ML1819 Research Assignment 1

Team 35

Task 106 – Hyper-Parameter Tuning: How strong is the impact on ML performance?

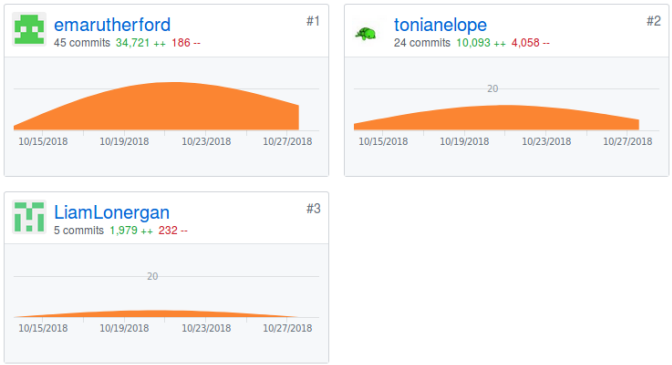
Antonia Baumann (15336726): Multi-Layer Perceptron Classifiers implementation, experiments & visualisations for HPT in MLPC, paper write-up
Liam Lonergan (15322130): Random Forest Classifier implementation, experiments for HPT in RandomForests & paper write-up
Eleanor Rutherford (15314908): Support Vector Machine (classifier) implementation, experiments & visualisations for HPT in SVM, paper write-up

Word Count: 982

Link to Git Repo:
https://github.com/tonianelope/ML1819--task-106--team-35

Link to Git Commit Activity:
https://github.com/tonianelope/ML1819--task-106--team-35/graphs/contributors

Commit Activity Screenshot:



An Investigation of Hyper Parameter Tuning Methods with Scikit-Learn

Antonia Baumann
Trinity College Dublin
Dublin 2, Ireland
abaumann@tcd.ie

Liam Lonergan
Trinity College Dublin
Dublin, Ireland
llonerga@tcd.ie

Eleanor Rutherford
Trinity College Dublin
Dublin, Ireland
erutherf@tcd.ie

1 INTRODUCTION

Hyperparameters are an essential part of every machine learning algorithm (MLA), however choosing them often requires experience and expertise with the MLA and datasets at hand. Hyperparameter tuning (HPT) provides an algorithmic approach to find optimized values.

In this paper, we set out to 1) investigate the impact of hyper-parameter tuning on the performance of a set of machine learning algorithms, whose hyperparameters have been optimized using four hyperparameter optimization methods (Grid Search, Randomized Search, Bayesian Optimization and Tree-structured Parzen Estimator) and 2) compare the performance of the aforementioned optimization methods in terms of efficacy and efficiency.

2 RELATED WORK

That tuning hyperparameters improves the performance of ML algorithms is a well-established fact. [5] [3] [4] Research in this area predominantly focuses on choosing the best algorithm for tuning hyperparameters, in terms of both efficacy and efficiency. Bergstra and Bengio [2] compared grid search and random search, empirically and theoretically showing that "randomly chosen trials are more efficient for hyper-parameter optimization than trials on a grid," as GR wastes time optimizing parameters which are not relevant to performance. In another paper Bergstra et al. [1] introduce Sequential Model-based Global Optimization (SMBO) methods, which choose the parameters by sequentially modeling the hyperparameter performance based on previous data, showing that these methods perform better, compared to Random Search, given the same amount of trials.

3 METHODOLOGY

3.1 Algorithms

We chose several machine learning algorithms in order to ascertain whether the effects of hyper parameter tuning are exhibited universally by ML algorithms. This work focuses on the following classification algorithms: Support Vector Machine (SVM), Random Forest (RM) and Multi-Layer Perceptron Classifiers (MLPC), all of which have an extensive set of hyper-parameters to choose from [1]. We selected the hyper-parameter space for tuning based on the default values provided in scikit-learn and the related literature.[2] [1] Due to our limitations in processing power we limited the size of these search-spaces.

[1]https://bit.ly/2O9UXtT, https://bit.ly/2CNw3hS, https://bit.ly/2Oe9TYd, https://bit.ly/1V0iBuM

[2]parameters given in the tables are for grid/random search , SMBO algorithm select logunifrom values in the same range.

