International Journal of Information Technology and Knowledge Management
July-December 2012, Volume 5, No. 2, pp. 371-375

EMPIRICAL ANALYSIS OF FAULT PREDICATION TECHNIQUES FOR IMPROVING SOFTWARE PROCESS CONTROL

Kamaljit Kaur*

ABSTRACT: In this paper, we present the application of the neural network for the identification of Reusable Software modules in Oriented Software System. Metrics are used for the structural analysis of the different procedures. The values of Metrics will become the input dataset for the neural network systems and Fuzzy Systems. Training Algorithm based on Neural Network and fuzzy clustering are experimented and the results are recorded in terms of Accuracy, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). The results show that Fuzzy Clustering based technique have shown better results than Resilient Backpropagation on the basis of testing data in terms of prediction technique.Hence the proposed model can be used to improve the productivity and quality of software development.

Keywords: Software reusability, Neural Networks, MAE, RMSE, Accuracy.

1. INTRODUCTION

A software bug is an error, flaw, mistake, failure, or fault in a program that prevents it from behaving as intended(e.g., producing an incorrect result). Most bugs arise from mistakes and errors made by people in either a program's source code or its design, and a few are caused by compilers producing incorrect code [1]. The cost of ?nding and ?xing faults in software typically rises as the development project progresses into a new phase. Faults that are found after the system has been delivered to the customer are many times more expensive to track down and correct than if found during an earlier phase[2].

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” [3]. Software metrics is all about measurement and these are applicable to all the phases of software development life cycle from initiation to maintenance.

The main aim of this work is to model the impact of faults in function based software modules. The main objectives are described as follows:

- To find the structural code and design attributes of software systems
- Find the best algorithms that can be used to model impact of faults in object oriented i.e. the predict the level of impact of the faults in the software system.

This paper is organized as follows: Section two describes the Methodology part of work done, which shows the steps used in order to reach the objectives and carry out the results. In the section three, results of the implementation are discussed. In the last section, on the basis of the discussion various Conclusions are drawn and the future scope for the present work is discussed.

2. PROPOSED METHODOLOGY

The methodology consists of the following steps:

2.1 Find the Structural Code and Design Attributes

The first step is to find the structural code and design attributes of software systems i.e. software metrics. The realtime defect data sets are taken from the NASA's MDP (Metric Data Program) data repository. The dataset is related to the safety critical software systems being developed by NASA.

2.2 Collection of Metric Values

The suitable metrics like product module metrics out of these data sets are considered. The term product is used referring to module level data. The term metrics data applies to any finite numeric values, which describe measured qualities and characteristics of a product. The term product refers to anything to which defect data and metrics data can be associated. In most cases products will be synonymous with code related items such a functions and systems/sub-systems.

2.3 Analyze and Refine Metrics the Metric Values

In the next step the metrics are analyzed, refined and normalized and then used for modeling of fault tolerance in software systems. In the step, table of module levels metrics

* RIMT, Mandi Gobindgarh, Punjab,
Er.kamaljitkaur@yahoo.com

PC5_product_module_metrics is joined with PC5_defect_product_relations and thereafter again the join operation of the resultant table is performed with PC5_static_defect_data. An Entity-Relationship diagram relates Modules to Defects and Defects to Severity of Defects is shown in figure 1.

