ONE HIDDEN LAYER ANFIS MODEL FOR OOS DEVELOPMENT EFFORT ESTIMATION

Sheenu Rizvi[1], Dr S.Q.Abbas[2], Dr Rizwan Beg[3]
[1]Department of Computer Science, Amity University, Lucknow, India
[2]A.I.M.T, Lucknow, India
[3]Integral University, Lucknow, India

Abstract: In the present paper, applicability and capability of A.I techniques for effort estimation prediction has been investigated. It is seen that neuro fuzzy models are very robust, characterized by fast computation, capable of handling the distorted data. Due to the presence of data non-linearity, it is an efficient quantitative tool to predict effort estimation. The one hidden layer network has been developed named as OHLANFIS using MATLAB simulation environment.

Here the initial parameters of the OHLANFIS are identified using the subtractive clustering method. Parameters of the Gaussian membership function are optimally determined using the hybrid learning algorithm. From the analysis it is seen that the Effort Estimation prediction model developed using OHLANFIS technique has been able to perform well over normal ANFIS Model.

Keywords: ANFIS, Neural network, COCOMO-II, effort estimation, OOS

I. INTRODUCTION

Effort estimation of computer system software projects design play very significant role in the expansion of the software projects. When a estimated effort is less than the actual effort involved then it may generate number of issue that can directly effects the project quality, purposed time of completion, pressure on developing experts in terms of working overtime. A general issue in programming effort estimation is that verifiable information regularly holds an inclination to the goal that these information cannot directly be utilized for advancement of expense estimation devices, individually for the adjustment of such instruments and looking at estimations against its true or recorded effort value. Accurate effort and scheduling estimations provide very highly valuable medium in a number of designing decisions, financial expense decisions, and team work allocations and in supporting reliable bids for contract competition[1].

Evaluating programming improvement sets back the finances with correctness is generally a very troublesome regular approach for enhancing programming or software performance. The extra objective of having the capacity to anticipate the expenses and calendar at the start of the venture can end up being all the more testing. Unanticipated forecast of finishing time is completely vital for fitting development arranging and abhorrence of the project[2,3].

Nevertheless, utilizing the whole database of accessible programming effort estimation displays, researchers found that there is no confirmation that programming models are adequate at assessing ventures at an early phase of framework improvement.

A. *Problem of object-oriented software effort estimation:*

Many object-oriented software effort estimation methods have been proposed over the last decade. Effort prediction models using object-oriented design metrics can be used for obtaining estimates about software project performance and quality. In practice, effort estimation means either estimating reliability or maintainability [4]. Reliability is generally represented as the number of pre-release or post-release defects. Hence by effort estimated the number of active defects can also be normalized by a size measure to obtain a defect density estimate. Another parameter is related to issue of maintainability which is normally represented as a change effort. Change effort is explained as: 'either the average effort to make a change to a class or the total effort spent on changing a class'. There is great interest in the use of object-oriented based approach in software engineering. With the increasing use of object-oriented methods in new software development there is a growing demand to both in documents and current practices in object-oriented design and development.

Many methods have been proposed in the last two decades to estimate the quality of object-oriented software code and design efforts and used for detecting fault-proneness of classes. Most of the OOS effort estimation methods have some limitations that the organizing heads and developers need to be aware of [3, 5] so that the effort estimation may be viewed as valuable tools in the software engineering process.

A common approach concerning the OOS effort estimation based models is that they base their effort and scheduling predictions on the estimated size of the software project at hand , in terms of number of lines of code (LOC), or thousands of lines of code (KLOC). In most of the methodologies the OOS effort estimation is based upon an equation similar to:

$$E = A + B*(KLOC)^{C} \qquad 1(a)$$

where E stands for estimated effort (usually in many months), A, B, and C are constants, and KLOC is the expected number of thousands of lines of code in the final system. From the above equation it is easy to see that a given percentage error in the size (KLOC) may cause an even larger percentage error in the estimated effort. For instance, in COCOMO a 50% error in the size estimate will roughly result in a 63% error in the effort estimate. [3, 4, 6, 8, 15]. Many investigations using statistical methods had been made to predict software quality. In this paper we focused on the object-oriented software effort estimation approach. This document explores object-oriented paradigm that exhibits different characteristics from the procedural paradigm and the different software metrics that has been defined and used. We proposed various artificial intelligence based optimization approach that aims to predict object oriented software design effort by using the number of lines other software design related parameters. We have considered software design metrics concerning with inheritance related measures, cohesion measures, coupling measures, complexity measures and size measure. We discussed about network and fuzzy based predictors to improve estimation results for OOS design. Neural network based method is a back propagation network with different activation functions. They are applied to hidden layer slabs to detect different features in a pattern processed through a network to lead to better prediction. By using proper functions in one hidden slab we can detect feature in the mid-range of the data and the complement of chosen function in another hidden slab is used to detect features for the upper and lower extremes of the data. Thus, the output layer will get different "views of the data". Combining the two feature sets in the output layer leads to a better estimation.

