Software Testing And Defect Analysis Using Soft Computing Concepts

Ayush Raj, Aviral Upadhyay, Vikrant Nakhate

Abstract: Software testing is a wide field which is still a subject of much research. It encompassing several methods and fields related to it include Software defect analysis and fault prediction. Software fault prediction is the process of developing models that can be used by the software practitioners in the early phases of software development life cycle for detecting faulty constructs such as modules or classes. There are various machine learning techniques used in the past for predicting fault, testing software and analysis of defects. This paper aims to provide a comprehensive discussion on the recent trends in these fields and the application of Soft Computing concepts in the for testing and fault detection.

ABBREVATIONS

1. ANN -Artificial Neural networks
2. ANFIS -Adaptive network-based fuzzy inference system
3. SVM -Support vector machine
4. ML -Machine learning
5. SMO -Sequential Minimal Optimization
6. GA -Genetic Algorithms
7. MTBF -Mean time before failure
8. SOM -Self Organizing Maps

1 INTRODUCTION

The real desire from good software is its unwavering quality; accordingly, the quantity of failures that happen when the software is running must be limited to guarantee reliable software. Soft computing methods can be used to predict fault proneness of software using previous data. One way for software fault prediction (SFP) is labeling the modules/classes as fault prone and not fault prone. Using previous data to build a model and test the software on it Machine learning can adapt various techniques; integrate them to form a model to precisely predict the fault in software. Every method can have different metrics for results and different criteria for testing and hence the tester should identify the bestsuited model for their application. Broadly, AUC that is area under ROC curve can be used to judge the outcome of a method. Some of these methods are

1. Genetic Algorithms
2. Adaptive network-based fuzzy inference system
3. Support vector machine
4. Artificial Neural networks

The paper now describes various soft computing based classification techniques for software testing every section has its metrics and methodology described. The result also includes comparison from techniques not described in this paper. Mutation analysis is a software testing procedure utilized to evaluate the ampleness of a test set as far as its capacity to distinguish software deficiencies while Hardware-based fault injection method permits to infuse physical faults.

- *Ayush Raj; Vellore Institute Of Technology – Vellore, Mrayushraj2@gmail.com*
- *Aviral Upadhyay; Vellore Institute Of Technology - Vellore, aviralupadhyay@ymail.com*
- *Vikrant Nakhate; Vellore Institute of Technology - Vellore, vikrant.nakhate2014@vit.ac.in*

The simulation-based fault injections do not operate on the physical device under evaluation, but they target a model of the hardware described using a simulation language, such as VHDL or Verilog. We have also discussed 2 data mining based approaches. Apart from this we have discussed ALPA and a hybrid self-organization map technique. In section 3 of the paper, we have discussed a variety of specific and specialised approaches for software testing and fault detection. These include feed forward networks, State Transition Matrix for smart homes and software testing using computer vision amongst others.

2. SOME BASIC METHODOLOGIES:

2.1 Mutation Analysis

Mutation analysis is a software testing procedure utilized to evaluate the ampleness of a test set as far as its capacity to distinguish software deficiencies. The principle thought comprises in altering the first program with a specific end goal to get a defective program conduct. Mutation analysis comprises in utilizing supporting apparatuses to seed manufactured blames in the first code of the software keeping in mind the end goal to create broken projects that should create mistaken yields. The shortcomings focused by mutation analysis speak to slip-ups that the software engineer makes amid the execution or the particular of the program.

Hardware Based Fault Injection

To assess the system reliability, fault injection is an outstanding and capable strategy to watch the effect of created blunders on the system conduct. It depends on the acknowledgment of controlled analyses to assess the system conduct within the sight of counterfeit faults. Hardware-based fault injection method permits to infuse physical faults (e.g., bit flip fault, stuck at fault) in the objective hardware system

Simulation Based Fault Injection

The simulation-based fault injections do not operate on the physical device under evaluation, but they target a model of the hardware described using a simulation language, such as VHDL or Verilog. They inject faults in the models either at run-

time or at compile-time. Compared to the physical fault-injections, the simulation-based fault-injection techniques are cheaper in term of set-ups and involved hardware, and can better control the fault location. However, their application creates a computational overhead depending on the complexity of the system design.

