International Journal of Engineering Research and Development
e-ISSN : 2278-067X, p-ISSN : 2278-800X, www.ijerd.com
Volume 2, Issue 5 (July 2012), PP. 78-82

Design of Hybrid Neural Network Model for Quality Evaluation of Object Oriented Software Modules

Amandeep Kaur[1], Arjan Singh[2], Baljit Singh[3]
[1,3]*Department of Computer Science Engineering, BBSB Engineering College, Fatehgarh Sahib. Punjab, India*
[2]*Department of Mathematics, Punjabi University, Patiala-14002, Punjab, India.*

Abstract—The aim of this paper is to evaluate the quality of object oriented modules. In this work neural network approach is used along with PSO (particle swarm optimization) to find out fault prone components of software. The evaluation measures used are Accuracy, MAE (Mean absolute error), RMSE (Root mean square error) and the results are calculated on different iterations. The Accuracy, MAE, RMSE is improved as number of iterations increases.

Keywords—Software engineering, Fault proneness, Neural Network, Particle Swarm Optimization.

I. INTRODUCTION

Software quality is the degree to which software possesses a desired combination of attributes such as reliability, maintainability, efficiency, portability, usability, and reusability [1]. This desired combination of attributes must be clearly defined. Various software metrics have been identified for the purpose of evaluating these characteristics of software systems. The aim of software metrics is to predict the quality of the object oriented software products.

Fault-proneness of a software module is the probability that the module contains faults. A corelation exists between the fault-proneness of the software and the measurable attributes of the code and testing [9]. Accurate prediction of fault-prone modules enables the verification and validation activities focused on the critical software components Yan Ma and Bojan Cukic [3]. Therefore, software developers have a keen interest in software quality models. It is required that fault-prone prediction models should be efficient and accurate.

Predicting the number of faults in each module or identifying the fault-prone modules in the early stages of the software development life cycle will help the development Renu Kumar Louisiana [2]. Software complexity metrics play a useful role in this regard because they are numerical measures that can be obtained early in the software life cycle. There exists a relationship between measures of software complexity and the number of errors later found in test and validation.

Faults in software systems continue to be a major problem. Many systems are delivered to users with excessive faults. It has long been recognized that seeking out fault-prone parts of the system and targeting those parts for increased quality control and testing is an effective approach to fault reduction. It is difficult to identify a reliable approach to identifying fault-prone software components Parvinder S. Sandhu [9].

Basili et al. [4] empirically validated CK metrics and found no correlation among metrics. The paper stated that all metrics were effective in predicting fault proneness except LCOM (Lack of Cohesion of Methods). However the metric suite was applied to the source code because some of their metrics measure an inner complexity of a class, and such information cannot be obtained until the algorithm and structure of the class are determined at the end of design phase.

Khoshgoftarr et al. [6] introduced the use of the neural networks as a tool for predicting software quality. They compared the neural-network model with a nonparametric discriminant model, and found the neural network model had better predictive accuracy. This model used domain metrics derived from the complexity metric data. These metrics are not adequate for detecting object-oriented faults. Since the object-oriented paradigm exhibits different characteristics from the procedural paradigm, software metrics in object-oriented paradigm need to be used. With the increasing use of object-oriented methods in new software development there is a growing need to improve current practice in object-oriented design and development

Toshihiro Kamiya et al. [7] presented a method to estimate the fault-proneness of the class in the early phase, using several complexity metrics for object oriented software. They introduced four checkpoints into the analysis/design/implementation phase, and estimate the fault-prone classes using the applicable metrics at each checkpoint. They estimate the fault-proneness by using the multivariate logistic regression analysis.

Atchara Mahaweerawat [8] proposed fault prediction model based on supervised learning using Multilayer Perceptron Neural Network and the results are analyzed in terms of classification correctness and based on the results of classification, faulty classes are further analyzed and classified according to the particular type of fault.

Munson & Khoshgoftaar [10] use discriminant analysis technique to investigate some aspects of the relationship between program complexity measures and program faults.

