A Defect Prediction Model for Software Product based on ANFIS

Deepak Kumar Verma[1] H. S. Shukla[2]
[1,2]Department of Computer Science
[1,2]D.D.U. Gorakhpur University, Gorakhpur, India

Abstract— Artificial intelligence techniques are day by day getting involvement in all the classification and prediction based process like environmental monitoring, stock exchange conditions, biomedical diagnosis, software engineering etc. However still there are yet to be simplify the challenges of selecting training criteria for design of artificial intelligence models used for prediction of results. This work focus on the defect prediction mechanism development using software metric data of KC1.We have taken subtractive clustering approach for generation of fuzzy inference system (FIS).The FIS rules are generated at different radius of influence of input attribute vectors and the developed rules are further modified by ANFIS technique to obtain the prediction of number of defects in software project using fuzzy logic system.

Key words: Defect Prediction, ANFIS, Artificial Intelligence, Fuzzy Logic, ANN

I. INTRODUCTION

Adaptive neuro-fuzzy inference system is a fuzzy inference system developed in the structure of an adaptive neural network [1]. By using a hybrid learning procedure, ANFIS can develop an input-output relation based on both human-knowledge as fuzzy if-then rules and approximate membership functions from the stipulated input-output data pairs for neural network training. This procedure of developing a FIS using the framework of adaptive neural network is called an adaptive neuro fuzzy inference system (ANFIS)[1][2]. There are two techniques employed by ANFIS to update membership function parameters: 1) for all the parameters using backpropagation (a steepest descent method), and 2) a hybrid method made up of back propagation for the parameters attached with the input membership functions and least squares estimation for the parameters linked with the output membership function. Thus, the training error diminishes, for a smallest amount locally, all through the learning process. It applies the least-squares method to identify the consequent parameters that define the coefficients of each output equation in the Sugeno-type fuzzy rule base. The training process continues till the desired number of training steps (epochs) or the desired root mean square error (RMSE) between the desired and the generated output is achieved. This study uses a hybrid learning algorithm, to identify premise and consequent parameters of first order Takagi-Sugeno type fuzzy system for predicting software error.

Defective software modules cause software failures, increase development and maintenance costs, and decrease customer satisfaction. In ones work one strives to better software excellence and testing effectiveness by developing forecasting models from code attribute to allow an appropriate detection of fault-prone module. To identify and locate defects in software projects is a difficult task. Particularly, when project sizes grow up, this task becomes expensive with sophisticated testing and evaluation mechanisms. On the other hand measuring of softwares in an uninterrupted and regimented style brings many merits such as precise evaluation of project costs and schedules, in improving qualities of product and process. Detailed analysis of software metric data also gives important clue about the locations of possible defects in a programming code.

In the present work an Adaptive Neuro Fuzzy Inference System (ANFIS) Approach will be applied for the development of an efficient predictive model using Substractive Clustering Algorithm. For this NASA's Metrics Data Program (MDP) containing software metric data and error data at the function/method level has been used to validate the algorithm [3]. This includes a numeric attribute (NUMDEFECTS) to indicate defectiveness. The objective in the construction of models of software error prediction is to use measures that may be obtained relatively early in the software development life cycle to provide reasonable initial estimates of quality of an evolving software system.

II. RELATED WORK

In this section we have discussed recent advancement in the field of software defect prediction in the last decades. Xiao-dong Mu et. al.,(2012)[8], in their work to improve the accuracy of software defect prediction, a coevolutionary algorithm based on the competitive organization is put forward for software defect prediction. During this algorithm, firstly, competition mechanism is introduced to organization coevolutionary algorithm. Then, three evolution operators which are reduced operator, allied operators and disturbed operators are developed for evolution of population. And competition is considered for calculate the fitness function. When the algorithm applied into software defect prediction, it improves the accuracy of software prediction through increases the diversity of population.