2.2 Data Mining Approach

Rule induction techniques on the other hand construct very comprehensible results but have the disadvantage of reduced predictive power. Rule extraction is a technique that will compromise between these two requirements by building a simple rule set that mimics the performance of the complex model.

ALPA-An Extraction Algorithm

Keeping in mind the end goal to enhance a rule set as far as either prescient power or loyalty, we utilize one of the already prepared rule acceptance procedures to mimic the yield of the more intricate model that performs better. The path in which this impersonation is acknowledged varies between various rule extraction strategies, however is significant to the execution of the methods.

Classification ALPA

In arrangement, a model ordinarily actuates some sort of choice limit. This choice limit denote the move from one class to another. For arrangement, this locale has been observed to be the best area while upgrading loyalty. For Random Forests the correct parametric function of this limit district is obscure, yet this can be dodged by the era of data in this district utilizing a proxy in light of the supposed instability function for Random Forests.

Semi-supervised Hybrid Self-Organizing Map (HySOM) Metric Thresholds for HySOM

There are three noteworthy ways to deal with decide the metrics threshold values. The primary approach depends on understanding and observational reviews that have been performed in the writing. Tuning machine is the second approach that uses the vault of the tricky things, for example, faulty modules and afterward picks the threshold esteem in light of the likelihood of expanding the quantity of effectively anticipated things. And at last, the last technique looks at different forms of the software program; this strategy does not parameterize an approach with numerous threshold values, but rather an imperative time point of view is added to any suspicious element.

Self-Organizing Maps (SOM)

SOM is utilized for clustering data without knowing the class enrolments of the information data. SOM gives a topology protecting mapping from the high dimensional space to guide units and jelly the relative separation between the focuses, i.e., focuses that are close to each other in the info space are assembled to close-by guide units in the SOM yield space, and the other way around. Keeping in mind the end goal to save properties of topological information space, this model uses an area function.

Hybrid SOM

Adjusting SOM to fault forecast modelling with a specific end goal to rearrange the proposed model, all means are in three unique stages. Stage 1 demonstrates the means amid SOM clustering and stage 2 exhibits how neurons not winning amid viewing for info vectors are expelled from the SOM outline. At last, in stage 3, the means identified with mark the weights in light of the software estimation thresholds are clarified. Yields from stage 3 are nourished to ANN for the preparation procedure.

3. REVIEW OF SOME METHODOLOGIES

3.1 Comparative Analysis of Neural Network and Genetic Programming

Problem Addressed

Software fault prediction can be more useful if, besides predicting software modules being faulty or non-faulty, number of faults can also be predicted accurately. In this paper, the authors present an approach to predict the number of faults in the software system.

Technique Used

The authors have developed fault prediction model using neural network and genetic programming and compare the effectiveness of these techniques over ten project fault datasets collected from the PROMISE data repository. The results of the prediction are evaluated using error rate, recall and completeness parameters.

Metric Used

First, the authors divide software fault datasets into training data and testing data randomly. Subsequently, train the learning algorithms using training data and optimize the control parameters. The trained learning algorithms are then applied over the testing data and finally, they evaluate and validate the results using various evaluation measures.

Result Obtained

The results found that for small datasets, neural network produced better results, while for large datasets genetic programming produced better results. In terms of error values, neural network outperformed genetic programming, while for recall and completeness analysis, genetic programming produced the result better than neural network.

Limitation of the work

The results showed that genetic programing produced the better results compare to neural network. In future, the authors would try to replicate our study over some more datasets to generalize the finding of our experimental investigation.

3.2 Improved Software Defect Prediction using ANN

Problem Addressed

Software defect prediction (SDP) is a most dynamic research area in software engineering. SDP is a process used to predict the deformities in the software. To identifying the defects before the arrival of item or aimed the software improvement, to make software dependable, defect prediction model is utilized.

