Software Fault Prediction Techniques: A Study

Kirti Purswani
M-Tech Scholar (Software Engineering)
Shrinathji Institute of technology and Engineering

Pankaj Dalal
Associate Professor, Computer Engineering
Shrinathji Institute of technology and Engineering

Abstract: **Success of a software system depends not only on cost and schedule, but also on quality. The prediction of software faults, i.e., deviations from specifications or expectations which might lead to failures in operation has been an important research topic in the field of software engineering for more than 30 years. The quality of software components should be tracked continuously during the development of high-assurance systems such as telecommunication infrastructures, medical devices, and avionics systems. Quality assurance group can improve the product quality by allocating necessary budget and human resources to low quality modules identified with different quality estimation models. In this paper we investigate various fault prediction techniques and to conclude which one is best.**

Keywords: **Quality assurance, Quality of software components, Fault Prediction**

I. INTRODUCTION

In the last five decades data from various natural and social sources are stored in massive and complex databases for fast access of information and communication technologies. The clustered data holds various significant parameters to make it compatible in many ranges. For example, the code of biological information is stored in the sequence of DNA and RNA [1]. While the web documents are structured in the format of XML and HTML [2]. However it is nearly impossible to analyze the data by handwork considering speed and accuracy hence various data mining algorithms are designed in order to fetch data by pre-defined computational work. It is necessary that the developers test the system efficiency at regular intervals of time. The companies have a dedicated department of Quality assurance that deals to improve the product quality by allocating necessary budget and human resources. 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.

A software fault prediction is a proven technique in achieving high software reliability and software quality through improved scheduling and project control. A software having defects not only affect quality of a system but also it will reduce age of software. 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.

Some benefits of fault prediction models are:

- Identification of fault prone modules.
- Selection of best design approach from design alternatives.
- Reaching a highly dependable system.

II. RELATED WORK

1. Lanubile & Lonigro [3] presented an empirical investigation of the modeling techniques for identifying fault-prone software components early in the software life cycle.
2. Runeson & Magnus [4] 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.
3. Briand [5] 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
4. P. Bellini [6] compared Fault-Proneness Estimation Models that are developed using logistic regression and the discriminate analyses.
5. Yan Ma [7] suggested that accurate prediction of fault prone modules in software development process enables effective discovery and identification of the defects.

III. FAULT PREDICTION TECHNIQUES

Software engineering data, like any other data, becomes useful only when it is turned into information through analysis. This information can be used to make predictions; thus forming a potential decision support system. Such decisions can ultimately affect scheduling, cost and quality of the end product. However, it is worth keeping in mind that the nature of a typical software engineering data is such that different machine learning techniques might be conducive to play a part in understanding a rather complex and changing software engineering process. Some of the techniques used for software fault prediction:

A. Classification and Regression Trees

Regression trees are parallel to regression/ANOVA modeling, in which the dependent variable is quantitative. Classification trees are parallel to discriminate analysis and algebraic classification methods. Kass (1980) proposed a modification to AID called CHAID for categorized dependent and independent variables. His algorithm incorporated a sequential merge-and-split procedure based on a chi- square test statistic. Kass was concerned about computation time (although this has since proved an unnecessary worry), so he decided to settle for a suboptimal split on each predictor instead of searching for all possible combinations of the categories. Kass's algorithm is like sequential cross tabulation. For each predictor:

- Cross tabulate the m categories of the predictor with the k categories of the dependent variable.
- Find the pair of categories of the predictor whose subtable is least significantly different on a chi-square test and merge these two categories.
- If the chi-square test statistic is not "significant" according to a preset critical value, repeat this merging process for the selected predictor until no non significant chisquare is found for a subtable.
- Choose the predictor variable whose chi-square is the largest and split the sample into subsets, where l is the number of categories resulting from the merging process on that predictor.
- „ Continue splitting, as with AID, until no significant chi-squares result.

B. Case Based Reasoning

Cased-based Reasoning (CBR) is an artificial intelligence method that utilizes knowledge from a past situation to help deal with new complex problems. The beginnings of CBR can be traced back to the work of Shank and Abelson in their introduction of "scripts" for representing knowledge about abstract problem situations. They proposed that these scripts or problem cases could be used to solve new problems by finding and appropriately modifying the closest matching item in memory. Shank's work produced a cognitive model upon which many CBR applications are based today.

The creation of a CBR system involves the indexing of information and the use of domain knowledge to improve search. CBR often works with a smaller hand-tailored collection of data where cases are structured according to domain-specific rules. Indexing and knowledge representation are the two initial stages of CBR, determining the ultimate performance of a CBR engine. The CBR cycle has many variations depending on the application, but at a high level, CBR can be summarized by the steps below:

- Retrieve the most similar cases from memory
- Reuse the retrieved cases and attempt to solve the problem at hand
- Revise the proposed solution (if necessary)
- Retain the new solution in the case memory for future use

C. Fuzzy Clustering

Integration of fuzzy logic with data mining techniques has become one of the key constituents of soft computing in handling the challenges posed by massive collections of natural data [1]. The central idea in fuzzy clustering is the non-unique partitioning of the data in a collection of clusters. The data points are assigned membership values for each of the clusters. The fuzzy clustering algorithms allow the clusters to grow into their natural shapes. In some cases the membership value may be zero indicating that the data point is not a member of the cluster under consideration. Many crisp clustering techniques have difficulties in handling extreme outliers but fuzzy clustering algorithms tend to give them very small membership degree in surrounding clusters [8]

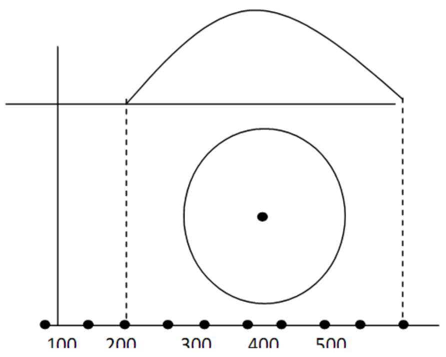

Figure 1: Fuzzy membership in a cluster between zero and one.