Mie mie thet thwin, tong-seng quah [11] proposed neural network model along with object oriented metrics. This empirical study presents the prediction of faults in three industrial real time system using multiple regression model and a neural network model and found that neural network model can predict the number of faults more exactly than multiple regression model.

A variety of software fault predictions techniques have been proposed, but none has proven to be consistently accurate. Therefore, there is a need to find the best prediction techniques for a given prediction problem.

1. The techniques used in Khoshgoftarr et al. [6] used to predict faults that use traditional metrics are not generally applicable to object oriented system.
2 The data collected in Basili et al. [4] is not of industry. So there can be increased complexity when it comes to large object oriented system of organization.
3 CK metrics evaluate static complexity of object oriented software in Basili et al. [4], V. R. Basili [5]. Several OOD specifications include dynamic information. It is necessary to evaluate such dynamic complexity.
4. Models used in Khoshgoftaar [12] take Increase time to design and develop quality cases.

So, in this study a PSO Technique is used along with Neural Network to predict fault prone modules in object oriented.

II. MATERIALS AND METHODS

The proposed study find out the fault proneness of object oriented modules. The technique used in this study is Particle swarm optimization along with Neural Network. The evaluation measures used are Accuracy, Mean Absolute Error, Root Mean Squared Error.

Step-1: Find the structural code and design attributes of software. The dataset taken is from NASA KC1.The dataset is related to software defect prediction which contains 2109 modules and 22 metrics.
Step-2: Load the training data .In this step the 80% of KC1 dataset is used for training.
Step-3: Perform Particle Swarm Optimization based Neural Network training. In this step Particle Swarm Optimization toolbox for Matlab is used and the training is performed with the help of this toolbox.
Step-4: Load testing data. In this step 20% of kc1 dataset is used for testing.
Step-5: In this step testing is done and the performance is calculated on the basis of Accuracy, MAE (Mean Absolute Error), RMSE (Root Mean Squared Error)

The evaluation measures that are used to carry out the results are discussed below:

- Mean absolute error (MAE)
- Root mean square error (RMSE)
- Accuracy

- **Mean absolute error**

Mean absolute error, *MAE* is the average of the difference between predicted and actual value in all test cases; it is the average prediction error. The formula for calculating *MAE* is given in equation shown below:

$$\frac{|a_1 - c_1| + |a_2 - c_2| + \ldots + |a_n - c_n|}{n} \quad (1)$$

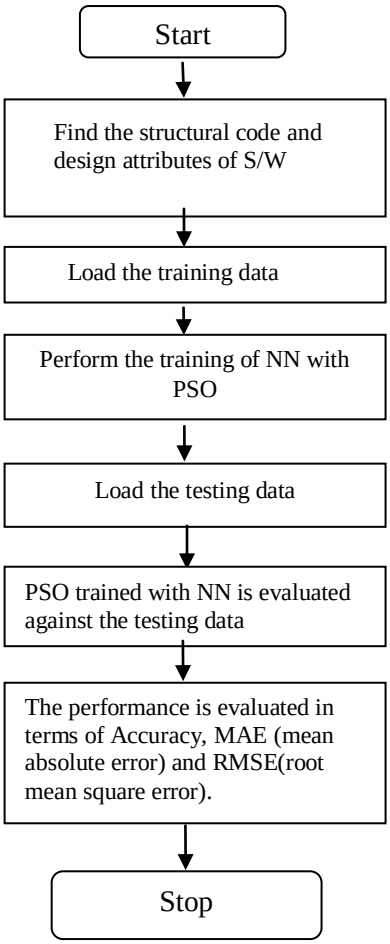
Assuming that the actual output is *a*, expected output is *c*

- **Root mean-squared error**

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.

$$\sqrt{\frac{(a_1 - c_1)^2 + (a_2 - c_2)^2 + \ldots + (a_n - c_n)^2}{n}} \quad (2)$$

The basic flow of proposed model is given below:



III. RESULTS AND DISCUSSIONS

The PSO technique along with Neural network is used to find the fault prone components of object oriented software modules .The results are calculated on different iterations and neural network gives better results as umber of iterations increases.

