Software Effort Estimation Using Neuro-Fuzzy Approach

[1]Urvashi Rahul Saxena, [2]S.P Singh

[1]*Assistant Professor, Department of Computer Science and Engineering, JSS Academy of Technical Education, Noida, India*

[2]*Assistant Professor, Computer Science Department, Birla Institute of Technology, Noida, India*

[1]urvashirahulsaxena@gmail.com

[2]spsingh11in@yahoo.co.in

***Abstract*- A successful project is one that is delivered on time, within budget and with the required quality. Accurate software estimation such as cost estimation, quality estimation and risk analysis is a major issue in software project management. A number of estimation models exist for effort prediction. However, there is a need for novel model to obtain more accurate estimations. As Artificial Neural Networks (ANN's) are universal approximators, Neuro-fuzzy system is able to approximate the non-linear function with more precision by formulating the relationship based on its training. In this paper we explore Neuro-fuzzy techniques to design a suitable model to utilize improved estimation of software effort for NASA software projects. Comparative Analysis between Neuro-fuzzy model and the traditional software model(s) such as Halstead, Walston-Felix, Bailey-Basili and Doty models is provided. The evaluation criteria are based upon MMRE (Mean Magnitude of Relative Error) and RMSE (Root mean Square Error). Integration of neural networks, fuzzy logic and algorithmic models into one scheme has resulted in providing robustness to imprecise and uncertain inputs.**

***Keywords*— Effort Estimation, Artificial Neural Networks, Halstead Model, Walston-Felix Model, Bailey-Basili Model, Doty Model.**

I. Introduction

As software development has become an essential investment for many organizations, accurate software cost estimation models are needed to effectively predict, monitor, control and assess software development.

Need for Estimates:

(1) Strategic Planning
(2) Feasibility Study
(3) Project Planning
(4) System Specifications
(5) Evaluating Supplier's Proposal.

Accurate estimates are critical both to developers and customers. Underestimation of costs may result in management approving proposed systems which can exceed their budgets, with underdeveloped functions and poor quality, and failure to complete on time. Cost overestimation may result in too many resources committed to the project, or, when contract bidding is done, result in not winning the contract, which can lead to loss of jobs. According to Parkinson's Law *work expands to fill the time available*. Estimate calculation should be the preliminary step at the beginning of the software life cycle; this fact is also referred to *Cone of Uncertainty* as mentioned in Fig.1; the factor gradually decreases over time from initially four to zero at the end forming a cone-like structure.

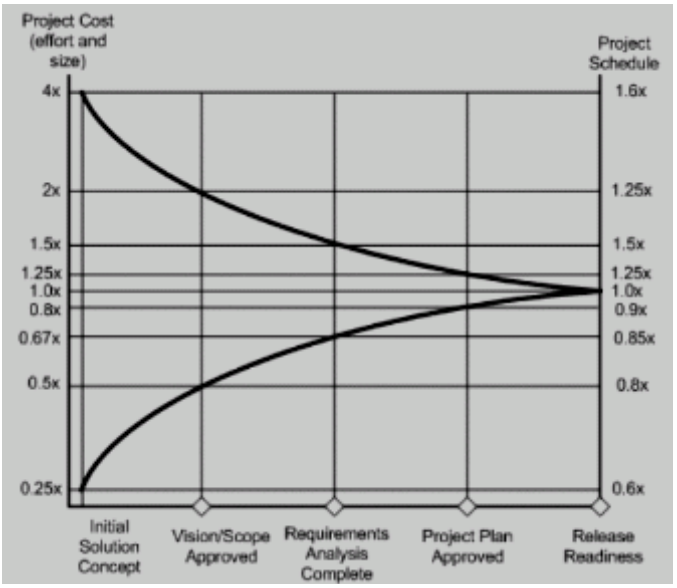

Fig.2 Cone of Uncertainty [15]

Thus, accurate cost estimation is important and Software cost estimation involves the determination of effort (usually in person-months), project duration (in calendar time) and cost (in dollars). Literature review explains that there are two major types of cost estimation models, Algorithmic and Non-Algorithmic models as discussed in [2],[3],[4],[5],[6],[8],[10-12].