The non-zero membership values, with a maximum of one, show the degree to which the data point represents a cluster. As shown in Fig. 1, the points at the centre of the cluster have maximum membership values and the membership gradually decreases when we move away from the cluster centre. Thus fuzzy clustering provides a flexible and robust method for handling natural data with vagueness and uncertainty. In fuzzy clustering, each data point will have an associated degree of membership for each cluster. The membership value is in the range zero to one and indicates the strength of its association in that cluster.

Various types of fuzzy clustering algorithms

K-means: One of the simplest clustering algorithms is K-means clustering method. In the initialization phase, clusters are initialized with random instances and in the iteration phase, instances are assigned to clusters according to the distances, computed between the centroid of the cluster and the instance. his iteration phase goes on until no changes occur in the clusters.

X-means: One drawback of k-means algorithm is the selection of the number of clusters, k, as an input parameter. Rather than choosing the specific number of clusters, k, x-means needs kmin and kmax values. The algorithm starts with kmin value and adds centroids if needed. The BIC or Schwarz criterion is applied to split some centroids into two and hence new centroids are created [9]. Final centroid set is the one that has the best score.

Fuzzy C-means: Fuzzy c-means clustering method was developed by Bezdek [13]. Each instance can belong to every cluster with a different membership grades between 0 and 1 for this algorithm. 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. Fuzzy clustering is useful in handling unclear boundaries of clusters. Fuzzy c-means has been a very important tool for image processing in clustering objects in an image.

D. Neural Network:

An artificial neural network consists of a collection of processing elements that are highly interconnected and transform a set of inputs to a set of desired outputs. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. By modifying the connections between the nodes the network is able to adapt to the desired outputs [10, 11].

Unlike expert systems, which can provide the user with a definitive answer if the characteristics which are reviewed exactly match those which have been coded in the rulebase, a neural network conducts an analysis of the information and provides a probability estimate that the data matches the characteristics which it has been trained to recognize. While the probability of a match determined by a neural network can be 100%, the accuracy of its decisions relies totally on the experience the system gains in analyzing examples of the stated problem.

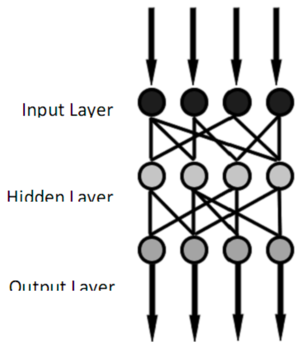

Figure 2: Structure of neural network

Neural network may be classified as:

Artificial neural networks: artificial neural networks are computational models inspired by animals' centralnervous systems (in particular the brain) that are capable of machine learning and pattern recognition. They are usually presented as systems ofinterconnected "neurons" that can compute values from inputs by feeding information through the network. Neural networks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output.

Feed forward neural network: A feed forward neural network is a biologically inspired classification algorithm. It consists of a (possibly large) number of simple neuron-like processing *units*, organized in *layers*. Every unit in a layer is connected with all the units in the previous layer. These connections are not all equal, each connection may have a different strength or *weight*. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called *nodes*.

Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called *feed forward* neural networks.

IV. CONCLUSION

We presented various techniques of fault prediction to estimate software reliability and software quality. In order to achieve software quality faults must be known prior to development so that more emphasis can be made on fault prone areas. By studying about various methods we conclude fuzzy c-means clustering will give appropriate results for fault prediction but neural networks will give accurate results.

V. REFERENCES

[1] Jaakkola T., and Haussler D., “Exploiting generative models in discriminative classifiers”, In Advances in Neural Information Processing Systems 1, MIT Press, pp. 487–493, 1998.

[2] Kazama J., and Tsujii J., “Evaluation and extension of maximum entropy models with in equality constraints”, Proceedings of 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP2003), pp. 137–144, 2003.

[3] 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, 1995.

[4] Runeson, laesWohlin, Magnus C. Ohlsson, “A Proposal for omparison of Models for Identification of FaultProneness”, Dept. of ommunication Systems, Lund University, Profes 2001, pp. 341-355, 2001

[5] Jiang, Y., Cukic, B., Menzies, T., “Fault Prediction Using Early Lifecycle Data”, In the 18th IEEE Symposium on Software Reliability Engineering (ISSRE 2007), IEEE Computer Society, Sweden, pp. 237-246, 2007.

[6] Bellini, P., “Comparing Fault-Proneness Estimation Models”, 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05), vol. 0, 2005, pp. 205-214, 2005. [7] Ma, Y., Guo, L., “A Statistical Framework for the Prediction of Fault-Proneness”, West Virginia University, Morgantown, 2006. [8] Binu Thomas, Raju G., and Sonam Wangmo, “A Modified Fuzzy C-Means Algorithm for Natural Data Exploration” World Academy of Science, Engineering and Technology 25 2009.

[9]J. C. Bezdek, “Pattern recognition with fuzzy objective function algorithms”, Plenum Press, New York, 1981.

[10]Nam Hun Park Won Suk Lee, "Statistical Grid-based Clustering over Data Streams". SIGMOD Record, Vol. 33, No. 1, March 2004.

[11] Peter Cheeseman, John Stutz, “Bayesian Classification (Auto Class): Theory and Results”, 1996.