Automatic Exploration of Machine Learning Experiments on OpenML

by Daniel Kühn*, Philipp Probst*, Janek Thomas and Bernd Bischl

June 19, 2018

Abstract

Understanding the influence of hyperparameters on the performance of a machine learning algorithm is an important requirement for well performing and adequately tuned algorithms. As to date, data to help this required understanding of state-of-the-art algorithms like gradient boosting or random forest are rare, this paper presents a large, free and open dataset addressing this problem (see [Kühn et al., 2018] to access the dataset). The dataset contains the performance (AUC, accuracy and Brier score) of six different machine learning algorithms with randomly sampled hyperparameters, runtime and meta-data for 38 datasets. Each algorithm was cross-validated up to 500000 times with different hyperparameters resulting in a dataset of around 2.5 million experiments overall. Such data can be invaluable for meta-learning, benchmarking and other tasks related to hyperparameters like finding good defaults or measuring the tunability of algorithms and hyperparameters.

PP: Satz klingt komisch, es kann auch mal "well performen" ohne dieses Wissen

1 Introduction

When applying machine learning algorithms on real world datasets, users have to choose from a large selection of different algorithms with many of them offering a set of hyperparameters. Even though sometimes default values exist, they can be suboptimal and need to be specified by the user which often has large influence on the performance of the algorithm. The *no free lunch theorem* [Wolpert, 2002] states that in algorithm selection one algorithm can not consistently outperform all others algorithms for every dataset, so a crucial question practitioners have to face on a daily basis therefore is the selection of the best algorithm with optimal hyperparameters for a given dataset. This problem is very hard to solve, since many algorithms exist, the evaluation of a single machine learning run is often computationally expensive and the hyperparameter-space is complex [Claesen and Moor, 2015]. The usual approach is to run a hyperparameter tuning algorithm such as random search, grid search or Bayesian optimization [Snoek et al., 2012] to find the best hyperparameter setting. These methods have the drawback that a large number of runs might be necessary which can result in very high computational costs.

Meta-learning tries to decrease this cost [Feurer et al., 2015], by reusing information of previous runs of the algorithm on other datasets. A requirement for this, is to have a meta-learning dataset, that contains such information. With this paper we provide a freely accessible dataset that contains around 2.5 million runs of six different machine learning algorithms on 38 classification datasets. Large, freely available datasets like Imagenet [Deng et al., 2009] are important for the progress of machine learning, so we hope to support the development in the area of meta-learning and benchmarking with this dataset. While similar meta-datasets were created in the past, we were not able to access them by the links provided in their respective papers: Smith et al. [2014] provides a repository with Weka-based machine learning experiments on 72 data sets, 9 machine learning algorithms, 10 hyperparameter settings for each algorithm, and several meta-features of each data set.

Reif [2012] created a meta-dataset based on machine learning experiments on 83 datasets, 6 classification algorithms, and 49 meta-features.

In this paper we first describe how the meta-dataset is created by executing random machine learning experiments and storing the results on OpenML [Vanschoren et al., 2013], an open source database for machine learning problems. Then the possibilities of accessing this dataset are shortly presented. Finally we briefly discuss potential usage of the dataset.

2 Creating the dataset

To create the dataset the following six supervised machine learning algorithms implemented in R are run on 38 classification tasks with predefined ranges for their relevant hyperparameters: elastic net (`glmnet` [Friedman et al., 2010]), decision tree (`rpart`, [Therneau and Atkinson, 2018]), k-nearest neighbors (`kknn`, [Schliep and Hechenbichler, 2016]), support vector machines (`svm`, [Meyer et al., 2017]), random forest (`ranger`, [Wright and Ziegler, 2017]) and gradient boosting (`xgboost`, [Chen and Guestrin, 2016]). These algorithms cover a wide range of approaches to machine learning. For each algorithm the available hyperparameters are explored in a predefined range (see Table 1). Some of these hyperparameters are transformed by the function found in column *trafo* to sample from the range non-uniformly. This is an often performed procedure, if e.g. minor changes for bigger values of a hyperparameter are not expected to have a significant impact on the performance of an algorithm.