Table 1: Chosen hyper-parameters for SVM

Parameter	Values	Count
Kernel	Linear, rbf	2
C	0.001, 0.01, 0.1, 1, 10	5
Gamma	0.001, 0.01, 0.1, 1	4
total		~40

Table 2: Chosen hyper-parameters for Random Forest

Parameter	Values	Count
Max depth	3,None	2
Max features	1,3,10	3
Min samples split	2,3,10	3
Bootstrap	True, False	2
Criterion	Gini, entropy	2
Total		~72

Table 3: Chosen hyper-parameters for MLPC

Parameter	Values	Count
Hidden Layer sizes	1/2/5*(n_features/n_features*4)	6
Alpha	0.0001, 0.001, 0.01, 0.1	4
Learning rate init value	0.00001, 0.0001, 0.001, 0.01, 0.1	5
Total		~120

3.2 Datasets

Based on the assumption that HPT can be applied in any context, we selected scikit-learn's digits dataset[3] and publicly available census data [4]; both requiring minimal preprocessing. The census data was cleaned by removing all rows with missing features and converting all entries to numerical values. This resulted in a truncated census dataset with 1443 samples with 8 features. The digits dataset in turn had 1797 rows with 64 features (the pixels of each image). To avoid overfitting, cross-validation was used.

3.3 HPT-Methods

Grid-search provides a "baseline" for tuning methods. By brute-forcing the complete search-space it finds the optimal solution (within the space given) but at the cost of efficiency. *Random-search*

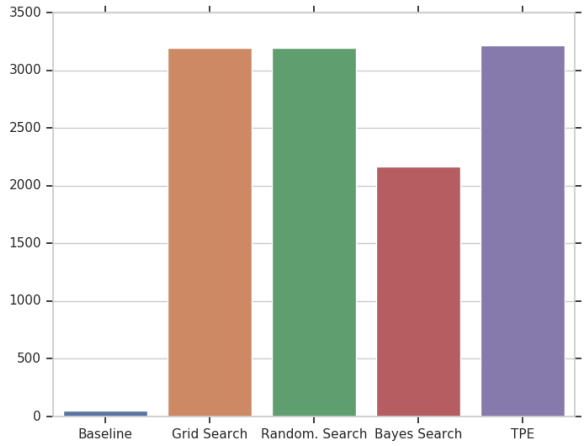
[3]http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits

[4]https://www.kaggle.com/uciml/adult-census-income

in contrast, chooses hyper-parameters to try at random. Both GS and RS scikit learn implementations were used in this work. We then compared these against two sequential model-based optimization (SMBO) approaches.*Bayes-search* is implemented by scikit-optimize which uses the Gaussian Process approach as its probability model as a means to select parameters. [1] *Tree-structured Parzen Estimator*(TPE) selects the next iteration on parameters by modeling P(x|y), the probability of the hyper-parameters given the performance, and P(y), the probability of that performance, based on the previous measurements.

4 RESULTS AND DISCUSSION

Figure 1: SVM on Census data - Tuning time



Our experiments shows that HPT improves ML performance, but also highlight some of its limitations. SVM and MLP (tuned on the census and digits datasets respectively) improved, as displayed in Table 4 & Figures 1 and 2. With a base accuracy of 0.78 and 0.96, their increase of upto 0.18 and 0.02 is significant, given the small hyperparameter spaces (40 and 120).

GridSearch (GS) and Randomized Search (RS) performed the same for SVM, since both sampled the maximum amount of parameters. For RandomForest, GS outperformed RS by 0.05% sampling only slightly more parameters (72 vs. 60) In comparison, Table 5 confirms Bergstra et al. [1]'s research hypothesis that RS is more efficient than GS, since it achieves the same score sampling a third of the parameters.

