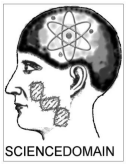
British Journal of Mathematics & Computer Science
19(2): 1-12, 2016; Article no.BJMCS.29644
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org



Defect Prediction Framework Using Adaptive Neuro-Fuzzy Inference System (ANFIS) for Software Enhancement Projects

Vipul Vashisht[1*], Manohar Lal[1] and G. S. Sureshchandar[2]

[1]SOCIS, IGNOU, New Delhi, India.
[2]ASQ India Pvt Ltd., Chennai, India.

Authors' contributions

This work was carried out in collaboration among all authors. Author VV conceived and designed the work and wrote the first draft of the manuscript. Authors ML and GSS helped perform the analysis of study with constructive discussions. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJMCS/2016/29644
Editor(s):
(1) Dariusz Jacek Jakóbczak, Chair of Computer Science and Management in this Department, Technical University of Koszalin, Poland.
(2) Tian-Xiao He, Department of Mathematics and Computer Science, Illinois Wesleyan University, USA.
Reviewers:
(1) Adeel H. Suhail, Baghdad Oil Training Institute (BOTI), Iraq.
(2) Shaik Nafeez Umar, ANGRAU, India.
Complete Peer review History: http://www.sciencedomain.org/review-history/16582

Original Research Article

Received: 22nd September 2016
Accepted: 13th October 2016
Published: 17th October 2016

Abstract

Software Defect Prediction is the process of forecasting the defect count during various phases of software development life cycle. Defect prediction is vital to successful software project execution since the output is used to proactively plan defect prevention activities. During initial phases of software development life cycle, prediction is quite challenging due to the presence of uncertainty in input parameters, which constitute major component of estimated effort. Multiple attempts have been made by researchers in past to design an appropriate defect prediction model but so far none has found widespread adoption in software industry. In this communication, Adaptive Neuro-fuzzy Inference system (ANFIS) approach has been proposed for designing a defect prediction model. In order to achieve complexity reduction and to increase model adoption, an easy-to-use graphical user interface is designed. The proposed ANFIS based model makes use of organization's historical projects' data for building the model. The model provides a defect range (minimum, maximum) as a prediction output. The effectiveness and superiority of proposed ANFIS model is demonstrated through analysis of results achieved.

**Corresponding author: E-mail: vipulvashisht@gmail.com;*

Keywords: Software defect; software defect prediction model, neural network (NN); quality management; fuzzy logic; adaptive Neuro-fuzzy inference system (ANFIS).

1 Introduction

In view of the well-known fact of our being more and more dependent on the software-based technology, there is a growing demand for developing reliable, affordable and faster software systems. This fact is setting higher expectations for the organizations to improve the overall product quality. In IT industry, defects detected in software are most commonly considered as one of the major metrics for quality of deliverables. A higher residual defect leakage could not only result in budget overrun but can also result in impacting the overall customer satisfaction index. During the signing of large engagement contracts, more and more customers are enforcing financial penalty clauses to counter damages due to residual defect leakage from supplier end. Hence planning for defect prevention is considered vital for the organization. It is always beneficial to focus on preventing defects in software development life cycle rather than expending effort in fixing defects found by end users while software has gone live in production.

In view of the increasing complexity in systems, it is almost impossible to deliver software with zero production defects. The occurrence of defects is considered almost inevitable and is the one of the significant contributors to rise in overall project costs due to defect fixing effort. In order to prevent defects from occurrence and to detect the leakage in the initial phases, most organizations plan to implement defect prediction model. The Capability Maturity Model Integration (CMMI) is a globally-adopted capability improvement framework that advocates use of appropriate defect prediction model as one of the high maturity practices for process improvement under Quantitative Project Management process area at Level 4. In order to achieve CMMI level 5 certification, organizations need to showcase the benefits achieved by implementing quantitative techniques like statistical process control charts and defect prediction model [1,2]. Defect count prediction enables project manager (PM) to take data driven decisions. Based on outcome of model, IT organizations can perform contingency planning for areas that need necessary attention and investment [3]. Prevention activities may include setting up multiple review channels, usage of automation tools and performing process audits.

In the recent past, there has been an increased usage of computational intelligence (Cl) technologies such as fuzzy logic, neural networks, and ANFIS, to solve issues in the field of software engineering. The Cl technologies have been found to be particularly successful in solving problems such as effort and defect prediction that arise as a result of measurement which is not precise and inaccurate [4,5]. In this communication, ANFIS approach has been proposed for designing a defect prediction model for software enhancement projects.

Next section describes about software enhancement methodology.