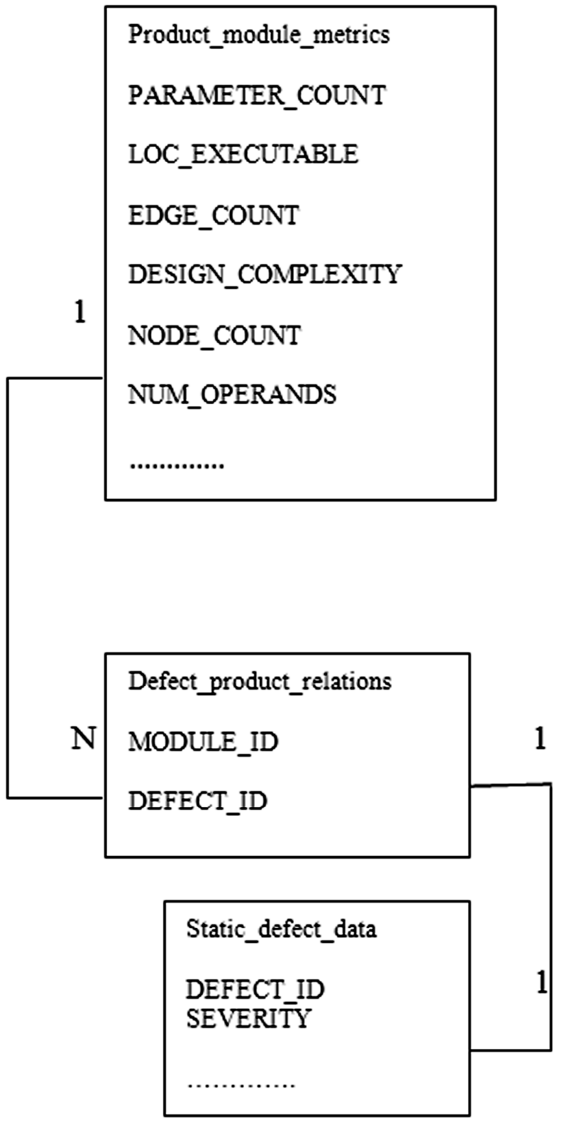

Figure 1: An Entity-Relationship Diagram Relates Modules to Defects and Defects to Severity of Defects

In the figure 1 the MODULE_ID is the unique numeric identifier of the module and DEFECT_ID is the unique numeric identifier of the associated defect. The SEVERITY field in the PC5_static_defect_data table shows the value that quantifies the impact of the defect on the overall environment in the range of 1 to 5. Where, 1 means most severe and 5 being least severe. For example, severity 1 may imply that the defect caused a loss of functionality without a workaround where severity 5 may mean that the impact is superficial and did not cause any major disruptions to the system.

2.4 Explore Techniques Based on Neural Network and Fuzzy

2.4.1 Resilient Back-propagation Algorithm

It is a supervised learning method, and is a generalization of the delta rule. It requires a dataset of the desired output for many inputs, making up the training set. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for "backward propagation of errors". Backpropagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable. The algorithm acts on each weight separately. For each weight, if there was a sign change of the partial derivative of the total error function compared to the last iteration, the update value for that weight is multiplied by a factor η^-, where $0 < \eta^- < 1$. If the last iteration produces the same sign, the update value is multiplied by a factor of η^+, where $\eta^+ > 1$. The update values are calculated for each weight in the above manner, and finally each weight is changed by its own update value, in the opposite direction of that weight's partial derivative. This is to minimize the total error function. η^+ is empirically set to 1.2 and η^- to 0.5.

Phase 1: Propagation

Each propagation involves the following steps:

1. Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
2. Backward propagation of the propagation's output activations through the neural network using the training pattern's target in order to generate the deltas of all output and hidden neurons.

Phase 2: Weight update

For each weight-synapse follow the following steps:

1. Multiply its output delta and input activation to get the gradient of the weight.
2. Bring the weight in the opposite direction of the gradient by subtracting a ratio of it from the weight.

This ratio influences the speed and quality of learning; it is called the *learning rate*. The sign of the gradient of a weight indicates where the error is increasing, this is why the weight must be updated in the opposite direction.

Repeat phase 1 and 2 until the performance of the network is satisfactory.

2.4.2 Fuzzy Clustering

Clustering can be a very effective technique to identify natural groupings in data from a large data set, thereby allowing concise representation of relationships embedded in the data. In our study, clustering allows us to group software modules into faulty and non-faulty categories hence allowing for easier understandability. Main Steps of fuzzy clustering algorithm[6] :

Step 1: Calculate the input data to be clusters.