Another architecture that we have chosen is the Adaptive neuro-fuzzy inference system (ANFIS).It is a memory-based network that provide estimates of continuous variables and converges to the underlying (linear or nonlinear) regression surface. This is a one-pass learning algorithm with a highly parallel structure. Even if the data is sparse in a multi-dimensional measurement space; the ANFIS algorithm [6] provides smooth transitions from one observed value to another. Most of these prediction models can be built using statistical models. A.I techniques have seen an explosion of interest over the years, and are being magnificently applied over a large range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics. Indeed, in the cases there are problems of prediction, classification or control, these methods are being introduced. We have found that these methods can be used as a predictive model because it is very sophisticated modeling techniques capable of modeling complex functions.

B. *Framework work for object oriented s/w effort estimation*

The COCOMO model database is used due to its popularity for software effort estimation. This dataset has been validated on estimation of large projects at consulting firm, Teen Red Week (TRW) software production system (SPS) in California, USA [8]. The structure of the dataset has been divided on the basis of the parameters of projects to be handled. The projects are categorized as organic, semidetached, and embedded. The model structure that we have followed is represented as follows:

Effort=$a*(\text{KLOC})^{b}$ (1b)

Here a and b are domain specific constants. For estimating the object oriented software development effort, a and b have been adjusted on the past data set of various projects. Five scale factors are considered to generalize and demonstrate the effects of the development mode in COCOMO II [9,10]. There are fifteen constraints which acts as parameter that can affect the effort involved in the software development. These constraints are analyst capability ($acap$), programmer developer capability ($pcab$), application related expertise (aexp), modern based programming expertise (modexp), software tools involved in development, design and testing ($tool$), virtual memory experience (Vexp), language expertise ($lexp$), scheduling constraint ($sced$), main memory constraint ($stor$), database size requirement ($data$), CPU related time constraint ($time$), turnaround time ($turn$), machine volatility (Virt), process complexity ($cplx$), and reliability of the required software ($rely$):

Effort=$a*(\text{KLOC})^{b}*c$ (2)

KLOC is directly computed from a function point analysis and c is the product of fifteen effort multipliers hence effort can be represented as:

Effort=$a*(\text{KLOC})b*(\text{EM1}*\text{EM2}*\cdots*\text{EM15})$ (3)

Above prediction model of software development effort estimation is applied to estimate the software development effort by using sixteen independent specifications named as $rely$, $data$, $cplx$, $time$, $stor$, Virt, $turn$, $acap$, $aexp$, $pcab$,Vexp, $lexp$, $modp$, $tool$, $sced$ and $kloc$.

All these sixteen parameters are used as input vector in one hidden layer feed forward ANFIS network discussed in next section.

C. *Effort Estimation Using One Hidden Layer ANFIS Network (OHLANFIS)*

Through back propagation with gradient descent training, mapping between input vectors and output vectors an ANFIS network has been developed with the target of minimization of the sum of squared error at output layer. The optimal weight vectors are evaluated for the network to estimate the software development effort of database given in COCOMO II model for object oriented software projects. The optimal weight vector obtained from ANFIS network is being used as testing the results for an optimized network with minimum root mean square error value and high regression.

OHLANFIS is developed with gradient descent back propagation learning method for optimizing this model in reference of estimation of the object oriented software development effort.

We have considered input vector $XTk=(x1;x2;...xn)$ where $n=16$ and output vector $DTk=(d1;d2;...dp)$. The OHLANFIS network is trained by using the input and output vector mapping. P is set of Q training vector pairs:

$P=\{Xk, \text{Dk}\}^{Q}{}_{k=1}$ (4)

$Xk\in \mathrm{R}^{n}$ and $k\in R^{p}$, where$n$=16, p=1, and Q=40.

Here neuro-fuzzy rules are generated for k output signal vector(Yk)andYk is vector of activations of output layer fuzzy rules. Error at kth training pair (Xk, Dk) is evaluated as follows:

$Ek=Dk-\mathrm{f}(Yk)$, (5)

Where, $Ek=(ek1,...ekp)^{T}$

$=(dk_{1}-\mathrm{f}(yk_{1}),...,dk_{p}-f(yk_{p}))^{T}$ (6)

The squared error is considered taken as sum of squares of every individual output error ekj given as:

$\xi k=1/2.\sum(dk_{j}-f(yk_{j}))^{2}=1/2.Ek^{T}Ek$ (7)

The MSE is computed over the entire training set P:

$\text{MSE}=1/Q\sum\xi k.$ (8)

The weights between hidden and output layer are updated as

$w_{k+1}{}^{hj}=w_{k}{}^{hj}+\Delta w_{k}{}^{hj}$ (9)

and the weights between input and hidden layer are updated as

$w_{k+1}{}^{ih}=w_{k}{}^{ih}+\Delta w_{k}{}^{ih}$ (10)

where $\Delta w_{k}{}^{hj}$ and $\Delta w_{k}{}^{ih}$ are weight changes computed in previous step.