Technique Used

It is always desirable to predict the defects at early stages of life cycle. Hence to predict the defects before testing the SDP

is done at end of each phase of SDLC. It helps to reduce the cost as well as time. To produce high quality software, the artificial neural network approach is applied to predict the defect.

Metric Used
Nine metrics are applied to the multiple phases of SDLC and twenty genuine software projects are used. The software project data were collected from a team of organization and their responses were recorded in linguistic terms. For assessment of model the mean magnitude of relative error (MMRE) and balanced mean magnitude of relative error (BMMRE) measures are used.

Result Obtained
In this research work, the implementation of neural network based software defect prediction is compared with the results of fuzzy logic basic approach. In the proposed approach, it is found that the neural network based training model is providing better and effective results on multiple parameters.

Limitation of the work
The proposed ANN based approach is effective than the classical fuzzy based approach and giving more accurate and precision based values which are more effective and be used in hybrid approach to a large dataset for better performance aware. In future, following techniques can and efficient results – Decision Tree, Bayesian Net, Random Forest, Honey Bee Algorithm, Swarm Intelligence, Simulated Annealing, Genetic Algorithm.

3.3 Feedforward Neural Network for Predicting Duration of Maintained Projects

Problem Addressed
Once a software project has been developed and delivered, any modification to it corresponds to maintenance. Software maintenance (SM) involves modifications to keep a software project usable in a changed or a changing environment, reactive modifications to correct discovered faults, and modifications to improve performance or maintainability.

Technique Used
The research paper presents related studies on SM duration prediction.,details the statistical analysis of software projects selected based upon the guidelines suggested by the ISBSG, section 4 describes the multilayer feedforward neural network (multilayer perceptron) applied in this study, whereas then it presents the criterion for evaluating the accuracy of the models, describes how the models were trained and tested and then finally compares the accuracy results obtained for the models

Metric Used
Duration prediction is the metric used which is needed (a) for budgeting purposes to consider payments of facilities rental, services, taxes or other kind of payment monthly needed, and (b) for identifying risks of a software project in the sense of knowing the likelihood of finishing a project on time established for the customer.

Result Obtained
Results based on Wilcoxon statistical tests show that prediction accuracy with the MLP is statistically better than that with the statistical regression models when software projects were maintained on (1) Mid-Range platform and coded in programming languages of third generation, and (2) Multiplatform and coded in programming languages of fourth generation.

Limitation of the work
Regarding limitations of this study, although ISBSG dataset contained 5052 projects, only two small data sets of 23 and 17 software projects resulted statistically significant to train and test the MLR and MLP models.

3.4 Software Testing with Computer Vision

Problem Addressed
The blind spot of software testing is the assessment of the actual behavior of the system under test in the real, physical world. In this paper the authors show this inherent restriction of software testing to the “cyber world” can be overcome with the use of methods and techniques from computer vision. It augments conventional software testing and allows making observations about states and events in the physical world as well as the system’s real-world context.

Technique Used
The author implemented a demonstration scenario that shows how visual test automation can be combined with computer vision techniques to include observations of the physical properties of a mechatronic system in software testing. The successful application of the approach in a lab setting revealed several benefits and also some limitations.

Metric Used
The researchers show an illustrative study where they used visual test scripts to specify the expected behaviour at the system-level without the need to intercept signals or to access internal states via test interfaces.

Result Obtained
The proposed use of CV shows a high potential to extend conventional software testing into the physical world and to increase the context-awareness of the tests.
Limitation of the work
Following the limitations are,
Observation but no actuation.
Complexity of image-based observations.
Robustness and maintainability of system tests.

3.5 Software Testing based on State Transition Matrix for Smart Homes

Problem Addressed
The smart home has been developed and widely used with household internet of things (IoT), but the security test of a smart house is still a major problem. Specifically, the smart TV, one of the most important terminals in the smart house, compared to a traditional TV, is larger in scale, more powerful in function and more complex in structure. The software test of the smart TV is more time-consuming than previously, which delays bringing smart TVs to the market. A novel automatic

software testing method based on system design specifications is proposed to improve the smart TV software test efficiency.