Manu Banga, (2013) [7], here a new computational intelligence sequential hybrid architectures involving Genetic Programming (GP) and Group Method of Data Handling (GMDH) viz. GPGMDH have been discussed. Besides GP and GMDH, a host of techniques on the ISBSG dataset has been tested. The proposed GP- GMDH and GMDH-GP hybrids outperformed all other stand-alone and hybrid techniques. It is concluded that the GPGMDH or GMDH-GP model is the best model among all other techniques for software cost estimation.

Mohamad Mahdi Askari and Vahid Khatibi Bardsiri (2014) [4] for the prediction of software defects used artificial neural network inorder to better the generalization capability of the algorithm. Further support vector machine technique was used along with the learning algorithm and evolutionary technique. Thus this led to the maximization classification margin and prevented overfitting problem. This algorithm was tested with eleven machine learning models from NASA datasets. The conclusion drawn was that it provided better accuracy and precision than other models.

Mrs.Agasta Adline, Ramachandran. M(2014) [5] Predicting the fault-proneness of program modules when the fault labels for modules are unavailable is a challenging task frequently raised in the software industry. They attempted to predict the fault–proneness of a program modules when fault labels for modules are not present. Supervised techniques like Genetic algorithm based software fault prediction approach for classification has been proposed.

Xiaoxing Yang, et.al. (2014) [6] Used the rank performance optimization technique for software forecasting model development. For this rank to learning approach was used. The model was developed on previous work and was later studied for improving the performance of the model. . The work includes two aspects: one is a novel application of the learning-to-rank approach to real-world data sets for software defect prediction, and the other is a comprehensive evaluation and comparison of the learning-to-rank method against other algorithms that have been used for predicting the order of software modules according to the predicted number of defects. This study shows that the effect of optimization of the model performance using rank to learning approach truly improve the prediction accuracy.

III. METHODOLOGY

Fuzzy logic is conceptualized as a generalization of classical logic. modern fuzzy logic was developed by Lotfi Zadeh [8] within the mid-1960s to model those issues during which inaccurate data should be used or in which the rules of inference are developed during a} very general means making use of diffuse categories [8]. In fuzzy logic, that is additionally generally known as diffuse logic, there aren't simply 2 alternatives however a full continuum of truth values for logical propositions. A proposition A will have the truth value zero.4 and its complement Ac the reality worth zero.5. in line with the kind of negation operator that's used, the 2 truth values should not be essentially add up to one. fuzzy logic features a weak association to probability theory. fuzzy logic doesn't have to be compelled to be even employing a probabilistic approach. The common route is to generalize the findings of multivalued logic in such some way on preserve a part of the algebraical structure [8]. A fuzzy set theory corresponds to fuzzy logic and therefore the semantic of fuzzy operators is understood employing a geometric model. The geometric visualization of fuzzy logic can provide us a touch on the potential reference to neural networks. fuzzy logic is used as an interpretation model for the properties of neural networks, moreover as for giving a additional precise description of their performance. fuzzy logic may be used to specify networks directly while not having to apply a learning algorithmic rule. an expert in a certain field will generally produce an easy set of control rules for a dynamical system with less effort than the work concerned in training a neural network. A classical example proposed by Zadeh to the neural network community is developing a system to park a car. it's simple to formulate a collection of fuzzy rules for this task, however it's not immediately obvious the way to build a network to try and do the same nor how to train it. fuzzy logic is currently being used in several product of business and consumer electronics for which a decent system is ample and wherever the question of optimal control doesn't essentially arise.

The process of fuzzy logic is explained in algorithmic rule 1: foremost, a crisp set of input data ar gathered and converted to a fuzzy set using fuzzy linguistic variables, fuzzy linguistic terms and membership functions. This step is understood as fuzzification. Afterwards, an inference is created supported a collection of rules. Lastly, the ensuing fuzzy output is mapped to a crisp output using the membership functions, within the defuzzification step. So as to exemplify the usage of a FLS, consider an air conditioner system controlled by a FLS. The system adjusts the temperature of the room in line with this temperature of the room and therefore the target value. The fuzzy engine sporadically compares the room temperature and therefore the target temperature, and produces a command to heat or cool the room.

