# Effects of Random Search Augmentation with Predictive Termination Criterion on Simple Classification Tasks

Irina-Mona Epure Luiza Dzhidzhavadze Jonathan Collu

LIACS, Leiden University, The Netherlands

**Abstract**

Early termination techniques are a powerful tool for decreasing the runtime of automated machine learning methods. The purpose of this project was to analyse the effects of using the predictive termination criterion to speed up model selection from a pool of 17 different classifiers for 13 datasets. Our experimental results showed that searching for the best model using the predictive termination criterion takes more time than a simple random search over the set of classifiers in the case of simple learning tasks.

## 1 Introduction

Automated methods for model selection and hyperparameter optimization are becoming increasingly popular due to requiring less time and knowledge than human experts, and yielding competitive accuracy results. However, a disadvantage of automated techniques is the inability to perform early termination while training a model that doesn't show a promising trend in performance improvement. Learning curve modelling can be used to approximate the final accuracy achieved by the model being trained after only a few training iterations. Then, early stopping methods can be used to interrupt training for models deemed not competitive with the current known best.

The predictive termination criterion was tested as an early stopping method for neural network hyperparameter optimization. However, experiments in [1] did not extend to simpler machine learning models. In our project, we aimed to evaluate the changes in performance caused by using PTC during model selection for more varied machine learning tasks. To this end, we observe and report the variations in runtime and accuracy produced by augmenting a random search with PTC for 13 different classification problems.

## 2 Background

The scientific paper that we are basing our project on is "Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves" by Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter[1]. It proposes a predictive termination criterion (PTC) that works as a method for speeding up optimization, particularly for deep neural networks. It models the learning curve of a learner over multiple training iterations, starting from the known initial segment of the curve. In order to accurately model the learning curve, 11 different curve models are combined in a linear combination. By using Markov Chain Monte-Carlo (MCMC) sampling, a probability distribution for the curve is obtained. PTC is always used in combination with an optimizer. In every PTC iteration, the optimizer selects a learner or configuration to be trained for a few epochs. Then, the rest of the learning curve is modelled using the weighted linear combination mentioned above. If the expected accuracy at the end of training is lower than the known best with a certainty of at least 95%, then training is not continued for the learner. Otherwise, training continues for another few epochs, and then the expected accuracy is calculated again.

The main purpose of using PTC is to reduce the time spent training networks that are expected to perform poorly. In the experiments performed in [1], PTC sped up two state-of-the-art hyperparameter optimizers (SMAC and TPE) by a factor of roughly two for deep neural networks, and it also resulted in state-of-the-art accuracy on the CIFAR-10 dataset.

## 3 Related Work

This project entails comparing the results of random search with and without PTC for model selection in the case of low-complexity classification tasks. We followed the experimental procedure of [3]. They compared their proposed Learning Curve Cross-Validation (LCCV) method with 10-Fold Cross Validation (10CV) by measuring the runtime and resulting accuracy of model selection with a random search AutoML tool that chooses 1000 candidate classifiers to be evaluated on 67 datasets from the AutoML benchmark suite described in [2]. In order to simulate a proper random search, they generated a randomly ordered sequence of 1000 of learners which are copies of 17 Scikit-learn classifiers with default parameters. Both model selection techniques were applied to each classifier by iterating on the sequence mentioned before. We used the same approach to make a comparison between regular and PTC-augmented random search.

