Comparison of Resilient Backpropagation & Fuzzy Clustering Based Approach for Prediction of Level of Severity of Faults in Software Systems

Kritika Gupta, Sandeep Kang, Dr. Parvinder S. Sandhu

Abstract—There is lot of work done in prediction of the fault proneness of the software systems. But, it is the severity of the faults that is more important than number of faults existing in the developed system as the major faults matters most for a developer and those major faults needs immediate attention. As, Neural networks, which have been already applied in software engineering applications to build reliability growth models predict the gross change or reusability metrics. In which majority of faults are found in a few of its modules so there is a need to investigate the modules that are affected severely as compared to other modules and proper maintenance need to be done in time especially for the critical applications. In this paper, Resilient Backpropagation based Neural Network & fuzzy clustering Based techniques are discussed and comparative analysis is performed in order to predict level of impact of faults in NASA’s public domain defect dataset. Predicting faults in the software life cycle can be used to improve software process control and achieve high software reliability. The results show that when the best prediction techniques are evaluated.

Keywords—Neural Network, Software Faults, Software Metric, Fuzzy Clustering, Software Quality.

I. INTRODUCTION

This Faults in software systems continue to be a major problem [1]. A software bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result) [2]. A software fault is a defect that causes software failure in an executable product. In software engineering, the non-conformance of software to its requirements is commonly called a bug. 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. Knowing the causes of possible defects as well as identifying general software process areas that may need attention from the initialization of a project could save money, time and work. The possibility of early estimating the potential faultiness of software could help on planning, controlling and executing software development activities [3]. 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”[4]. Software metrics is all about measurement and these are applicable to all the phases of software development life cycle from initiation to maintenance.

The IEEE Standard Glossary defines metric as a Quantitative measure of degree to which a system, component or process possess a given attribute [5].

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.

II. PROPOSED METHODOLOGY

The methodology consists of the following steps:

A. 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 real-time defect data sets are taken from the NASA’s MDP (Metric Data Program) data repository [6]. The dataset is related to the safety critical software systems being developed by NASA.

B. Collection & Processing 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. The description of the module level metrics is shown in Table 1.

C. Analyze and refine metrics the metric values

In the next step table of module levels metrics PC4_product_module_metrics is joined with PC4_defect_product_relations and thereafter again the join operation of the resultant table is performed with PC4_static_defect_data. An Entity-Relationship diagram relates Modules to Defects and Defects to Severity of Defects is shown in figure 1.

D. Explore Resilient Backpropagation & Fuzzy Clustering Techniques

It is very important to find the suitable algorithm for modeling of software components into different levels of fault severity in software systems. The following resilient backpropagation & fuzzy clustering algorithm are experimented.

Resilient Backpropagation

Multilayer networks typically use sigmoid transfer functions in the hidden layers. These functions are often called "squashing" functions, because they compress an infinite input range into a finite output range. Sigmoid functions are characterized by the fact that their slopes must approach zero as the input gets large. This causes a problem when you use steepest descent to train a multilayer network with sigmoid functions, because the gradient can have a very small magnitude and, therefore, cause small changes in the weights and biases, even though the weights and biases are far from their optimal values.

The purpose of the resilient backpropagation training algorithm is to eliminate these harmful effects of the magnitudes of the partial derivatives. Only the sign of the derivative is used to determine the direction of the weight update; the magnitude of the derivative has no effect on the weight update. The size of the weight change is determined by a separate update value. The update value for each weight and bias is increased by some factor whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations. The update value is decreased by some factor whenever the derivative with respect to that weight changes sign from the previous iteration. If the derivative is zero, then the update value remains the same. Whenever the weights are oscillating, the weight change is reduced. If the weight continues to change in the same direction for several iterations, then the magnitude of the weight change increases.

Resilient Backpropagation can train any network as long as its weight, net input, and transfer functions have derivative functions. In this algorithm Backpropagation is used to calculate derivatives of performance PERF with respect to the weight and bias variables X. Each variable is adjusted according to the following equation:

\[ \Delta X = \Delta Y \times \text{sign}(g_X) \]  

Where the elements of \( \Delta X \) are all initialized to 0 and \( g_X \) is the gradient. For each iteration, the elements of \( \Delta X \) are modified. If elements of \( gX \) changes sign from one iteration to the next, then the corresponding element of deltaX is decreased by delta_dec. If an element of \( gX \) maintains the same sign from one iteration to the next, then the corresponding element of deltaX is increased by delta_inc [9].