$$X_{ij}, i = 1, 2, ..., n; j = 1, 2, ..., m$$

n is the number of data

m is the type of data

Step 2: Set the variables value:

i- $r_j, j = 1, 2, ..., m$

ii- η quash factor

iii- ε^* accept ratio

iv- ε reject ratio

v- $X_{j\min}$

vi- $X_{j\max}$

Step 3: Set the normal data value based on *Xj*-min dan *Xj*-max use with the following model:

$$X_{ij}^{norm} = \frac{X_{ij} - X_{j-\min}}{X_{j-\min} - X_{j-\min}}, i = 1,2,...,n; j = 1,2,...,m$$

Step 4: Set the potential of each data point by the formula:

$a = 1$, revise to $a = n$

if $m = 1$, set

$$P'_i = \sum_{i=1}^{n} e^{-4\left\|\frac{x_t - x_k}{r_a}\right\|}, i = 1,2,...,n; k = 1,2,...,n; i \neq k$$

Step 5: Set the highest potential value of data:

$$M = {}_{\max} \lfloor P_i' | i = 1,2,...,n \rfloor$$

$$h - i, \text{ so that } D_i - M$$

Step 6: Set cluster centre and update the potential value that correspond to another data:

i- $Cnt = []$

ii- $V_j = X_{hj}, j = 1, 2, ...m$

iii- $C = 0$ (number of clusters)

iv- $Cnd = 1$

v- $z = m$

vi- Do $Cnd \neq 0$ and $Z \neq 0$

Step 7: Put the real data:

$$Cnt_{ij} = Cnt_{ij} * (X_{j-\max} - X_{j-\min}) + X_{j-\min}$$

Step 8: Set the cluster sigma:

$$\sigma_j = \frac{r_j^* (X_{j-\max} - X_{j-\min})}{\sqrt{8}}$$

2.5 Comparison of the Algorithms

The comparisons are made on the basis of the more accuracy and least value of MAE and RMSE error values. Accuracy value of the prediction model is the major criteria used for comparison. The mean absolute error is chosen as the standard error. The technique having lower value of mean absolute error is chosen as the best fault prediction technique.

- **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 [4]. The formula for calculating MAE is given in equation shown below:

$$\frac{|a_1 - c_1| + |a_2 - c_2| + ... + |a_n - c_n|}{n}$$

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 [7]. It is just the square root of the mean square error as shown in equation given below:

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

The mean-squared error is one of the most commonly used measures of success for numeric prediction. This value is computed by taking the average of the squared differences between each computed value and its corresponding correct value. The root mean-squared error is simply the square root of the mean-squared-error. The root mean-squared error gives the error value the same dimensionality as the actual and predicted values. The mean absolute error and root mean squared error is calculated for each machine learning algorithm i.e. Neural Network.

3. EXPERIMENTAL RESULTS AND DISCUSSIONS

In this section, we discuss the experimental results obtained using Resilient Backpropagation training algorithm, Such algorithm is explored not only for accuracy point of view but also for comparing it with fuzzy clustering. Experiments are conducted on NASA's public domain defect dataset.

3.1 Data Set

As mentioned above, experiments are conducted on NASA's public domain defect dataset. The real time defect data set used is taken from the NASA's MDP (Metric Data Program) data repository, the details of that dataset contains 293 Object Oriented modules with different values of impact of faults labeled as 1, 2, 3, 4 and 5. The severity level 5 are not present in the dataset. So, the level 1 represents the highest severity, level 2 represents of less severity as compared to level 1 and level 3 represents the medium fault and level 4 represent the minor fault that can be overlooked to save time. Details of the Type of Modules in the Dataset are shown in Table 1 in tabular form and Figure 2 in graphical form.

Table 1
Details of the Type of Modules in the Dataset Severity of Impact

Level	*Count*
1	48
2	207
3	28
4	10

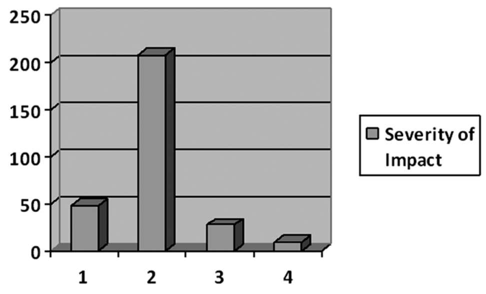

Figure 2: Graphical Representation of Details of the Type of Modules in the Dataset

X axis belongs to Level of Severity of Faults

Y axis belongs to Number of Modules

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

The developed 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.