## 4 Experiments

### 4.1 Datasets and Classifiers

This subsection shortly describes the datasets we experimented with, and the classifiers considered for the random search. We performed our experiments on 13 datasets from the OpenML benchmark suite. The main characteristics of each of them are presented in Table 1. We used 17 classifiers with their default parameters from the scikit-learn library: Linear SVC, Decision Tree Classifier, Extra Tree Classifier, Logistic Regression, Passive Aggressive Classifier, Perceptron, Ridge Classifier, SGD Classifier, MLP Classifier, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Bernoulli NB, Multinomial NB, K Neighbors Classifier, Extra Trees Classifier, Random Forest Classifier, and Gradient Boosting Classifier.

| ID | Dataset Name | No. of Instances | No. of Attributes |
|---|---|---|---|
| 12 | mfeat-factors | 2000 | 217 |
| 54 | vehicle | 846 | 19 |
| 1464 | blood-transfusion-service-center | 748 | 5 |
| 1468 | cnae-9 | 1080 | 857 |
| 4134 | Bioresponse | 3751 | 1777 |
| 4135 | Amazon$_employee_access$ | 32769 | 10 |
| 23517 | numerai28.6 | 96320 | 22 |
| 40670 | dna | 3186 | 181 |
| 40900 | Satellite | 5100 | 37 |
| 40996 | Fashion-MNIST | 70000 | 785 |
| 41027 | jungle-chess-2pcs-raw-endgame-complete | 44819 | 7 |
| 41138 | APSFailure | 76000 | 171 |
| 41146 | sylvine | 5124 | 21 |

Table 1: Descriptions of datasets

## 4.2 Experimental Procedure

The implementation of PTC is already available online, but we wanted to test it on a wider range of classifiers and datasets than in [1]. Because the original paper's implementation of PTC is specifically designated to be used for neural network hyperparameter optimization, we implemented a different procedure for performing model selection for simple classification tasks. We repeated the experiment for each of the datasets specified in Table 1.

Our experiment consisted of first performing a simple random search over the set of classifiers. The search ran for 30 iterations, each time selecting a random model from the list and and training it fully on the training set, before evaluating its classification accuracy on the test set. Then, we repeated this procedure by adding the usage of PTC for early termination of bad learning processes. Because epochs are not commonly used for training low-complexity machine learning models, we chose to base training iterations on the amount of data used for training a classifier. During every iteration, the training set was divided into 10 folds containing and equal number of datapoints. After training the randomly selected model on each fold, the current classification accuracy was evaluated on the test set and a learning curve was modelled for the learner based on previous accuracy values. The predictive termination criterion was then applied such that training was discontinued on the remaining folds of the dataset if at any point the predicted final accuracy of the model was expected to be lower than the known best with a certainty of at least 95%.

During both approaches, the best performing model, and the runtime and accuracy achieved were saved for ulterior analysis. The entirety of the code used for the implementation of this experiment is linked in the footnotes [1].

[1]https://github.com/JonathanCollu/Group4AMLProject

## 5 Results

The chosen classification models and their performances on the corresponding datasets are reproduced in Table 2 for the simple random search, and in Table 3 for the PTC-augmented search. A visual comparison between the two approaches is shown in Figure 1. From the obtained data, it becomes apparent that augmenting the random search with PTC causes the runtime to increase exponentially, leading to poor performance.

The accuracy also diminished with up to 15% when using PTC, and the true best learner was only found in one case. It is expected for the simple random search to result in higher accuracy than the PTC-augmented one. Due to using a set random seed, both of the searches selected the same classifier to train at each iteration. But the two methods differ because, during the simple random search, model training was carried out until the end, leading to certain knowledge of the best classifier for each task. Meanwhile, training was often dropped for many learners when PTC was used together with the random search. This always involves some risk of performing early termination on high-perfomring learners.

## 6 Discussion

From our results, we can observe that searching for the best model while using PTC is more time-consuming than without it in the case of simple machine learning tasks. This can be explained by the fact that all of the datasets included in the experiments are relatively small "toy" datasets, and are usually fast to train on for low complexity machine learning models. However, modelling learning curves through the method proposed by [1] is quite time consuming. Thus, modelling a learning curve for use with PTC multiple times while training a classifier actually ends up requiring a lot more time than directly training it on the entire dataset. This renders the method inefficient