Observations predict that cost and effort estimation models [1],[2] helps in project exploration and are one of the major contributors to map project sizes into project efforts. Thus, in this paper we integrate and compare several models for analysis of effort estimation using neuro-fuzzy approach.

Typical models that are being used as benchmarks for software effort estimation are:

(i) Halstead
(ii) Walston-Felix
(iii) Bailey-Basili
(iv) Doty (for KLOC > 9)
(v) Neuro-fuzzy Model

As Neural based system is able to predict approximately the non-linear function with more precision and none of the researchers have explored Neuro approach for the Effort Estimation and there is still scope of exploring statistical modeling approach. Thus, this comparative study is an attempt to build Neuro-fuzzy systems that can improve accuracy estimates of effort required to build a software system.

The remainder of this paper is organized as follows: The next section contains a description of the methodology used; Sections III discusses the implementation results of proposed and existing models. At last, conclusions are drawn.

II. Methodology Used

The following steps of the methodology are proposed for modeling of effort estimation:

A. Data Collection

First, Survey of the existing Models of Effort Estimation is to be performed and Secondly, Historical Data being used by various existing models for the cost estimation is collected.

B. Neuro-Fuzzy Framework

An intuitive attempt to solve problem related to the ratings of contributing factors as per input values are concerned is to use a *Five Layer Neuro-Fuzzy Model* for software estimation (Gray and MacDonell 1997). In this model, the inputs are the rating values or measurements of the contributing factors and the output is the estimated software output metric. That is, they use a neuro-fuzzy model to handle the dependencies among contributing factors and to calibrate all the parameters of fuzzy if then rules. While the learning capability is an advantage from the viewpoint of FIS, the formation of linguistic rule base will be the advantage from the viewpoint of ANN.

For many software estimation problems, the effects of contributing factors on the estimated software output metrics are usually not strongly coupled together, and strong effect dependency exists only on a small number of combinations of contributing factors. Consequently, given a specific software estimation problem, we can find the combinations of contributing factors with strong effect dependency based on expert knowledge and analysis of the characteristics of the problem, and then only handle dependency on this small part of combinations. This will greatly reduce the complexity of the estimation problem and make the problem tractable.

In a simplest way, a cooperative model can be considered as a pre-processor wherein ANN learning mechanism determines the FIS membership functions or fuzzy rules from the training data.

In this framework as depicted in Fig. 2, we initially propose a pre-processing neuro-fuzzy inference system (PNFIS) to handle the dependencies among contributing factors. We then use a neuro-fuzzy bank to map the adjusted rating values of contributing factors to the corresponding numerical multiplier values. Finally, output is evaluated using algorithmic model(s).

There are three major components in our framework: Pre-processing Neuro-Fuzzy Inference System (PNFIS), Neuro-Fuzzy Bank (NFB) and algorithmic model.

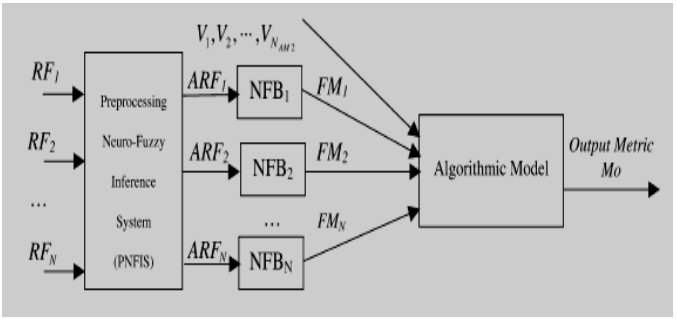

Fig.2 PNFIS Architecture [16]

This architecture is applicable to various applications and allows the utilization of more information from different sources such as expert knowledge and project data. The change of the application will only cause the redefinition of the fuzzy if-then rules in the first part (PNFIS) and the replacement of the algorithmic model in the third part.

The expert knowledge on contributing factor dependency is encoded in the first part: preprocessing neuro-fuzzy inference system. The conventional algorithmic model reflects important knowledge on a specific software estimation problem in which we use the neuro-fuzzy bank to calibrate the parameters of the algorithmic model through learning from industry project data. Thus, we can integrate the numerical project data, expert knowledge, and conventional algorithmic model into one framework. According to information theory and collected data, this process should yield more accurate estimation results.

