PREDICTION OF SEVERITIES OF BUGS WITH LEVELS AND TYPES OF INHERITANCE

Varuna Gupta
Research Scholar
Christ University

Dr. N. Ganeshan
Director MCA
RICM, Bangalore

Dr. Tarun Kumar Singhal
Professor
Symbiosis Institute of Telecom Management (SITM)
(A Constituent of Symbiosis International University)

ABSTRACT

Past researches have attributed level of inheritance as major contributor of effectiveness in prediction of severities of bugs. This research initially attempts to correlate effectiveness of prediction of severities of bugs with levels of inheritance. This research also attempts to take a step further by correlating the prediction of severities of bugs with types of inheritance as well.

This present research has considered different levels of inheritance and has established a correlation framework for severities of bugs (non trivial bugs, major bugs, and critical bugs) with types and levels of inheritance.

This research has successfully revealed that the severities of bugs can be associated with different levels and types of inheritance and has further concluded that with increasing levels as well as complexity of types of inheritance, the severity of bugs will also increase.

In this research work, two back-propagation training functions such as Broyden–Fletcher–Goldfarb–Shanno (BFG) and Levenberg-Marquardt (LM) have been selected for evaluation. The present research work has used these two training functions to validate the results on the basis of mean square error (MSE), prediction accuracy, R on testing, R on training and R on validation.

The present research has generated sufficient interest with the help of correlation framework associating levels and types of inheritance with severities of bugs. The present research work has also resulted in development of a tool for demonstrating type of inheritance (single inheritance, multilevel inheritance, hierarchical inheritance, and multiple inheritances) associated with each file containing bugs.

Furthermore, the findings are of growing importance suggesting that levels and types of inheritance need to be rationalized in order to contain severities of bugs for effective quality control in software project.

Keywords: Level & type of inheritance, Severities of software bugs, Neural Network.

1. INTRODUCTION

In order to ensure reasonable quality and reliability in software development, bug prediction has to play a noticeable role. Software bug is generally known as an error, flaw, mistake, failure, or defect in a software program or system producing an incorrect or unexpected result, or making it to behave in unintended ways. A large number of bugs interfering with functionality of software can make it bug prone. Bug prediction process tries to locate and identify the defective modules in software. The traditional processes involved in identifying bugs are as code review, unit testing, integration testing and system testing. With increasing code size, complexity and depth of inheritance, the process of finding and fixing bugs becomes more difficult and expensive using sophisticated testing and evaluation procedures. With growing complexity and depth of inheritance, the severities of bugs also rise to higher levels. This observation has formed the very basis of present research work highlighting inadequacies of existing bug finding approaches. This motivation has directed for the underlying research work to predict the severities of bugs.

In this research, the researchers have used artificial neural network (ANN) for prediction. An ANN is a biologically inspired computational model composed of various processing elements called artificial neurons. They are connected with coefficients or weights which construct the neural network's structure [24]. The processing elements have weighted inputs, transfer function and outputs for processing information. There are many types of neural networks with different structures have been designed, but all are described by the transfer functions used in processing elements (neurons), the way of training given or learning rule and by the connection formula. In a feed forward multilayer perceptron network, the inputs signals are multiplied by the connection weights are first summed and then directed to a transfer function to give output for that neuron. The transfer function (purelin, hardlim, sigmoid, logistic) executes on the weighted sum of the neuron's inputs.

Some files in the training set of software metrics have zero or near zero values of each type of severity of bug. So, training data can be classified into two clusters as buggy and non-buggy sets. This partitioning enhances the performance of learning process and enables neural network to work only on training data consisting of files that are having any number of bugs.

In this research, the researchers have used two training functions of neural network such as trainlm and trainbfg. Post analysis, the research has found that these functions are more effective in prediction of severities of bugs with almost zero possibility of errors.

According to the above research most of the work has been done using C&K metrics and type of inheritance as the input of the network and number of bugs has been used as the output of the network. Some researchers have concluded that DIT and other inheritance related metrics have a strong relationship with bugs or the significant reasons of bugs in the software [1, 2, 3, 26 & 27]. However, previous researches haven't associated/correlated levels and types of inheritance with severities of bugs.

