An Empirical Study of Hyperparameter Importance Across Datasets

Jan N. van Rijn and Frank Hutter

University of Freiburg, Germany
{vanrijn,fh}@cs.uni-freiburg.de

Abstract. With the advent of automated machine learning, automated hyperparameter optimization methods are by now routinely used. However, this progress is not yet matched by equal progress on automatic analyses that yield information beyond performance-optimizing hyperparameter settings. Various post-hoc analysis techniques exist to analyze hyperparameter importance, but to the best of our knowledge, so far these have only been applied at a very small scale. To fill this gap, we conduct a large scale experiment to discover general trends across 100 datasets. The results in case studies with random forests and Adaboost show that the same hyperparameters typically remain most important across datasets. Overall, these results, obtained fully automatically, provide a quantitative basis to focus efforts in both manual algorithm design and in automated hyperparameter optimization.

Keywords: Hyperparameter importance, empirical study

1 Introduction

The performance of most machine learning algorithms highly depends on their hyperparameter settings. Various methods exist to automatically optimize hyperparameters, including random search [1], Bayesian optimization [13,16,22], evolutionary optimization [18], meta-learning [21,23] and bandit-based methods [17]. All of these techniques require a *search space* that specifies which hyperparameters to optimize and which ranges to consider for each of them. Currently these search spaces are designed by experienced machine learners, with decisions typically based on a combination of intuition and trial & error.

Various post-hoc analysis techniques exist that, for a given dataset and algorithm, determine what were the most important hyperparameters; examples include *Forward Selection* [14], *functional ANOVA* [12], and *Ablation Analysis* [2,7]. However, to the best of our knowledge, all of these techniques have only been used for individual data sets. Here, to obtain results that can be deemed more representative, we analyze hyperparameter importance across many different datasets. Specifically, we employ the 100 datasets from the OpenML [19,25] benchmark suite OpenML-100 [3] to determine the most important hyperparameters of random forests [4] and Adaboost [9]. In this preliminary work, we focus on the hyperparameter importance analysis framework of functional ANOVA [12], but in the future we would like to expand it to other methods.

2 Background: Functional ANOVA

The functional ANOVA framework for assessing hyperparameter importance of algorithms is based on efficient computations of marginal performance. Various hyperparameter configurations of an algorithm result in various performances. Functional ANOVA determines per hyperparameter how it contributes to the variance in performance.

After introducing notation, we first briefly describe these and then summarize how they can be plugged into the standard functional ANOVA framework to yield the importance of all hyperparameters. For details, we refer the reader to the original paper [12].

Notation. Following the notation of [12], algorithm A has n hyperparameters with domains $\Theta_1, \ldots, \Theta_n$ and configuration space $\boldsymbol{\Theta} = \Theta_1 \times \ldots \times \Theta_n$. Let N be the set of all hyperparameters of A. An instantiation of A is a vector $\boldsymbol{\theta} = \langle \theta_1, \ldots, \theta_n \rangle$ with $\theta_i \in \Theta_i$ (this is also called a configuration of A). A partial instantiation of A is a vector $\boldsymbol{\theta}_U = \langle \theta_1, \ldots, \theta_n \rangle$ with a subset $U \subseteq N$ of the hyperparameters fixed, and the values for other hyperparameters unspecified. (Note that from this it follows that $\boldsymbol{\theta}_N = \boldsymbol{\theta}$).

Efficient marginal predictions. The *marginal performance* $\hat{a}_U(\boldsymbol{\theta}_U)$ is defined as the average performance of all complete instantiations $\boldsymbol{\theta}$ that agree with $\boldsymbol{\theta}_U$ in the instantiations of hyperparameters U. We note that this marginal involves a very large number of terms (even in the case of finite hyperparameter ranges, it is exponential in the remaining number of hyperparameters $N \setminus U$). However, for a tree-based model for approximating the performance of configurations, the average over these terms can be computed exactly by a procedure that is linear in the number of leaves in the model [12].

