ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Predicting Test Execution Times with Asymmetric Random Forests

Francisco L. F. Pereira[1], Hélio M. S. Silva[1]
João P. P. Gomes[1] and Javam de C. Machado[1] *

1- Federal University of Ceará
Department of Computer Science, Brazil
{lucas.falcao, helio.sales, joao.pordeus, javam.machado}@lsbd.ufc.br

Abstract. Being able to estimate a test execution time is of fundamental importance when you need to prioritize tests. Furthermore, it is also important that an estimation algorithm do not underestimate the execution time, since time can be a hard constraint in many problems. If a test take longer than expected, some test that is planned to be executed in the future may have to be cancelled. Under such scenario, in this paper, we developed two simple variants of the Random Forest regression algorithm to predict test execution times in storage diagnostics tests. The proposed methods are compared to a baseline time estimation method (already available in a commercial product) and other machine learning based models. On the basis of our experiments we can state that the proposed variants achieved promising results when considering an asymmetric error metric.

1 Introduction

Software for failure diagnostics in electronic components is a key feature in many computational systems [1]. Nowadays, most computer manufacturers have developed their own diagnostics solution or incorporated third party tools as an important item in their products. Among all monitored components, storage devices are responsible for a significant amount of failures. Hence, being capable to detect incipient failures may avoid severe data losses that will incur in increasing costs and also increase service availability [2].

Although diagnostics systems are available is most computational platforms, in practice, testing faces limitations of time and budget [3]. Since testing resources (time, budget, testers, etc.) are limited, if all the test cases are executed, it may lead to the over usage of resources. Nevertheless, executing less number of test cases may lead to unidentified faults [4].

Given such scenario of resource limitations, it is important to identify criteria that can be used to select the subset of available tests that shall be executed. One can argue that execution time is one important criterion of the decision making problem [5]. If execution time can be predicted, it will help testers to plan the execution of test cases well in advance [4].

Thus, in this work we developed two computational algorithms to predict the execution time of a storage diagnostics test. The proposed methods are variants

*This research was partially funded by Lenovo, as part of its R&D investment under Brazilian Informatics Law, and by CAPES under the grant #88887.609134/2021-00.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

of the well known Random Forest (RF) regression algorithm. More specifically, we proposed two simple ways to combine the regression trees generated by the RF. The main objective of our methods is to provide execution time estimates that do not underestimate the real execution time. In this setting, the prediction algorithm is conservative since it considers the cost of underestimating to be higher than the cost of overestimating. Such characteristic is desirable given that the time may be a hard constraint in real life applications.

2 Background

In this section, we describe the storage test whose execution time we aim to predict and introduce the basic concepts of RF for regression.

2.1 Storage Diagnostics Test

The test we aim to predict the execution time is the Linear Read Test. It is a common test used in many storage diagnostic tools. This test scans storage devices checking for bad blocks, using two-step read sector block verification. The test can also be customized with parameters. As parameters of this test, we have start and end range, they define the testing area and coverage defines the percentage of logical sectors that will be tested within the defined area. These test parameters range from 0% to 100%, and the start parameter must be smaller than the end parameter.

The diagnostic application gives us an estimated time to complete the tests, what we will call the baseline estimation, and was granted by hard code following previous observation on many machines. This baseline was used on our machine learning models to improve results.

2.2 Random Forest

A Random Forest [6] is an ensemble averaging machine learning model defined by a collection of M decision tree predictors $t(x;\theta_m), m = 1,\ldots,M$, where x represents the data point of the input data D and θ are independent and identically distributed outputs of a randomizing variable. For the regression task, the prediction $\hat{y}$ of a data point x can be defined as the unweighted average of the predictions of the trees as the following:

$$\hat{y} = \frac{\sum_{m=1}^{M} t(x;\theta_m)}{M} \tag{1}$$

Each tree of the Random Forest is a set of decision nodes and leaf nodes. The decision nodes takes a feature of the input, evaluates the feature by a split condition and passes the data sample to its branches accordingly. In the training phase, the parameters θ of the split are optimized to make the best split. In the regression task, usually the splits are evaluated by the residual sum of squares (RSS)[6].

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

To improve the performance of an ensemble model we need decision trees to be diversified, this is done by using two well known strategies: bagging and feature randomness. The first strategy is a statistical re-sampling technique that takes random samples with replacement to construct a new dataset with the same size as the original dataset. The second, takes a random set of features. By using these two techniques, each decision tree in the collection will train with a different sampling of the dataset and with an different feature set, thus producing different trees.

