International Journal of Information Technology and Knowledge Management
July-December 2008, Volume 1, No. 2, pp. 259-266

MODELING OF FAULT PREDICTION IN SOFTWARE SYSTEMS

Deepali Gupta

Abstract

Prediction of fault-prone modules provides one way to support software quality engineering through improved scheduling and project control. Methodologies and techniques for predicting the testing effort, monitoring process costs, and measuring results can help in increasing efficiency of software testing. Predicting faults early in the software life cycle can be used to improve software process control and achieve high software reliability.

In the present work, different machine learning algorithms and neural network techniques are evaluated on two different real-time software defect datasets. The results show that when all the prediction techniques are evaluated, then best algorithm for classification of the software components into faulty/fault-free systems is found to be Generalized Regression Neural Networks.

Keywords: Fault prediction, Software metrics, Software quality, Machine learning techniques and Neural Network algorithms.

1. Introduction

A software fault is a defect that causes software failure in an executable product. When a software system is developed, the majority of faults are found in a few of its modules. In most of the cases, 55% of faults exist within 20% of source code. It is, therefore, much of interest is to find out fault-prone software modules at early stage of a project [7]. Timely predictions of faults in software modules can be used to direct cost-effective quality enhancement efforts to modules that are likely to have a high number of faults [3]. Using software complexity measures, the techniques build models, which classify components as likely to contain faults or not. Early detection of fault-prone software components enables verification experts to concentrate their time and resources on the problem areas of the software system under development [4].

Metrics is defined as "The continuous application of measurement based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products". The attributes of effective software metrics are:

- Simple and Computable.
- Empirically and Intuitively Persuasive.

- Consistent Results.
- Consistent in the Use of Units and Dimensions.
- Programming Language Independent.

To predict the fault in software data a variety of techniques have been proposed which includes statistical method, machine learning methods and neural network techniques.

- Statistical methods are used to find an explicit numerical formula, which determines completely how classification is performed.
- Machine learning is concerned with the design and development of algorithms and techniques to extract rules and patterns out of massive data sets.
- Neural networks, which have been already applied in software engineering applications, to build reliability growth models predict the gross change or reusability metrics. A neural network is trained to reproduce a given set of correct classification examples, instead to produce formulas or rules [8]. Neural networks are non-linear sophisticated modeling techniques that are able to model complex functions. Neural network techniques are used when exact nature of input and outputs is not known. A key feature is that they learn the relationship between input and output through training.

Machine learning capabilities create applications that are rugged, self-adapting, easier to maintain and often more fault tolerant than conventional systems. Learning systems also provide the core mechanism for powerful predictive and classification models that fine tune their abilities as they gather more and more experience. Machine learning deals with the issue of how to build programs that improve their performance at some task through experience. There is an existence of a correlation between a reasonable set of static metrics and software fault proneness [1].

In this paper, various machine learning algorithms and Neural Network techniques are explored and comparative analysis is performed for the prediction of faults in software systems. The rest of the paper is organized as follows. Section 2 gives overview of the problem formulation. Results and Discussion is presented in Section 3. Section 4 focuses on conclusion of this work.

2. PROBLEM FORMULATION

Software quality prediction models seek to predict quality factors such as whether a component is fault prone or not. Methods for identifying fault-prone software modules support helps to improve resource planning and scheduling as well as facilitating cost avoidance by effective verification [5]. Software quality models seek to predict quality factors of software components based on product and process attributes. The ability of software quality models to accurately identify critical components allows for the application of focused verification activities ranging from manual inspection to automated formal analysis methods [9].

Prediction of fault-prone modules:

(1) Supports software quality engineering through improved scheduling and project control.

(2) Can be a key step towards steering the software testing and improving the effectiveness of the whole process by planning and executing testing activities.

(3) Enables effective discovery and identification of defects.

(4) Enables the verification and validation activities focused on critical software components.

(5) Used to improve software process control and achieve high software reliability.

(6) Can be used to direct cost-effective quality enhancement efforts to modules that are likely to have a high number of faults.

A wide variety of techniques have been proposed [2]. The modeling techniques cover the main classification paradigms, including principal component analysis, discriminate analysis, logistic regression, logical classification models and layered neural networks [6].