These weights are iteratively updated in output and hidden layers by the following equations:

$\Delta w_{k+1}^{hj} = \Delta w_k^{hj} + \eta(zkh)$,

$\Delta w_{k+1}^{ih} = \Delta w_k^{ih} + \eta\delta kXxki$ (11)

We can introduce the momentum into back propagation with the help of the following equations:

$\Delta w_k^{hj} = \eta(zkh) + \alpha\Delta w_{k-1}^{h}$,

$\Delta w_k^{ih} = \eta\delta khxki + \alpha\Delta w_{k-1}^{ih}$ (12)

Back propagation propagates changes back because it can do substantial good thing. The change in Oj should be proportional to $(1-Oj)$ the slope of the threshold function, at node k.The change to Oj should also be proportional to Wjk the weight on the link connecting node j to node k.
Summing over all nodes in layer $k = \sum kw(1-Ok)\beta k$.
At the output layer, the benefit has been given by the error at the output node. The output layer z will be benefited as $\beta z = dz - oz$.
Here a rate parameter r has been introduced for controlling the learning rate. So change in wij is proportional to r; that is, $\Delta wij = r(1 - Oj)\beta$
j and $\beta j = \sum kWjk(1 - Ok)\beta k$ for nodes in hidden layers and $\beta z = dz - oz$ for nodes in the output layer. The output of the network is compared with desired output; if it deviates from desired output, the difference between actual output and the desired output is propagated back from the output layer to previous layer to modify the strength or weight of connection.

II. RESULTS FOR OHLANFIS BASED OOS EFFORT ESTIMATION

Here the OHLANFIS model has been trained tested by neuro-fuzzy based algorithm and their performance for the best prediction model are evaluated and compared for training and testing data sets separately. The RMSE performances of the model both for training and testing datasets have been plotted separately in Fig. 1 & Fig.2 and their corresponding range of values (minimum and maximum) are summarized in Table 1.

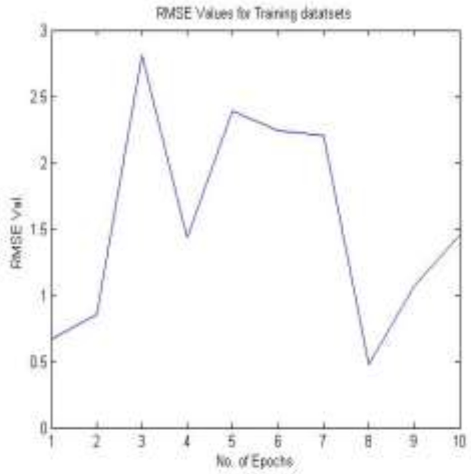

Fig. 1 Graphical plot of RMSE value variation during training

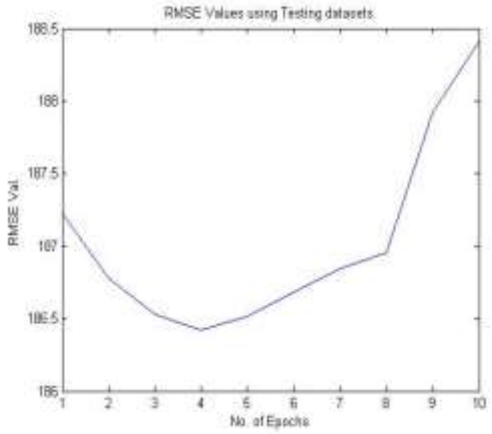

Fig. 2 Graphical plot of RMSE value variation during testing

TABLE 1: RANGE OF RMSE VAL. DURING TRAINING AND TESTING PHASE

	RMSE Value	
	Minimum	**Maximum**
Training datasets	0.4824	2.8096
Testing datasets	186.41	188.41

Further Table 2 gives the RMSE values using both the normal ANFIS and OHLANFIS techniques.