In this study, researchers have selected, two back-propagation training functions such as Broyden–Fletcher–Goldfarb–Shanno (BFG) and Levenberg-Marquardt (LM) for evaluating the result. The present research work has used these two training functions to validate the results on the basis of mean square error (MSE), prediction accuracy, R on testing, R on training and R on validation.

2. OBJECTIVES

The research work has generally tried to associate different levels and types of inheritance through neural network by establishing correlation with severities of bugs. Specifically, the research has tried to fulfill the following objectives:

1. To establish the correlation among levels of inheritance, types of inheritance and severities of bugs.

2. To predict various types of severities of bugs (Non Trivial, Major and Critical) corresponding to increasing levels of inheritance.
3. To assess the impact of types of inheritance on severities of bugs.

3. LITERATURE REVIEW

In this research, researchers have used Machine learning techniques (neural networks) to predict the types of severities of the bugs using the proprietary software. A research [1] based on C & K metrics suits claimed that most CK metrics were effective bug predictors, and among them, independent variables DIT and RFC were having significant influence on the dependent variable (bugs). In another research, researchers have concluded [2] that WMC (Weighted methods per class**)** and DIT were significant indicators for finding buggy files.
In this study [3], researchers have emphasized on various sized oriented metrics and proved that DIT, WMC, CBO (coupling between objects) and LOC (Line of code) are significant bug indicators for predicting number of bugs.
A significant amount of empirical work had been already done for finding out buggy data using object oriented design metrics and C&K metrics suits. These research works have basically proved that there is a significant relationship between metrics and number of bugs [4, 5, 7, 11, 12, 13]. After that some more research studies [6, 8, 9, 10] have emphasized on predicting the number of bugs using machine learning methods. In machine learning methods [14], Bayesian approach was used by the researchers to demonstrate a strong relationship between product metrics and number of bugs.
However, these studies failed to focus on the severities of bugs. There are only a few studies which were based on the severities of bugs. A popular study [15] has focused on three levels of severities of bugs (Low, Medium, and High) and concluded that predictions using ANN methods are better than predictions using statistical methods. The researchers have confirmed that CBO, WMC, RFC and SLOC metrics are more significant than DIT for bug prediction at all severity levels. Somewhat same results were also derived in another research paper [16], which demonstrated that the purposed model was performing well with low and medium level of severity rather than high level of severity. This research also investigated the fault-proneness prediction performance of OO design metrics with respect to ungraded, high, and low severity faults by employing statistical (LR) and machine learning (Naïve Bayes, Random Forest, and NNge) methods.
In another research [25], the researchers firstly predicted the nature of file (buggy/not buggy) and secondly predicted the magnitude of the possible bugs with respect to various viewpoints such as density, severity or priority. Finally, the researchers concluded that ANN was successful in predicting the really defected items. According to their results, the MLP algorithm approximates the bug severity values well only when defected items reside in the input data set.

4. PROPOSED WORK

The experiments reported in this research work, involve software named as work force scheduler (WFS). WFS is the object oriented programming based project, which is used widely in work force management projects. However, WFS presently doesn't accommodate types of inheritance. The researchers intend to develop a new tool to calculate the types of inheritance associated with files containing bugs. Also the researchers intend to modify the Boolean values received as a result of type of inheritance and associate them with types of inheritance.

Our objective was to predict severities of bugs on the basis of types and levels of inheritance using neural network (NN) algorithms. Our experiments with neural network consisted of 172 inputs with two inheritance related data inputs.

This research work basically used two experiments. In the first experiment, researchers tried to find out the correlation among levels of inheritance, types of inheritance and severities of bugs. After that, in the second experiment, the researchers tried to predict the type of severities of bugs using levels and types of inheritance as network input with the help of two training functions of back-propagation neural network algorithms. The researchers used trainlm and trainbfg training functions with different number of neurons (10 and 20). Validation of these two training functions was done on the basis of mean square error (MSE), prediction accuracy, R on testing, R on training and R on validation.