3. RESULTS AND DISCUSSION

The first step is to find the structural code and design attributes of software systems i.e. software metrics. So, the real-time defect data sets used are taken from the NASA's MDP (Metric Data Program) data repository, available online at *http://mdp.ivv.nasa.gov.* The CM1data is obtained from a spacecraft instrument, written in C, containing approximately 505 modules. The PC1 data is collected from a flight software system coded in C, containing 1107 modules.

Figure 1 shows the CM1 Graphical representation of details of the output of software where label i.e. error count is meant for output and is equal to the number of errors and the count tells the number of occurrences of that label in the CM1 data set. And the PC1 Graphical representation of details of the output of software is shown in Figure 2.

The next step is to find the suitable algorithm for classification of the software components into faulty/fault-free systems. The algorithms which are explored are already built java classes in WEKA project [10]. For this a variety of many machine learning algorithms and neural network techniques are analyzed. The model is implemented and then best algorithm is found.

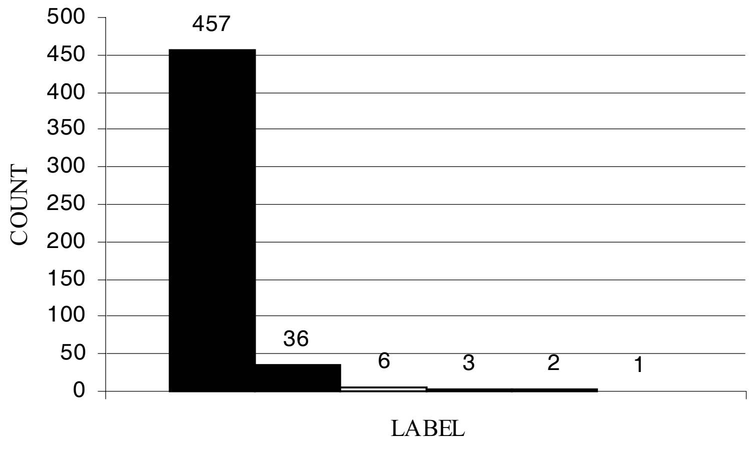


Figure 1: CM1 Graphical Details of the Output of Software

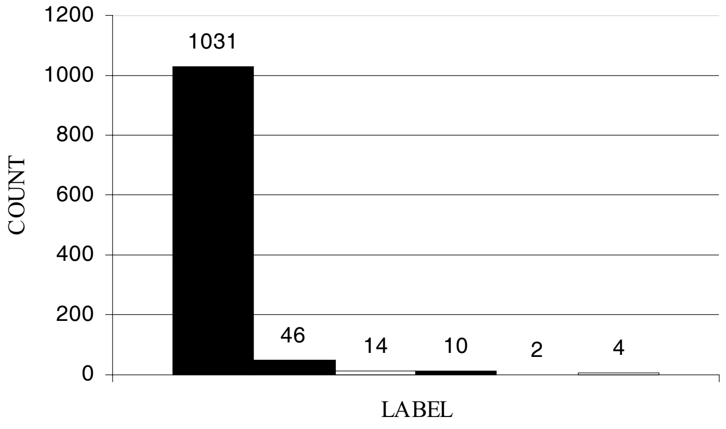


Figure 2: PC1 Graphical Details of the Output of Software

The algorithms are evaluated on the basis of the following criteria:

The WEKA software computes the mean absolute error, root mean squared error, relative absolute error and root relative squared error. However, the most commonly reported error is the mean absolute error and root mean squared error.

The root mean squared error is more sensitive to outliers in the data than the mean absolute error. In order to minimize the effect of outliers, mean absolute error is chosen as the standard error. The prediction technique having least value of mean absolute error is chosen as the best prediction technique.

The mean absolute error and root mean squared error is calculated for all the classes of machine learning algorithms and neural network techniques. The graphical user interface is designed in MATLAB for various neural network techniques as shown in Figure 3.