PP: Zitierungen stören den Textfluss

These algorithms are run on a subset of the OpenML100 Benchmark suite [Bischl et al., 2017], which consists of 100 classification datasets carefully curated from the thousands of datasets available on OpenML [Vanschoren et al., 2013]. We only include datasets without missing data and with a binary outcome resulting in 38 datasets. The datasets with their specific characteristics can be found in Table 2.

3 Random Experimentation Bot

To conduct a large number of of experiments a bot was implemented to automatically plan and execute runs. Following the search paradigm of random search the bot iteratively executes several steps:

1. Randomly draw one of the six algorithms.
2. Randomly draw a hyperparameter setting of the chosen algorithm based on the ranges of Table 1.
3. Randomly draw one of the 38 binary classification benchmark datasets from Table 2.
4. Load the dataset from cache or download it from OpenML and cache it.
5. Evaluate the algorithm with the sampled hyperparameters on the selected dataset with 10-fold cross-validation. Consistent cross-validation splits are provided by OpenML.
6. Upload the benchmark results with hyperparameters, performance measures and time measurements to OpenML. The tag `mlrRandomBot` is used for identification.

An advantage of using random search instead of other tuning methods like grid search is that additional experiments can be easily added. One could for example easily increase the hyperparameter range for a specific algorithm or add a new algorithm simply by adding new experiments to the existing ones. Moreover, random search is more efficient than grid search in covering multidimensional hyperparameter spaces and finding optimal hyperparameter settings [Bergstra and Bengio, 2012].

algorithm	hyperparameter	type	lower	upper	trafo
glmnet	alpha	numeric	0	1	-
	lambda	numeric	-10	10	2^x
rpart	cp	numeric	0	1	-
	maxdepth	integer	1	30	-
	minbucket	integer	1	60	-
	minsplit	integer	1	60	-
kknn	k	integer	1	30	-
svm	kernel	discrete	-	-	-
	cost	numeric	-10	10	2^x
	gamma	numeric	-10	10	2^x
	degree	integer	2	5	-
ranger	num.trees	integer	1	2000	-
	replace	logical	-	-	-
	sample.fraction	numeric	0	1	-
	mtry	numeric	0	1	$x \cdot p$
	respect.unordered.factors	logical	-	-	-
	min.node.size	numeric	0	1	n^x
xgboost	nrounds	integer	1	5000	-
	eta	numeric	-10	0	2^x
	subsample	numeric	0	1	-
	booster	discrete	-	-	-
	max_depth	integer	1	15	-
	min_child_weight	numeric	0	7	2^x
	colsample_bytree	numeric	0	1	-
	colsample_bylevel	numeric	0	1	-
	lambda	numeric	-10	10	2^x
	alpha	numeric	-10	10	2^x

Table 1: Hyperparameters of the algorithms. p refers to the number of variables and n to the number of observations.

The bot is developed open source and can be found on GitHub (`https://github.com/ja-thomas/OMLbots`). To add a new algorithm it has to be included in the file `R/botSetLearnerParamPairs.R` of the GitHub repository with with its hyperparameter ranges. The main function `runBot` executes the bot with a predefined number of experiments. The bot is based on the R packages `mlr` [Bischl et al., 2016] and `OpenML` [Casalicchio et al., 2017] and written in modular form such that it can be extended with new sampling strategies for hyperparameters, algorithms and datasets in the future.

After more than 6 million benchmark experiments the results of the bot are downloaded from OpenML. Since on dataset 4135 all algorithms except of `rpart` and `ranger` crashed, it is excluded and 38 datasets remain.

For each of the algorithms 500000 experiments are used to obtain a final dataset. The experiments are chosen by the following procedure: for each algorithm, a threshold B is set (see below) and, if the number of results for a dataset exceeds B, we draw randomly B of the results obtained for this algorithm and this dataset. The threshold value B is chosen for each algorithm separately to exactly obtain 500000 results for each algorithm.

For `kknn` we only execute 30 experiments per dataset because this number of experiments is high enough to cover the hyperparameter space (that only consists of the parameter k for $k \in \{1, ..., 30\}$) appropriately, resulting in 1140 experiments. In total this results in around 2.5 million experiments.

The distribution of the runs on the datasets and algorithms can be seen in table 3.