Mean absolute error, MAE is the average of the difference between predicted and actual value in all test cases. The root mean-squared error i.e. RMSE is simply the square root of the mean-squared-error. The root mean-squared error gives the error value as the same dimensionality as the actual and predicted values.

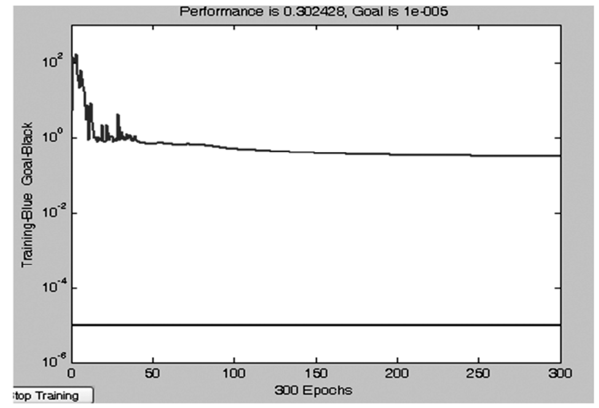

Figure 3: The Training Phase Performance of the Resilient Backpropagation

In the present work the five Neural Network based algorithms experimented in Matlab 7.7 and after the training each trained network is tested with testing dataset of 15 values derived from the PC4 dataset. The overall testing performance of the different algorithms is shown in table 2. The results reveal that the Resilient Backpropagation algorithm have outperformed all other algorithm under study with 0.3980, 0.5385 and 80% as MAE, RMSE and Accuracy values respectively.

Table 2
Performance Results of Different Resilient Backpropagation and Fuzzy Clustering Algorithms

Sr. No.	*Algorithm*	*MAE*	*RMSE*	*Accuracy%*
1	Resilient Backpropagation	0.3408	0.5385	82.2526
2	Fuzzy clustering	0.2982	0.4200	94.2262

REFERENCE

[1] http://puretest.blogspot.in/2009/11/1.html

[2] Barry Boehm and Victor R. Basili. "Software Defect Reduction Top 10 List. Computer", **34(1)**, 135-147, January 2001.

[3] Bibi S., Tsoumakas G., Stamelos I., Vlahavas I., "Software Defect Prediction Using Regression via Classification", *IEEE International Conference on Computer Systems and Applications*, Issue Date: March 8, 2006, pp. 330-336, (http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1618375)

[4] Challagulla V.U.B., Bastani, F.B., I-Ling Yen , Paul, (2005). "Empirical Assessment of Machine Learning Based Software Defect Prediction Techniques", *10th IEEE International Workshop on Object- Oriented Real-Time*

Dependable Systems, WORDS 2005, 2-4 Feb 2005, pp. 263-270.

[5] 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.

[6] Agus Priyono, Muhammad Ridwan, Ahmad Jais Alias, Riza Atiq, O.K. Rahmat, Azmi Hassan and Mohd. Alauddin Mohd. Ali, "Generation of Fuzzy Rules with Subtractive Clustering", *Jurnal Teknologi*, 43(D) Dis. 2005, pp. 143-153, Universiti Teknologi Malaysia.

[7] Hudepohl J.P., S.J. Aud, T.M. Khoshgoftaar, E.B. Allen, and J.E. Mayrand, (1996). "Software Metrics and Models on the Desktop", *IEEE Software*, **13(5)**, 56-60.

[8] Satwinder Singh, K.S. Kahlon, "Effectiveness of Encapsulation and Object-oriented metrics to Refactor Code and Identify Error Prone Classes Using Bad Smells", *ACM SIGSOFT Software Engineering Notes*, **36(5)**, September 2011.

[9] Puneet Mittal, Satwinder Singh, and K.S. Kahlon, "Identification of Error Prone Classes for Fault Prediction Using Object Oriented Metrics", ACC 2, Vol. 191 Springer (2011) , pp. 58-68.