IV. PROPOSED MODEL

The Proposed model for software defect prediction is based on Adaptive neuro fuzzy inference system. For this NASA's Metrics Data Program (MDP) containing software metric data and error data at the function/method level has been used. The process of fuzzy logic is explained in previous section. For the development of the model a crisp set of input data are gathered and converted to a fuzzy set using fuzzy linguistic variables, fuzzy linguistic terms and membership functions. This step is understood as fuzzification. Afterwards, an inference is created supported a collection of rules. Lastly, the ensuing fuzzy output is mapped to a crisp output using the membership functions, within the defuzzification step.

A. Algorithm for Proposed Model:

- Step 1: Define Input /Output variables as software attributes /number of defects.
- Step 2: Select the Significant attributes from all inputs.
- Step3: Generates Fuzzy variables Membership function for each input and output.
- Step4: Perform clustering of input data at particulat specified raddi of influence.
- Step 5 : Perform fuzzification of input data.
- Step 6: Define Parameters of Membership function.
- Step 7: Design the fuzzy rules for mapping of output to input variables.
- Step 8: Evaluates fuzzy output for input data.
- Step 9: Check the MSE of result.
- Step 10: If performance satisfies stopping criteria stop the generation of FIS system else repeat step 3 to 9.
- Step 11: Generate the modified FIS system using training data by ANFIS.
- Step 12: Generate output for testing data using ANFIS based modified FIS system.

In the present work ANFIS Network Structure model consisting of one input layer with five input variables and an output layer consisting of weld bead width as the output variable. This is shown in Fig. 1 below.

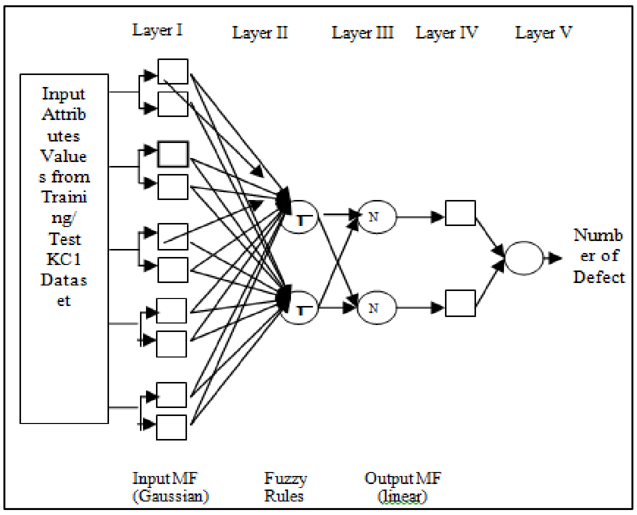

Fig. 1: Block structure of Defect Prediction Model using ANFIS

V. ILLUSTRATION OF PROPOSED MODEL

ANFIS model having twenty input variables are trained and tested by ANFIS method and their performances compared and evaluated based on training and testing data. The best fit model structure is determined according to criteria of performance evaluation. The performances of the ANFIS model are shown in Fig. 2 & 3 and their best RMSE values based on radius of influence r=0.5,0.75 and 1, both for the r=0.75 training and testing data are 0.01 and 13.25 respectively (Table 1 below).

Training Data	0.014	0.0101	1.844
Testing Data	25.197	13.256	31.372
Overall Data	15.518	8.164	19.2