TABLE 2 : PERFORMANCE EVALUATION USING RMSE CRITERIA

	Using Normal ANFIS	**Using OHLANFIS**
RMSE Val.	532.2147	112.638

From analysis of Fig. 1 & Fig. 2 and perusal of the data given in tables 1 and 2 it is inferred that during training phase (Fig.1), there is zig zag variation in the RMSE values, having a minimum value of 0.4824 (at epoch 8) and a maximum value of 2.8096 (epoch 3). Hence during training phase there is initially a rise in the RMSE value and then there is a fall at epoch no. 8, after which there is again a slight increase. On the other hand, during testing phase (Fig.2) of OHLANFIS training initially upto epoch 4 the RMSE value decreases and reaches upto a minimum of 186.41 and then there is steep rise in the RMSE value upto 10 epochs, where the maximum value reached is 188.41. From the above analysis it can be inferred that ANFIS has performed better during training phase than testing phase but its overall RMSE value is 112.638. which shows a marked improvement than those calculated in COCOMO model i.e. 532.2147 (given above in Table 2).

III. CONCLUSION

The absolute values of Mean of Relative Error (MRE) calculated both for normal ANFIS and OHLANFIS and their comparative plot, both for training and testing are compared. From the perusal of both the data and the graphical plot, it is seen that during the training as well as testing phase of the OHLANFIS model development, the absolute values of the

MRE are very less as compared to normal ANFIS model, especially during training phase. Since Absolute MRE computes the absolute percentage of error between the actual and predicted effort for each referenced project, hence from the above data analysis it can be inferred that the absolute percentage of error between the actual and predicted effort using OHLANFIS technique is far less than those using normal ANFIS model. Thus, it is clear that proper selection of influential radius which affects the cluster results directly in ANFIS using substractive clustering rule extraction method, has resulted in reduction of RMSE and MRE both for training and testing data sets. Hence, it is seen that for small size training data, OHLANFIS has outperformed normal ANFIS model.

REFERENCES

[1] Mohammad Saber Iraji, Et.Al., (2012), " Object Oriented Software Effort Estimate With Adaptive Neuro Fuzzy Use Case Size Point (Anfusp)", *I.J. Intelligent Systems And Applications,* 6, 14-24.

[2] K.K.Aggarwal,Et.Al.,(2006),**"**Software Design Metrics For Object-Oriented Software", *Journal Of Object Technology*, Vol. 6. No. 1, Pp. 121-138.

[3] Somesh Kumar, Manu Pratap Singh Et Al, "Hybrid Evolutionary Techniques In Feed Forward Neural Network With Distributed Error For Classification Of Handwritten Hindi (Swars)", Connection Science, 2013 Taylor & Francis Pp. 197-215

[4] Yuming Zhou, Baowenxu, Hareton Leung, "On The Ability Of Complexity Metrics To Predict Fault-Prone Classes In Object-Oriented Systems", The Journal Of Systems And Software 83(2010) 660-674, Elsevier.

[5] Alaa F. Sheta, Alaa Al-Afeef, "A Gp Effort Estimation Model Utilizing Line Of Code And Methodology For Nasa Software Projects" 978-1-4244-8136-1 Ieee Transaction, 2010 Pp. 284-289.E. Praynlin, P. Latha, "Performance Analysis Of Software Effort Estimation Models Using Neural Networks", I.J. Information Technology And Computer Science, 2013, 09,101-107.

[6] SOMESH KUMAR, MANU PRATAP SINGH ET AL ,"HYBRID EVOLUTIONARY TECHNIQUES IN FEED FORWARD NEURAL NETWORK WITH DISTRIBUTED ERROR FOR CLASSIFICATION OF HANDWRITTEN HINDI (SWARS)",CONNECTION SCIENCE, 2013 TAYLOR & FRANCIS PP.197-215.

[7] CHANDRA SHEKHAR, RAGHURAJ SINGH, "TUNING OF COCOMO81 MODEL PARAMETERS FOR ESTIMATING SOFTWARE DEVELOPMENT EFFORT USING GA FOR

[8] PROMISE PROJECT DATA SET", INTERNATIONAL JOURNAL OF COMPUTER APPLICATIONS (0975-8887) VOL. 90, NO. 1 FOUNDATION OF COMPUTER SCIENCE, NEW YORK, USA,2014 PP. 37-43.

[9] TAGHI M. KHOSHGOFTAAR, ROBERT M. SZABO, "IMPROVING NEURAL NETWORK PREDICTIONS OF SOFTWARE QUALITY USING PRINCIPAL COMPONENTS ANALYSIS", IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE, 1994 IEEE INTERNATIONAL CONFERENCE VOLUME:5, PP. 3295-3300 DOI:10.1109/ICNN.1994.374764.

[10] PING YU, T. SYSTA, AND H. MULLER, "PREDICTING FAULT-PRONENESS USING OO METRICS. AN INDUSTRIAL CASE STUDY", PROCEEDINGS. OF 6TH EUROPEAN CONFERENCE ON SOFTWARE AINTENANCE AND REENGINEERING, 2002, PP. 99 –107, 2002.