To design a learning system, the data set in this work is divided into two parts: the training data set and the testing data set. Some predictor functions are defined and trained with respect to Multi-Layer Perceptron.

Multi-layer perceptron method (feedforward neural networks trained with the standard back propagation algorithm) have been used for the basic advantage offered in the form of a general framework using non-linear functional mappings between two sets (one for input variables and second for output variables). For this, activation functions have been used for representing the nonlinear function of many variables in terms of compositions of nonlinear functions of a single variable [17].

4.1 Multilayer Feed Forward Neural Networks

Multilayer feed forward neural networks (MLF) consisting of multiple layers of computational units are generally interconnected in a feed-forward way known as the input layer. The last layer is represented as the output layer and all other layers in between are known as hidden layers. Each layer is connected to the next layer through connections to the neurons. Multilayer Feed Forward Neural Networks (non-parametric regression methods) approximate the underlying functionality in data through reduction in the loss function. Specified items of data records used as input to neural network and weights are respectively changed to ensure that output approximates the values in the data set during training activity [18].

4.2 Neural Network Training Algorithms

There are number of training algorithms which can be used to train a network. In the present research work, the researchers have used basically two best training functions of neural network algorithms as trainbfg and trainlm [23].

i. ***BFGS (Broyden–Fletcher–Goldfarb–Shanno) algorithm (trainbfg):*** approximates Newton's method, a class of hill-climbing optimization techniques that seeks a stationary point of a function. For such problems, a necessary condition for optimality is that the gradient be zero [20]. BFGS have good performance even for non smooth optimizations and an efficient training function for smaller networks.

ii. **Levenberg–Marquardt backpropagation (trainlm) algorithm** locates the minimum of a multivariate function that can be expressed as the sum of squares of non-linear real-valued functions. It is an iterative technique to reduce performance function in each iteration of the algorithm. This feature makes trainlm the fastest training algorithm for networks of moderate size [21, 22].

5. EXPERIMENTAL RESULTS AND CONCLUSIONS

To achieve the objectives of the study, the researchers have performed two experiments.

Experiment-1:
Experiment 1 is depicted the correlation between level & type of inheritance and severities of bugs

Table 1- Correlation Table

Project	**Indicators**	**Correlation with severity of bugs**
WFS (Object Oriented)	Levels of Inheritance	.746
	Types of Inheritance	.536

In the present research work, the researchers have observed significant correlations of levels and types of inheritance with severities of bugs.
This is further interesting to witness that above two indicators are correlated significantly with severities of bugs.

Experiment-2:
The training functions used were coded in MATLAB using ANN toolbox [19]. The experimental data consisted of 172 bug data entries with two inputs. Experiment 2 demonstrated the 5 levels of inheritance and related tables with the training functions and types of severities of bugs.
In this experiment, table 2, table 3, table 4, table 5 and table 6 had shown that with amplification in levels of inheritance the severities of bugs would also amplify accordingly.

Table 2: Predicted Severities of Bugs at Inheritance Level-1

Inheritance Level -1			
Training Function	**nontrivial Bugs**	**Major Bugs**	**Critical Bugs**
LM(10)	26	0	0
LM(20)	26	0	0
BFG(10)	26	0	0
BFG(20)	26	0	0

Table 3: Predicted Severities of Bugs at Inheritance Level –2

Inheritance Level -2			
Training Functions	**nontrivial Bugs**	**Major Bugs**	**Critical Bugs**
LM(10)	0	24	0
LM(20)	0	24	0
BFG(10)	0	24	0
BFG(20)	20	4	0

Table 4: Predicted Severities of Bugs at Inheritance Level –3

Inheritance Level -3			
Training Functions	**nontrivial Bugs**	**Major Bugs**	**Critical Bugs**
LM(10)	0	41	3
LM(20)	0	41	3
BFG(10)	0	41	3
BFG(20)	0	0	44

Table 5: Predicted Severities of Bugs at Inheritance Level –4

Inheritance Level -4			
Training Functions	**nontrivial Bugs**	**Major Bugs**	**Critical Bugs**
LM(10)	0	2	63
LM(20)	0	2	63
BFG(10)	0	0	65
BFG(20)	0	0	65