1) Pre-processing Neuro-fuzzy inference system (PNFIS): It is assumed in many of the approaches that the effects of contributing factors on the estimated software output metric are independent, but this assumption is often arguable. Due to issues in widely used models such as COCOMO, experts diverge from this assumption. In this paper we propose the use of a preprocessing neuro-fuzzy inference system (PNFIS) that encodes expert knowledge into fuzzy if then rules; the inputs of PNFIS are the ratings of the contributing factors (RF_is) and its output is the adjusted ratings of contributing factors (ARF_is) and numerical value for the contributing factors (FM_is).

2) The Neuro-fuzzy bank (NFB): In the neuro-fuzzy bank, element *i* is a neuro-fuzzy subsystem ($NFB_{i)}$, which is

associated with the contributing factor *i*. Each contributing factor has several qualitative rating levels: very low(VL),low(L),nominal(N),high(H),very high(VH) and extra high(XH) for most cost drivers. Thus, for every contributing factor, each rating level relates to a quantitative value called parameter value, which is used in the algorithmic model. One goal of building a software estimation model is to calibrate its parameter values based on the expert knowledge and project data.

As far as the functionality of NFB_i is concerned two broad perspectives can be analyzed:

(i) From the learning perspective, we can treat NFB_i as a neural network so that we can use available learning algorithms for neural networks to calibrate the corresponding parameters, and NFB_i has learning and adaptation capability.

(ii) From the reasoning perspective, NFB_i can be considered as a fuzzy logic system. Its output is derived using fuzzy if-then rules and the reasoning process is transparent in the way similar to the decision-making process of human beings. Thus, the neuro-fuzzy subsystem NFBi is not a black box. The whole reasoning process is clear to the user and can be traced and validated by users/experts. Consequently, our neuro-fuzzy model is more easily accepted for project management.

Remark 1: In the neuro-fuzzy bank, each element is a 'standard' neuro-fuzzy subsystem. The number of elements is equal to the number of rating contributing factors. In each element, the number of fuzzy rules equals the number of rating levels of the corresponding factor.

Remark 2: For a given problem, once the number of rating contributing factors and the number of rating levels of each factor are determined, the structure and fuzzy rules of the neuro-fuzzy bank are determined. Only the fuzzy rule parameters are needed to fine-tune through the learning process from numerical project data.

Remark 3: *Guideline for determining rating levels* specifies how to determine the appropriate number of rating levels. On one hand, fewer rating levels imply coarse and less accurate description of contributing factors, but the system is easy to use since users can give more accurate rating values. On the other hand, too many rating levels for one factor indicate more accurate descriptions of contributing factors, but the definition and rating criteria of each level are more complicated .Users generally have difficulty rating the factors. That is, the rating errors tend to be larger. Therefore, there is a balance between the number of rating levels and ease of use.

On wide categorization there are two basic types of Neuro-fuzzy systems: Mamdani Neuro-fuzzy systems and Takagi-Sugeno Neuro-fuzzy system.

The Sugeno based Neuro-fuzzy model has the following advantages over Mamdani Model:

(a) Expressing a clearer concept of inference process.
(b) Less number of fuzzy rules
(c) Faster learning
(d) Less memory space.

In this Neuro-fuzzy system the first Sugeno Based Fuzzy Inference System is designed that needs the initialization of the Membership Function of the different attributes and linear Membership Function for the output and deducing the fuzzy rules from the data. Then the Sugeno Based FIS is trained with the neural Network using the hybrid training algorithm. In the forward pass, Backpropagation learning algorithm and in the backward pass Least Mean Square Error (LMS) learning algorithm is used to update the non-linear and linear parameters of the Neuro-fuzzy system respectively.

According to the inference rule base concept, two Takagi-Sugeno if-then rules are as follows:

Rule 1: If x is A_1 and y is B_1, then $f_1=P_1x+q_1y+r_1$.

Rule 2: If x is A_2 and y is B_2, then $f_2=P_2x+q_2y+r_2$.