1.1 Software enhancement methodology

Enhancement life cycle methodology aims at achieving improvements to existing software in terms of functionality / technologies. The enhancement methodology is most suited to the situations in which there are constantly changing customer requirements, involving partially or fully completed projects. The changes may involve functionality or technology upgrade. Once the software is delivered; the maintenance and enhancement of application software consume a major portion of the total life cycle resources that includes cost of the system [6,7]. Most legacy software systems do not have well documented requirements, which pose a great challenge for project manager to plan for software enhancement to existing product. Most of the times, the person who would make changes to code is different from the person who initially authored the code. In such cases, defect prevention through defect prediction becomes all the more vital. It has been suggested that the ongoing maintenance of legacy software is becoming more difficult year by year since software updates gradually changes the original architecture of the applications [8].

It has also been observed that most research work done in past regarding software defect prediction is focused on development projects, instead of software enhancements or software maintenance projects [9]. To address this issue, a defect prediction model for software enhancement projects is being proposed, justifying the investigations reported through this communication. In this communication, three distinct phases (Requirement Gathering, Construction and Testing) of projects regarding software enhancement life cycle are considered for designing the model. In view of the fact that effort and duration for analysis phase and design phase play a minimal role, the defect prediction for these phases is not taken into consideration.

In the next section, ANFIS overview is provided followed by literature review, section IV discusses proposed framework followed by results discussion and conclusion.

2 Adaptive Neuro Fuzzy Inference System (ANFIS)

In the last few decades, study in the field related to artificial neural networks and fuzzy inference systems has been a major area of consideration, especially in the areas involving specific type of uncertain knowledge. Fuzzy logic based systems have the ability to represent, and to draw inferences regarding comprehensive linguistic knowledge, though vague yet understandable to human experts [10]. Nevertheless, fuzzy systems lack ability to obtain and tune the rules automatically. While, neural networks based systems are known to be adaptive and can be easily trained and tuned from provided data set. Fuzzy systems and neural-networks are known to be complementary paradigms for addressing such complex problems; hence, it is natural to combine these technologies to create hybrid systems [11].

An adaptive Neuro-fuzzy inference system (ANFIS) technique integrates both neural networks and fuzzy logic principles in a single framework. ANFIS makes use of training algorithm supported by NN architectures [12,13]. This technique got developed in the early 1990s and makes use of Takagi–Sugeno fuzzy model which is known to be more compact and computationally efficient than Mamdani model. In this approach, ANFIS inference system corresponds to a set of fuzzy IF–THEN rules that have learning capability with appropriate membership functions to generate the result as stipulated input output pairs [14,15].

In a Sugeno model, a typical fuzzy rule is described as:

$$\boldsymbol{if\ x\ is\ in\ A\ and\ y\ is\ in\ B\ then\ z = f(x, y)} \tag{1}$$

In equation (1), A and *B* are fuzzy sets in the antecedent, and f is a crisp function mapping the ordered pair (x, y) to z the consequent. The function f is a polynomial function which describes the model output within fuzzy region specified by the antecedent of the rule.

In Fig. 1, a two-input first-order Sugeno Fuzzy Model with two rules is shown with corresponding ANFIS Architecture presented in Fig. 2.

Rule 1: if $x\ is\ A1$ and

$$y\ is\ B1, \text{ then } (f1 = p1x + q1y + r1) \tag{2}$$

Rule 2: if $x\ is\ A2$ and

$$y\ is\ B2\ , \text{then} (f2 = p2x + q2y + r2) \tag{3}$$

The five layers of ANFIS architecture are explained in Table 1 (Jang, 1993).

Table 1. Description of layers in ANFIS architecture

Layer 1	Every node i in this layer is an adaptive node with a node function $O_i^1 = \mu_{A_i}(x)$ (4) Where x is the input to node *i* and A_i is a linguistic label associated with this node. In other words, O_i^1 is the membership grade of a A_i and it specifies the degree to which the given input x satisfies the quantifier A_i . Here the membership function for A can be any appropriate parameterized membership function: $\mu_{A_i}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}}$ (5) $\mu_{A_i}(x) = \exp\left\{-\left(\frac{x-c_i}{a_i}\right)^2\right\}$ (6) Where $\{ai, bi, ci\}$ is the parameter set referred as *premise parameters.*
Layer 2	In this layer every node is a fixed node whose output is the product of all the incoming signals. $O_i^2 = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y)$, $i = 1, 2$ (7) Here, output of each node represents the firing strength of a fuzzy rule.
Layer 3	In this layer every node is a fixed node labeled N. The *i*th node calculates the ratio of the *i*th rule's firing strength to the sum of all rules' firing strengths, output is known as *normalized firing strengths*: $O_i^3 = \overline{w}_i = \frac{w_i}{w_1+w_2}$, $i = 1, 2$ (8)
Layer 4	Every node i in this layer is a square node with a node function: $O_i^4 = \overline{w}_i f_i = \overline{w_i}\,(p_i x + q_i y + r_i)$, $i = 1, 2$ (9) Where $\overline{w}_i$ is a normalized firing strength from layer 3 , and $\{p_i , q_i, r_i\}$ is the parameter set of this node. Parameters in this layer are referred to as *consequent parameters*.
Layer 5	The single node in this layer is a fixed node labeled $\sum$, which computes the overall output as the summation of all incoming signals: $O_i^5 = overall\ output = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$ (10)