Table 6: Predicted Severities of Bugs at Inheritance Level –5

Inheritance Level -5			
Training Functions	**nontrivial Bugs**	**Major Bugs**	**Critical Bugs**
LM(10)	0	0	13
LM(20)	0	0	13
BFG(10)	0	0	13
BFG(20)	0	0	13

Table 7 is showing the comparative results of two training functions with different number of

neurons. The parameters of comparisons are number of neurons, MSE, R on training, R on testing and R on validation. All the parameters are checked for 10 and 20 number of neurons in the hidden layer. The network is trained until the MSE is equal to zero. According to this resultant table training function trainlm with 10 & 20 neurons is performing well than the training function trainbfg.
Trainlm is giving almost zero MSE.

Table 7: Accuracy Measurement of Training Functions

Training Functions	**Neurons**	**Best validation MSE**	**R on training**	**R on testing**	**R on validation**
Trainlm	10	0.105	0.766	0.715	0.843
	20	0.119	0.763	0.673	0.838
Trainbfg	10	0.174	0.726	0.816	0.865
	20	0.166	0.746	0.740	0.855

Networks simulated using these two training functions are affected according to the number of neurons in their hidden layer. The regression analysis function compares the actual outputs of the neural network with the corresponding desired outputs (targets).
It returns the correlation coefficient (R) in the range of 0 to 1. A value towards 1 represents a perfect positive correlation between actual and desired output. Regression value (R) on training, testing and validation in table 7 clearly indicates trainlm qualifies best.
Table 8 indicates the impact of types of inheritance on the severities of bugs corresponding to the levels of inheritance.
It has also been observed that with increasing levels of inheritance, types of inheritance are getting more complex and the severities of bugs are also escalating.

Table 8 Impact of Types of Inheritance on Severities of Bugs

Levels of Inheritance	**Types of Inheritance**	**Severities of Bugs**
1	Single Inheritance	Non trivial
2	Multilevel Inheritance	Major
3	Multilevel	Major, Critical
4	Multilevel, Hierarchical	Critical
5	Hierarchical, Multiple	Critical

The ANNs were simulated and trained with both the training functions using the training dataset. In the proposed work we did not found much difference between trainlm and trainbfg. They are in acceptable range. On the basis of MSE, trainlm and trainbfg both performed well. Considering the sample size of input patterns, we found that trainlm suits to larger data set. It converges in less number of iterations and in lesser time than the other training functions.

6. FUTURE WORK

The tool developed during the course of these experiments looks promising and adds one more dimension of associating types of inheritance with severities of bugs, which was earlier associated with levels of inheritance only.
However, this tool has definite scope of refinement so as to include higher levels of inheritance and investigating the impact after the level of inheritance goes above 5.
The researchers intend to refine this tool further on the above mentioned lines and also look for association of other relevant factors in addressing severities of bugs.

References

[1] Basili, V.R., L.C. Briand and W.L. Melo, 1996. A validation of object-oriented design metrics as quality indicators. IEEE Trans. Software Eng., 22: 751-761. DOI: 10.1109/32.544352.
[2] Tang, M.H., M.H. Kao and M.H. Chen, 1999. An empirical study on object-oriented metrics. Proceedings of the 6th International

Symposium on Software Metrics, Oct. 04-06, IEEE Computer Society, Boca Raton, FL., USA., pp: 242-249. DOI: 10.1109/METRIC.1999.809745.

[3] Gupta. V, Ganeshan N. and Singhal T.K., "Developing Software Bug Prediction Model Using Various Software Metrics " IJACSA, vol.6, No. 2, pp. 60-65, 2015

[4] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Systems with Applications*,vol.36, pp. 7346-7354, 2009.

[5] K.E. Emam and W. Melo, "The Prediction of Faulty Classes Using Object-Oriented Design Metrics," *Technical report: NRC 43609*, 1999.

[6] I. Gondra, "Applying machine learning to software fault-proneness prediction," *The Journal of Systems and Software*, vol.81, pp. 186-195, 2008.