The SMBO performance varied strongly for the MLAs in question - for SVM, TPE had the longest run time of all the HTP methods but the lowest score (with a max_eval of 10 vs. 40 iterations for RS & Baye's), whereas the other SMBO method, Bayes' search performed, as expected, most efficiently and with an accuracy score only 0.03 smaller than GS or RS. In contrast TPE was the most efficient method and Bayes the most performant method for MLP, see Table (5). However, the tuning-time for Bayes was similar or worse than GS, while sampling only a third of the parameters. This can be explained by the Gaussian Process that is implemented

Figure 2: SVM on Census data - Accuracy

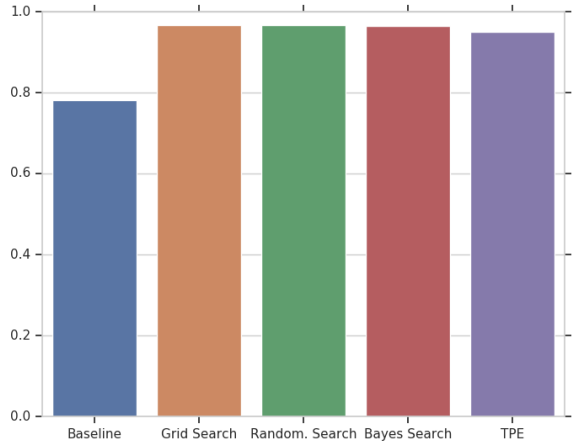


Table 4: Summary of SVM performance on census data

Method	Time (in s.)	# Iters	Accuracy	Precision	Recall
Baseline	54	n/a	0.781	0.738	0.898
Grid	3196	n/a	0.968	0.965	0.973
Random	3193	40	0.968	0.965	0.973
Bayes	2169	40	0.965	0.965	0.969
TPE	3217	10	0.949	0.968	0.934

by Bayes, which scales cubically with each iteration [1].This is normally overshadowed by training/cross-validating time; in this case our dataset is too small to do so. The results for the census dataset reveal that the hyperparameter space does not generalise to other datasets. MLP starts from a low baseline of 55% accuracy, which is matched by the HPT methods +- 0.06. The accuracy drop for GS and Bayes might be explained by the test and train data-split, namely the parameters perform best in cross-validation but not on the test-set.

Table 5: Summary of MLPC performance on digits data

HPT method	Time (in sec)	Params sampled	Accuracy
Baseline	5	1	0.961
Grid	1376	120	0.972
Random	492	40	0.972
Bayes	1206	40	0.983
TPE	417	40	0.972

Overall we found that HPT improves performance, but is limited by the hyper-parameter space and dataset size chosen. For instance, MLP tuned well on digits but not on the census dataset. Regarding the HPT methods, RS and TPE showed the best results in terms of efficiency and efficacy across the board.

