Survey on Test Generation Using Machine Learning Technique

Naresh E, Vijaya Kumar B P, Madhuri D Naik

Abstract— Software testing and maintenance requires a considerable amount of time in the software lifecycle in order to have a quality product. Recent usage of machine learning algorithms in testing has rapidly increased and different testing type uses different machine learning algorithm depending on the test suits, which helps in automating the test cases in order to decrease the manual effort, which in turn helps in reduction of cost and time spent on the project by the software engineer. An overview of advantages are discussed for various software testing and machine learning techniques is provided along with some of the research on various combination used in order to generate test cases and test suites which are more optimized and performance enhanced.

Keywords: software testing, machine learning, software engineering.

1. INTRODUCTION

Software testing can be defined as a process to validate and verify the different attributes and functionality of the software system that is considered, in-depth investigation takes place to achieve the intended goal, currently as the software system grows it becomes more complex hence the validation and verification is not possible by the only through manual methods which are very primitive, hence the automation of test case methods are emerging [1]. In order to have a successful and optimized automated test case generation the machine learning techniques are very useful. For a given framework the first step is to construct a concrete guidelines that helps in finding a fitting learning algorithm which can be later applied for automation purpose in software life-cycle. The main motivation to optimize these test suite approach is to reduce the time and cost invested during the life-cycle of the software [2].it can significantly enhance the testing process performance due to which there would be reduction in labor intensive and cost process can end up to 50% of the development resource [3]. Lot of research work suggest that employing the techniques of machine learning is giving a very fruitful result.

The conjunction of software testing process and the machine learning algorithms needs to have these dimensions in order to achieve the required optimized results. We are familiar with the different approach for testing i.e. the black-box testing which mainly focuses on the external attributes/description of the given software system, similarly another approach is the white box testing in which the properties are considered internal to the system such as the code length, source code, number of modules, so on. The third approach is the gray box which is conjunction of both white and black box.

Revised Version Manuscript Received on March 08, 2019.

Sony kanhaiyalal Ahuja, Shri Ramdeobaba College of Engineering and Management, Ramdeo Tekdi, Gittikhadan, Katol Road, Nagpur, Maharashtra, India.

When the standard testing procedure is considered for any software life cycle [4], we need to start with basic planning in which the test manager can make a rough estimations on time and cost of the test process, along with this we require a test case management such as the prioritization of test cases, refinement of test suites, selection of test cases, covering of all the different scenario, test case design and evaluation. One more important and useful step is debugging in which localizing the errors can provide the location of the defects in the program exactly.

Some of the existing work have some widely accepted levels for testing such as regression testing, where the new changes in software works along with the older programming which consist of three steps retesting all the test cases, selection of the test cases for regression test, prioritizing the test cases. Acceptance testing concentrates more on systems acquiescence of requirements regarding a specification are met and delivery of software can be forwarded (meeting the software standard and it is acceptable for delivery). Integration testing combines the individual component and then testing takes place for the whole combined component this helps in identifying the errors during the interaction between the units that is integrated. System testing is the higher level testing that happen after integration of all component and final software assembled.

Looking into the machine learning concept which is mainly divided into 3 types called supervised learning, unsupervised learning and reinforcement learning. These algorithms under machine learning whose performance will improve as they are exposed to more data over time. In supervised learning it begins with data that already knows the desired output and data have been labelled, here we use the training data to derive the relationships between the input features and respective desired output. In terms of unsupervised learning it begins with very little or no idea on wat the expected results should looks like, this derives structure from data where we don't necessarily understand the effect of the given variables. Classification and regression come under the supervised learning some main examples are fraud detection, customer retention and attrition, image classification. For regression we have consumption forecast, estimating the most probable outcome advertising popularity prediction.

Dimensionality reduction and clustering are the part of unsupervised learning. Dimensionality reduction used converting the data having a high dimension space set into a lower dimension space set, examples are following big data visualization, meaningful compression, feature elicitation, structure discovery.

For clustering some example are document clustering, target marketing, segmentation.

Reinforcement learning tries to find the best suitable way in order to make specific decision, the software is exposed to an environment where it will get trained continuously using a trial and error approach.

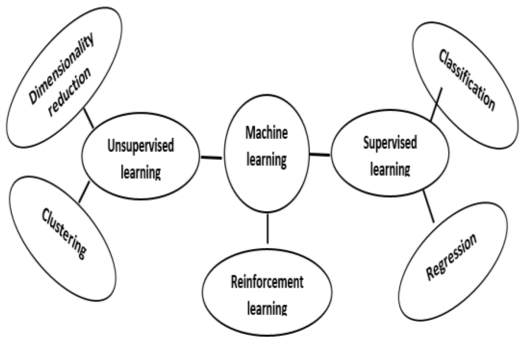


Fig. 1 Three types machine learning approch

II. METHODOLOGY

Study on few types of testing is collected from various papers below on roles of test generation

A. Regression testing

During testing many factor can be dependent such as size, type of testing, reason for testing, available test case/test suite dataset, different performance metrics, etc.