[7] T. Gyimothy, R. Ferenc and I. Siket, "Empirical validation of object oriented metrics on open source software for fault prediction," *IEEE Transactions on Software Engineering*, vol.31, no.10, pp. 897-910, 2005.

[8] R. Malhotra and Y. Singh, "On the Applicability of Machine Learning Techniques for Object- Oriented Software Fault Prediction," *Software Engineering: An International Journal*, vol.1, no.1, pp. 24-37, 2011.

[9] R. Malhotra and A. Jain, "Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality," *Journal of Information Processing Systems*,vol. 8, no.2, pp. 241- 262, 2012.

[10] N. Ohlsson, M. Zhao, M and M. Helander, "Application of multivariate analysis for software fault prediction," *Software Quality Journal*, vol.7, pp.51-66, 1998.

[11] H. Olague, L. Etzkorn, S. Gholston and S. Quattlebaum, "Empirical validation of three software metrics suites to predict fault-proneness of object oriented classes developed using highly iterative or agile software development processes," *IEEE Transactions on Software Engineering*, vol.33, no.8, pp. 402-419, 2007.

[12] P. Yu, T. Systa, and H. Muller, "Predicting fault-proneness using OO metrics: An industrial case study," *In Proceedings of Sixth European Conference on Software Maintenance and Reengineering*, Budapest, Hungary, pp.99-107, 2002.

[13] Y. Zhou, B. Xu, and H. Leung, "On the ability of complexity metrics to predict fault-prone classes in object -oriented systems," *The journal of Systems and Software*, vol.83, pp.660-674, 2010.

[14] G. Pai, "Empirical analysis of software fault content and fault proneness using Bayesian methods," *IEEE Transactions on Software Engineering*, vol. 33, no. 10, pp. 675-686, 2007.

[15] Y. Singh, A. Kaur and R. Malhotra,"Empirical validation of object oriented metrics for predicting fault proneness models," *Software Quality Journal*, vol.18, pp. 3-35, 2010.

[16] Y. Zhou, and H. Leung,"Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults," IEEE Transactions on Software Engineering, vol. 32, no. 10, pp. 771-789, 2006.

[17] Bishop, M., 1995, Neural Networks for Pattern Recognition, Oxford University Press.

[18] Gayathri M, A. Sudha ,"Software Defect Prediction System using Multilayer Perceptron Neural Network with Data Mining" , International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Vol. 3, no. 2, 2014.

[19]. M. Beale, M. Hagan, H. Demut, "Neural Network Toolbox User's Guide," 2010.

[20]'Mathworks', 2015, available: http://www.mathworks.in/help/nnet/ref/trainbfg.html

[21] D.Pham, S. Sagiroglu, "Training multilayered perceptrons for pattern recognition: a comparative study of four training algorithms", *International Journal of Machine Tools and Manufacture*, vol.41, pp. 419–430, 2001

[22]B. Sharma, K. Venugopalan, "Comparison of Neural Network Training Functions for Hematoma Classification in Brain CT Images ", *IOSR Journal of Computer Engineering*, vol 16, pp- 31-35, -2014

[23] S. Ali and K. A. Smith, “On learning algorithm selection for classification”, Applied Soft Computing, (6), pp.119–138, 2006.
[24] S. Haykin, “Neural Networks- A Comprehensive Foundation,” 2nd ed., Pearson Prentice Hall, 2005.
[25] Kutlubay O. and A. Bener, "A Machine Learning Based Model For Software Defect Prediction," Boğaziçi University, Computer Engineering Department, 2005.
[26] Succi, G., W. Pedrycz, M. Stefanovic and J. Miller, “Practical assessment of the models for identification of defect-prone classes in objectoriented commercial systems using design metrics”. J. Syst. Software, 65: 1-12. DOI: 10.1016/S0164- 1212(02)00024-9, 2003.
[27] Madhu Rohilla1, P. K. Bhatia2 Prediction of Fault-Proneness UsingCK Metrics”, International Journal of Emerging Technology and Advanced Engineering ISSN 2250-2459, ISO 9001:2008 Certified Journal, Vol 3, Issue 8, 2013.