Automatic Exploration of Machine Learning Experiments on OpenML

Daniel Kühn*[a], Philipp Probst*[a], Janek Thomas[a], Bernd Bischl[a]

[a]*Ludwig-Maximilians-Universität München, Germany*

Abstract

Understanding the influence of hyperparameters on the performance of a machine learning algorithm is an important scientific topic in itself and can help to improve automatic hyperparameter tuning procedures. Unfortunately, experimental meta data for this purpose is still rare. This paper presents a large, free and open dataset addressing this problem, containing results on 38 OpenML data sets, six different machine learning algorithms and many different hyperparameter configurations. Results where generated by an automated random sampling strategy, termed the *OpenML Random Bot*. Each algorithm was cross-validated up to 20.000 times per dataset with different hyperparameters settings, resulting in a meta dataset of around 2.5 million experiments overall.

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 to control algorithmic performance. Although sometimes default values exist, there is no agreed upon principle for their definition (but see our recent work in in (Probst et al., 2018) for a potential approach). Automatic tuning of such parameters is a possible solution (Claesen and Moor, 2015), but comes with a considerable computational burden.

Meta-learning tries to decrease this cost (Feurer et al., 2015), by reusing information of previous runs of the algorithm on similar datasets, which obviously requires access to such prior empirical results. With this paper we provide a freely accessible meta 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, we hope to support developments in the area of meta-learning and benchmarking, meta-learning and hyperparameter tuning with our work here.

While similar meta-datasets have been 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 describe our experimental setup, to specify how our meta-dataset is created by running random machine learning experiments through the OpenML platform (Vanschoren et al., 2013) and how to access our results.

Email addresses: `daniel.kuehn.87@gmail.com` (Daniel Kühn*), `philipp_probst@gmx.de` (Philipp Probst*), `janek.thomas@stat.uni-muenchen.de` (Janek Thomas), `bernd_bischl@gmx.net` (Bernd Bischl)

2. Considered ML data sets, algorithms and hyperparameters

To create the meta dataset, six supervised machine learning algorithms are run on 38 classification tasks. 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* of Table 1 to allow non-uniform sampling, a usual procedure in tuning.

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 used algorithms are `glmnet` (Friedman et al., 2010), `rpart` (Therneau and Atkinson, 2018), `kknn` (Schliep and Hechenbichler, 2016), `svm` (Meyer et al., 2017), `ranger` (Wright and Ziegler, 2017) and `xgboost` (Chen and Guestrin, 2016).

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 and their respective characteristics can be found in Table 2.

3. Random Experimentation Bot

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

Data_id	Task_id	Name	n	p	majPerc	numFeat	catFeat
3	3	kr-vs-kp	3196	37	0.52	0	37
31	31	credit-g	1000	21	0.70	7	14
37	37	diabetes	768	9	0.65	8	1
44	43	spambase	4601	58	0.61	57	1
50	49	tic-tac-toe	958	10	0.65	0	10
151	219	electricity	45312	9	0.58	7	2
312	3485	scene	2407	300	0.82	294	6
333	3492	monks-problems-1	556	7	0.50	0	7
334	3493	monks-problems-2	601	7	0.66	0	7
335	3494	monks-problems-3	554	7	0.52	0	7
1036	3889	sylva_agnostic	14395	217	0.94	216	1
1038	3891	gina_agnostic	3468	971	0.51	970	1
1043	3896	ada_agnostic	4562	49	0.75	48	1
1046	3899	mozilla4	15545	6	0.67	5	1
1049	3902	pc4	1458	38	0.88	37	1
1050	3903	pc3	1563	38	0.90	37	1
1063	3913	kc2	522	22	0.80	21	1
1067	3917	kc1	2109	22	0.85	21	1
1068	3918	pc1	1109	22	0.93	21	1
1120	3954	MagicTelescope	19020	12	0.65	11	1
1461	14965	bank-marketing	45211	17	0.88	7	10
1462	10093	banknote-authentication	1372	5	0.56	4	1
1464	10101	blood-transfusion-service-center	748	5	0.76	4	1
1467	9980	climate-model-simulation-crashes	540	21	0.91	20	1
1471	9983	eeg-eye-state	14980	15	0.55	14	1
1479	9970	hill-valley	1212	101	0.50	100	1
1480	9971	ilpd	583	11	0.71	9	2
1485	9976	madelon	2600	501	0.50	500	1
1486	9977	nomao	34465	119	0.71	89	30
1487	9978	ozone-level-8hr	2534	73	0.94	72	1
1489	9952	phoneme	5404	6	0.71	5	1
1494	9957	qsar-biodeg	1055	42	0.66	41	1
1504	9967	steel-plates-fault	1941	34	0.65	33	1
1510	9946	wdbc	569	31	0.63	30	1
1570	9914	wilt	4839	6	0.95	5	1
4134	14966	Bioresponse	3751	1777	0.54	1776	1
4534	34537	PhishingWebsites	11055	31	0.56	0	31

Table 2: Included datasets and respective characteristics. n are the number of observations, p the number of features, *maj.class* the percentage of observations in the largest class, *numFeat* the number of numeric features and *catFeat* the number of categorical features.

1. Randomly sample a task T (with an associated data set) from Table 2.
2. Randomly sample one ML algorithm A.
3. Randomly sample a hyperparameter setting θ of algorithm A, uniformly from the ranges specified in Table 1, then transform, if a transformation function is given.
4. Obtain task T (and dataset) from OpenML and store it locally.
5. Evaluate algorithm A with configuration θ on task T, with associated 10-fold cross-validation from OpenML.
6. Upload run results to OpenML, including hyperparameter configuration and time measurements.
7. OpenML now calculates various performance metrics for the uploaded cross-validated predictions.
8. The OpenML-ID of the bot (2702) and the tag `mlrRandomBot` is used for identification.