The task of the learning algorithm for this architecture is to tune all the modifiable parameters, namely $\{a_i\ b_i\ c_i\}$ and $\{p_i\ q_i\ r_i\}$ to make the ANFIS output match the training data. When the premise parameters a_i b_i and c_i of the membership function are fixed, the output of the ANFIS model can be written as:

$$f = \frac{w_1}{w_1+w_2} f_1 + \frac{w_2}{w_1+w_2} f_2 \quad (11)$$

$$f = \overline{w}_1 f_1 + \overline{w}_2 f_2 \quad (12)$$

$$f = (\overline{w}_1\, x)p_1 + (\overline{w}_1\, y)q_1 + (\overline{w}_1\,)r_1 + (\overline{w}_2\, x)p_2 + (\overline{w}_2\, y)q_2 + (\overline{w}_2\,)r_2 \quad (13)$$

Next section describes the related literature review.

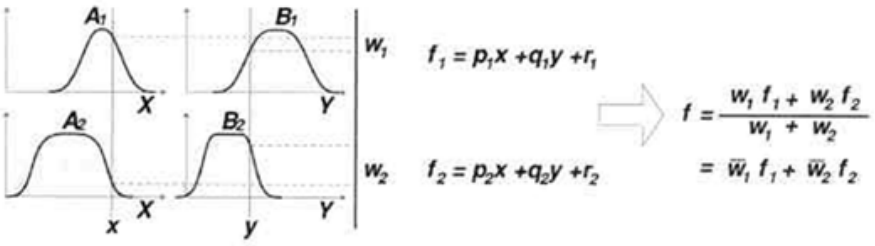


Fig. 1. A two-input first-order Sugeno Fuzzy model with two rules [14]

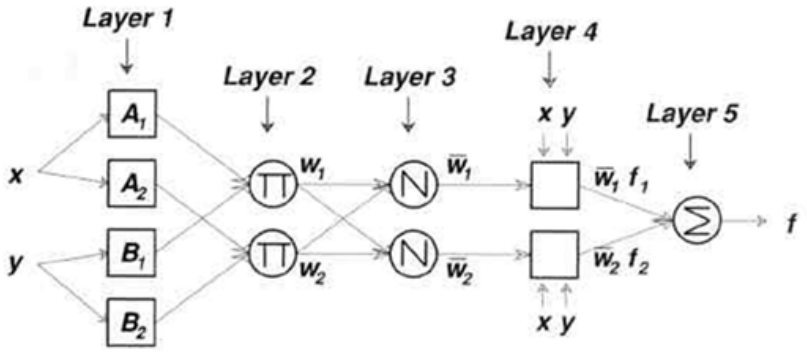


Fig. 2. Equivalent ANFIS architecture [14]

3 Related Literature Review

Few of the areas where ANFIS approach, which is used for this investigation, has been successfully used:

- Prediction of software maintenance effort of commercial software systems has been evaluated using various techniques and results of ANFIS were observed to be the best [16].
- ANFIS has been used for predicting the reliability of the software using attributes such as size of software, number of failures and Total time [17].
- ANFIS has been used for Tamil speech word recognition system [18].
- A method based on ANFIS has been used to evaluate the software reliability. The model makes use of the reliability data of one software project as an input data, and use the prediction of reliability as output data [19].
- Neuro-fuzzy technique has been used for estimating software development time. The results showed that Neuro-fuzzy system is much better than fuzzy logic and neural network, when used separately [20].
- ANFIS based technique has been implemented for predicting software effort. The same was compared with neural network based technique and was found to be performing better [21].
- A comparative analysis using ANFIS was done to predict level of impact of faults in NASA's public domain defect dataset coded in Perl programming language. The accuracy value of trained Neuro-fuzzy system was found to be 93.33% [22].

