AutoML - Hyperparameter Optimization

LanGe Chen - LIACS, Leiden University

1 Introduction

There are large data volumes generated by customers every day for a tech company, which can be used to analyse and optimize business processes. While there are different types of machine learning techniques to learn meaningful functions from such data, each technique requires a set of hyperparameters that may have large impact on the algorithm performance. This assignment aims to investigate different strategies to automatically perform hyperparameter optimization and evaluate the efficiency and quality of the performance on the given datasets. A key question is how well certain algorithms can scale under different dataset sizes, and how robustly and efficiently such sets of suitable hyperparameters can be found. In particular, the larger the datasets, parameter spaces and the more expensive the evaluation function become, aiming to find an approximate solution of hyperparameters may be more efficient than aiming to find the optimal solution.

2 Background

Hyperparameters play a crucial role for machine learning algorithms as they directly control the behavior and expressiveness of the model to fit the data. However, tuning each parameter manually can be a labor-intensive and time-consuming process, especially for larger spaces, and can sometimes require expert knowledge. Among popular techniques such as *Random Search*, *Grid Search*, *Bayesian Optimization*, *Evolutionary Strategies* or *Deep Learning*, the following paragraphs will describe the former three as relevant part of this project:

- **Random Search** uses a probability distribution, typically uniform, to randomly select a set of values from the parameter space to train the model. The sampling strategy should enable every possible parameter combination to be drawn under infinite iterations and can use both discrete and continuous feature dimensions.

- **Grid Search** is to deterministically search through the parameter space to test every possible combination one after another. It requires the search space to be *discretized* to a finite set of arguments in each dimension, and to heuristically define the sampling resolution.

- **Bayesian Optimization**[1] utilizes a probabilistic model of the learner, directly predicting the mapping from parameters to score. The aim is to shift expensive slow *computations* of the true evaluation of the learner to cheaper and faster *estimations* by the probabilistic model while iteratively updating the predictive function with more points being sampled.

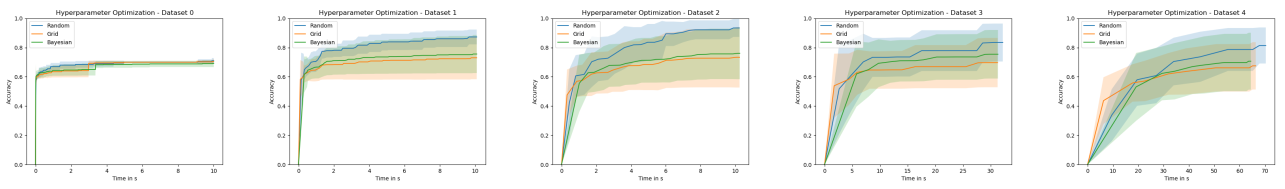

Figure 1: Performance of Random, Grid and Bayesian search on different sizes of datasets.

3 Experiments

In the implementation[1], the dataset is first divided into 90% of the training set and 10% of the test set, while training the model against the former and evaluating it against the latter to avoid overfitting. The hyperparameter space for training the SVM model includes *C, kernel, degree, gamma, coef0, shrinking, probability*. The range of hyperparameters was chosen with meaningful lower and upper bounds for continuous (i.e. regularization C) and discrete values (i.e. degree). Boolean parameters are expressed as 0 or 1, and categorial features are encoded as one-hot vectors. For reasonable evaluation, each model is run over 10 iterations with a timeout limit of $[10, 30, 60]$ seconds depending on the dataset size. Fig 1 shows the performance of the different optimization techniques for the given datasets. It can be observed that Random search performs consistently well across all datasets, Bayesian search improving with the size of the dataset, and Grid search performing least efficient. One important property about Random search is that it aims to cover the distribution of parameter selections equally well independent from the time, while Grid search instead linearly iterates through the search space. Therefore, Grid search can find the optimal solution within its predefined search space, but often suffers from a combinatorial explosion of required iterations to test each parameter set. Random search, however, can be stopped at any time, which suggests scaling better under large dimensionality. Bayesian optimization can be considered an enhancement over pure random search as it guides the sampling strategy to draw more promising samples from the search space that would most likely maximize the score. It can be particularly efficient when evaluations become computationally expensive, such as for very large datasets.

4 Discussion and Conclusions

In conclusion, Grid search iterates all parameter combinations and leads to slow performance, often only covering a small percentage of possible solutions within the given time limits. Random search has been found to perform more efficiently than Grid search, even though the sampling strategy is relatively simple compared to the other methods. Also, in practice there can be multiple parameters that lead to an equally good score, making Random search a suitable technique for this problem. Bayesian optimization in contrast formulates the parameter search as an additional function to be optimized and can provide a more structured sampling strategy to enhance Random search. However, one should keep in mind that the hyperparameter optimizer should remain less sensitive to its own parameters in order to avoid an additional loop of black box parameter choices. In summary, Bayesian optimization and Random search both perform consistently robust for the given datasets, while Grid search only appears useful in low-dimensional parameter spaces.

[1]https://github.com/LanGe-Chen/AutoML/tree/main/Hyperparameter_Optimization

References

[1] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms, 2012.