3 Proposed Methods

In this section, we propose two random forest variants that can produce biased predictions. Such strategies are useful in our application since underestimating and overestimating the test execution time may have significantly different impacts.

3.1 Asymmetric Random Forest with Weighted Trees (ARFWT)

Let $\hat{y}_j$, $j \in 1 \dots M$ be the predicted value of each tree of the Random Forest and let $\hat{y}$ be the default predicted value of the Random Forest defined as the mean of the outputs of the trees. We can define $\hat{y}_k^-$, $k \in 1 \dots K, K \leq M$ as the values of $\hat{y}_j$ such that $\hat{y}_j \leq \hat{y}$. And we can define $\hat{y}_l^+$, $l \in 1 \dots L, L \leq M$ as the values of $\hat{y}_j$ such that $\hat{y}_j > \hat{y}$.

In other words, we can split the predicted values of the trees of the Random Forest in two sets: the trees that predicted a value smaller than the mean ($\hat{y}_k^-$) and the trees that predicted a value bigger than the mean $\hat{y}_k^+$. With this we can define a new combination of the output of the trees as defined in the following equation:

$$\hat{y}' = \frac{\sum_{k=1}^{K} \hat{y}_k^- + \left(\sum_{l=1}^{L} \hat{y}_k^+ * W \right)}{M} \tag{2}$$

where $W > 1.0$ is a hyper-parameter defining the weight we want to consider to the values above the mean value of the prediction. For example, a value of $W = 1.10$ means we are adding 10% to the predicted output of the trees that produce output that are bigger than the average prediction.

3.2 Asymmetric Random Forest with Biased Estimates (ARFBE)

Let $\hat{y}_j$, $j \in 1 \dots M$ be the predicted value of each tree of the Random Forest and let $\hat{y}$ be the default predicted value of the Random Forest defined as the mean of the outputs of trees. We can define a method of combination of the outputs as:

$$\hat{y}'' = W * \hat{y} \tag{3}$$

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

where $W > 1.0$ is a hyper-parameter defining how much of the original prediction value we want to increase. For example, a value of $W = 1.10$ mean we are adding 10% to the original predicted output.

4 Experiments and Results

4.1 Dataset

The dataset is compound by 14 machines each has 40 executions of the Linear Read Test described above, diversifying the parameters start, end and coverage. The storage size of the machines vary in the range of 128Gb to 1Tb. The features used in the model are: "Start" test parameter, "End" test parameter, "Coverage" test parameter, Storage size, Logical sector size, Logical sectors, Used storage space, Free storage space and Machine memory size.

The test parameters are percentages and therefore can range from a large domain. In order to build the dataset, these parameters were discretized from the domain 0%-100% to $[0\%, 25\%, 50\%, 75\%, 100\%]$ and for each combination an execution of the test was performed to retrieve the baseline (application estimation) and the actual execution time.

4.2 Experimental Procedure

To simulate real scenarios of the use of execution time prediction, instead of the usual split of the examples row-wise, the train-test split was done on machines, separating the diagnostic executions of 9 machines as the train set and the diagnostic executions of 5 machines is the test set.

To evaluate the use of machine learning in this task to improve the execution time estimation, we choose a linear regression with L2 regularization, a regular Random Forest and our two proposed methods. We chose Random Forests as a basis prediction algorithm because its averaging combination of predictions reduces prediction variance [7]. Besides that Random Forest is a powerful non-linear regression model that have succeeded in many applications.

In our experiments we used 100 trees in all the types of Random Forests. Also, we tried three different values of the weight W to our proposed methods: 1.10, 1.15, and 1.20. Those values were used to verify the impact in the results.

To measure the performance of all methods we used two metrics. The first one is the Root Mean Squared Error (RMSE), a standard metric for regression tasks. Also, in regard to one of our motivations, we wanted to assess the performance to penalize underestimates. To do so, we modified the RMSE to create a new metric named Asymmetric Root Mean Squared Error (ARMSE). This new metric is similar to RMSE but it increases the weights associated to underestimated predictions ($Y > \hat{Y}$). In our experiments, we set the weight (λ) value to 10.