| # | Dataset | Chosen Learner | Accuracy | Run time (sec) |
|---|---|---|---|---|
| 1 | 12 | LogisticRegression | 0.722 | 75.15 |
| 2 | 54 | QuadraticDiscriminantAnalysis | 0.694 | 2.93 |
| 3 | 1464 | MultinomialNB | 0.766 | 2.22 |
| 4 | 1468 | BernoulliNB | 0.777 | 69.39 |
| 5 | 4134 | GradientBoostingClassifier | 0.704 | 256.168 |
| 6 | 4135 | BernoulliNB | 0.942 | 59.492 |
| 7 | 23517 | LogisticRegression | 0.518 | 516.99 |
| 8 | 40670 | MultinomialNB | 0.851 | 19.79 |
| 9 | 40900 | MultinomialNB | 0.993 | 6.89 |
| 10 | 40996 | GradientBoostingClassifier | 0.756 | 12961.30 |
| 11 | 41027 | MLPClassifier | 0.754 | 113.079 |
| 12 | 41138 | SGDClassifier | 0.921 | 4.7894 |
| 13 | 41146 | GradientBoostingClassifier | 0.8478 | 9.0434 |

Table 2: Experimental results of runs without PTC

| # | **Dataset** | **Chosen Learner** | **Accuracy** | **Run time (sec)** |
|---|---|---|---|---|
| 1 | 12 | MultinomialNB | 0.65 | 14710.18 |
| 2 | 54 | LogisticRegression | 0.611 | 20879.66 |
| 3 | 1464 | LogisticRegression | 0.766 | 29043.408 |
| 4 | 1468 | LogisticRegression | 0.634 | 21941.35 |
| 5 | 4134 | MultinomialNB | 0.657 | 33092.54 |
| 6 | 4135 | BernoulliNB | 0.942 | 24812.325 |
| 7 | 23517 | LogisticRegression | 0.513 | 25321.37 |
| 8 | 40670 | BernoulliNB | 0.797 | 14990.05 |
| 9 | 40900 | LinearDiscriminantAnalysis | 0.99 | 23234.984 |
| 10 | 40996 | GradientBoostingClassifier | 0.722 | 19481.359 |
| 11 | 41027 | MLPClassifier | 0.747 | 27085.057 |
| 12 | 41138 | ExtraTreesClassifier | 0.894 | 31101.358 |
| 13 | 41146 | RandomForestClassifier | 0.832 | 24202.225 |

Table 3: Experimental results of runs with PTC

for use with such small-scale classification tasks.

In contrast, when PTC is used for hyperparameter optimization in [1], the runtime is shortened because the experiments focus on DNNs training on large image datasets (MNIST, CIFAR-10, and CIFAR-100). This type of classification problem takes a much longer time to train a DNN for. Thus, it makes sense to take some time to model the learning curve of a network instead of taking a longer amount of time to train it on the entire dataset from the beginning. In essence, this is the reason why PTC performs so much worse in our experiments with simple classification problems, than in Domhan's experiments with DNNs.

## 7 Conclusions and Further Research

In this project, we performed a comparison between simple and PTC-augmented random search over a set of machine learning models in the case of 13 simple classification tasks. Our experimental results showed that searching for the best model using a simple random search outperforms the PTC method in terms of runtime in 100% of cases and in terms of accuracy in 85% of experiments. As future work, we would like to try to reduce the complexity of learning curve modelling for the PTC algorithm in order to make it applicable to simple classification tasks.

## References

[1] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[2] Pieter Gijsbers, Erin LeDell, Janek Thomas, Sébastien Poirier, Bernd Bischl, and Joaquin Vanschoren. An open source automl benchmark. *arXiv preprint arXiv:1907.00909*, 2019.

[3] Felix Mohr and Jan N van Rijn. Towards model selection using learning curve cross-validation. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021.

