International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 1 (2018) pp. 394-397
© Research India Publications. http://www.ripublication.com

Software Defect Prediction using Adaptive Neuro Fuzzy Inference System

Satya Srinivas Maddipati
Research Scholar, Department of Computer Science & Engineering,
Koneru Lakshmayya University, Vaddeswaram, Guntur District, Andhra Pradesh, India.
Orcid Id: 0000-0001-9608-9481

Dr. G Pradeepini
Professor, Department of Computer Science & Engineering,
Koneru Lakshmayya University, Vaddeswaram, Guntur District, Andhra Pradesh, India.
Orcid Id: 0000-0001-7757-6559

Dr. A Yesubabu
Professor, Department of Computer Science & Engineering,
Sir C R Reddy College of Engineering, Eluru, Andhra Pradesh, India.
Orcid Id: 0000-0001-8072-5252

Abstract

Software Defect Prediction is a major challenge in Software Development process, to reduce the cost of software implementation. Predicting Defective prone modules in software industry greatly reduces the software development cost. Most of the researchers applied various data mining techniques like Adaboost, Neural networks, Random Forest and support vector machines for software defect prediction datasets downloaded from NASA repositories. These datasets are imbalanced in nature. In this paper software defects are predicted using Adaptive Neuro Fuzzy Inference System(ANFIS). Initial Fuzzy Inference System(FIS) was derived using Subtractive Clustering method and then FIS was trained using hybrid learning rule. The performance of the classifier is measured in terms of AuC values for these imbalanced datasets. We compared the results of ANFIS with cost sensitive neural networks. The Receiver operating characteristics (ROC) curves are generated and presented in Result section. The ROC values of ANFIS are found satisfactory compared to cost sensitive Neural networks.

Keywords: Adaptive Neuro Fuzzy Inference System, Receiver operating characteristics, Software Defect Prediction, Subtractive Clustering, hybrid learning

INTRODUCTION

In the process of software development, predicting software defects plays a major role. Predicting a software defect in advance reduces the cost of software development and improves the quality of software product. There are various approaches for software defect prediction

1. Simple metric and defect estimation model : According to Akiyama, Number of defects depends on software metric, lines of code (LOC). He derived an equation No. of defects(N)=4.86+0.018*LOC. But LOC is not enough to capture software complexity.
2. Complexity metrics and fitting models: In 1976, McCube found cyclometric metrics for determining software complexity. Cyclometric complexity of a program V(G)= E – V + 2, here E is number of edges, V is number of vertices. In 1977, Halsted introduced Halsted complexity measures, which reflects implementation of algorithms in different languages. He observed that the number of defects depends on effort, which in turn depends on difficulty and volume. Defects(D)=$E^{2/3}$/3000. But the limitation of this model is it just fit the known data but not validated for new entries.
3. Regression model: Shen et al.'s empirical study showed that Linear regression model can be validated on actual new modules. He found Mean magnitude of relative error(MRE) between actual and predicted number of defects as 0.48. Munson et al. Applied Discriminative analysis using Logistic regression with Halsted and cyclometric complexity metrics and obtained accuracy of 92%.
4. Just in time Prediction model: A large scale empirical study of just in time quality assures 68% accuracy, 64% recall on 11 open source and commercial projects. The limitation of JIT model is practical validation difficult.
5. Practical Models: Chidamber & Kemerer introduced CK metrics which are Object oriented for software defect prediction. The metrics are weighted methods per class(WMC), Depth of inheritance tree (DIT) , Number of children (NOC), Coupling between objects(CBO) and Response for a class (RFC)
6. History metrics prediction models: History metrics do not extract particular program characteristics such as developer social network, Component network and anti pattern. It is not applicable for new projects and projects lacking in historical data.
7. Cross Project defect prediction: These models are applicable for new projects lacking in historical data.

The remainder of this paper is organized as follows. Section 2 discuss about Related work. Section 3 discuss methodology. In section 4 we discuss results and comparisons.