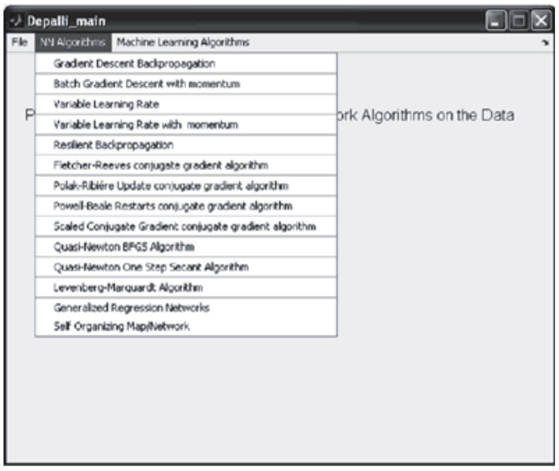

Depalli_main
File
NN Algorithms
Machine Learning Algorithms
Gradient Descent Backpropagation
Batch Gradient Descent with momentum
Variable Learning Rate
Variable Learning Rate with momentum
Resilient Backpropagation
Fletcher-Reeves conjugate gradient algorithm
Polak-Ribiére Update conjugate gradient algorithm
Powell-Beale Restarts conjugate gradient algorithm
Scaled Conjugate Gradient conjugate gradient algorithm
Quasi-Newton BFGS Algorithm
Quasi-Newton One Step Secant Algorithm
Levenberg-Marquardt Algorithm
Generalized Regression Networks
Self Organizing Map/Network
P
ork Algorithms on the Data

Figure 3: GUI for Neural Network Techniques

The graphical user interface designed for all the classes of machine learning algorithms is shown in Figure 4.

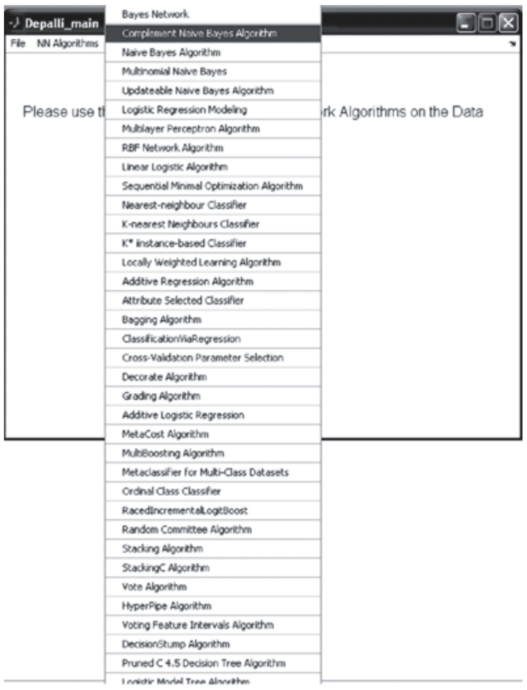

Depalli_main
File
NN Algorithms
Bayes Network
Complement Naive Bayes Algorithm
Naive Bayes Algorithm
Multinomial Naive Bayes
Updateable Naive Bayes Algorithm
Logistic Regression Modeling
Multilayer Perceptron Algorithm
RBF Network Algorithm
Linear Logistic Algorithm
Sequential Minimal Optimization Algorithm
Nearest-neighbour Classifier
K-nearest Neighbours Classifier
K* instance-based Classifier
Locally Weighted Learning Algorithm
Additive Regression Algorithm
Attribute Selected Classifier
Bagging Algorithm
ClassificationViaRegression
Cross-Validation Parameter Selection
Decorate Algorithm
Grading Algorithm
Additive Logistic Regression
MetaCost Algorithm
MultiBoosting Algorithm
Metaclassifier for Multi-Class Datasets
Ordinal Class Classifier
RacedIncrementalLogitBoost
Random Committee Algorithm
Stacking Algorithm
StackingC Algorithm
Vote Algorithm
HyperPipe Algorithm
Voting Feature Intervals Algorithm
DecisionStump Algorithm
Pruned C 4.5 Decision Tree Algorithm
Logistic Model Tree Algorithm
Please use tl
rk Algorithms on the Data

Figure 4: GUI for Machine Learning Algorithms

When all the classes of prediction techniques were evaluated then:

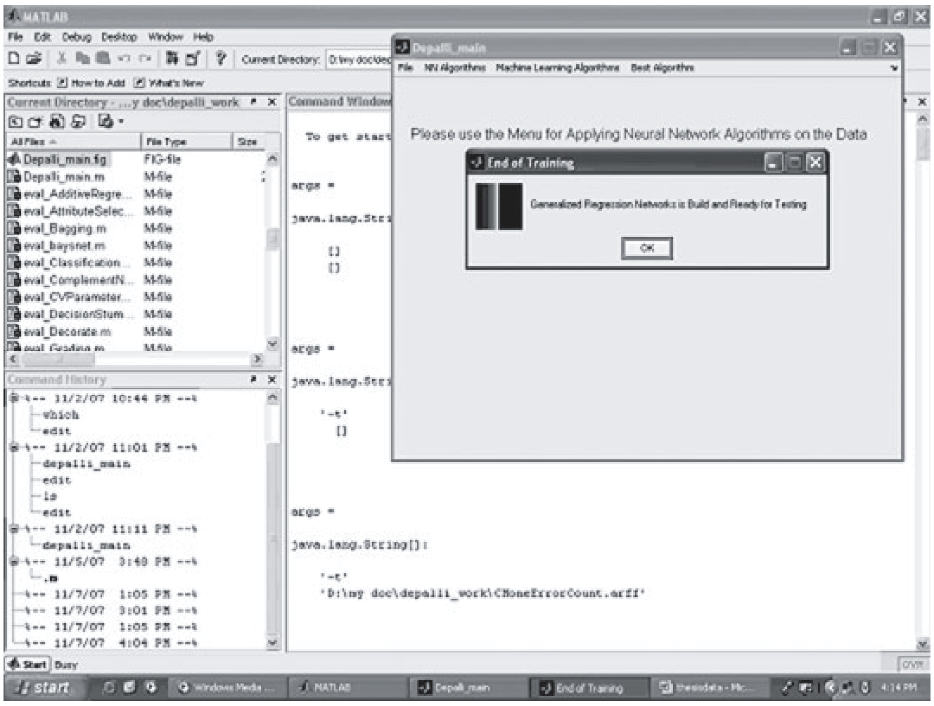
- For Bayes class of machine learning algorithms, the best algorithm comes out to be ComplementNaiveBayes with *MAE* value 0.062 for CM1 project and 0.0337 for PC1 project.
- For Functions class of machine learning algorithms, the best algorithm comes out to be MultilayerPerceptron with *MAE* value 0.04 for CM1 project and 0.0327 for PC1 project.
- For Lazy class of machine learning algorithms, the best algorithm comes out to be IB1 with *MAE* value 0.0528 for CM1 project and 0.0289 for PC1 project.
- For Meta class of machine learning algorithms, the best algorithm comes out to be Grading with *MAE* value 0.0317 for CM1 project and 0.0229 for PC1 project.
- For Misc class of machine learning algorithms, the best algorithm comes out to be DecisionStump with *MAE* value 0.0565 for CM1 project and 0.0417 for PC1 project.
- For Tree class of machine learning algorithms, the best algorithm comes out to be RandomTree with *MAE* value 0.0502 for CM1 project and 0.0316 for PC1 project.
- Among the Neural Network Techniques, the best algorithm comes out to be GRNN with *MAE* value 0.0020 for CM1 project and 0.0127 for PC1 project.

Finally the best model is build and evaluated which comes out to be Generalized Regression Networks as shown in Table 1.

Table 1
Results of All Classes of Machine Learning and Neural Network Techniques

Algorithm	*Projects*			
	CM1		*PC1*	
	MAE	*RMSE*	*MAE*	*RMSE*
Complement Naive Bayes	0.062	0.2491	0.0337	0.1836
Multilayer Perceptron	0.04	0.1477	0.0327	0.1468
IB1	0.0528	0.2298	0.0289	0.17
Grading	0.0317	0.178	0.0229	0.1513
Decision Stump	0.0565	0.1706	0.0417	0.1448
Random Tree	0.0502	0.224	0.0316	0.1763
GRNN	0.0020	0.0009	0.0127	0.0138

The graphical user interface for GRNN shows detail of Generalized Regression Neural Networks as shown in Figure 5.