The detailed functioning of each layer of Takagi Sugeno Neuro-fuzzy Inference system [1] as depicted in Fig. 3 is as follows:

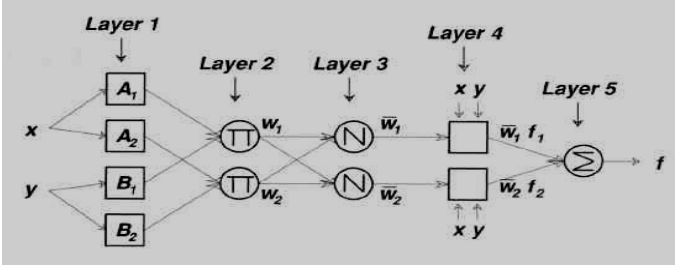

Fig. 3 Five layer ANFIS Structure [1]

Layer-1(Input layer): No computation is done in this layer. Each node in this layer, which corresponds to one input variable, only transmits input values to the next layer directly. The link weight in layer 1 is unity.

Layer-2 (Production of the Intermediate result): This layer is also called the fuzzification layer. Each node in this layer corresponds to one linguistic label, (excellent, good, bad, etc). to one of the input variables in layer 1. The output link represents the membership value, which specifies the degree to which an input value belongs to a fuzzy set, is calculated in this layer. The membership functions are tuned according to the design of the training algorithm.

Layer-3 (Rule Antecedent layer): Every node i in this layer represents the antecedent part of a rule. Usually a T-norm operator is used in this node. The concept of triangular norm was introduced in order to generalize the triangular inequality of a metric-norms generalize the conjunctive 'AND' operator and T-conorms generalize the disjunctive 'OR' operator. The output of a layer 3 node represents strength of the corresponding fuzzy rule.

$$\overline{wi} = \frac{wi}{w1 + w2}, i = 1,2 \ldots . \qquad (1)$$

Layer-4 (Rule strength normalization): Every node in this layer calculates the ratio of the ith rule's firing strength to the sum of all rules firing strength.

Layer-5 (Rule consequent layer): Every node i in this layer is with a node function:

$$\overline{w}ifi = \overline{w}i(pix1 + qix2 + ri) \qquad (2)$$

Where, w_i is the output of layer 4, and {fi;qi;ri} is the parameter set .Most prominent way is to determine the consequent parameters using the least mean squares algorithm.

Overall output, X_i is the summation of all incoming signals.

$$Xi = \sum_i \overline{w} fi = \frac{\sum_i \overline{w}ifi}{\sum_i wi} \qquad (3)$$

Fuzzy if-then rule inference mechanism is depicted in Fig.4.

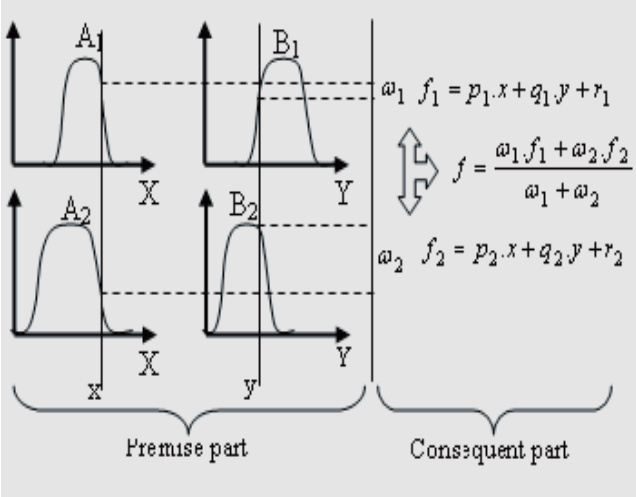

Fig.4 Fuzzy if-then Inference System [17]

C) Effort Calculation using different modes

The use of algorithmic models to predict software output metrics is one of the most popular and traditional software estimation techniques. An algorithmic estimation model can be built by analyzing a software output metric and attributes of completed projects, and used to predict the software output metric based on the attributes of software product and development process.

After data collection, following models are used for effort estimation:

(i) Halstead
(ii) Walston-Felix
(iii) Bailey-Basili
(iv) Doty (KLOC>9)
(v) Neuro-fuzzy Model.