Table 1: RMSE Values for Datasets after using ANFIS

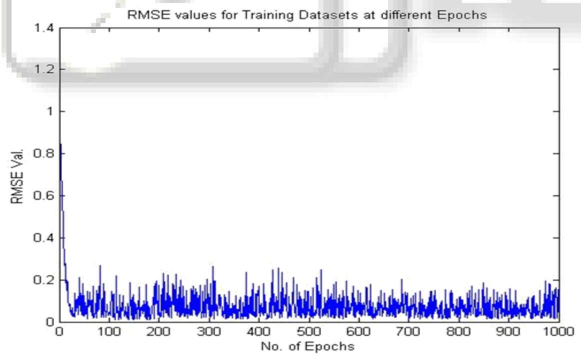

Fig. 2: RMSE Plot of Training Datasets during ANFIS Training

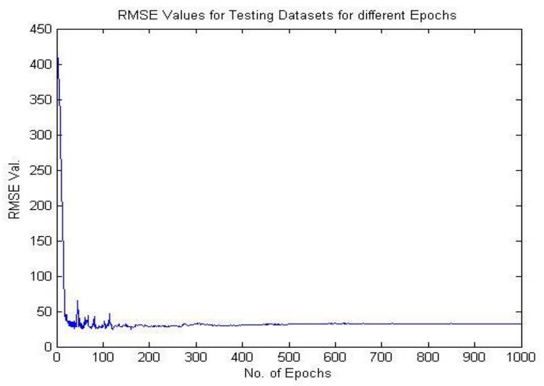

Fig. 3: RMSE Plot of Testing Datasets during ANFIS Training

Further in order to determine the capability and effectiveness of the model to predict the software error values MRE has been used. MRE is an indication of the average deviation of the predicted values from the corresponding measured data and can provide information on long term performance of the models; the lower MRE the better is the long term model prediction. A positive MRE value indicates the amount of overestimation in the predicated software error and vice versa.

The MRE of training and testing data sets for software errors are shown in fig. 4, 5 and 6 below for different radius of influence values.

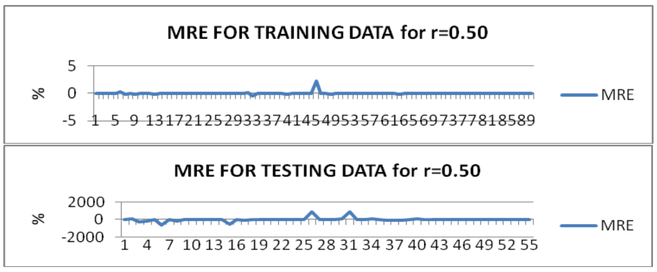

Fig. 4: Magnitude of Relative Error for r=0.50

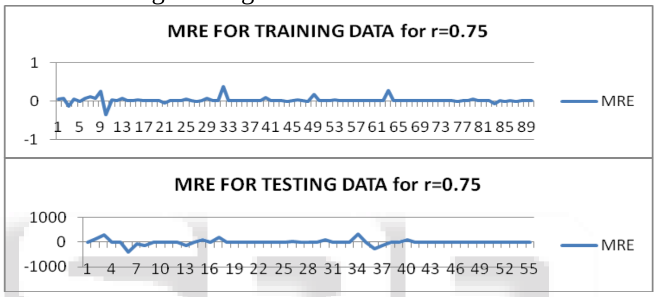

Fig. 5: Magnitude of Relative Error for r=0.75

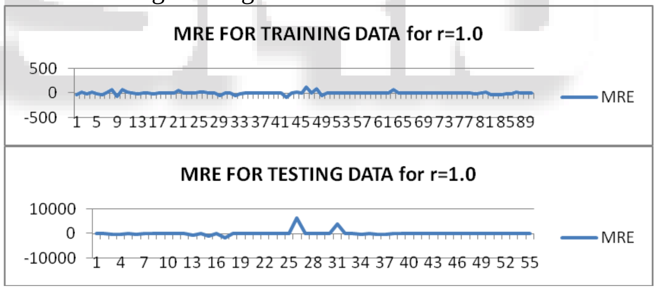

Fig. 6: Magnitude of Relative Error for r=1.0

MMRE for diff. radius of influence (r)			
	r=0.5	**r=0.75**	**r=1.0**
Training Data	0.0181	0.0144	3.396
Testing Data	5.63	2.47	120.53

Table 2: MMRE values corresponding to various radius of influence

The mean MRE for software error prediction of training data and testing data for different values of radius of influence are given in table 8 above. It is an indication of deviation of the predicted values from the corresponding measured data and can provide information on long term performance of the models. The lower deviation, the better is the long term model prediction. A positive value indicates the amount of overestimation in the predicted values and vise-versa. Here also the MMRE value for r=0.75, both for training and testing datasets are the least.