Technique Used

Firstly, the behavior of the smart TV is modeled, based on the system design specification with HSTMs (Hierarchical State Transition Matrixes). The scale of the state model of the smart TV is lowered by setting the group state according to the choice of the key nodes based on the importance of the nodes in the network; then, the HSTM model is converted into an expanded regular expression (ERE) with the memory property. Secondly, every closure operator "*" in the ERE is replaced recursively with a certain integral value with the cyclomatic complexity of an ERE in the closure to generate a simplified ERE. Then, a test case is generated from the simplified ERE. Finally, the test cases are converted into python script, and a platform is designed to send the python script to the Android smart TV automatically through its ADB (Android Debug Bridge) interface.

Metric Used

A black testing approach of generating test cases with embedded software based on the state transition matrix (STM) was proposed.

Result Obtained

The experimental results showed that this method not only can cover all of the designing of system functions but also can improve the testing efficiency by the automatic testing platform and reduce the cost.

Limitation of the work

First, the future research will continue to improve the extended regular expression to make the time parameter, probability parameter and cycle parameter more reasonable. Second, the future research will find a new method to test the smart TV in agreement with the complex structure of the smart TV.

4. ARTIFICIAL NETWORK BASED FUZZY INFERENCE SYSTEM

ANFIS is a supervised learning method that combines the advantages of soft computing techniques such as fuzzy inference system (FIS) and Artificial Neural Network (ANN) methods. ANFIS has six layers, from layer 0 to layer 5, the latter produces the output, which in this case is faultiness. ANFIS uses hybrid-learning algorithm, combination of least square estimator and gradient descent. As ANFIS is an ANN, after every use, the model learns and improves its prediction.

4.1 Metrics

McCabe metrics, which is a type of method metric, is used. Primarily because metrics focus on programming concepts and collecting them from the program is relatively easy for the developer or tester. Four McCabe metrics, namely

1. McCabe number of code line
2. Cyclomatic complexity
3. Essential complexity
4. Design complexity

Are used to test the outcomes of these methods.

4.2 Methodology

As discussed, ANFIS has 6 layers with the final objective to obtain the faultiness. Their functions are described briefly

1. Layer 0: Take inputs from the environment (crisp).
2. Layer 1: Fuzzifies the input from layer 0, using a bell shaped function.
3. Layer 2: Incorporates a specific node for every rule, and every node calculates the firing strength of the associated rule using the product operator.
4. Layer 3: !Normalize firing strength
5. Layer 4: Fires the rules
6. Layer 5: Defuzzifies (as contains only one node) to produce the wanted output

5. SUPPORT MACHINE VECTOR

SVM as a method for classification can be used for both, linear and non-linear data. It transforms the training data to a higher dimension using non-linear mapping.

5.1 Metrics

Metrics classified in section 2.1 that are McCabe metrics are to be used to classification using SVM.

5.2 Methodology

SVM finds the hyperplane (the best linear separator hyperplane using new dimensions), and using sequential minimal optimization (SMO) to find the hyperplane that separates the class of data, i.e. faulty or non faulty. Optimization methods are used to find the best suiting support vectors (s), weights (w) and, bias (b). To classify x, to a particular class. Using the formula

$$c = \Sigma wik(si,x)b$$

Where k is kernel function and if $c \geq 0$ then class one otherwise class 2.

6. ARTIFICIAL NUERAL NETWORKS

Also known as ANN is a very primitive model of human brain and is used to learn and then classify vectors into different classes. ANNs are little tolerant to imperfect data encountered while training and hence are a good method to classify software, or programs, which sometimes can be imperfect.

6.1 Metrics

Metrics classified in section 2.1 that are McCabe metrics are to be used to classification using ANN. Using three or four McCabe metrics as inputs and one output that is faultiness.

6.2 Methodology

Scaled conjugate gradient is used, as the model to develop the ANN. It's a feed forward network used to classify input into classes, which in this case is just one, faultiness.

7. GENETIC ALGORITHM

Genetic algorithm is a metahueritic technique based on "natural selection" and survival of the fittest in which best amongst the population are selected and crossed and then their offspring are treated as new population to breed.