Over the last 20 years, research efforts have focused on the development of techniques that are quantitatively based, in an effort to remove or reduce subjectivity in the estimation process. Wittig and Finnie [9], describe their use of back-propagation learning algorithms on multilayer perceptron in order to predict development effort. Although, other techniques for the exploratory data analysis, such as clustering, case-based reasoning and ANN have been effective as a means of predicting software project effort. Use of clustering techniques to predict software quality is described in [13].

Khoshgoftaar [14] presented a case study considering real time software to predict the testability of each module from source code static measures. They consider ANN's as one of the most promising modes to build predictive models, as they are capable of modelling non-linear relationships.

ANN's are massively parallel systems inspired by the architecture of biological neural networks, comprising simple interconnected units (artificial neurons). An Artificial neuron computes a weighted sum of its inputs and generates an output if the sum exceeds a certain threshold; this output then becomes an excitatory (positive) or inhibitory (negative) input to other neurons in the network. Fig. 5 shows an artificial neuron, which computes the weighted sum of its n inputs, and generates an output of y.

The neural network is a resultant of the arrangement of various input-output units in layers, which are interconnected one to another. The resulting architecture is a solution to several issues by learning the characteristics of the available data of related to the problem.

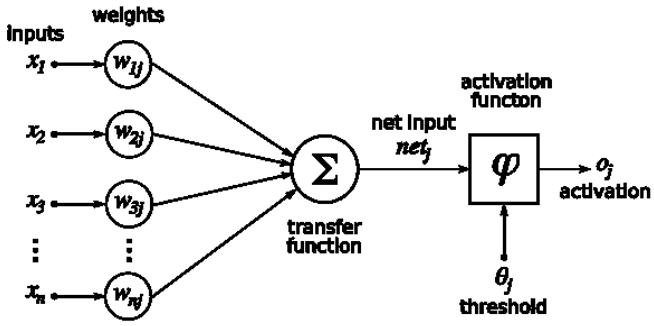

Fig.5 An Artificial Neuron [18]

To elaborate these simulations Back-Propagation artificial neural networks were used; at each neuron in the network the error function is used to adjust the weights and threshold values of the neuron , so that at the next epoch the error in the network response.

Results on comparing different models are based on the following factors:

(i) Mean Magnitude of Relative Error (MMRE)
(ii) Root Mean Square Error (RMSE)

RMSE is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated.

The mean-squared error is one of the most commonly used measures of success for numeric prediction. This value is computed by taking the average of the squared differences between each computed value and its corresponding correct value. The root mean-squared error is simply the square root of the mean-squared-error as shown in equation given below:

$$RMSE= \sqrt{\frac{1}{N}\sum_{i=1}^{n}(yi-\hat{y})^2} \quad (4)$$

Where actual value i.e., the i^{th} value is represented by y_i and $\hat{y}i$ is the estimated effort.

Literature review of software effort estimation using Neural Networks [19] considers *mean magnitude of relative error (MMRE)* as the main performance measure. The value of an effort predictor can be reported many ways including MMRE. Relative error helps in computing, which is the relative size of the difference between the actual and estimated value.

Given a data set of size "D", a "Training set of size "(X=|Train|) <= D", and a "test" set of size "T=D-|Train|", then the mean magnitude of the relative error, or MMRE, is the percentage of the absolute values of the relative errors, averaged over the "T" items in the "Test" set.

MMRE is another measure and is the percentage of the absolute values of the relative errors, averaged over the N items in the "Test" set and can be written as:

$$MMRE=\frac{1}{N}\sum_{i=1}^{N}|yi-\hat{y}i|/yi \quad (5)$$

III. Results And Discussion

The data set used for the comparison of different models is from the data source of NASA [7]. In this dataset, there is empirical data in terms of *DKLOC, Methodology* and *Effort* values of 18 projects as shown in Table 1.