If the testing is black-box then mainly the test cases plays vital role and should cover all test requirement so that acceptance of the software is possible. Process called as re-engineering for the test suite is used which refers to test suites that has been developed with no proper ground and may need refining to get the dependability or to minimize the deadlines that has to be met. The methodology Lionel [5] used is to analyze the drawback of test suit and iteratively provide a better solution, the category-partition is used to abstract the test cases in the form of category and combination, machine learning is applied to get the relationship among the inputs provided and output shown. This gives clear detail to the tester about the capabilities and weakness in the test cases. This methodology ease the improvement of the test case and test suites specification. Consideration of classification [6] in the machine learning is focused as they tried to get the relationship between the input and output. Mainly approached the decision tree algorithm [7] and also the ripper rule induction algorithm. The advantage of the above algorithms is the interoperability of the models ie the certain cases implied gives a particular output equivalence class. The ripper algorithm focus mainly supply the generating rules in a continuous fashion, removing the rules/cases that is covered already. Since the result of the above algorithm are interdependent and forms a decision list and the rules needs to included and applicable in the same stepwise that it was generated. They considered the weka tool [8] for the build and for assessment of the decision tree.

B. System testing

System testing is also important and most of the work is spent on maintenance of a software around 50 to 70% of the time and cost for software is spent on this phase, considering ashish [9] proneness is measured then the probability if class will face any change during the maintenance & these change can affect other modules too. The author is trying to establish a connection between metrics and proneness, the analysis is done on 10 different machine learning technique and also logistic regression to predict how the proneness changed the method used. In first the data is collected. The empirical verification and validation has considered open source OO security software called the XML Security which has a large number of classes so testing needs to be met for acceptance specification. Mainly the procedure can be divided into statistical and Machine learning technic. Logistic regression has been selected for statistical model to give prediction to get various metrics such as CBO,NPRM.the dependent variables are predicted from a set of independent variables, for the later part they have considered 10 algorithm and used weka tool to get results using which the basic performance measurement such as sensitivity, specificity, precision or accuracy of the result obtained.

Sensitivity = [(number of classes correctly predicted as true positive)/ (total number of actual changes for true positive case)]*100

Sensitivity = TP/TP+FN (1)

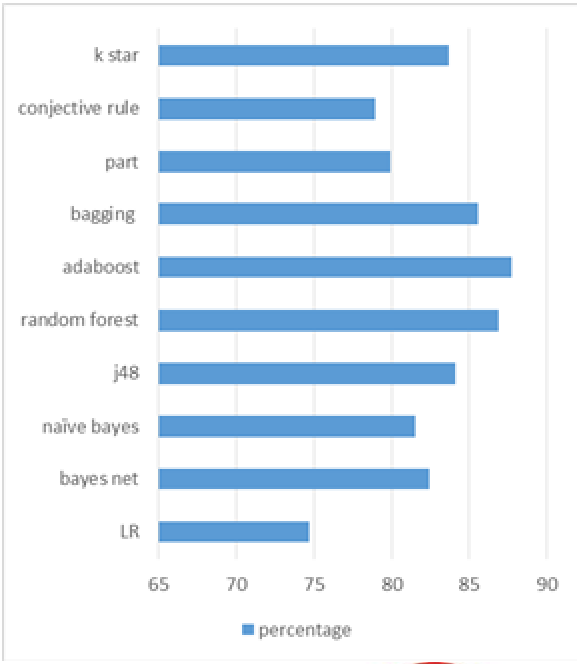
Specificity = [(number of classes correctly predicted as true negative)/ (total number of actual changes for true negative case)]*100

Specificity = TN/TN+FP (2)

The ROC (Receiver operation characteristic) analysis is used to evaluate the models performance for the prediction. Following is a conclusion bar graph of the 10 ML used for the model.

The x axis gives the percentage of area under the curve and y axis shows the different algorithm used for the system testing.

TABLE I. Comparision of different algorithms



From the graph we can analyze that adaboost and Random forest methods resulted better for other machine learning methods, since the area under the curve of ROC is higher [10].

For system level testing author [11] considered a test case history dataset and also the natural language description of the test cases to prioritize, the have used the support vector machine Rank to obtain the prioritizing of test case, mainly by doing this step they are able to shift the focus to important test case which are mandatory and bare minimum.