RELATED WORK

Faruk Arar et.al applied Artificial Neural Network (ANN) for constructing a model for Software Defect Prediction. He applied Artificial Bee Colony(ABC) for optimizing connection weights in ANN. Model was applied to five publicly available datasets from the NASA repository[1.] Jun Zheng et al, studied three cost sensitive algorithms for software defect prediction. The performances of the three algorithms are evaluated by using four datasets from NASA projects[2.] A comparison of soft computing algorithms for software defect prediction is done by Ertruk[3.] E Erturk applied Adaptive Neuro fuzzy inference system for software defect prediction. Rodriguez used datasets from PROMISE repository and applied feature selection and genetic algorithms for predicting defective modules[8]. Guo suggested the use of Random forest for predicting software modules[9]. Decision tree learners are used to predict the defect densities in SDP datasets[12].

METHODOLOGY

Adaptive Neuro Fuzzzy Inference System(ANFIS)

Generating Initial FIS

The problem with fuzzy inference system is identification of rules. Rules are generated either by using grid partitioning or subtractive clustering methods.

Grid Partitioning

Grid partitioning divides each input variables into sub intervals and forms the rule for each possible interval of input variables. For the variables with continuous values, this method generates a huge collection of rules which over fits the data. Hence this method is not suitable for the datasets consisting of continuous variables.

Subtractive Clustering

Subtractive clustering generates the fine tuned clusters for each input variable. A rule is generated for each cluster of input variables. Subtractive clustering algorithm considers each data point as a candidate for cluster centre. First, density measure for all data points is calculated. For x_i it is defined as

$$D_i = \sum_{j=1}^{n} e\left[\frac{-(x_i - x_j)}{(r_a/2)}\right]$$

A data point with many neighbouring points have high density measure. r_a is the range of data point that the minimum number of neighbouring data points lies. Data point with highest density value is selected as first cluster centre. r_b is then used as radius which defines the neighbourhood where density reduction is done.

$$D_i = D_i - D_{c1}\, e\left[\frac{-(x_i - x_{c1})}{(r_b/2)}\right]$$

After reduction is done data point with highest density measure is selected as cluster x_{c2}. Next we use cluster x_{c2} for density reduction. This procedure is repeated until $D_{ck} < AD_1$.

Training FIS

There are various methods that are proposed to construct the model for software defect prediction. In this paper we are proposing Adaptive Neuro Fuzzy Inference system for constructing the classifier.

Adaptive Fuzzy Inference System(ANFIS)

ANFIS is a 5 layered architecture which derives Sugeno type Fuzzy Inference system using hybrid learning rule. Figure 1 shows the architecture of ANFIS.

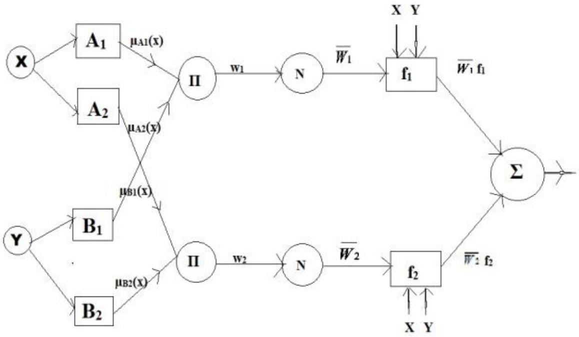

Figure 1: Adaptive Neuro Fuzzy Inference System

Layer 1 is an adaptive layer that computes the membership values of input variables. In this paper we are using Gaussian membership function.

Layer 2 is a fixed layer that computes the firing strength of input variables using product operator.

Layer 3 is a fixed layer that computes the normalized weight of the firing rule

Layer 4 is an adaptive layer that uses neural networks for best fitting of consequent parameters(p,q,r) of node function f(x,y)=p*x+q*y+r.

Layer 5 is a fixed node that computes the overall sum of input signals.