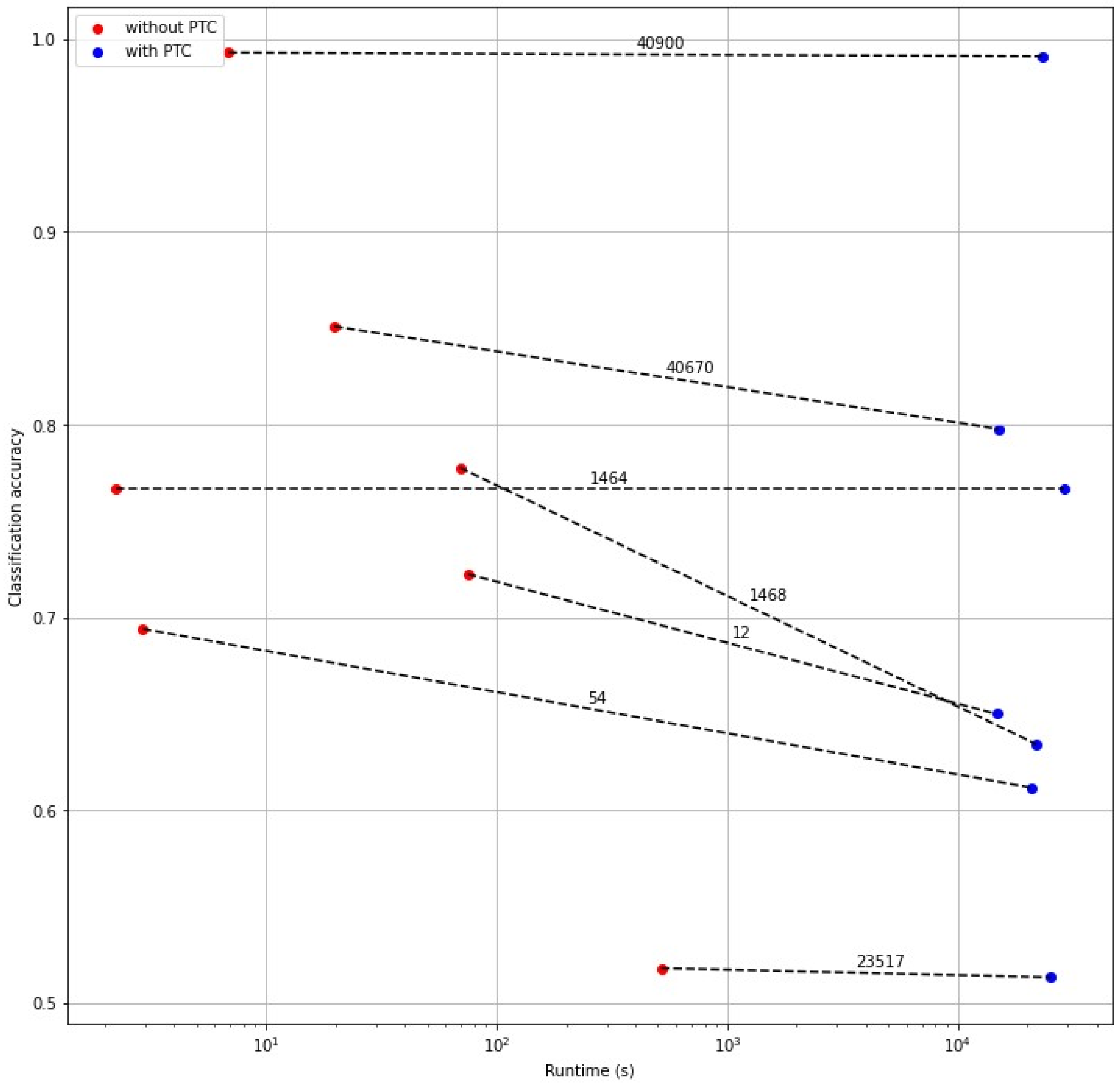

Figure 1: Comparison of the runtimes and accuracies achieved using a random search without (red) and with PTC (red). Each point is a visual representantion of the running time nessary for the random search (on the horizontal axis), and the best accuracy reached (on the vertical axis). Results referring to corresponding datasets are united through dashed lines. The numeric labels represent the dataset IDs.