7.1 Selecting test cases using GA

Genetic algorithms cannot only be used to predict the faultiness of the software but also selecting test case for software testing, especially black box testing. In black box testing, the key issue is to select the best test cases with minimal cost. Neural network such as back propagation method is used to develop a model for the test case and then use genetic algorithm to improvise and test the model. MATLAB, “trainbr” can be used to train the initial back propagation network with some initial test data. One advantage of using ANN is that they improve with time if used. Now using a fitness function and one can select the code of software to generate test cases and their offspring, which are the best will be the parents for next generation and so on. In every iteration, test cases improve as they mix within each other to provide better results and then they the best ones are picked. The paper tests two software on various parameters [3] and generates test cases using ANN and GA. Results show that using this method one can develop test cases for black box testing effectively.

7.2 Predicating MTBF for software

Mean time before failure series is a way to judge how much failure will occur in software over a given period of time.

7.3 Ant colony optimizer for fuzzy rules design

This module portrays the primary piece of the proposed model, which coordinates both of process and item quality attributes by means of ACO-based fuzzy space. Inside the fuzzy rationale approach, the initial step is to change the discrete qualities into consistent one, this process is called fuzzification. These are then controlled in the fuzzy space by a surmising engine in light of the information base (rule base and database) provided by area specialists. At last, the way toward making an interpretation of back fuzzy individuals into single qualities is named defuzzification. This unit considers every single quality property's estimations gotten from the past stride as inputs and gives a fresh estimation of certification level as yield utilizing the ACO-based rule base. All information qualities can be grouped into fuzzy sets as low (L), medium (M) and high (H). With a specific end goal to fuzzify the inputs, the trapezoidal function is picked. Likewise, the yield variable (quality level) is ordered into fuzzy sets as poor, essential, good and excellent utilizing the triangular membership function. It has been understood that the fuzzy systems create an extensive rule set, thusly changing the translate capacity of the system. Additionally, the created rule base is troublesome, hence bringing about the general surmising system complex. One approach to improve the translate capacity of a fuzzy model comprises of endeavoring to find out rules as general as conceivable so that each rule grasps the highest number of illustrations and, thusly, the span of the rule base decreases. Nonetheless, the target of disclosure the perfect arrangement of such broad rules is not a simple occupation. It is probably going to expand the language structure of the rules by including more than one name to each info variable in the precursor of the rule and by tolerating other social operators various from the typical equal to operator.

8. COMPARISON ON SOFT COMPUTING TECHNIQUES

Software fault prediction done with ANN, ANFIS and, SVM is compared. AUC that is, area under an ROC curve is used as metrics for accuracy. In experiment, five groups of data is presented to the model. With each of the group (train) have 17% of faulty programs (or software), whereas test data has about 18% of faulty programs. SVM had the most error rate, whereas ANN and ANFIS both had a comparable and better than SVM, error rate. With group 4 having the best results for the test, one with three McCabe metrics and the other with four McCabe metrics. Various other methods such as CHAID: Chi-squared Automatic Interaction Detection, CART: Classification and Regression Trees, ADT: Alternating Decision Tree, RF: Random Forest, NB: Naïve Bayes, BN: Bayesian Networks, MLP: Multilayer Perceptron, PNN: Probabilistic Neural Network, RBF: Radial Basis Function, LB: Logit Boost, AB: AdaBoost, NNGE: Neighbor With Generalization, GP: Genetic Programming, ACO: Ant Colony Optimization, RP: Recursive Partitioning, AIRS: Artificial Immune System, KNN: K- Nearest Neighbor, VFI: Voting Feature Intervals, EDER-SD: Evolutionary Decision Rules For Subgroup Discovery, SAPNN: Simulated Annealing Probabilistic Neural Network, VP: Voted Perceptron Can be used to test software or generate test cases.