Functional ANOVA. Functional ANOVA [10,11,15,24] decomposes a function $\hat{y} : \Theta_1 \times \cdots \times \Theta_n \to \mathbb{R}$ into additive components that only depend on subsets of the hyperparameters N:

$$\hat{y}(\boldsymbol{\theta}) = \sum_{U \subseteq N} \hat{f}_U(\boldsymbol{\theta}_U) \tag{1}$$

The components $\hat{f}_U(\boldsymbol{\theta}_U)$ are defined as follows:

$$\hat{f}_U(\boldsymbol{\theta}_U) = \begin{cases} \hat{f}_\emptyset & \text{if } U = \emptyset. \\ \hat{a}_U(\boldsymbol{\theta}_U) - \sum_{W \subsetneq U} \hat{f}_W(\boldsymbol{\theta}_W) & \text{otherwise,} \end{cases} \tag{2}$$

where the constant $\hat{f}_\emptyset$ is the mean value of the function over its domain. Our main interest is the result of the unary function $\hat{f}_{\{j\}}(\boldsymbol{\theta}_{\{j\}})$, which captures the effect of varying hyperparameter j, averaging across all possible values of all other hyperparameters. Additionally, the function $\hat{f}_U(\boldsymbol{\theta}_U)$ for $|U| > 1$ captures the interaction effects between all variables in U (excluding effects of subsets

$W \subsetneq U$), but studying these higher-order effects is beyond the scope of this preliminary work.

Given the individual components, functional ANOVA decomposes the variance $\mathbb{V}$ of $\hat{y}$ into the contributions by all subsets of hyperparameters $\mathbb{V}_U$:

$$\mathbb{V} = \sum_{U \subset N} \mathbb{V}_U, \text{ where } \mathbb{V}_U = \frac{1}{||\boldsymbol{\Theta}_U||} \int \hat{f}_U(\boldsymbol{\theta}_U)^2 d\boldsymbol{\theta}_U, \tag{3}$$

where $\frac{1}{||\boldsymbol{\Theta}_U||}$ is the probability density of the uniform distribution across $\boldsymbol{\Theta}_U$.

Putting it all together. To apply functional ANOVA, we first collect performance data $\langle \boldsymbol{\theta}_i, y_i \rangle_{k=1}^K$ that captures the performance y_i (e.g., misclassification rate or RMSE) of an algorithm A with hyperparameter settings $\boldsymbol{\theta}_i$. We then fit a random forest model to this data and use functional ANOVA to decompose the variance of each of the forest's trees $\hat{y}$ into contributions due to each subset of hyperparameters. Importantly, based on the fast prediction of marginal performance available for tree-based models, this is an efficient operation requiring only seconds in the experiments for this paper. Overall, based on the performance data $\langle \boldsymbol{\theta}_i, y_i \rangle_{k=1}^K$, functional ANOVA thus provides us with the relative variance contributions of each individual hyperparameter (with the relative variance contributions of all subsets of hyperparameters summing to one).

3 Hyperparameter Importance Across Datasets

We report the results of running functional ANOVA on 100 datasets for two classifiers implemented in scikit-learn [5]: random forests [4] and Adaboost (using decision trees as base-classifier) [9]. We use almost the same hyperparameters and ranges as used by auto-sklearn [8], described in detail in Tables 1 and 2, respectively. The only difference with the auto-sklearn search space is the maximal features hyperparameter of random forests, which is modelled as a fraction of the number of available features (range $[0.1, 0.9]$).

Both algorithms apply the following data preprocessing steps. Missing values are imputed (categorical features with the mode; for numerical features, the imputation strategy was one of the hyperparameters), categorical hyperparameters are one-hot-encoded, and constant features are removed.

For each of the 100 datasets in the OpenML-100 benchmark suite [3], we created performance data for these two algorithms by executing random configurations on a large compute cluster. We ensured that each dataset had at least 200 runs to make functional ANOVA's model reliable enough for the small hyperparameter spaces considered here. (We note that for larger hyperparameter spaces, more sophisticated data gathering strategies are likely required to accurately model the performance of the best configurations.) The exact performance data we used is available on OpenML[1].

[1] RF: https://www.openml.org/f/6969; Adaboost: https://www.openml.org/f/6970

Random Forest Case Study

Table 1. Random Forest Hyperparameters.

hyperparameter	values	description
bootstrap	{true, false}	Whether to use bootstrap samples or full train set.
split criterion	{gini, entropy}	Function to determine the quality of a possible split.
max. features	$[0.1, 0.9]$	Fraction of random features sampled per node.
min. samples leaf	$[1, 20]$	The minimal number of data points required to split an internal node.
min. samples split	$[2, 20]$	The minimal number of data points required per leaf.
imputation	{mean, median, mode}	Strategy for imputing numeric variables.