- ANFIS has been successfully applied for predicting software change-prone classes in the early phases of software development [23].
- ANFIS has been used for estimating software effort. Comparison of various membership functions was done and the results showed the trapezoidal membership function was better compared to other kind of membership functions [24].

In the next section proposed framework is discussed.

4 The Proposed Framework

ANFIS is a hybrid AI technique, which combines best features of fuzzy logic and parallel processing neural networks. Since ANFIS possesses fast convergence and has more accuracy than back propagation neural network, it is considered to be a universal estimator [25]. The experiments reported here involve data sets taken from 50 real projects from a large software organization. Out of this dataset, 40 projects are used for training the model and the rest 10 projects data is used for validating/testing the trained model in MATLAB environment. Considering the fact that the degree to which historical data is similar to future data determines the degree to which the model predicts the future events; the data has been segregated appropriately. Table 2 provides information regarding the structure of the adaptive Neuro-fuzzy based inference system.

Table 2. Configuration settings for ANFIS based modeling

S. no.	Parameters	Description
1	Training Samples	40 samples of 4 elements (4 X 40) [Production, Review, Rework & Prevention]
2	Target Samples	40 samples of 1 element (1 X 40) [Defect]
3	FIS (Fuzzy Inference System) Method	Grid Partitioning
4	Number of Membership Function, MF	3, 3, 3, 3 (for each of the four inputs)
5	MF Type	Gaussian, Linear
6	Number of rules	81
7	Training Optimization Method	Hybrid
8	Average Testing error at epoch 3	0.28771 (Requirement) 2.2389 (Construction) 1.9647 (Testing)
9	Defuzzification	Wtaver, (Weighted Average)

ANFIS makes use of a hybrid learning algorithm for parameter identification of Sugeno type FIS. Sugeno model is computationally efficient, works well with linear optimization and adaptive techniques. It has guaranteed continuity of the output surface and is well suited to mathematical analysis [26].

Fig. 3 provides ANFIS structure for the Requirement Gathering phase of software enhancement projects. The structure would be the same for other two phases of enhancement life cycle. The branches shown in the ANFIS architecture are color coded, which characterize the fuzzy rules used.

The *anfis* functionality in Fuzzy Logic Toolbox accomplishes the membership function parameter adjustment. This adjustment allows fuzzy systems to learn from the data being modeled. ANFIS uses two datasets, one for training and the other for testing. In our case, first dataset of 40 projects is used for training and other dataset of 10 projects is used for testing the trained model. Both datasets are from the historical projects, which were completed at the organization. Considering the stability of system and minimum training error, the number of epochs for training purpose is set to 3. Each of the three SDLC phases, four inputs (viz. Production effort, Review effort, Rework effort and Prevention effort) has Gaussian membership function with 3 membership function for each input.

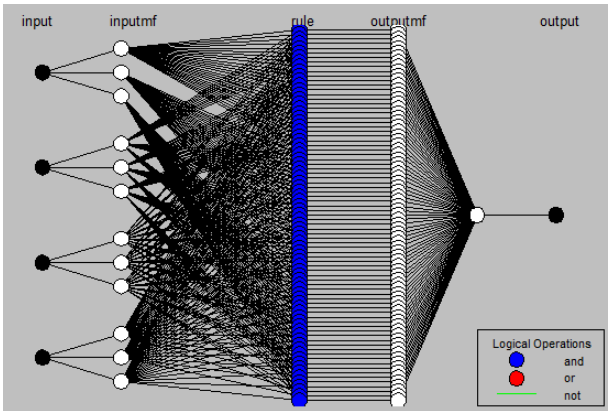


Fig. 3. ANFIS structure for requirement gathering phase with four inputs

The fuzzy inference system is shown in Figs. 4 and 5 depicts membership function for Production effort, which is one of the input parameter.

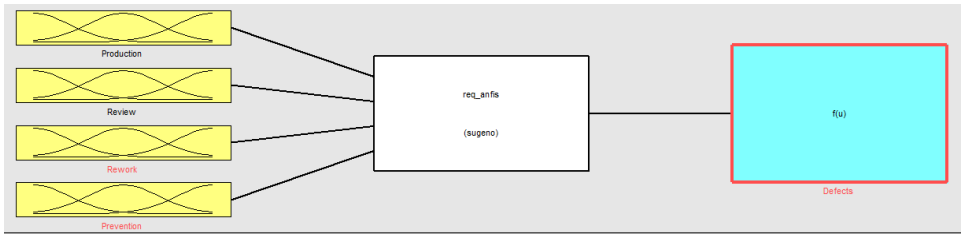


Fig. 4. Fuzzy inference system

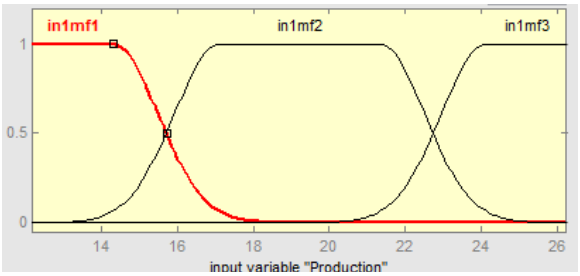


Fig. 5. Membership functions for inputs (Production effort)