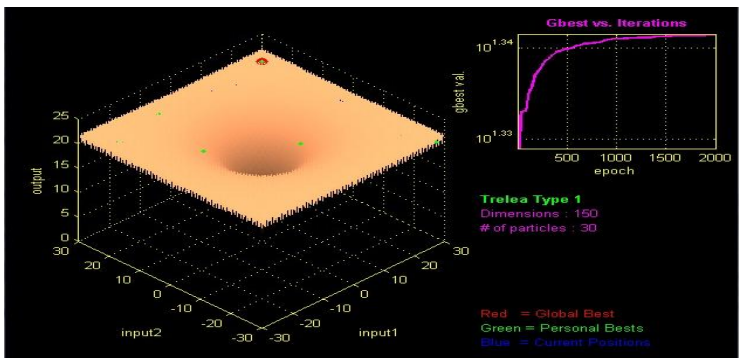
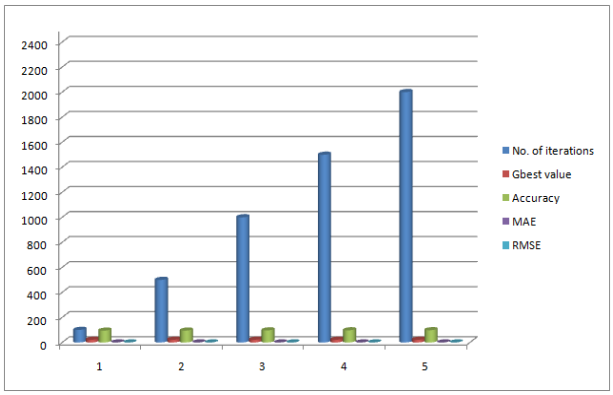


Fig1: **Finding Gbest value by PSO by different iteration**

No. of particle	Dimensions	No. of iterations	Gbest value	Accuracy	MAE	RMSE
30	150	100	21.4691	94.22	0.0299	0.1109
		500	21.8190	94.78	0.0290	0.1102
		1000	21.8173	96.25	0.0279	0.1022
		1500	21.8609	96.98	0.0197	0.0990
		2000	21.8709	97.75	0.0178	0.0957

Table1: **Performance on the basis of different iterations by PSO**



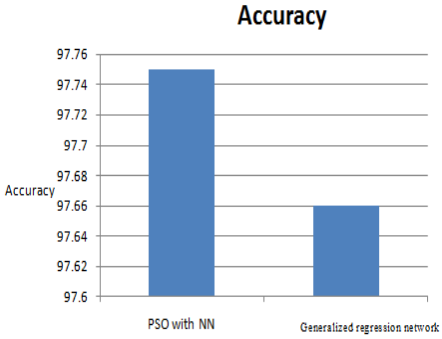
***Graph1:*Showing performance by PSO on different iterations**

PSO used with neural network is giving better results as the number of iterations increases.

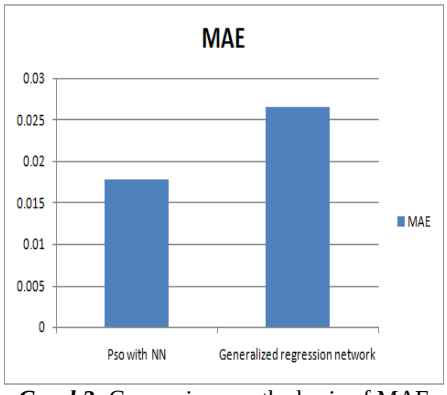
The comparison is shown in table given below:

S.no.	Algorithm	Accuracy	MAE	RMSE
1	PSO with NN	97.75	0.0178	0.0957
2	Generalized regression network	97.66	0.0265	0.1056