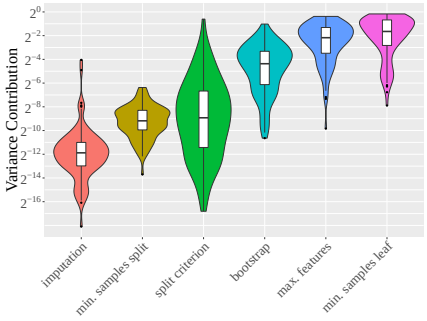

Fig. 1. Marginal contribution of random forest hyperparameters per dataset.

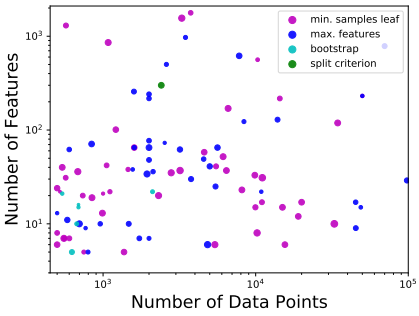

Fig. 2. Most important hyperparameter plotted against dataset dimensions.

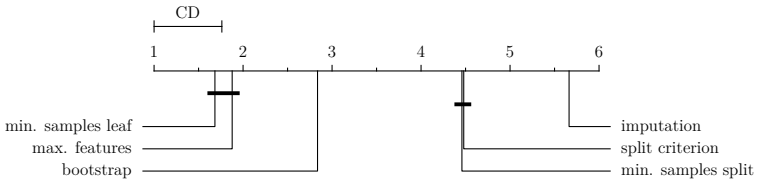

Fig. 3. Ranked hyperparameter importance, critical distance based on $\alpha = 0.05$.

3.1 Hyperparameter Importance of Random Forests

Table 1 and Figures 1–3 present the results of the random forest case study.

Table 1 shows the precise hyperparameters and ranges we used. Figure 1 shows violinplots of the marginal contribution per hyperparameter per dataset. The x-axis shows the hyperparameter under investigation, and each data-point represents $\mathbb{V}_j/\mathbb{V}$ for hyperparameter j. A high value implies that this hyperparameter accounted for a large fraction of variance, and therefore would account for high accuracy-loss if not set properly. For two datasets ('tamilnadu-electricity' and 'Mice Protein') we could not perform a functional ANOVA anal-

ysis, because the random forest classifier performed very similarly across the various hyperparameter values (almost always a perfect accuracy) and functional ANOVA's internal model predicted zero variance.

The results reveal that most of the variance could be attributed to a small set of hyperparameters: the minimal samples per leaf and maximal number of features for determining the split were the most important hyperparameters. It is reasonable to assume that these hyperparameters have some regions that are clearly sub-optimal. For example, at every split point, the random forest should always have at least some features to choose from; if this number is too low this clearly affects performance. This was also conjectured in earlier work, see Figure 4.8 of [20].

Figure 2 shows the most important hyperparameters per dataset, plotted against the dataset dimensions (number of data points on the x-axis; number of features on the y-axis). Each data point represents a dataset, the color indicates which of the hyperparameters was most important, and the size indicates the marginal contribution (formally, $\mathbb{V}_j/\mathbb{V}$).

The results of Figure 2 add to the results presented in Figure 1: minimal samples per leaf and maximal features dominate the other hyperparameters. Only in a few cases bootstrap ('balance-scale', 'credit-a', 'kc1', 'Australian', 'profb' and 'climate-model-simulation-crashes') or the split criterion ('scene') was most important. The imputation strategy was never the most important hyperparameter. This is slightly surprising, since the benchmark suite also contains datasets that have many missing values.

Figure 3 shows the result of a Nemenyi test over the average ranks of the hyperparameters (for details, see [6]). A statistically significant difference was measured for every pair of classifiers that are not connected by the horizontal black line. The results support the observations made in Figure 1, giving statistical evidence that the minimal samples per leaf and maximal number of features for determining the split are more important than the other hyperparameters.

3.2 Hyperparameter Importance of Adaboost

Table 2 and Figures 4–6 present the results for the Adaboost case study, structured exactly like the study for random forests.