Gaussian combination membership function is represented as:

$$y = gauss2mf(x, [sig1\ c1\ sig2\ c2]) \tag{14}$$

The Gaussian function depends on two parameters sig and c as given by:

$$f(x; \sigma, c) = e^{\frac{-(x-c)^2}{2\sigma^2}} \tag{15}$$

The function **gauss2mf** is a combination of two of these two parameters. The first function, specified by *sig1* and *c1*, determines the shape of the left-most curve. The second function specified by *sig2* and *c2*

determines the shape of the right-most curve. Whenever $c1 < c2$, the **gauss2mf** function reaches a maximum value of 1. Otherwise, the maximum value is less than one. The Gaussian function has been used for specifying the fuzzy sets due to their smoothness and concise notation property and having the advantage of being smooth and nonzero at all points.

The parameters are listed in the order:

$$[sig1, c1, sig2, c2] \tag{16}$$

The model is validated by using remaining 10 data sets. For the purpose of validation, the training error (refer Fig. 6) is defined as the difference between the training data output value and the output of the fuzzy inference. The average testing error achieved at epoch 3 is 0.28771 for Requirement Gathering phase, 2.2389 for Construction phase and 1.9647 for Testing phase. The lower values of testing error show better quality of prediction. The key point to observe here is that by emulating the fuzzy rules in neural network architecture, the network can now be trained with standard back propagation methods in response to training patterns. This means that the shape of the membership functions and the strength of the connection for the rules can be adjusted and learned. When the training is completed, the neural network can simply be converted back to fuzzy rules, if desired [27].

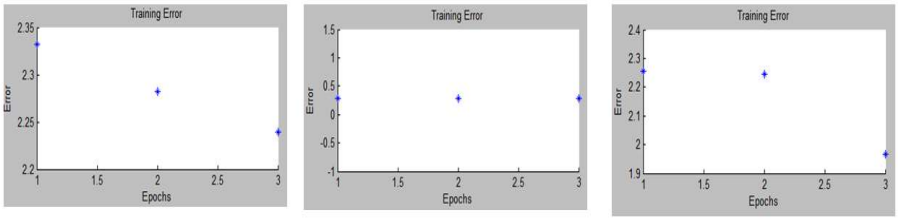


Fig. 6. Training error for three phases

4.1 Graphical user interface development

The execution of the defect prediction model could become a deterrent if the operation is complex and is not end-user friendly. In this communication, to facilitate the operation, Matlab toolbox is used to design and develop a graphical user interface to input the data. The tool uses only two windows, the first for identifying the SDLC phase for which prediction is required and the second to input the planned effort for activities. Output of the defect predictions is displayed in a separate window (refer Fig. 7).

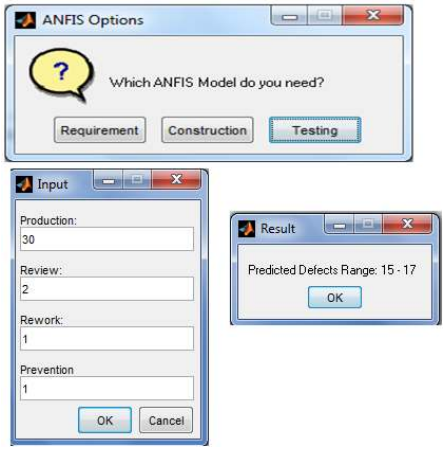


Fig. 7. Defect prediction system UI

The functioning of GUI is as follows:

- For any new software enhancement project, the end user will provide the inputs to the GUI. The inputs include planned efforts in person days for that specific SDLC phase. Apart from production effort, the planned review effort, planned prevention effort and the planned rework effort are also required as a feed to the model.
- Model would process the inputs and predict the defect count for that particular SDLC phase. The model would provide the defect count range , that includes minimum and maximum number of predicted defects.

The defect forecast for other two phases is done using the same steps as listed above.