RESULTS & DISCUSSION

Knowledge Extraction based on Evolutionary Learning (keel) is an open source tool to conduct experiments on different datasets to evaluate the performance of various learning algorithms. We conducted experiments on software defect prediction using 4 datasets downloaded from NASA data repository to identify defective prone modules. Neural networks are one of the methodologies for best fitting non linear relationships between attributes. Cost sensitivity is applied to neural networks to improve the performance of the classifier.

In Keel experimentation section, Cost Sensitive Neural Networks are implemented using JAVA. For experimentation, we imported 4 SDP datasets in Data Management section. In the experimentation section, we included dataset and Cost Sensitive Neural Network algorithm is imported and the results are showed using imbalance check methods. This algorithm is repeated with different parameter values. In the first iteration, number of layers are fixed to two with a total of 15 neurons. In the next iterations the number of neurons are increased to 30 followed by 60. Finally the algorithm is tested by 3 layers with 90 neurons. These results are presented in Table 1.

We implemented Adaptive Neuro Fuzzy Inference system for Software defect prediction. In MAT lab, there is a toolbox Neuro Fuzzy tool box which implements ANFIS. In the Data section, training data was loaded. In FIS section, Initial Sugeno Fuzzy Inference system was derived by using subtractive clustering method. Subtractive clustering method takes four parameters Range of Influence(0.3), Squash factor(1.25) Accept ratio(0.5) & Reject Ratio(0.15). In training section, FIS was trained using ANFIS algorithm. In testing section, FIS was tested with unseen data. The Receiver Operating Characteristics(ROC) was plotted against true positive rate with false positive rate. AuC values are determined for each dataset of Software defect prediction and results are compared with cost sensitive neural networks.The performance of the classifier is measured in terms of Area under ROC Curve (AuC) values for imbalanced datasets. In Table 1 we compared the results of cost sensitive neural networks (considering different parameter values) with ANFIS.

For all datasets, the performance of ANFIS found satisfactory. ROC curve are generated by plotting false positive rate against true positive rate. Figures 2-5 illustrates these ROC curves on various SDP datasets.

Table 1: AuC values for Software Defect Prediction

Dataset\Algorithm	CS NN (l:2, n:15)	CS NN (l:2,n:30)	CS NN (l:2,n:60)	CS NN (l:3,n=90)	ANFIS
kc1	0.5	0.5	0.5	0.5	0.8482
kc2	0.64958	0.66451	0.660324	0.534307	0.8998
cm1	0.5	0.5	0.5	0.5	0.8904
pc1	0.589767	0.548134	0.637584	0.561159	0.8416