TABLE 1

NASA DATASET [8] OF EFFORT ESTIMATION

Project No.	DLOC	Methodology	Actual Effort
1	90.2	30	115.8
2	46.2	20	96
3	46.5	19	79
4	54.5	20	90.8
5	31.1	35	39.6
6	67.5	29	98.4
7	12.8	26	18.9
8	10.5	34	10.3
9	21.5	31	28.5
10	3.1	26	7.0
11	4.2	19	9.0
12	7.8	31	7.3
13	2.1	28	5.0
14	5.0	29	8.4
15	78.6	35	98.7
16	9.7	27	15.6
17	12.5	27	23.9
18	100.8	34	138.3

The Neuro-fuzzy system is trained for nearly about 500 epochs and then tested. The plot depicting variations of data index v/s expected output is shown in fig. 6

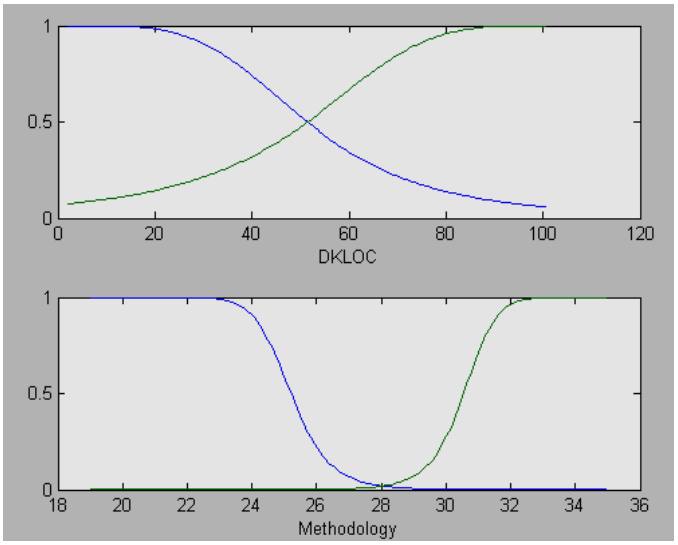

Fig .6 Changed Parameters of Membership Functions of the input attributes [19].

After training actual effort is calculated over 13 projects as shown in Table II and 5 projects are tested for all models as depicted in Table III.

TABLE II

COMPUTED EFFORT FOR NASA SOFTWARE PROJECTS- TRAINING CASE

Project No.	Actual Effort	Neuro-fuzzy Model	Halstead Model	Walston-Felix Model	Bailey-Basili Model	Doty Model
1	115.8	115.7	599.66	312.78	140.82	589.38
2	96	96.069	219.82	170.15	67.774	292.53
3	79	78.903	221.96	171.15	68.243	294.52
4	90.8	90.813	281.64	197.75	80.93	347.78
5	39.6	38.875	121.41	118.69	44.848	193.29
6	98.4	98.447	388.2	240.25	102.18	435.08
7	18.9	20.789	32.056	52.913	19.55	76.303
8	10.3	9.9815	23.817	44.186	16.666	62.012
9	28.5	29.343	69.784	84.825	31.142	131.33
10	7.0	7.3786	3.8207	14.559	8.2121	17.288
11	9.0	8.6934	6.0252	19.194	9.3574	23.759
12	7.3	8.9732	15.249	33.714	13.41	45.427
13	5.0	4.6705	2.1302	10.215	7.2262	11.499

Simulations show that after analysing the *Training* and *Testing* cases for the back-propagation neural network, performance evaluation is simplified. Thus, we calculate the results on the basis of RMSE and MMRE, this helps in comparing the results for each developed model.

It is clearly evident from Table II that predicted values of effort are very close to the expected or actual values. Calculation of RMSE and MMRE values over the complete data set of models is portrayed in Table IV.

It can be seen that the Neuro-fuzzy model outperforms the Halstead, Walston-Felix, Bailey-Basili and Doty Models.