5 Comparison of Results

The performance of the defect prediction model for software enhancement projects is validated by comparing the trend of the outputs; one obtained through ANFIS based model and the other being the actual output. The defect trend chart (refer Fig. 8) shows that, in most cases, actual defects follows the ANFIS prediction trend.

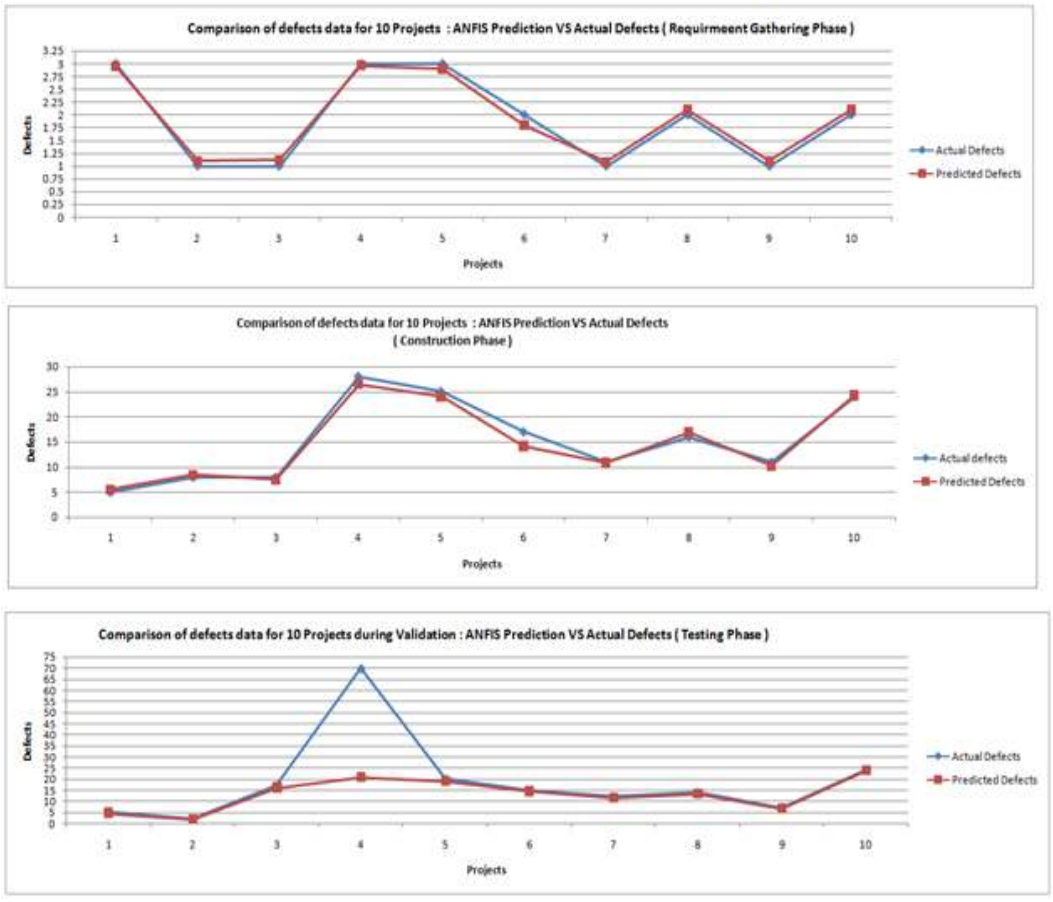


Fig. 8. Comparative trend of defects predicted by ANFIS Vs actual defects

The overall accuracy of ANFIS based defect prediction model is calculated based on the count of predicted defects and actual defects. The overall accuracy on the validation data of 10 projects is 91.8% while the overall accuracy considering 50 data sets comes around 88% (refer Fig. 9). The accuracy of validation data sets of 10 projects during requirement gathering and construction phase is 93.4% which is comparable to 93.33% accuracy achieved on NASA's MDP (Metric Data Program) data repository [22].