This approach reduces the work of test expert input and make automatic prioritizing of the test case, the effectiveness has increases and by issuing SVM the test cases are getting ranked by analyzing the result, the fault rate of fault revealing improved and time taken is lesser. The author considered test cases which are labelled as important or not. The features that is considered shows a significant or not, if the dimensionality is higher, which leads to larger computation time and also reduces the performance of prediction [12]. In order to make test case descriptor effective for machine learning, natural language processor (NLP) technique was used, some common features selected are requirement coverage, revealed failure, test execution cost. For measuring the effectiveness they have used average percentage of fault detection. The APFD give value in the range of 0 to 1 where closers to 1 means earlier the error was detected.
APFD can be defined for set of n test case and having m failures found. Following equation [13] gives the value.

$$\text{APFD} = 1 - \frac{\sum_{i=1}^{m} \text{TF}\, i}{nm} + \frac{1}{2n} \qquad (3)$$

The dataset of body comfort system was used and observed that using SVM rank helped to increase the APFD meaning closer to fault detection.

APFD = > 0.8

Another data set of real time situation was used and was observed to have higher around 0.9, the technique used scaled very well as the dictionaries expanded along with larger qualities of input data. These results provides with finding error, this can give more efficient regression testing. When model based software testing is considered, this testing is widely used recently in order to automatically generate test suites.

C. Model-based testing

The author [14] concentrates strongly on infeasible test cases of various scenarios considered in graphical user interface, where all the possible sequences are considered for the event. Some reasons for test cases not available are interdependency between the events, constraints in the event, any bugs in the flow. Earlier some genetic algorithms [15] were used in order to overcome the issues. The method considered for predicting the test case generation are SVM and induced grammar, these come under the supervised learning, these algorithm are having higher effectiveness as the result of the prediction helps in prioritizing the test case and also to remove infeasible test cases. SVM comes under classification and regression which maximize the margin for the labelled data in order to predict which the better binary label that can be applied is. The nearest vector value from the classification's algorithm are named as support vector.

We use some linear algebra in order to achieve the hyperplane in linear SVM, following equation gives prediction for the product of the input considered and the support vector.

Function(y) = B(0) + SUM [mi * (y,yi)] (4)

Where y represents the new input vector that in included in the training data. B(0) gives the coefficient and yi is the vector support that was obtained by the learning algorithm. Since the support vector machine algorithm requires vector values the test cases that is considered also needs to be a vector valued. Three features are considered. The next approach was induced grammar in which each test event was considered and grammar F was created. The F is the base sentence for the grammar, F1 will be the sub-part. Those grammar that violates the constraints of the event are the infeasible test cases which need to remove or give less priority. This helps support vector machine to obtain the vectored dataset for the testing.

The grammar inductive are generally a hard problem and basic idea is

$$G= (A,\sum,B,S) \qquad (5)$$

Where A gives non terminal symbol, ∑ terminal symbol, B rules that give idea on how to work with terminal and non-terminal symbols, S represent the start symbol. The algorithm starts with considering the regular expression and iteratively modifies them by replacing events with regular expression. The result observed was that induced grammar was more limited as cost of computation is high, but it was able to show the event sequence that caused infeasible test cases, overall the classification was able to help the testers to learn about the GUI constraints. Using of SVM helped in extraction of feature and also robustness of training test case for different length was better [16].

D. Metamorphic testing

Oracle problem needs to be tackled in software testing, the oracle plays a vital role in order to automate the test process.

As the oracle is tough to generate an unavailable for many scientific and mathematical software/function, testing becomes major concern. Lack of knowledge on software testing leads to difficulty in detecting errors/faults that is subtle [17]. Metamorphic testing can be used in case of absence of oracle, it specifies changes in the input that can affect the output. The relationship is drawn between the input and the output, so that the difference in these can be accounted as violation. The limitation is that tester can miss few relation that could cause fault.

The Author [1] mainly concentrates on extracting the feature from the software and build a predictive model with the help of machine learning, here supervised learning method is considered as the data is getting labelled. The Machine learning algorithms used are decision tree and support vector machine to check if the functions and module considered in software will satisfy the metamorphic relationship or not.

First the source code was used to obtain the control flow graph, then features of the software was extracted from the control flow graph.

The final step was that the predictive model was created using the Machine learning algorithm for the function.

From the generated control flow the extracted feature are the following

1. Node feature
2. Path feature

The number of occurrence of node and number of occurrence of the path was calculated.

The author considered the following metamorphic relation

1. Additive : which meant changing the value
2. inclusive : which meant including/excluding elements
3. permutation : which meant changing order of the input

The total input for the predicting model was 48 mathematical function along with its feature that is path and node.