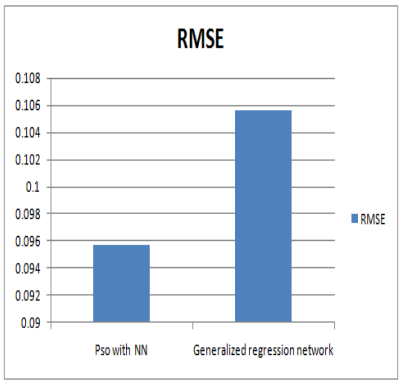
Table 2: Comparison of PSO with NN with generalized regression network on the basis of different measures



***Graph 2 :*Comparison on the basis of accuracy**



Graph3: Comparison on the basis of MAE



Graph 4: Comparison on the basis of RMSE

IV. CONCLUSION

A variety of software fault prediction techniques have been proposed, which have been already applied in software engineering applications. A neural network is trained to reproduce a given set of correct classification examples, instead to produce formula or rules. Neural networks are non linear techniques that are able to model complex functions. In this work NN approach is used along with PSO to find out fault prone components of software. The evaluation measures used are Accuracy, MAE, RMSE and the results are calculated on different iterations. Result is calculated on different iterations. The accuracy is improved as the number of iterations increases. The results showed that proposed technique is better as compared to traditional approach and give better results.

REFERENCES

[1]. [1]IEEE Standard for a Software Quality Metrics Methodology, Institute of Electrical and electronics engineers Inc.1993.

[2]. [2] Renu Kumar Louisiana State University Baton Rouge Suresh Rai Louisiana State University Baton Rouge Jerry L. Trahan Louisiana State University Baton Rouge Neural-Network Techniques for Software-Quality Evaluation

[3]. [3] Yan Ma, Lan Guo and Bojan Cukic, "A Statistical Framework for the Prediction of Fault-Proneness", Department of Statistics, West Virginia University, Morgantown.

[4]. [4] S. Chidamber, and C. Kemerer, "A metrics suite for object-oriented design", IEEE Transactions on Software Engineering, 20(6), 1994, pp.476-493.

[5]. [5] V. R. Basili, L. C. Briand, W. L. Mélo((1996)): A validation of object-oriented design metrics as quality indicators, IEEE Trans. on Software Eng., Vol. 20, No. 22, pp. 751-761.

[6]. [6] P.M. Khoshgoftaar, E.B. Allen, J.P. Hudepohl, and S.J.Aud, "Application of neural networks to software quality modeling of a very large telecommunications system", IEEE Transactions on Neural Network, vol. 8, pp. 902-909, 1997.

[7]. [7] T. Kamiya, Kusumoto, S., K. Inoue,"Prediction of fault proneness at early phase in object-oriented development", Proceedings of the Second IEEE International Svmposium on Object-Oriented Real-Time Distributed Computing, pp., 253-258,1999.

[8]. [8]Mahaweerawat, A. (2004), "Fault-Prediction in object oriented software's using neural network techniques", Advanced Virtual and Intelligent Computing Center (AVIC), Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok, Thailand, pp.1-8.Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'OZ) , Vol. 5 Lip0 W a g Jagath C. Rajapakse, Kunihiko Fukushima,Soo-Young Lee, and Xin Yao (Editors)

[9]. [9] Parvinder S. Sandhu, Satish Kumar Dhiman, Anmol Goyal World Academy of Science, Engineering and Technology 60 2009.

[10]. A Genetic Algorithm Based Classification Approach for Finding Fault Prone Classes

[11]. [10] Munson, J. C. & Khoshgoftaar, T. M., "The detection of fault-prone programs", IEEE Transactions on Software Engineering, (1992), 18(5), 423-433.

[12]. [11] Mie Mie Thet Thwin Tong-Seng Quah School of Electronic & Electrical Engineering Nanyang Technological University Application of neural network for predicting software development faults using object-oriented design metrics.

[13]. [12] Khoshgoftaar, T. M., Allen, E. B., Ross, F. D., Munikoti, R., Goel, N. & Nandi, A., "Predicting fault-prone modules with case-based reasoning". ISSRE 1997, the Eighth International Symposium on Software Engineering (pp. 27-35), IEEE Computer Society (1997).