VI. CONCLUSION

The objective in the construction of models of software error prediction is to use measures that may be obtained relatively early in the software development life cycle to provide reasonable initial estimates of quality of an evolving software system. For this use of artificial intelligence technique, viz. ANFIS for the development of software defect prediction model is a very appropriate technique because predicting the defective modules in a software system prior to project deployment is a very crucial activity, since it leads to a decrease in the total cost of the project and an increase in overall project success rate. Defect prediction will give one more chance to the development team to retest the modules or files for which the defectiveness probability is high. In the present work it has been observed that the radius of influence effects the subtractive-clustering performance and at proper selection of parameters at r=0.75 we have obtained very low prediction error.

Note: For the development and illustration of proposed model NASA's Metrics Data Program (MDP) containing software metric data and error data at the function/method level has been used.

ANNEXURE-1

A. *Procedure of the Work Carried Out:*

1) Initiate MATLAB.
2) Load data. For this NASA's Metrics Data Program (MDP) containing software metric data and error data at the function/method level has been used.
3) Divide the data into training and testing datasets using Matlab commands.
4) Start ANFIS Editor using commands.
5) Load training data into ANFIS editor.
6) Generate Fuzzy Inference System (FIS) using Substractive clustering algorithm.
   – Input Selection:- Number and type of input / output membership functions.
7) ANFIS Training
   – Optimization method selection: Error tolerence, no. of epochs.
8) ANFIS Testing
   – Plot ANFIS output against Observed training and testing data.
9) Record the plot of training and testing datasets.
10) ANFIS analysis based on RMSE i.e. RMSE < RMSE ref ?
11) NO, goto step 5 and repeat step 6 to 10, else
12) END.

ANNEXURE-2

A. *Program for Defect Prediction Model:*

1) *GENFIS1*

```
shufflingtrgdata=data(randperm(145),1:22);
a = shufflingtrgdata;
a=data;
A=a(:,1);
B= a(:, 2);
C= a(:, 3);
D= a(:, 4);
E= a(:, 5);
F= a(:, 6);
G= a(:, 7);
H= a(:, 8);
I= a(:, 9);
J= a(:, 10);
K= a(:, 11);
L= a(:, 12);
M= a(:, 13);
N= a(:, 14);
O= a(:, 15);
P= a(:, 16);
Q= a(:, 17);
R= a(:, 18);
S= a(:, 19);
T= a(:, 20);
U= a(:, 21);
V= a(:, 22);
a)      >> % Prepare Training Data
trn_data(:, 1) = A(1:90);
trn_data(:, 2) = B(1:90);
trn_data(:, 3) = C(1:90);
trn_data(:, 4) = D(1:90);
trn_data(:, 5) = E(1:90);
trn_data(:, 6) = F(1:90);
trn_data(:, 7) = G(1:90);
trn_data(:, 8) = H(1:90);
trn_data(:, 9) = I(1:90);
trn_data(:, 10) = J(1:90);
trn_data(:, 11) = K(1:90);
trn_data(:, 12) = L(1:90);
trn_data(:, 13) = M(1:90);
trn_data(:, 14) = N(1:90);
trn_data(:, 15) = O(1:90);
trn_data(:, 16) = P(1:90);
trn_data(:, 17) = Q(1:90);
trn_data(:, 18) = R(1:90);
trn_data(:, 19) = S(1:90);
trn_data(:, 20) = T(1:90);
trn_data(:, 21) = U(1:90);
trn_data(:, 22) = V(1:90);
b)      >> % Prepare Checking Data
chk_data(:, 1) = A(91:145);
chk_data(:, 2) = B(91:145);
chk_data(:, 3) = C(91:145);
chk_data(:, 4) = D(91:145);
chk_data(:, 5) = E(91:145);
chk_data(:, 6) = F(91:145);
chk_data(:, 7) = G(91:145);
chk_data(:, 8) = H(91:145);
chk_data(:, 9) = I(91:145);
chk_data(:, 10) = J(91:145);
chk_data(:, 11) = K(91:145);
chk_data(:, 12) = L(91:145);
chk_data(:, 13) = M(91:145);
chk_data(:, 14) = N(91:145);
chk_data(:, 15) = O(91:145);
chk_data(:, 16) = P(91:145);
chk_data(:, 17) = Q(91:145);
chk_data(:, 18) = R(91:145);
chk_data(:, 19) = S(91:145);
chk_data(:, 20) = T(91:145);
chk_data(:, 21) = U(91:145);
chk_data(:, 22) = V(91:145);
fismat = genfis1(trn_data,2,'gaussmf');
fismat = genfis1(trn_data,2,'gbellmf');
XIN=[trn_data(:,1) trn_data(:,2) trn_data(:,3) trn_data(:,4)
trn_data(:,5) trn_data(:,6) trn_data(:,7) trn_data(:,8)
trn_data(:,9) trn_data(:,10) trn_data(:,11) trn_data(:,12)
trn_data(:,13) trn_data(:,14) trn_data(:,15) trn_data(:,16)
trn_data(:,17) trn_data(:,18) trn_data(:,19) trn_data(:,20)
trn_data(:,21)]
```

