Software Fault Prediction Using Fuzzy C-Means Clustering and Feed Forward Neural Network

Kriti Purswani  
Kirtipurswani2750@gmail.com

Pankaj Dalal  
pkjdalal@gmail.com

Dr. Avinash Panwar  
avinashpanwar@gmail.com

Kushagra Dashora  
kushagrashdora@ymail.com

Abstract: Modern systems are primarily based on the software-based systems. Software quality and reliability have become the main concern during the software development. It is very difficult to develop software without any fault. Fault-proneness of a software module is the probability that the module contains faults and a software fault is a defect that causes software failures in an executable project. Early detection of fault-prone software components enables verification experts to concentrate their time and resources on the problem areas of the software systems under development. In this research, a hybrid approach based on K-means clustering-based approach and feed-forward neural network-based approach has been performed with the real-time data set named PC1 taken from NASA MDP software projects. The performance is recorded on the basis of accuracy, MAE, and RMSE values. The performance is better in case of our hybrid approach as compared with the existing approaches.

Keywords: K-means, Neural Network, Fault Prediction

I. Introduction

It is necessary that the developers test the system efficiency at regular intervals of time. For the softwares that are installed in systems like transmission systems, medical units need to be precise and accurate according to the pre-defined parameters. The companies have a dedicated department of Quality assurance that deals to improve the product quality by allocating necessary budget and human resources. Recent advances in software quality estimation yield building defect predictors with a mean probability of detection of 71 percent and mean false alarms rates of 25 percent. Software quality estimation has opened its branches not only in reliability, but also the other quality characteristics such as usability, efficiency, maintainability, functionality, and portability. Software metrics are used as independent variables and fault data are regarded as dependent variable in software fault prediction models [1].

High assurance and complex mission-critical software systems are heavily dependent on reliability of their underlying software applications. A software fault prediction is a proven technique in achieving high software reliability. Prediction of fault-prone modules provides one way to support software quality engineering through improved scheduling and project control. Quality of software is increasingly important and testing-related issues are becoming crucial for software. Although there is diversity in the definition of software quality, it is widely accepted that a project with many defects lacks quality. Methodologies and techniques for predicting the testing effort, monitoring process costs, and measuring results can help in increasing efficiency of software testing. Being able to measure the fault-proneness of software can be a key step towards steering the software testing and improving the effectiveness of the whole process.

II. Related Work

Until now, software engineering researchers have used Case-based Reasoning, Neural Networks, Genetic Programming, Fuzzy Logic, Decision Trees, Naïve Bayes, Dempster-Shafer Networks, Artificial Immune Systems, and several statistical methods to build a robust software fault prediction model. Some researchers have applied different software metrics to build a better prediction model, but recent papers [3] have shown that the prediction technique is much more important than the chosen metric set. The use of public datasets for software fault prediction studies is a critical issue. However, a recent systematic review study has shown that only 30% of software fault prediction papers have used public datasets [2].

Other papers include research work using various classification techniques as

Lanubile & Lonigro [4] presented an empirical investigation of the modeling techniques for
identifying fault-prone software components early in the software life cycle.


SaidaBenlarbi [6] surveyed that the basic premise behind the development of object-oriented metrics is that they can serve, as early predictors of classes that contain faults or that are costly to maintain.

Runeson & Magnus [7] proposed that in striving for high quality software, the management of faults plays an important role. The faults that reside in software products are not evenly distributed over the software modules; some modules are more fault-prone than others.

Briand [8] extracted 49 metrics to identify a suitable model for predicting fault proneness of classes. The system under investigation was medium sized C++ software system developed by undergraduate or graduate students. The eight systems under study consisted of a total of 180 classes.

A. Mahaweerawat [9] analyzed that to remain competitive in the dynamic world of software development; organizations must optimize the usage of their limited resources to deliver quality products on time and within budget.

P. Bellini [10] compared Fault-Proneness Estimation Models that are developed using logistic regression and the discriminate analyses.


G. Pai [12] validated the public domain NASA data set as used in their study to predict fault proneness models with respect to two categories of faults: high and low. In this also used the same data set using a Bayesian approach to predict fault proneness models.


### III. Fuzzy C-Means Clustering

In fuzzy clustering, every point has a degree of belonging to clusters, as in fuzzy logic, rather than belonging completely to just one cluster. Thus, points on the edge of a cluster, may be in the cluster to a lesser degree than points in the center of cluster. An overview and comparison of different fuzzy clustering algorithms is available.

Any point \( x \) has a set of coefficients giving the degree of being in the \( k \)th cluster \( w_k(x) \). With fuzzy \( c \)-means, the centroid of a cluster is the mean of all points, weighted by their degree of belonging to the cluster [14].

Fuzzy clustering is useful in handling unclear boundaries of clusters. One of the most widely used fuzzy clustering algorithms is the Fuzzy C-Means (FCM) Algorithm (Bezdek 1981) [24]. Fuzzy \( c \)-means has been a very important tool for image processing in clustering objects in an image.

Let \( \{x_1, x_2, \ldots, x_N\} \) be a set of \( N \) data objects represented by \( n \)-dimensional feature vectors.

\[ x_k = [x_{1k}, \ldots, x_{nk}]^T \in \mathbb{R}^n \]

A set of \( N \) feature vectors is then represented as a \( n \times N \) data matrix

\[ X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nN} \end{bmatrix} \]

Figure 1: Cluster centers obtained with the FCM algorithm for data with two groups. The larger and
the smaller groups have 1000 and 15 points, respectively.