9. RELIABILITY ASSESSMENT BASED ON NEURAL NETWORK AND FAULT DATA CLUSTERING FOR CLOUD WITH BIG DATA - AN EXAMPLE

Neural networks and fault data clustering is taken as an example to test the theory of neural networks in software fault prediction. Most of the industries are moving towards cloud and the amount of data produced each day shows no sign of slowing down and with emergence of IOT, the amount of data and use of cloud computing is going to be immense in the near future. Hence, more need for lets faults in a cloud system. In the paper [5], number of faults is clustered using time series and then analyzed by neural networks. Faults are clustered together and hence future prediction of faults is easier for the future and if a cloud/big data/software shows symptoms of falling to any such cluster it can be easily picked out and rectified. The same technique can also be used for other developments such as software, websites and not is restricted to cloud.

10. ACKNOWLEDGEMENT

I would like to take this opportunity to thank VIT University for providing me the opportunity to explore new horizons and complete a research paper in the field of Software Engineering. It was an unparalleled learning experience and I will like to thank Dr. Illango P our subject faculty for providing support and guidance throughout the duration of this project.

11. CONCLUSION

We have discussed a variety of methodologies and techniques used for software fault detection in this paper. We have also discussed some relatively common methods and some relatively specific methods. The techniques used for smart home systems and the kind that uses computer vision provide a unique perspective to software testing. The performance accomplished by ANFIS (both three and four parameters) is

good enough for one to consider it for testing software's fault proneness, and is considerably better than SVM. It also enables tester to include vagueness in the input mainly due to use of fuzzy logic, the same "vagueness" cannot be included in many other machine learning algorithms. Several other methods such as genetic algorithm can be used to devise test cases and also test the software. Various combinations of these techniques and improvement is their respective subjects such as neural networks will also lead to an improvement in software testing if these methods are used. The methods mentioned are comparatively nascent with a lot of chance of improvement.

12. REFERENCES

[1]. Ruchika Malhotra, A systematic review of machine learning techniques for software fault prediction – ScienceDirect 2013

[2]. Ezgi Erturk, Ebru Akcapinar Sezer, A comparison of some soft computing methods for software fault prediction- ScienceDirect 2014

[3]. Ruilian Zhao, Shanshan Lv, Neural-Network Based Test Cases Generation Using Genetic Algorithm - IEEE2013

[4]. ZHANG Yongqiang, CHEN Huashan, Predicting for MTBF Failure Data Series of Software Reliability by Genetic Programming Algorithm - IEEE2006

[5]. Yoshinobu Tamura, Yumi Nobukawa, Shigeru Yamada, A Method of Reliability Assessment Based on Neural Network and Fault Data Clustering for Cloud with Big Data, IEEE2015

[6]. Comparative Analysis of Neural Network and Genetic Programming for Number of Software Faults Prediction By Santosh Singh Rathore, Sandeep Kumar

[7]. Improved Approach for Software Defect Prediction using Artificial Neural Networks By Tanvi Sethi, Gagandeep

[8]. Feedforward Neural Networks for Predicting the Duration of Maintained Software Projects By Cuauhtémoc López-Martín

[9]. Poster: What You See Is What You Test - Augmenting Software Testing with Computer Vision by Rudolf Ramler and Thomas Ziebermayr

[10]. Automated Software Testing Based on Hierarchical State Transition Matrix for Smart Home by Kai Cui, Kuanjiu Zhou, Houbing Song and Mingchu Li

[11]. Analyzing and Interpreting the Fault Localized Using PCA with CK Metric by Manpreet Singh Bajwa, Pradeep Kumar Singh, Arun Prakash Agarwal

[12]. Comprehensible software fault and effort prediction: A data mining approach by Julie Moeyersoms, Enric Junqué de Fortuny, Karel Dejaeger, Bart Baesens, David Martens

[13]. Computing with the collective intelligence of honey bees – A survey by Anguluri Rajasekhar, Nandar Lynn, Swagatam Das, P.N. Suganthan

[14]. An empirical study based on semi-supervised hybrid selforganizing map for software fault prediction by Golnoush Abaei, Ali Selamat, Hamido Fujita

[15]. Software test quality rating: A paradigm shift in swarm computing for software certification by Saad M. Darwish

[16]. A comparison of some soft computing methods for software fault prediction by Ezgi Erturk, Ebru Akcapinar Sezer