Data_id	Name	nObs	nFeat	majPerc	numFeat	catFeat
3	kr-vs-kp	3196	37	0.52	0	37
31	credit-g	1000	21	0.70	7	14
37	diabetes	768	9	0.65	8	1
44	spambase	4601	58	0.61	57	1
50	tic-tac-toe	958	10	0.65	0	10
151	electricity	45312	9	0.58	7	2
312	scene	2407	300	0.82	294	6
333	monks-problems-1	556	7	0.50	0	7
334	monks-problems-2	601	7	0.66	0	7
335	monks-problems-3	554	7	0.52	0	7
1036	sylva_agnostic	14395	217	0.94	216	1
1038	gina_agnostic	3468	971	0.51	970	1
1043	ada_agnostic	4562	49	0.75	48	1
1046	mozilla4	15545	6	0.67	5	1
1049	pc4	1458	38	0.88	37	1
1050	pc3	1563	38	0.90	37	1
1063	kc2	522	22	0.80	21	1
1067	kc1	2109	22	0.85	21	1
1068	pc1	1109	22	0.93	21	1
1120	MagicTelescope	19020	12	0.65	11	1
1176	Internet-Advertisements	3279	1559	0.86	1558	1
1220	Click_prediction_small	39948	12	0.83	11	1
1461	bank-marketing	45211	17	0.88	7	10
1462	banknote-authentication	1372	5	0.56	4	1
1464	blood-transfusion-service-center	748	5	0.76	4	1
1467	climate-model-simulation-crashes	540	21	0.91	20	1
1471	eeg-eye-state	14980	15	0.55	14	1
1479	hill-valley	1212	101	0.50	100	1
1480	ilpd	583	11	0.71	9	2
1485	madelon	2600	501	0.50	500	1
1486	nomao	34465	119	0.71	89	30
1487	ozone-level-8hr	2534	73	0.94	72	1
1489	phoneme	5404	6	0.71	5	1
1494	qsar-biodeg	1055	42	0.66	41	1
1510	wdbc	569	31	0.63	30	1
4134	Bioresponse	3751	1777	0.54	1776	1
4135	Amazon_employee_access	32769	10	0.94	0	10
4534	PhishingWebsites	11055	31	0.56	0	31

Table 2: Included datasets with meta-data. *nObs* are the number of observations, *nFeat* the number of Features, *majPerc* the percentage of observations with the most common class, *numFeat* the number of numeric features and *catFeat* the number of categorical features.

4 Access to the results

The results of the benchmark can be accessed in different ways:

- The easiest way to access them is to go to the figshare repository [Kühn et al., 2018] and download the `.csv` files or the `.RData` file.

PP: Format ändern auf feather (?) und auf OpenML hochladen (?)

- Alternatively the code for the extraction of the data from the nightly database snapshot of OpenML can be found here: `https://github.com/ja-thomas/OMLbots/blob/master/snapshot_database/database_extraction.R`

5 Discussion and potential usage of the results

The presented data can be used to discover effects of hyperparameters on performances of different algorithms over different datasets.

Possible applications are:

- Find good defaults for the algorithms that work well on many datasets.
- Measure and investigate differences between algorithms.
- Improve hyperparameter tuning algorithms:

Data_id	glmnet	rpart	kknn	svm	ranger	xgboost	Total
3	15547	14633	30	19644	15139	16867	81860
31	15547	14633	30	19644	15139	16867	81860
37	15546	14633	30	15985	15139	16866	78199
44	15547	14633	30	19644	15139	16867	81860
50	15547	14633	30	19644	15139	16866	81859
151	15547	14632	30	2384	12517	16866	61976
312	6613	13455	30	18740	12985	15886	67709
333	15546	14632	30	19644	15139	16867	81858
334	15547	14633	30	19644	14492	16867	81213
335	15547	14633	30	15123	15139	10002	70474
1036	14937	14633	30	2338	7397	2581	41916
1038	15547	5151	30	5716	4827	1370	32641
1043	6466	14633	30	10121	3788	16867	51905
1046	15547	14633	30	5422	8842	11812	56286
1049	7423	14632	30	12064	15139	4453	53741
1050	15547	14633	30	19644	11357	13758	74969
1063	15547	14633	30	19644	7914	16866	74634
1067	15546	14632	30	10229	7386	16866	64689
1068	15546	14633	30	13893	8173	16866	69141
1120	15531	7477	30	3908	9760	8143	44849
1176	13005	14632	30	14451	15140	13047	70305
1220	6970	14073	30	2678	14323	2215	40289
1461	8955	14633	30	6320	15139	16867	61944
1462	15547	14632	30	19644	15139	16867	81859
1464	15547	14633	30	4441	15139	16866	66656
1467	15547	14633	30	9725	13523	16866	70324
1471	15546	14633	30	19644	15140	16867	81860
1479	15024	14633	30	19644	15139	16254	80724
1480	8247	10923	30	10334	15139	9237	53910
1485	3866	11389	30	1490	15139	5813	37727
1486	15547	6005	30	19644	15139	11194	67559
1487	15547	14633	30	17298	15139	16867	79514
1489	15547	14632	30	19644	15139	16867	81859
1494	15547	14633	30	19644	15140	16867	81861
1510	15547	14633	30	19644	15139	16867	81860
4134	15546	14632	30	19644	15139	16867	81858
4135	1493	3947	30	560	14516	2222	22768
4534	2801	3231	30	2476	15139	947	24624
Total	500000	500000	1140	500000	500000	500000	2501140

Table 3: Number of experiments for each combination of dataset and algorithm.

  - Measure the tunability of algorithms and find out which parameters should be tuned [see ?]
  - Get priors for tuning algorithms to search important regions of the hyperparameter space with higher probability. [see van Rijn and Hutter, 2017, ?]

- Train models based on dataset characteristics and propose hyperparameter settings that perform good on a new dataset.

A potential weakness of this dataset is that the dimension of hyperparameter spaces for example for `xgboost` can be very high and the number of experiments is not sufficient to explore the space appropriately, especially in regions in which very high performance can be achieved. This could potentially be improved by using smarter sampling strategies similar to tuning algorithms to explore these regions more thoroughly.

References

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones. mlr: Machine learning in r. *Journal of Machine Learning Research*, 17(170): 1–5, 2016. URL `http://jmlr.org/papers/v17/15-066.html`.

B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, and J. Vanschoren. OpenML benchmarking suites and the OpenML100. *ArXiv preprint arXiv:1708.03731*, Aug. 2017. URL https://arxiv.org/abs/1708.03731.

G. Casalicchio, J. Bossek, M. Lang, D. Kirchhoff, P. Kerschke, B. Hofner, H. Seibold, J. Vanschoren, and B. Bischl. OpenML: An R package to connect to the machine learning platform OpenML. *Computational Statistics*, 32(3):1–15, 2017.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2.

M. Claesen and B. D. Moor. Hyperparameter search in machine learning. *MIC 2015: The XI Metaheuristics International Conference*, 2015.

J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

M. Feurer, J. T. Springenberg, and F. Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1128–1135. AAAI Press, 2015.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

D. Kühn, P. Probst, J. Thomas, and B. Bischl. OpenML R bot benchmark data (final subset). 2018. URL https://figshare.com/articles/OpenML_R_Bot_Benchmark_Data_final_subset_/5882230.

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. L. h. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. URL https://CRAN.R-project.org/package=e1071. R package version 1.6-8.

M. Reif. A comprehensive dataset for evaluating approaches of various meta-learning tasks. In *ICPRAM*, 2012.

K. Schliep and K. Hechenbichler. *kknn: Weighted k-Nearest Neighbors*, 2016. URL https://CRAN.R-project.org/package=kknn. R package version 1.3.1.

M. R. Smith, A. White, C. Giraud-Carrier, and T. Martinez. An Easy to Use Repository for Comparing and Improving Machine Learning Algorithm Usage. *ArXiv preprint arXiv:1405.7292*, 2014. URL https://arxiv.org/abs/1405.7292.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

T. Therneau and B. Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2018. URL https://CRAN.R-project.org/package=rpart. R package version 4.1-12.

J. N. van Rijn and F. Hutter. Hyperparameter importance across datasets. *ArXiv preprint arXiv:1710.04725*, 2017. URL https://arxiv.org/abs/1710.04725.

J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

D. H. Wolpert. The supervised learning no-free-lunch theorems. In *Soft computing and industry*, pages 25–42. Springer, 2002.

M. N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017.