A clear advantage of random sampling is that all bot runs are completely independent of each other, making all experiments embarrassingly parallel. Furthermore, more experiments can easily and conveniently added later on, without introducing any kind of bias into the sampling method.

The bot is developed open source in R and can be found on GitHub[1]. 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

[1] https://github.com/ja-thomas/OMLbots

for hyperparameters, algorithms and datasets in the future. Parallelization was performed with R package `batchtools` (Lang et al., 2017).

After more than 6 million benchmark experiments the results of the bot are downloaded from OpenML. For each of the algorithms 500000 experiments are used to obtain the 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 in total 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. All in all this results in around 2.5 million experiments.

The distribution of the runs on the datasets and algorithms is displayed in Table 3.

Data_id	Task_id	glmnet	rpart	kknn	svm	ranger	xgboost	Total
3	3	15547	14633	30	19644	15139	16867	81860
31	31	15547	14633	30	19644	15139	16867	81860
37	37	15546	14633	30	15985	15139	16866	78199
44	43	15547	14633	30	19644	15139	16867	81860
50	49	15547	14633	30	19644	15139	16866	81859
151	219	15547	14632	30	2384	12517	16866	61976
312	3485	6613	13455	30	18740	12985	15886	67709
333	3492	15546	14632	30	19644	15139	16867	81858
334	3493	15547	14633	30	19644	14492	16867	81213
335	3494	15547	14633	30	15123	15139	10002	70474
1036	3889	14937	14633	30	2338	7397	2581	41916
1038	3891	15547	5151	30	5716	4827	1370	32641
1043	3896	6466	14633	30	10121	3788	16867	51905
1046	3899	15547	14633	30	5422	8842	11812	56286
1049	3902	7423	14632	30	12064	15139	4453	53741
1050	3903	15547	14633	30	19644	11357	13758	74969
1063	3913	15547	14633	30	19644	7914	16866	74634
1067	3917	15546	14632	30	10229	7386	16866	64689
1068	3918	15546	14633	30	13893	8173	16866	69141
1120	3954	15531	7477	30	3908	9760	8143	44849
1461	14965	6970	14073	30	2678	14323	2215	40289
1462	10093	8955	14633	30	6320	15139	16867	61944
1464	10101	15547	14632	30	19644	15139	16867	81859
1467	9980	15547	14633	30	4441	15139	16866	66656
1471	9983	15547	14633	30	9725	13523	16866	70324
1479	9970	15546	14633	30	19644	15140	16867	81860
1480	9971	15024	14633	30	19644	15139	16254	80724
1485	9976	8247	10923	30	10334	15139	9237	53910
1486	9977	3866	11389	30	1490	15139	5813	37727
1487	9978	15547	6005	30	19644	15139	11194	67559
1489	9952	15547	14633	30	17298	15139	16867	79514
1494	9957	15547	14632	30	19644	15139	16867	81859
1504	9967	15547	14633	30	19644	15140	16867	81861
1510	9946	15547	14633	30	19644	15139	16867	81860
1570	9914	15546	14632	30	19644	15139	16867	81858
4134	14966	1493	3947	30	560	14516	2222	22768
4534	34537	2801	3231	30	2476	15139	947	24624
Total	257661	486995	485368	1110	485549	484860	486953	2430835

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

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 to download the `.csv` files. For each algorithm there is a csv file that contains a row for each algorithm run with the columns `Data_id`, the hyperparameter settings, the performance measures (auc, accuracy and brier score), the runtime, the scimark reference runtime and some characteristics of the dataset such as the number of features or the number of observations.

- 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`. With this script all results that were created by the random bot (OpenML-ID 2702) are downloaded and the final dataset is created. (Warning: As the OpenML database is updated daily, changes can occur.)

5. Discussion and potential usage of the results

The presented data can be used to study the effect and influence of hyperparameter setting on performance in various ways. Possible applications are:

- Obtaining defaults for ML algorithm that work well across many datasets (Probst et al., 2018);
- Measuring the importance of hyperparameters, to investigate which should be tuned (see van Rijn and Hutter, 2017; Probst et al., 2018);
- Obtaining ranges or priors of tuning parameters to focus on important regions of the search space (see van Rijn and Hutter, 2017; Probst et al., 2018);
- Meta-Learning;
- Investigating, debugging and improving the robustness of algorithms.

Possible weaknesses of the approach, which we would like to address in the future, are:

- For each ML algorithm, a set of considered hyperparameters and their initial ranges has to be provided. It would be much more convenient if the bot could handle the set of all technical hyperparameters, with infinite ranges.
- Smarter, sequential sampling might be required to scale to high-dimensional hyperparameter spaces. But note that we not only care about optimal configurations but much rather would like to learn as much as possible about the considered parameter space, including areas of bad performance. So simply switching to Bayesian optimization or related search techniques might not be appropriate.

References

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.

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.

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.

M. Lang, B. Bischl, and D. Surmann. batchtools: Tools for R to work on batch systems. *The Journal of Open Source Software*, 2(10), 2017.

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.

P. Probst, B. Bischl, and A.-L. Boulesteix. Tunability: Importance of hyperparameters of machine learning algorithms. *ArXiv preprint arXiv:1802.09596*, 2018. URL https://arxiv.org/abs/1802.09596.

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. In *Meta-Learning and algorithm selection workshop at ECAI 2014*, page 41, 2014.

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.

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.