	Requirement Phase			Construction Phase			Testing Phase		
Project #	Actual Defect Count	ANFIS Pedicted defects	Accuracy ANFIS	Actual Defect Count	ANFIS Pedicted defects	Accuracy ANFIS	Actual Defect Count	ANFIS Pedicted defects	Accuracy ANFIS
P1	3	2.8	93.1%	15	16.9	87.1%	15	13.3	88.6%
P2	4	3.4	84.7%	27	24.4	90.5%	26	23.2	89.2%
P3	3	2.7	88.5%	25	24.2	96.8%	22	22.1	99.5%
P4	1	1.3	67.1%	15	15.5	96.6%	13	12.7	97.7%
P5	1	1.0	99.9%	13	15.2	83.4%	9	7.8	86.4%
P6	1	1.3	74.8%	19	23.7	75.3%	15	15.3	97.7%
P7	1	0.7	70.8%	13	13.6	95.5%	11	12.0	90.5%
P8	1	1.0	99.6%	12	12.0	100.0%	10	10.1	99.1%
P9	4	3.7	91.4%	24	24.5	97.8%	26	20.0	77.1%
P10	1	1.2	78.7%	10	10.2	98.2%	6	7.1	81.7%
P11	1	0.8	83.5%	19	22.9	79.5%	25	23.0	91.9%
P12	1	0.8	76.1%	10	10.2	97.8%	5	7.6	48.4%
P13	1	0.8	78.8%	19	23.2	77.9%	18	20.2	87.8%
P14	1	1.7	25.9%	6	5.9	98.1%	10	8.4	83.7%
P15	1	1.1	91.7%	10	10.9	91.1%	13	12.9	99.2%
P16	2	1.7	82.9%	10	11.4	85.5%	6	8.9	50.9%
P17	4	3.9	96.7%	18	24.6	63.4%	15	15.1	99.6%
P18	2	1.8	91.7%	16	14.2	88.8%	10	10.2	98.3%
P19	2	2.4	80.0%	14	15.9	86.7%	12	12.5	96.2%
P20	1	1.7	29.4%	12	15.0	75.3%	11	10.5	95.8%
P21	3	3.2	93.2%	19	18.0	94.5%	13	12.8	98.4%
P22	2	1.7	85.8%	24	23.9	99.6%	18	16.2	90.1%
P23	3	2.6	86.7%	10	11.5	85.5%	8	7.7	96.6%
P24	2	1.7	86.9%	14	16.1	85.3%	15	15.8	94.8%
P25	5	5.3	94.2%	23	21.0	91.4%	19	18.7	98.5%
P26	2	2.3	87.0%	18	16.6	92.3%	12	11.5	95.6%
P27	2	1.9	96.5%	20	19.7	98.7%	11	13.8	74.7%
P28	2	3.1	47.4%	25	24.3	97.3%	24	23.8	99.4%
P29	1	0.9	88.9%	10	8.8	88.4%	7	7.6	91.4%
P30	1	1.0	99.0%	23	22.5	97.9%	24	23.8	99.2%
P31	1	1.6	38.4%	20	19.4	96.8%	14	14.1	99.0%
P32	2	2.1	96.5%	15	14.9	99.2%	12	11.2	93.0%
P33	2	1.6	81.4%	24	22.1	92.1%	22	22.8	96.6%
P34	2	1.8	90.6%	16	16.7	95.4%	9	14.1	43.6%
P35	2	1.6	80.4%	14	14.1	98.9%	20	14.1	70.3%
P36	1	0.9	90.5%	18	23.5	69.6%	18	17.0	94.3%
P37	1	0.8	83.4%	18	17.7	98.4%	17	15.6	91.9%
P38	1	1.4	55.8%	9	6.0	66.4%	16	15.2	95.2%
P39	1	0.7	73.3%	17	16.8	98.7%	16	15.6	97.2%
P40	2	1.8	88.0%	19	16.1	85.0%	13	12.0	92.1%
P41	3	2.9	98.0%	5	5.6	88.6%	5	4.7	94.0%
P42	1	1.1	90.0%	8	8.5	94.1%	2	1.9	95.0%
P43	1	1.1	88.0%	8	7.5	93.8%	17	16.1	94.9%
P44	3	3.0	98.7%	28	26.5	94.6%	70	20.7	29.6%
P45	3	2.9	96.7%	25	24.1	96.3%	20	19.0	95.0%
P46	2	1.8	90.0%	17	14.2	83.4%	15	14.5	96.3%
P47	1	1.1	91.7%	11	10.9	99.0%	12	11.5	95.5%
P48	2	2.1	95.0%	16	16.9	94.2%	14	13.3	95.0%
P49	1	1.1	90.0%	11	10.2	92.5%	7	6.8	96.9%
P50	2	2.1	95.0%	24	24.3	98.6%	24	23.8	99.4%

Fig. 9. Model accuracy calculation

6 Conclusions

The study was conducted to illustrate the potential effectiveness of ANFIS approach in respect of software defect prediction on data sets from software enhancement projects. The accuracy of validation with data sets from 10 projects during requirement gathering and construction phase is 93.4%. The results from experiments indicate that the proposed ANFIS based model has better defect prediction capability. The conclusions are based on investigations of software enhancement projects regarding a large software organization. In order to adapt the proposed prediction model to suit other software development methodologies like ERP, Agile, Production Support, etc, further effort is required.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Tamura S. Integrating CMMI and TSP/PSP: Using TSP data to create process performance models (No. CMU/SEI-2009-TN-033). Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst; 2009.