Figure 3: MLPC on Census data - Accuracy

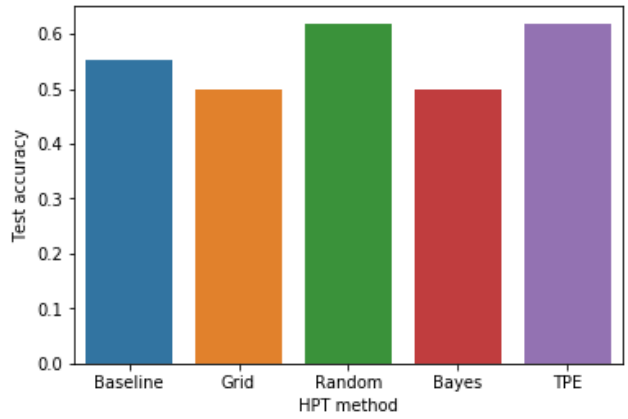


Figure 4: MLPC on Census data - Tuning time

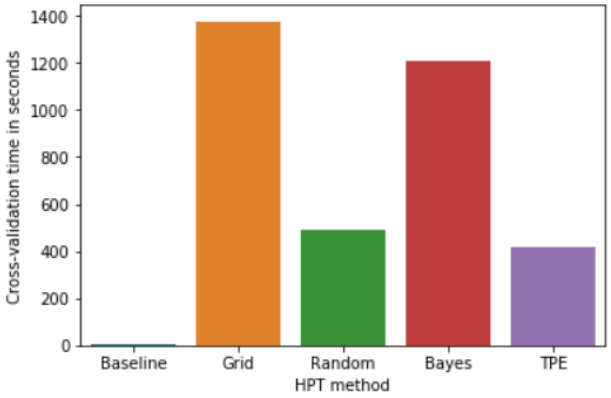


Figure 5: MLPC on Digits data - Tuning time

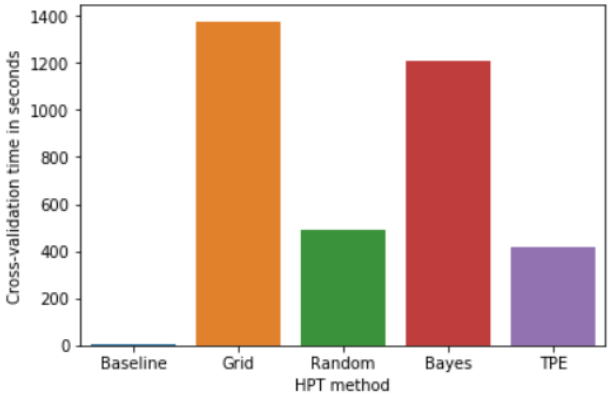


Figure 6: MLPC on Digits data - Accuracy

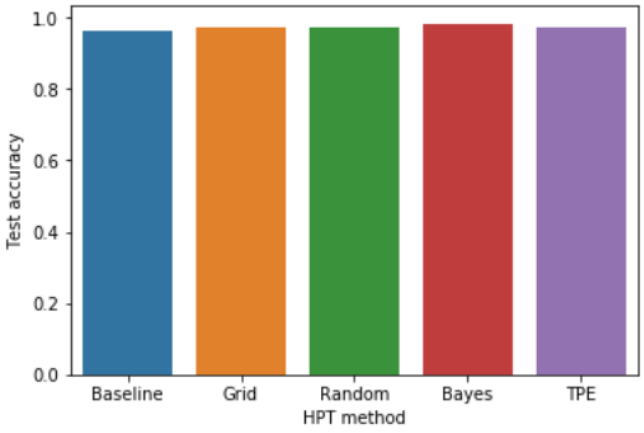


Table 6: Params chosen for SVM

	Kernel	C	Gamma	No. of iters/evals
Baseline	linear	1	auto	n/a
Grid	rbf	10	0.01	n/a
Random	rbf	10	0.01	40 iters
Bayes	rbf	3.821	0.011	40 iters
TPE	rbf	8.72	0.137	10 evals

Table 7: Params chosen for MLPC

	Alpha	hidden_layer_size	learning_rate_init
Baseline	0.001000	64	0.001000
Grid	0.001000	(256)*2	0.001000
Random	0.001000	(256)*2	0.001000
Bayes	0.001162	(256)*5	0.001599
TPE	0.001651	(256)*5	0.000552

5 LIMITATIONS AND OUTLOOK

The main limitation of this research was a lack of access to a cluster/supercomputer and experience, which meant we were limited in the size of the datasets and hyperparameter spaces for our experiments. Therefore, our experiments only show small improvements, and were prone to show discrepancies, such as tuning-time for Bayes and the performance drop from GS, see Table 3. In the future we would like to run these experiments on larger datasets and with bigger sample spaces (thus letting the algorithm find the best parameters without being limited by the space selection) and compare these results with our current findings.

REFERENCES

[1] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-parameter Optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11)*.

[2] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization? *Journal of Machine Learning Research* 13 (2012), 281–305.

[3] William Koehrsen. 2018. Automated Machine Learning Hyperparameter Tuning in Python. https://bit.ly/2KrPGkm

[4] Wolfgang Konen, Patrick Koch, Oliver Flasch, Thomas Bartz-Beielstein, Martina Friese, and Boris Naujoks. 2011. Tuned data mining: a benchmark study on different tuners. In *GECCO*.

[5] Pawel Matuszyk, Rene Tatua Castillo, Daniel Kottke, and Myra Spiliopoulou. 2016. A Comparative Study on Hyperparameter Optimization for Recommender Systems. In *Workshop on Recommender Systems and Big Data Analytics (RS-BDA'16) @ iKNOW 2016*, Elisabeth Lex, Roman Kern, Alexander Felfernig, Kris Jack, Dominik Kowald, and Emanuel Lacic (Eds.).