The end result was to get the predictive model along with detecting fault from the predicted metamorphic relation. The measurement calculated was accuracy, area under the curve of operating characteristic and false positive rate. The rate of false positive should be lower and the performance can be evaluated using k-fold cross validation. The support vector machine gave better performance because they are less prone to overfitting and accuracy was also high then 80% and area under curve was 90% and above.

For analyzing the fault detection they used mutation analysis and around 66% was able to detect the error. As the dateset was increased it was observed to reduce the fault detection and prediction was not accurate.by using smaller set for training makes it more cost effective.

III. RESULTS & CONCLUSION

From the above various results we can conclude the following observations in table below

Table II comparsion on various testing type

Sl.no	Testing type	Dataset	ML used	Result	size
1	Regression testing	JVM	1.Decision tree 2.Ripper rule induction	1.CP was improved Equivalent to two to three cycles 2.higher fault detection	small
2	System testing	XML security software	10 ML	Machine learning gives better result vs Statistical compared to a random order	medium
3	System testing	Automotive Industry Data Body Comfort System	SVM rank	1. Imply that technique improves the failure 2.detection rate significantly increased	Large
4	Model-based testing	GUI Application	SVM Induced grammar	SVM with pairwise extraction feature was very robust	small
5	Metamorphic testing	Scientific software and mathematical functions	SVM Decision tree	SVM predicted better on smaller dataset	small

The main objective of the paper was to survey on various algorithm’s effect on different type of test generation, in order to reduce the cost machine learning can be applied but the test suite size needs to be small. SVM gives a better result in most of the cases and has higher accuracy and lower false positive rate. The machine learning provides with more reliable result then a statistical method. Optimization of the test suite in order to reduce the time and cost spent on testing procedure has positive effect on the quality of the software and maintenance is also easier.

These finding and survey can be used in future in order to select the correct algorithms for different type of testing depending on the dataset size, environment, and availability of resource.

REFERENCES

1. Mahdi Noorian, Ebrahim Bagheri, Wheichang Du, “Machine Learning –based Software Testing: Towards a Classification Framework”
2. Paul Ammann and Jeff Offutt. Introduction to software testing Cambridge University Press, 2008.
3. Boris Beizer. Software testing techniques (2nd ed.). Van Nostrand Reinhold Co., New York, NY, USA, 1990.
4. Glenford J. Myers and Corey Sandler. The Art of Software Testing. John Wiley & Sons, 2004.
5. Lionel C. Briand, Yvan Labiche, Zaheer Bawar, “Using Machine Learning to refine Black-Box Test Specification and Test Suites” The Eight International Conference on Quality Software” 2008 IEEE.
6. Witten I. H. and Frank E., Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufman, 2005.
7. Quinlan J. R., C4.5: Programs for Machine Learning, Morgan Kaufmann.
8. Csallner C. and Smaragdakis Y., “DSD-Crasher: A Hybrid Analysis Tool for Bug Finding,” Proc. ISSTA, 2006.
9. Ashish Kumar Tripathi, Kapil Sharma, “Optimizing testing effort based on change proneness through machine learning techniques” 2014 IEEE.
10. D’Ambros M, L. M. (2009). On the relationship between change coupling and software defects. 16th working conference on reverse engineering ,pp 35-44.
11. Remo Lachmann, Manual Nieke, Christoph Seidl, Ina Schaefer, Sandro Schulze, “System-level test case prioritization using machne learning” 2016 15th IEEE International Conference on Machine Learning and Applications.

12. K. Gao, T. M. Khoshgoftaar, and A. Napolitano. A hybrid approach to coping with high dimensionality and class imbalance for software defect prediction. pages 281–288, 2012.
13. G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. Prioritizing test cases for regression testing. IEEE Trans. Soft. Eng., Vol.27 No.10:929– 948, 2001.
14. Robert Gove, Jorge Faytong, "Identifying infeasible GUI test cases using support vector machines and induced grammars" 2011 Fourth International Conference on software Testing, Verification and Validation Workshops.
15. S. Huang, M. Cohen, and A. M. Memon, "Repairing GUI test suites using a genetic algorithm," in Proc. 3rd IEEE Int. Conf. Software Testing, Verification and Validation. Washington, DC, USA: IEEE Computer Society, 2010.
16. G. J. Bex, F. Neven, T. Schwentick, and S. Vansummeren, "Inference of concise regular expressions and DTDs," ACM Trans. Database Syst., vol. 35, no. 2, pp. 1–47, 2010.
17. R. Sanders and D. Kelly, "The challenge of testing scientific software," in Proc. Conf. for the Association for Software Testing (CAST), Toronto, July 2008, pp. 30–36
18. [Upulee kanewala, James M Bieman, "Using machine learning technique to detect metamorphic relations for programs without test orcales" IEEE 2013