2) XOUT=[trn_data(:, 22)]

```
>> fismat = genfis2(XIN, XOUT,0.5); showfis(fismat);
>> fismat = genfis2(XIN, XOUT,0.75); showfis(fismat);
>> fismat = genfis2(XIN, XOUT,1.0); showfis(fismat);
% The initial MFs for training are shown in the plots.
for input_index=1:21,
subplot(3,7,input_index)
[x,y]=plotmf(fismat,'input',input_index);
plot(x,y)
axis([-inf inf 0 1.2]);
xlabel(['Input ' int2str(input_index)]);
end
>>[fismat1, error1, ss, fismat2, error2]=anfis(trn_data,
fismat, 1000, [], chk_data);
% The output MFs for training are shown in the plots.
for input_index=1:21,
subplot(3,7,input_index)
[x,y]=plotmf(fismat1,'input',input_index);
plot(x,y)
axis([-inf inf 0 1.2]);
xlabel(['Input ' int2str(input_index)]);
end
index=1:90
input=[trn_data(:,1:21)];
anfis_output1=evalfis(input, fismat1)
index=91:145
input=[chk_data(:,1:21)];
anfis_output2=evalfis(input, fismat2)
plot(1:55,chk_data(:,22), 1:55,anfis_output2)
legend('Actual','Predicted')
plot(1:90,trn_data(:,22), 1:90,anfis_output1)
legend('Actual','Predicted')
```

References

[1] "Fuzzy Logic Toolbox", MATLAB version R2012a.
[2] JANG, J-S. R., "ANFIS-Adaptive-Network Based Fuzzy Inference System", IEEE Transactions on Systems, Man and Cybernatics, 23(3), pp 665-685, 1993.
[3] http://mdp.ivv.nasa.gov/mdp_glossary.html.
[4] Mohamad Mahdi Askari and Vahid Khatibi Bardsiri (2014), "Software Defect Prediction using a High Performance Neural Network", International Journal of Software Engineering and Its Applications Vol. 8, No. 12 (2014), pp. 177-188. 14.
[5] Mrs.Agasta Adline, Ramachandran. M(2014), "Predicting the Software Fault Using the Method of Genetic Algorithm", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Special Issue 2,, pp 390-398.
[6] Xiaoxing Yang, et.al. (2014), IEEE TRANSACTIONS ON RELIABILITY, This article has been accepted for inclusion in a future issue of this journal.
[7] Manu Banga, "Computational Hybrids Towards Software Defect Predictions", International Journal of Scientific Engineering and Technology Volume 2 Issue 5, pp: 311-316, 2013.
[8] Xiao-dong Mu, Rui-hua Chang, Li Zhang, "Software Defect Prediction Based on Competitive Organization CoEvolutionary Algorithm", Journal of Convergence Information Technology (JCIT) Volume7, Number5, 2012.