IV. Neural Network

In mathematical notation, any neuron-k can be represented as follows;

\[ u_k = \sum_{j=1}^{m} w_{kj} x_j \]
\[ y_k = \varphi (u_k + b_k) \]

where \( X_1, X_2, \ldots, X_m \) are the inputs, \( W_{k1}, W_{k2}, \ldots, W_{km} \) are the synaptic weights of the corresponding neuron, \( u_k \) is the linear combiner output, \( b_k \) is bias, \( \varphi() \) is the activation function and \( y_k \) is the output.

A. Artificial Neural Network

Artificial Neural Network is biologically inspired network that are suitable for classification of biomedical signal. A combination of wavelets transform, FCMC and NNs is proposed to classify cardiac arrhythmias. The precision of classification results of the anomalies depends on the number of parameters selected; the number of neurons of input layer is equals to the numbers of FCMC clusters centres.

B. Feed Forward Neural Network

The feed forward neural network was the first and arguably simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

V. Proposed Methodology

The methodology consists of the following steps:

*Find the structural code and requirement attributes:*

The first step is to find the structural code and requirement attributes of software systems i.e. software metrics. PC1 has 320 requirements available and all of them are associated with program modules. All these data sets varied in the percentage of defect modules, with the PC1 dataset containing the least number of defect modules.

*Select the suitable metric values as representation of statement:*
The suitable metric values used are fault and without fault attributes, we set these values in database create in MATLAB R2010 A as 0 and 1. Means 0 for data with fault and 1 for data without fault. When reporting a fault prediction modelling experiment, it is important to describe the characteristics of the datasets. Each data set contains twenty-one software metrics, which describe product’s size, complexity and some structural properties. We use only fault and without attributes to classify the selected NASA MDP PC1 dataset.

![Figure 3: Proposed Flow Diagram](image)

**Classification Results in terms of:**

- Accuracy
- Mean Absolute Error
- Net Reliability
- Root Mean Squared Error
- Computational Time

**Fuzzy Clustering Block:**

Initial data is passed to Fuzzy Clustering classification block in order to classify data according to attributes.

**Feed Forward Neural Network:**

The classified data from Fuzzy clustering block is given to Neural Network to train neural network for these attributes in order to make a supervised classifier.

VI. Results and Discussion

In this section, in order to evaluate the performance and scalability of the proposed initialization method, some standard data sets are downloaded from the UCI Machine Learning Repository (2010). All missing attribute values are treated as special values.

**Soybean Large Dataset**

Relevant Information Paragraph:

There are 19 classes, only the first 15 of which have been used in prior work. The folklore seems to be that the last four classes are unjustified by the data since they have so few examples.

**Zoo Dataset**

Relevant Information:

A simple database containing 17 Boolean-valued attributes. The "type" attribute appears to be the class attribute. Here is a breakdown of which animals are in which type: (I find it unusual that there are 2 instances of "frog" and one of "girl"!)

Class# Set of animals:

1 (41) aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant, fruitbat, giraffe, girl, goat, gorilla, hamster, hare, leopard, lion, lynx, mink, mole, mongoose, opossum, oryx, platypus, polecats, pony, porpoise, puma, pussycat, raccoon, reindeer, seal, sealion, squirrel, vampire, vole, wallaby, wolf
2 (20) chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, sparrow, swan, vulture, wren
3 (5) pitviper, seasnake, slowworm, tortoise, tuatara
4 (13) bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, tuna
5 (4) frog, newt, toad
6 (8) flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp
7 (10) clam, crab, crayfish, lobster, octopus, scorpion, seawasp, slug, starfish, worm

Number of Instances: 101
Number of Attributes: 18 (animal name, 15 Boolean attributes, 2 numerics)

In this study, training and testing methodology is being used, wherein a project is chosen for training the system. The NASA MDP dataset named PC1 is used in this. Then our Fuzzy C-means Clustering and neural network based classification approach is applied on the same project and the final calculated values are then used to classify the modules of project as fault prone or fault free. The classification is based on values of accuracy, MAE and RMSE.

Table 1: Computed Result For Different Approaches

<table>
<thead>
<tr>
<th>Technique</th>
<th>Accuracy %</th>
<th>MAE</th>
<th>RMSE</th>
<th>Reliability %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchical Clustering</td>
<td>64.3678</td>
<td>1.0800</td>
<td>0.0636</td>
<td>63.871</td>
</tr>
<tr>
<td>Neural network</td>
<td>71.6561</td>
<td>1.0352</td>
<td>0.1200</td>
<td>69.154</td>
</tr>
<tr>
<td>Proposed Approach</td>
<td>81.8182</td>
<td>0.1818</td>
<td>0.2010</td>
<td>75.976</td>
</tr>
</tbody>
</table>

In training dataset input, 23 modules has data with positive attributes after the 77 rows of data with negative attributes, if we plot certain defect data, it look like as shown in figure 5. The x axis shows the data modules without fault, and y axis shows data value stores in 21 columns of input dataset.

You can increase the number of clusters to see if our classification can find further grouping structure in the data. This time, use the optional 'display' parameter to print out information about each iteration in the clustering algorithm.
The accuracy of a measurement system is the degree of closeness of measurements of a quantity to that quantity's actual (true) value. In our approach, the achieved accuracy is approximately 81.8182%. If we draw a semilogy plot with this accuracy value, the plot is shown in Figure 9. Semilogy plots data with logarithmic scale for the y-axis. Semilogy of data creates a plot using a base 10 logarithmic scale for the y-axis and a linear scale for the x-axis. It plots the columns of Y versus their index if Y contains real numbers. “Semilogy (data)” is equivalent to “semilogy (real (data), imag (data))” if Y contains complex numbers. Semilogy ignores the imaginary component in all other uses of this function.