MATLAB
File Edit Debug Desktop Window Help
Current Directory: D:\my doc\dep
Shortcuts How to Add What's New
Current Directory - ...y doc\depalli_work
All Files
File Type
Size
Command History
Command Window
Depalli_main
File NN Algorithms Machine Learning Algorithms Best Algorithms
End of Training
OK
Start Busy
start
Windows Media ...
MATLAB
Depali_main
End of Training
thesisdata - Mic...
4:14 PM
OVR

Please use the Menu for Applying Neural Network Algorithms on the Data

Generalized Regression Networks is Build and Ready for Testing

All Files	File Type
Depalli_main.fig	FIG-file
Depalli_main.m	M-file
eval_AdditiveRegre...	M-file
eval_AttributeSelec...	M-file
eval_Bagging.m	M-file
eval_baysnet.m	M-file
eval_Classification...	M-file
eval_ComplementN...	M-file
eval_CVParameter...	M-file
eval_DecisionStum...	M-file
eval_Decorate.m	M-file
eval_Grading.m	M-file

```
%-- 11/2/07 10:44 PM --%
  which
  edit
%-- 11/2/07 11:01 PM --%
  depalli_main
  edit
  ls
  edit
%-- 11/2/07 11:11 PM --%
  depalli_main
%-- 11/5/07  3:40 PM --%
  .m
%-- 11/7/07  1:05 PM --%
%-- 11/7/07  3:01 PM --%
%-- 11/7/07  1:05 PM --%
%-- 11/7/07  4:04 PM --%
```

```
  To get start

args =

java.lang.Str

      []
      []

args =

java.lang.Str

    '-t'
      []

args =

java.lang.String[]:

    '-t'
    'D:\my doc\depalli_work\CMoneErrorCount.arff'
```

Figure 5: Generalized Regression Neural Networks

The testing and training of data is done for Generalized Regression Neural Networks and results are analyzed and compared. So, Generalized Regression Networks gives the best performance.

4. CONCLUSION

Fault prediction is used to improve software process control and achieve high software reliability. On comparing all the classes of machine learning algorithms, it is observed that Grading is better technique as compared with other classes of machine learning algorithms. On comparing various neural network techniques, the mean absolute error in case of Generalized Regression Neural Networks comes out to be 0.0020 for CM1 project and 0.0127 for PC1 project, which is much lower as compared to other prediction techniques.

It is therefore, concluded the model is implemented and the best algorithm for classification of the software components into faulty/fault-free systems is found to be Generalized Regression Neural Networks.

The mean absolute error value of proposed derived model is far less than previously proposed algorithms in literature for production of fault tolerance in software systems. The developed model can be used for classifying a faulty system from non-faulty system on the basis of structural attributes of the software systems.

REFERENCES

[1] Fenton, N. E. and Neil, M., "A Critique of Software Defect Prediction Models", *IEEE Trans. Software Engineering,* Bellini, I. Bruno, P. Nesi, D. Rogai, University of Florence, **25**, (5) (Sep 1999), 675–689.

[2] Lanubile F., Lonigro A., and Visaggio G. "*Comparing Models for Identifying Fault-Prone Software Components*", *Proceedings of Seventh International Conference on Software Engineering and Knowledge Engineering*, (June 1995), 12–19.

[3] Giovanni Denaro, "Estimating Software Fault-Proneness for Tuning Testing Activities" *Proceedings of the 22nd International Conference on Software Engineering (ICSE2000)*, Limerick, Ireland, (June 2000).

[4] Mahaweerawat, A., "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, 1–8.

[5] Runeson, Claes Wohlin and Magnus C. Ohlsson, "A Proposal for Comparison of Models for Identification of Fault-Proneness", Dept. of Communication Systems, Lund University, LNLS 2188, (2001), 341–355.

[6] Quah, T. and Thwin, M., "Application of Neural Networks for Predicting Software Development Faults Using Object-Oriented Metrics" *Proceedings of Ninth International Conference on Neural Information Processing (ICONIP'02)*, **5**, (2002), 2312–2316.

[7] Saida Benlarbi, Khaled El Emam, Nishith Geol, "Issues in Validating Object-Oriented Metrics for Early Risk Prediction", by *Cistel Technology 210* Colonnade Road Suite 204 Nepean, Ontario Canada K2E 7L5, (1999).

[8] Eric Rotenberg, "AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors", *Proceedings of the Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, June 15–18, 1999, 84–90.

[9] Bellini P., "Comparing Fault-Proneness Estimation Models", *10th IEEE International Conference on Engineering of Complex Computer Systems* (ICECCS'05), (2005), 205–214.

[10] WEKA (2007), *www.cs.waikato.ac.nz/~ml/weka/.*

Deepali Gupta
Department of Computer Science
GIMT, Kurukshetra University
Haryana, India
E-mail: *deepali_gupta2000@yahoo.com*