In the implementation first the network is created and training is performed on the training data. Thereafter the trained network is tested by testing data in the testing phase. The results of the different algorithms are expressed in terms
of MAE, RMSE and Accuracy values. The details of the different criteria used are in next step. The following steps will be followed to train a Neural Network:

• Load the data
• Divide data into Training, Validation and Test data
• Set number of hidden neurons
• Training is accomplished by sending a given set of inputs through the network and comparing the results with a set of target outputs.
  • If there is a difference between the actual and target outputs, the weights are adjusted to produce a set of outputs closer to the target values.
  • Network weights are determined by adding an error correction value to the old weight.
  • The amount of correction is determined
  • This Training procedure is repeated until the network performance no longer improves.
  • If the network is successfully trained, it can then be given new sets of input and generally produce correct results on its own

E. Comparison of 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 [10]. The formula for calculating MAE is given in equation 2.

  \[ MAE = \frac{1}{n} \sum_{i=1}^{n} |a_i - c_i| \]  

  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 [10]. It is just the square root of the mean square error as shown in equation 3.

  \[ RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (a_i - c_i)^2} \]  

  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.

  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 [7]:

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 = 1, 2, ..., m
  ii- η adjust 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 \( X_{j_{min}} \) dan \( X_{j_{max}} \) use.

Step 4: Set the potential of each data point by the formula:
\[ X_{ij}^{norm} = \frac{X_{ij} - X_{j_{min}}}{X_{j_{max}} - X_{j_{min}}}, i = 1,2,...,n; j = 1,2,...,m \]

  a = 1, revise to a = n

  if m = 1, set

  \[ P_i = \sum_{t=1}^{n} \sqrt{\frac{a - c_i}{a}} \]  

  Step 5: Set the highest potential value of data:

  \[ M = \max \{ P_i | i = 1,2,...,n \} \]

  \[ h = i, \text{ so that } D_h = M \]

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

  i- Cnt = [ ]
  ii- \( V = X_{ij} \), j = 1,2,...m
  iii- \( C = 0 \) (number of clusters)
  iv- Cnd = 1
  v- z = m
  vi- Do Cnd ≠ 0 and Z ≠ 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:

  \[ \sigma_j = \frac{\left( X_{j_{max}} - X_{j_{min}} \right)}{\sqrt{8}} \]

Fuzzy logic is an effective paradigm to handle imprecision.
It can be used to take fuzzy or imprecise observations for inputs and yet arrive at crisp and precise values for outputs. Also, the Fuzzy Inference System (FIS) is a simple and commonsensical way to build systems without using complex analytical equations.

Here, fuzzy logic will be employed to capture the broad categories identified during clustering into a Fuzzy Inference System (FIS). The FIS will then act as a model that will reflect the relationship between the different input parameters.

III. RESULTS & DISCUSSION

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 178 modules of C Programming language with different values of software fault severity labeled present as 1, 2 and 3. The severity level 4 and 5 are not present in the PC4 dataset. So, the level 1 represents the highest severity, level 2 represents the medium and level 3 represents the minor fault that can be overlooked to save time. Details of the type of faults existing in different number of modules of the Dataset are shown in bar chart of figure 2 and numeric values are tabulated in table I.

Fig. 1 Graphical Representation of different Severity of Faults in Modules

<table>
<thead>
<tr>
<th>Severity Level</th>
<th>Count of Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>58</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
</tr>
</tbody>
</table>

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.

The training phase performance of the Resilient Backpropagation is shown in figure 3.

Fig. 3. Training Performance of Resilient Backpropagation

In the present work the five Neural Network based algorithms experimented in Matlab 7.4 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 II. 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>
<thead>
<tr>
<th>Sr. No.</th>
<th>Algorithm</th>
<th>MAE</th>
<th>RMSE</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Resilient Backpropagation</td>
<td>0.3980</td>
<td>0.5385</td>
<td>80</td>
</tr>
<tr>
<td>2</td>
<td>Fuzzy clustering</td>
<td>0.2982</td>
<td>0.4200</td>
<td>100</td>
</tr>
</tbody>
</table>

TABLE III

PERFORMANCE RESULTS OF DIFFERENT RESILIENT BACKPROPAGATION & FUZZY CLUSTERING ALGORITHMS
IV. CONCLUSION AND FUTURE SCOPE

In [19], resilient backpropagation based technique is introduced. The Resilient Backpropagation based modeling technique has outperformed Logistic Model Trees technique (as discussed in [8]) on the basis of the testing data with 80, 0.3980 and 0.5383 as Accuracy, Mean Absolute Error and Root Mean Square Error values. The proposed Fuzzy Clustering based technique have shown better results than Resilient Backpropagation on the basis of testing data with 100, 0.2982 and 0.4200 as Accuracy, Mean Absolute Error and Root Mean Square Error values.

It is therefore, concluded the model is implemented and the best algorithm for modeling of the function based software modules into different level of severity of impact of the fault is found to be fuzzy clustering technique. It is the best prediction technique as compare to the resilient backpropagation technique. It gives more accurate result as compare to the resilient backpropagation technique. Hence, the proposed algorithm can be used to identify modules that have major faults and require immediate attention.

REFERENCES


