# Using Machine Learning to Classify Test Outcomes

Marc Roper
Department of Computer and Information Sciences
University of Strathclyde, Glasgow, UK
Email: marc.roper@strath.ac.uk

***Abstract*—When testing software it has been shown that there are substantial benefits to be gained from approaches which exercise unusual or unexplored interactions with a system – techniques such as random testing, fuzzing, and exploratory testing. However, such approaches have a drawback in that the outputs of the tests need to be manually checked for correctness, representing a significant burden for the software engineer. This paper presents a strategy to support the process of identifying which tests have passed or failed by combining clustering and semi-supervised learning. We have shown that by using machine learning it is possible to cluster test cases in such a way that those corresponding to failures concentrate into smaller clusters. Examining the test outcomes in cluster-size order has the effect of prioritising the results: those that are checked early on have a much higher probability of being a failing test. As the software engineer examines the results (and confirms or refutes the initial classification), this information is employed to bootstrap a secondary learner to further improve the accuracy of the classification of the (as yet) unchecked tests. Results from experimenting with a range of systems demonstrate the substantial benefits that can be gained from this strategy, and how remarkably accurate test output classifications can be derived from examining a relatively small proportion of results.**

## I. Introduction

The importance of testing in the software development life-cycle is widely acknowledged and several quite different strategies exist to support the process. Amongst these are approaches like random testing, fuzz testing and exploratory testing, which seek to investigate in particular how a system responds to less predictable and unplanned or unforeseen events. A significant limitation of these approaches is that unless a reliable test oracle exists for the system under test (which is very unlikely), the outputs of the tests need to be checked manually – a problem exacerbated by the fact that these approaches tend to generate very large volumes of tests. Consequently, techniques such as fuzzing are limited to exposing very obvious automatically detectable failures such as security vulnerabilities in the form of buffer overflows and memory leaks (see the work of Miller et al. [1] for example). The approach presented in this paper aims to make such techniques viable for the detection of a much larger class of faults by combining unsupervised and semi-supervised learning.

## II. Overview of a Test Classification Strategy

The test classification strategy consists of two phases:

1) Unsupervised learning (clustering) is used to create an initial grouping of tests where the smallest clusters contain a greater proportion of failures. Manual checking of tests then focuses on these smallest clusters first as they are more likely to contain failing tests.
2) Having checked a small proportion of the test outcomes, semi-supervised learning is then employed to use this information to label an initial small set of data and derive automatic pass/fail classifications for the remainder of the tests.

The combined effect of these is to create a far more efficient process than just checking the outcome of every test in order. Failing outputs are far more interesting than passing ones so finding these earlier is both more important and cost effective. Clustering creates a small subset of tests in which failures are more prevalent, and using semi-supervised learning allows the tester to focus next on those outputs considered to be failures.

In an earlier study [2] we explored a range of clustering algorithms using either just test inputs and outputs, or inputs, outputs and execution traces, and found that small (less than average sized) clusters contained more than 60% of failures (and often a substantially higher proportion). Moreover, as well as having a higher failure density they also contained a spread of failures in the cases where there were multiple faults in the programs.

We have also explored the potential of semi-supervised learning to classify faulty test cases (again using the same data) using a variety of algorithms and found that the most reliable approach came from using inputs, outputs and execution traces along with a mixture of positive and negative labels (i.e. passing and failing test cases) [3].

This paper represents the first step to explore how these two results may be effectively combined.

## III. An Illustrative Case Study

The case study is based upon several versions of the NanoXML system which is available from the Software Infrastructure Repository (SIR)[1]. NanoXML is an XML parser written in Java and consisting of 24 classes and 5 versions (version 3 is examined here, which has 7 faults and 169 supplied test cases), with error rates (the proportion of the test cases which fail) in the range 31-39%. The inputs and outputs to the system along with the execution traces are encoded to make them more amenable to processing by the machine learning algorithms (see earlier papers for details).

[1]http://sir.unl.edu/portal/index.php

## A. *Unsupervised Learning (Clustering)*

The results of the initial clustering can be seen in figure 1 and show the spread of cluster sizes and distribution of failing and passing tests. Different numbers of clusters were explored and it was found that the best results were obtained when the number of clusters was between 15-25% of the number of test cases. The results here are for 15% of the number of tests which resulted in 31 clusters being formed. As can be seen there is a distinct tendency for the smaller clusters to be dominated by failures.