To run the experiments we used Python version 3.9, along with the scikit-learn [8] 0.24.2. The proposed methods were developed as modifications of the original RF in the scikit-learn, respecting their BSD 3-Clause License.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

4.3 Results

Method	RMSE	ARMSE
Application Baseline	**11.7542**	**36.5968**
Ridge Regression	38.4721	58.2050
Random Forest	57.7256	58.4508
ARFWT (10%)	65.8917	66.2140
ARFWT (15%)	70.0188	70.2347
ARFWT (20%)	74.1687	74.3202
ARFBE (10%)	68.6977	68.7861
ARFBE (15%)	74.2621	74.2850
ARFBE (20%)	79.8642	79.8673

Table 1: Execution time estimation errors of all regression methods without using the baseline estimation as an input. The best results are indicated in bold

Table 1 shows the RMSE and ARMSE metrics for all regression methods. As can be noticed, no machine learning model was able to outperform the baseline model. A possible explanation is that the selected features are not enough to explain the predicted variable. To overcome such problem, we decided to include the baseline prediction as an additional feature in all models. The results with this new feature set are presented in Table 2.

Method	RMSE	ARMSE
Application Baseline	11.7542	36.5968
Ridge Regression	10.5911	24.6585
Random Forest	**9.1881**	19.0371
ARFWT (10%)	10.8659	17.1199
ARFWT (15%)	12.0356	16.9647
ARFWT (20%)	13.3524	17.2103
ARFBE (10%)	12.5237	**16.1025**
ARFBE (15%)	15.0825	17.0681
ARFBE (20%)	17.9186	18.9406

Table 2: Execution time estimation errors of all regression methods using the baseline estimation as an input feature. The best results are indicated in bold

By observing the results presented in Table 2, we can notice that standard machine learning based models (Ridge regression and Random Forest) outperformed the application Baseline algorithm in both metrics. The fact that the Random Forest algorithm outperformed Ridge regression may indicate that the mapping between the input features and the execution time is nonlinear.

Analyzing the performance of our methods, we can verify that both ARFWT and ARFBE improved the ARMSE metric when compared to the standard

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Random Forest. On the other hand, the RMSEs of both variants in all tested scenarios are higher than the RMSE of standard the Random Forest. Such behavior is expected since our variants tend to penalize more underestimations, thus compromising the RMSE metric. As previously described, such characteristic is desirable in execution time prediction applications. We can also notice that increasing the hyper parameter to penalize underestimations will improve the ARMSE while degrading the RMSE. Thus, in practical applications the use of a validation set is recommended to find the optimal hyper-parameter.

Comparing ARFWT and ARFBE we can notice that ARFWT provide a better compromise between RMSE and ARMSE since ARMSE values are very similar to the ones of ARFWT and RMSE are significantly better. Such result is expected because the weighting strategy of ARFWT is less severe since it weigths just several trees and not the final prediction.

5 Conclusion

In this paper we designed two simple variants of the RF regression algorithm and used both methods for execution time prediction of a hardware diagnostic test. The variants were created to provide biased predictions since the methods shall not underestimate the real execution time. In both models the bias was introduced by modifying the combination function of RF models.

Based on the experiments we can see that both methods had promising results since they were able to outperform all other approaches when considering an asymmetric error metric. It is worth mentioning that the proposed variants do not affect the training procedure of random forest and only modify the ensemble combination strategy. As a consequence, no additional time is required for training of testing. In future works, we intend to evaluate the proposed method in execution prediction tasks on different hardware diagnostics tests.

References

[1] Michael G. Pecht. *Prognostics and Health Management of Electronics*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2008.

[2] Francisco Lucas F. Pereira, Daniel N. Teixeira, Joao Paulo P. Gomes, and Javam C. Machado. Evaluating one-class classifiers for fault detection in hard disk drives. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 586–591, 2019.

[3] Sahar Tahvili. A decision support system for integration test selection, October 2016.

[4] Hasan Ameerjan. Predicting and estimating execution time of manual test casesa case study in railway domain sharvathul. 2017.

[5] Sahar Tahvili, Mehrdad Saadatmand, and Markus Bohlin. Multi-criteria test case prioritization using fuzzy analytic hierarchy process. 11 2015.

[6] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] Gérard Biau. Analysis of a random forests model. *The Journal of Machine Learning Research*, 13(1):1063–1095, 2012.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.