE international Conference on Tools with Artificial Intelligence (ICTAI 2004), 2004.
[11] K.K.Aggarwal, Y.Singh, P.Chandra, M.Puri, "Sensitivity Analysis of Fuzzy and Neural Network Models", ACM SIGSOFT Software Engineering Notes, Vol. 30, Issue 4, 2005.
[12] S. Mitra, Y.Hayashi, "Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework", IEEE Transactions on Neural Networks, Vol.11, No.3, 2000, pp. 748-768.
[13] D. Nauck, F. Klawonn, R. Kruse, "Foundations of Neuro-Fuzzy Systems", Chichester, 97.
[14] D. Nauck, "A Fuzzy Perceptron as a Generic Model for Neuro-Fuzzy Approaches", In Proceedings of Fuzzy-Systeme'94, 2nd GI-Workshop, Munich, Semen Corporation, 1994.
[15] M.O. Saliu, "Adaptive Fuzzy Logic Based Framework for Software Development Ef-fort Prediction", A Thesis Presented to the DEANSHIP OF GRADUATE STUDIES, King Fahd University of Petroleum & Minerals Dhahran, April 2003.
[16] A. Abraham, "Adaptation of Fuzzy Inference System Using Neural Learning", Springer-Verlag Berlin Heidelberg, 2005, pp. 59-83.
[17] Y. Shi, M. Mizumoto, N.Yubazaki, M. Otani, "A Learning Algorithm for Tuning Fuzzy Rules Based on the Gradient Descent Method", Proceedings of Fifth IEEE International Confe-rence on Fuzzy Systems (FUZZ-IEEE'96), New Orleans, USA, Vol.1, 1996, pp.55-61.
[18] C.L. Martin, J.L. Pasquier, M.C. Yanez, T.A. Gutierrez, "Software Development Effort Estimation Using Fuzzy Logic: A Case Study", IEEE Proceedings of the Sixth Mexican Inter-national Conference on Computer Science (ENC'05), 2005, pp. 113-120.
[19] V. Xia, L. F. Capretz, D. Ho, "A Neuro-Fuzzy Model for Function Point Calibration", WSEAS Transactions on Information Science & Applications, Vol. 5, Issue 1, 2008, pp. 22-30.
[20] S. F. Ahmad, et al., "A Comparative Study of Software Quality Models," International Journal of Science, Engineering and Technology Research, vol. 2, pp. pp: 172-176, 2013.
[21] Sheikh Fahad Ahmad, Mohd Rizwan Beg, and Mohd Haleem. "Test Driven Development with Continuous Integration: A Literature Review." International Journal of Computer Applications Technology and Research 2.3: 281-285.
[22] Fatima, Mizbah, Sheikh Fahad Ahmad, and Mustafa Hasan. "Fuzzy Based Software Cost Estimation Methods: A Comparative Study." International Journal for Scientific Research and Development 1.7 (2015): 287-290.