[2] Vashisht VV. Enhancing software process management through control charts. Journal of Software Engineering and Applications. 2014;7(2):87.

[3] McDonald M, Musson R, Smith R. The practical guide to defect prevention. Microsoft Press; 2007.

[4] Boetticher GD. Applying machine learners to GUI specifications in formulating early life cycle project estimations. In Software Engineering with Computational Intelligence. Springer US. 2003;1-16.

[5] Khoshgoftaar TM, ed. Software engineering with computational intelligence. Springer Science & Business Media. 2012;731.

[6] Lientz BP, Swanson EB, Tompkins GE. Characteristics of application software maintenance. Communications of the ACM. 1978;21(6):466-471.

[7] Canning G. The maintenance “iceberg”. EDP Analyzer. 1972;10(10):1–14.

[8] Jones C. The economics of software maintenance in the twenty first century, 2006. Performance Engineering to Enhance the Maintenance; 2006.

[9] Pigoski TM. Practical software maintenance: Best practices for managing your software investment. John Wiley & Sons, Inc; 1996.

[10] Sridhar M, Gill NS. Imperfection of domain knowledge and its formalization in context of design of robust software systems. Journal of Software Engineering and Applications. 2015;8(9):489.

[11] Aldair A, Wang W. FPGA based adaptive Neuro fuzzy inference controller for full vehicle nonlinear active suspension systems. International Journal of Artificial Intelligence & Applications (IJAIA). 2010;1(4):1-15.

[12] Nauck D. September. Neuro-fuzzy systems: Review and prospects. In Proceedings of Fifth European Congress on Intelligent Techniques and Soft Computing (EUFIT’97). 1997;1044-1053.

[13] Del Campo I, Echanobe J, Bosque G, Tarela JM. Efficient hardware/software implementation of an adaptive Neuro-fuzzy system. Fuzzy Systems, IEEE Transactions. 2008;16(3):761-778.

[14] Jang JSR. ANFIS: Adaptive-network-based fuzzy inference system. Systems, Man and Cybernetics, IEEE Transactions. 1993;23(3):665-685.

[15] Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control. Systems, Man and Cybernetics, IEEE Transactions. 1985;1:116-132.

[16] Kaur DA, Kaur K, Malhotra DR. Soft computing approaches for prediction of software maintenance effort. International Journal of Computer Applications. 2010;1(16).

[17] Bhardwaj S, Sinha A. An improved computational software reliability model using ANFIS. IJCA. 2015;114(16):7–9.

[18] Rojathai S, Venkatesulu M. Investigation of ANFIS and FFBNN recognition methods performance in Tamil speech word recognition. International Journal of Software Innovation (IJSI). 2014;2(2):43-53.

[19] Yuan D, Zhang C. Evaluation strategy for software reliability based on ANFIS. In Electronics, Communications and Control (ICECC), 2011 International Conference. IEEE. 2011;3738-3741.

[20] Marza V, Teshnehlab M. Estimating development time and effort of software projects by using a Neuro_Fuzzy approach. INTECH Open Access Publisher; 2009.

[21] Mewada KM, Sinhal A, Verma B. Adaptive Neuro-Fuzzy Inference System (ANFIS) based software evaluation. IJCSI International Journal of Computer Science. 2013;10(1):244-250.

[22] Ardil E. A soft computing approach for modeling of severity of faults in software systems. International Journal of Physical Sciences. 2010;5(2):74-85.

[23] Peer A, Malhotra R. Application of adaptive Neuro-fuzzy inference system for predicting software change proneness. In Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference. IEEE. 2013;2026-2031.

[24] Praynlin E, Latha P. Estimating development effort of software projects using ANFIS. In IJCA Proceedings on International Conference in Recent trends in Computational Methods, Communication and Controls (ICON3C 2012). Foundation of Computer Science (FCS). 2012;5.

[25] Jang JSR, Sun CT, Mizutani E. Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence; 1997.

[26] The mathworks. Neuro-Adaptive Learning and ANFIS. Available:http://in.mathworks.com/help/fuzzy/neuro-adaptive-learning-and-anfis.html

[27] Badiru AB, Cheung J. Fuzzy engineering expert systems with neural network applications. John Wiley & Sons. 2002;11.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/16582