Figure 4 shows that similarly to the random forest, most of the variance could be explained by a small set of hyperparameters, in this case the maximal depth of the decision tree and, to a lesser degree, the learning rate. Figure 5 shows a similar result. The maximal depth and learning rate were the most important hyperparameters for almost all datasets; there are only a few exceptions where the boosting algorithm ('madelon' and 'LED-display-domain-7digit') or the number of iterations ('steel-plates-fault') were the most important hyperparameters.

The results of the Nemenyi test (Figure 6) support the notion that the aforementioned hyperparameters indeed contributed most to the total variance. The maximal depth hyperparameter contributed significantly more than the other hyperparameters, followed by learning rate. Again, the imputation strategy did not seem to matter.

Adaboost Case Study

Table 2. Adaboost Hyperparameters.

hyperparameter	values	description
algorithm	{SAMME, SAMME.R}	Boosting algorithm to use.
learning rate	$[0.01, 2.0]$	Learning rate shrinks the contribution of each classifier.
max. depth	$[1, 10]$	The maximal depth of the decision trees
iterations	$[50, 500]$	Number of estimators to build.
imputation	{mean, median, mode}	Strategy for imputing numeric variables.

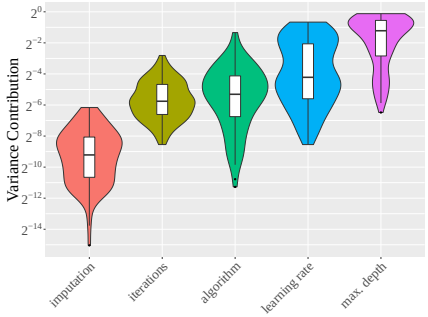

Fig. 4. Marginal contribution of Adaboost hyperparameters per dataset.

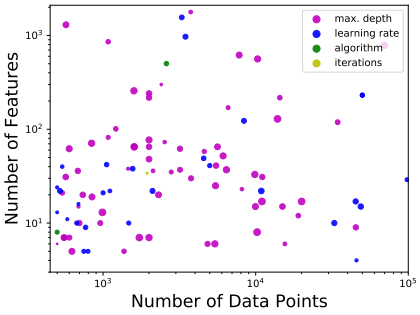

Fig. 5. Most important hyperparameter plotted against dataset dimensions.

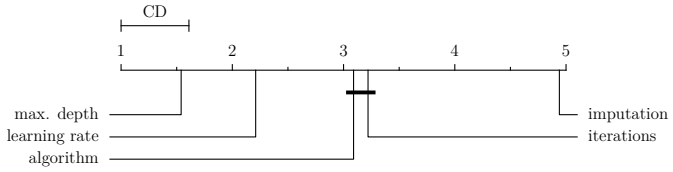

Fig. 6. Ranked hyperparameter importance, critical distance based on $\alpha = 0.05$.

We note that the results presented in this paper do by no means imply that it suffices to tune just the aforementioned hyperparameters. Contrarily, when enough budget is available it is still advisable to tune all hyperparameters. However, the results in [12] indicated that focusing only on tuning the most important hyperparameters yields improvements faster; thus, when the tuning budget is small this might be advisable. This could be tested by a multistage process, in which the tuning process initially focuses on the most important hyperparameter, and gradually changes its behaviour towards also incorporating less important hyperparameters. We leave it to future work to test under which circumstances focusing on a subset of hyperparameters indeed yields better results.

4 Conclusions and Future Work

We conducted a large scale study towards hyperparameter importance using functional ANOVA on OpenML datasets. Indeed, functional ANOVA and OpenML complement each other quite well: the experimental results available on OpenML can serve as an input for functional ANOVA (or any other model-based hyperparameter importance analysis technique) and can be used to assess which hyperparameters are most important.

To the best of our knowledge, this is the first large scale study over many datasets assessing which hyperparameters are commonly important. We did this for two popular tree-based methods, random forests and Adaboost, resulting in quantifiable measures of their hyperparameters' importance across datasets.

In future work, we aim to assess hyperparameter importance in good parts of the space (this is already supported by functional ANOVA by assessing improvements over a baseline [12]), to also use other analysis techniques for hyperparameter importance across datasets, and to extend this research towards other model types. In particular, deep neural networks are known to be very sensitive to some of their hyperparameters, and quantifiable recommendations about which hyperparameters to focus on would be very useful for the community.