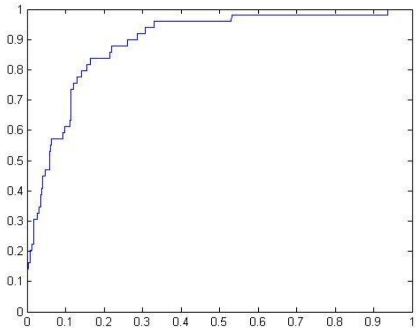

Figure 2: ROC Curve cm1 dataset

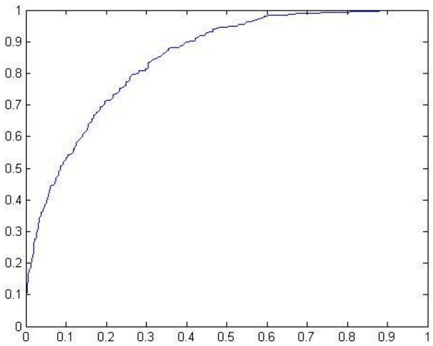

Figure 3: ROC Curve kc1 dataset

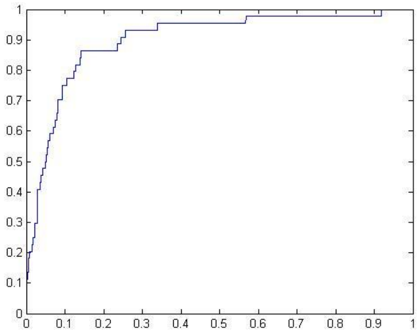

Figure 4: ROC Curve kc2 dataset

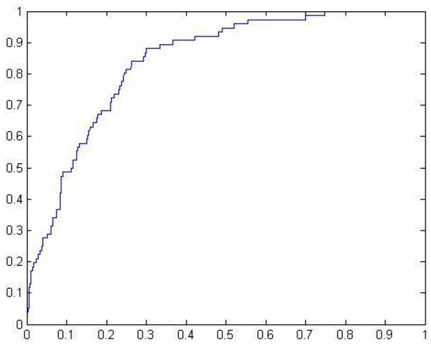

Figure 5: ROC Curve pc1 dataset

CONCLUSION

There are various methods for predicting defective prone modules using data mining like decision trees, Random forests, Support Vector Machines and Neural Networks. In this paper, We proposed Adaptive Neuro Fuzzy Inference System(ANFIS) for identifying defective prone modules. ANFIS method generates sugeno fuzzy model. Initially FIS was generated by subtractive clustering method and it was trained by ANFIS. We applied cost sensitive neural networks on SDP datasets with different parameters and the results are compared with ANFIS. The performance is measured in terms of AuC values and found satisfactory results with ANFIS.

REFERENCES

[1] Omer Faruk Arar, Kurşat Ayan,"Software defect prediction using cost-sensitive neural network" Applied Soft computing, April 2015.

[2] Jun Zheng," Cost-sensitive boosting neural networks for software defect prediction", Expert Systems with Applications. 2010 June;37(6),4537–4543.

[3] Ezgi Erturk, Ebru Akcapinar Sezer,"A comparison of some soft computing methods or software fault prediction", Expert systems with applications.2014

[4] J. Nam, S. J. Pan, and S. Kim,"Transfer defect learning", In Proceedings of the 2013 International Conference on Software Engineering. ICSE '13, 382–391.

[5] Song Q, Jia Z, Shepperd M, Ying S, Lin J, "A general software defect proneness prediction framework", IEEE Transations on Software Engineering. 2011;37,356-370.

[6] S. Shivaji, E. Whitehead, R. Akella, and S. Kim. "Reducing features to improve code change-based bug prediction", Software Engineering. IEEE Transactions on Systems.2013; 39(4),552–569.

[7] K O. Elish, M O. Elish, "Predicting defect prone software modules using support vector machines", Journal of systems & Softwares. 2008;649-660.

[8] Rodriguez D, Herraiz I, Harrson R, "On software engineering repositories and their open problems", In : First International workshop on realizing artificial Intelligence Synergies in Software engineering(RAISE).2012.

[9] Guo L, "Robust prediction of fault proneness by Random forests", In: 15th International symposium on software reliability engineering.ISSRE,IEEE;2004.

[10] Chen N, Hoi SCH,Xiao X. "Software process evaluation: A machine learning framework with application to defect management process", Empirical Software Engineering. 2013;19, 531-564.

[11] http//mdp.ivv.nasa.gov. Date accessed 15/11/2015

[12] Kanb P, Pinzger M, Bernstein, "A predicting defect densities in source code files with decision tree learners", Proc. of MSR New York, ACM press.2006;119-125.

[13] Shilpee Chaoli,Gil Tenne and Sanjay Bhatia, "Analysing Software Metrics for Accurate Dynamic Defect Prediction Models", Indian Journal of Science and Technology. 2015 Feb; 8(S4), 96-100

[14] C Balakrishna Moorthy, Ankur Agrawal and M K Deshmuh, "Artificial Intelligence Techniques for Wind Power Prediction: A Case Study", Indian Journal of Science and Technology. 2015 oct;8(25),1-10.

[15] S Meher Taj and A Kumaravel, "Survey on Fuzzy Petri Nets for Classification",Indian Journal of Science and Technology. 2015 July; 8(14).