TABLE III

COMPUTED EFFORT FOR NASA SOFTWARE PROJECTS-TESTING CASE

Project No.	Actual Effort	Neuro Fuzzy Model	Halstead Model	Walston-Felix Model	Bailey-Basili model	Doty Model
14	8.4	10.129	7.8262	22.494	10.222	28.518
15	98.7	113.81	487.79	275.95	120.85	510.27
16	15.6	18.126	21.147	41.112	15.685	57.074
17	23.9	22.647	30.936	51.783	19.169	74.431
18	138.3	141.6	708.42	346.06	159.43	662.09

TABLE IV

COMPUTED RMSE AND MMRE CRITERION FOR ALL MODELS

Performance-Criteria	Model Used				
	Neuro-Fuzzy Model	Halstead Model	Walston-Felix Model	Bailey-Basili Model	Doty Model
RMSE	7.0731	308.71	123.46	13.877	299.47
MMRE	0.1194	1.7566	1.5556	0.1595	3.025

IV. CONCLUSION

This paper has presented a general framework for software estimation. The framework concentrates on the Pre-processing Neuro-Fuzzy Inference System, the Neuro-Fuzzy Bank and Algorithmic models. Performance of the Neuro-fuzzy based Estimation Model and the other existing models such as: Halstead Model, Walston-Felix Model, Bailey-Basili Model and Doty Model is compared for effort dataset available in literature [7]. Results predict that Neuro-Fuzzy system has the lowest error values by observing RMSE and MMRE values in Table IV as 7.0731 and 0.1194 respectively. Second best performance is shown by Bailey-Basili Model with 13.877 and 0.1595 as RMSE and MMRE values. Hence, the proposed Neuro-Fuzzy System is able to provide good estimation capabilities.

REFERENCES

[1] A. Abraham, Adaptation of Fuzzy Inference System Using Neural Learning, Springer Berlin, ISSN: 1434-9922 (Print) 1860-0808 (Online), vol 181, 2005.

[2] B. W. Boehm, Software engineering economics, Englewood Cliffs, NJ: Prentice-Hall, 1981.

[3] C. E. Walston, C. P. Felix, A method of programming measurement and estimation, IBM Systems Journal, vol. 16, no. 1, pp. 54-73, 1977.

[4] G.N. Parkinson, Parkinson's Law and Other Studies in Administration, Houghton-Miffin, Boston, 1957.

[5] L. H. Putnam, A general empirical solution to the macro software sizing and estimating problem, IEEE Trans. Soft. Eng., pp. 345-361, July 1978.

[6] R J. R. Herd, J.N. Postak, W.E. Russell, K.R. Steward, Software cost estimation study: Study results, Final Technical Report, RADC-TR77-220, vol. I, Doty Associates, Inc., Rockville, MD, pp. 1-10, 1977.

[7] J. W. Bailey and V. R. Basili, "A meta model for software development resource expenditure," in Proceedings of the International Conference on Software Engineering, pp. 107–115, 1981.

[8] R. E. Park, PRICE S: The calculation within and why, Proceedings of ISPA Tenth Annual Conference, Brighton, England, pp. 231-240, July 1988.

[9] 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.

[10] R.K.D. Black, R. P. Curnow, R. Katz, M. D. Gray, BCS Software Production Data, Final Technical Report, RADC-TR-77-116, Boeing Computer Services, Inc., March, pp. 5-8, 1977.

[11] R. Tausworthe, Deep Space Network Software Cost Estimation Model, Jet Propulsion Laboratory Publication 81-7, pp. 67-78, 1981.

[12] W. S. Donelson, Project Planning and Control, Datamation, pp. 73-80, June 1976.

[13] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris, "Software productivity and effort prediction with ordinal regression," Journal Information and Software Technology, 2005, no. 47, pp.17-29.

[14] T. M. Khoshgoftaar, E.B. Allen, and Z. Xu, "Predicting testability of program modules using a neural network," Proc. 3rd IEEE Symposium on Application-Specific Systems and Sof. Eng. Technology, 2000, pp. 57-60.

[15] Microsoft corporation. Cone of Uncertainty 2003, http://www.microsoft.com/china/technet/images/itsolutions/techguide/innsol/images/msfpmd07.gif

[16] A Soft Computing Framework for Software Effort Estimation, Xishi, Huang, Danny Ho, Jing ren, Luiz F. Capretz,pp170-177

[17] ANFIS: Adaptive Neuro- Fuzzy Inference System, Adriano Cruz Mestrado NCE, IM, UFRJ.

[18] http://commons.wikimedia.org/wiki/file:ArtificialNeuronModelenglish.png.

[19] Software Effort Estimation Using Soft Computing Techniques,WASET, 2008.