Finally, we aim to use the knowledge obtained from hyperparameter importance analysis to prune the hyperparameter search space (see also [26]). Specifically, experiments across datasets could point out that for new datasets, certain hyperparameters or ranges are not worthwhile to explore; exploiting this could potentially lead to large speed-ups for hyperparameter optimization.

Acknowledgements. This work has partly been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant no. 716721.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13(Feb), 281–305 (2012)
2. Biedenkapp, A., Lindauer, M., Eggensperger, K., Fawcett, C., Hoos, H., Hutter, F.: Efficient parameter importance analysis via ablation with surrogates. In: Proc. of AAAI 2017. pp. 773–779 (2017)
3. Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R.G., van Rijn, J.N., Vanschoren, J.: OpenML Benchmarking Suites and the OpenML100. ArXiv [stat.ML] 1708.03731v1, 6 pages (2017)
4. Breiman, L.: Random Forests. Machine learning 45(1), 5–32 (2001)
5. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108–122 (2013)
6. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. The Journal of Machine Learning Research 7, 1–30 (2006)

7. Fawcett, C., Hoos, H.H.: Analysing differences between algorithm configurations through ablation. Journal of Heuristics 22(4), 431–458 (2016)
8. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Advances in Neural Information Processing Systems 28, pp. 2962–2970. Curran Associates, Inc. (2015)
9. Freund, Y., Schapire, R.E.: A desicion-theoretic generalization of on-line learning and an application to boosting. In: European conference on computational learning theory. pp. 23–37. Springer (1995)
10. Hooker, G.: Generalized functional anova diagnostics for high-dimensional functions of dependent variables. Journal of Computational and Graphical Statistics 16(3), 709–732 (2007)
11. Huang, J.Z., et al.: Projection estimation in multiple regression with application to functional anova models. The annals of statistics 26(1), 242–272 (1998)
12. Hutter, F., Hoos, H., Leyton-Brown, K.: An efficient approach for assessing hyperparameter importance. In: Proc. of ICML 2014. pp. 754–762 (2014)
13. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: International Conference on Learning and Intelligent Optimization. pp. 507–523. Springer (2011)
14. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Identifying key algorithm parameters and instance features using forward selection. In: International Conference on Learning and Intelligent Optimization. pp. 364–381. Springer (2013)
15. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global optimization 13(4), 455–492 (1998)
16. Klein, A., Falkner, S., Bartels, S., Hennig, P., Hutter, F.: Fast Bayesian optimization of machine learning hyperparameters on large datasets. In: Proc. of AISTATS 2017 (2017)
17. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: Bandit-Based Configuration Evaluation for Hyperparameter Optimization. In: Proc. of ICLR 2017 (2017)
18. Loshchilov, I., Hutter, F.: CMA-ES for hyperparameter optimization of deep neural networks. ArXiv [cs.NE] 1604.07269v1, 8 pages (2016)
19. van Rijn, J.N., Bischl, B., Torgo, L., Gao, B., Umaashankar, V., Fischer, S., Winter, P., Wiswedel, B., Berthold, M.R., Vanschoren, J.: OpenML: A Collaborative Science Platform. In: Proc. of ECML/PKDD 2013, pp. 645–649. Springer (2013)
20. van Rijn, J.N.: Massively Collaborative Machine Learning. Ph.D. thesis, Leiden University (2016)
21. van Rijn, J.N., Abdulrahman, S.M., Brazdil, P., Vanschoren, J.: Fast Algorithm Selection using Learning Curves. In: Advances in Intelligent Data Analysis XIV. pp. 298–309. Springer (2015)
22. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems 25. pp. 2951–2959 (2012)
23. Soares, C., Brazdil, P.B., Kuba, P.: A meta-learning method to select the kernel width in support vector regression. Machine learning 54(3), 195–209 (2004)
24. Sobol, I.M.: Sensitivity estimates for nonlinear mathematical models. Mathematical Modelling and Computational Experiments 1(4), 407–414 (1993)
25. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. ACM SIGKDD Explorations Newsletter 15(2), 49–60 (2014)
26. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization. In: Proc. of ECML/PKDD 2015. pp. 104–119. Springer (2015)