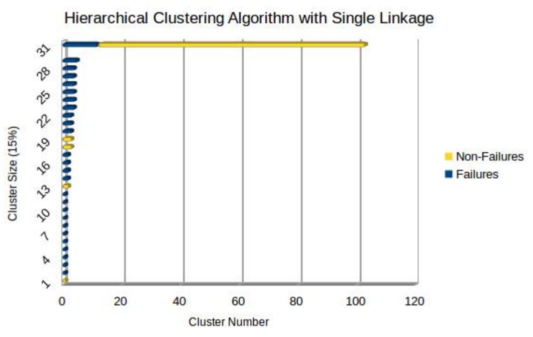

Fig. 1. Clustering Results using Hierarchical Single Linkage

In this version there are 7 distinct failures and 4 of these appear in the smallest clusters and a further 2 are revealed once the clusters of size 4 are examined. By this point just 55 test cases will have been examined. For full details see [2].

## B. *Semi-Supervised Learning*

In semi-supervised learning the learning algorithm is fed a labelled subset of the data - instances for which the correct classification is known - and uses this as the starting point in the construction of a model which classifies the remaining (unlabelled) data. There is a clear trade-off between the accuracy of the classifier and the volume of data used in training, and the challenge is to build the most effective classifier from the smallest amount of data. In the illustration here we look at the impact of labelling just the data associated with the smallest clusters (these are of size 1), and then increasing this to observe the change in performance. The results of this are shown in table I. The results show the cluster size being considered (which includes all smaller clusters), the proportion of data being labeled, the results in terms of the confusion matrix, and the accuracy rate. The confusion matrix is expressed in terms of true positives (TP: A passing test result classified as passing test), true negatives (TN: A failing test result classified as failing test) false positives (FP: A failing test result classified as passing test) and false negatives (FN: A passing test result classified as failing test). Accuracy is just the proportion of tests outcomes which were correctly classified.

The results for this case are particularly encouraging. Even after examining and labelling the clusters of size one (just 8 test cases in this case) a classification accuracy of almost 64% is achieved. This increases to nearly 86% once clusters of size 2 and 3 are also included (another 23 tests, amounting to 31 in total), and is based on less than 20% of the test cases overall. However following this the accuracy drops. Curiously this is caused by the emergence of 2 new failing test cases in the next cluster which appears to cause passing tests to be wrongly classified as failing.

TABLE I
THE IMPACT OF INCREASING LABEL SET SIZE ON SEMI-SUPERVISED LEARNING

| Cluster Size | Data Proportion | Confusion Matrix (TP,TN,FP,FN) | Accuracy |
|---|---|---|---|
| 1 | 4.7% | (39, 69, 0, 55) | 63.9% |
| 2 | 9.4% | (57, 69, 0, 43) | 74.6% |
| 3 | 18.3% | (76, 69, 0, 24) | 85.8% |
| 4 | 32.5% | (49, 69, 0, 51) | 69.8% |
| 5 | 35.5% | (47, 69, 0, 53) | 68.6% |
| 6 | 39.0% | (45, 69, 0, 55) | 67.5% |

## IV. CONCLUSIONS AND FUTURE WORK

We have presented a process for test classification based on clustering and semi-supervised learning. Using an illustrative case study we have shown how a remarkably accurate level of classification can be achieved by first clustering (using their inputs, outputs and execution traces), and then examining and labelling them in cluster-size order. The impact of clustering is to bring to the testers attention a diverse set of failing tests which provides a good basis for the labelling on which semi-supervised learning relies.

However, this is just an illustrative case study and by no means generalisable, and other cases examined reveal variable – but still encouraging – results. Future work will focus on improving the accuracy of classification but primarily the robustness and reliability. The main focus for this is the definition of more sensitive distance measures for the input, output and execution data. In this study we used euclidean distance which does not recognise how close quite similar attributes are – just that they are different. Another feature to explore is the use of confidence measures in the classification as further guidance to the tester.

## REFERENCES

[1] B. P. Miller, L. Fredriksen, and B. So, "An empirical study of the reliability of unix utilities," *Commun. ACM*, vol. 33, no. 12, pp. 32–44, Dec. 1990. [Online]. Available: http://doi.acm.org/10.1145/96267.96279

[2] R. Almaghairbe and M. Roper, "Separating passing and failing test executions by clustering anomalies," *Software Quality Journal*, vol. 25, no. 3, pp. 803–840, 2017. [Online]. Available: https://doi.org/10.1007/s11219-016-9339-1

[3] ——, "Automatically classifying test results by semi-supervised learning," in *27th IEEE International Symposium on Software Reliability Engineering, ISSRE 2016, Ottawa, ON, Canada, October 23-27, 2016*, 2016, pp. 116–126. [Online]. Available: https://doi.org/10.1109/ISSRE.2016.22