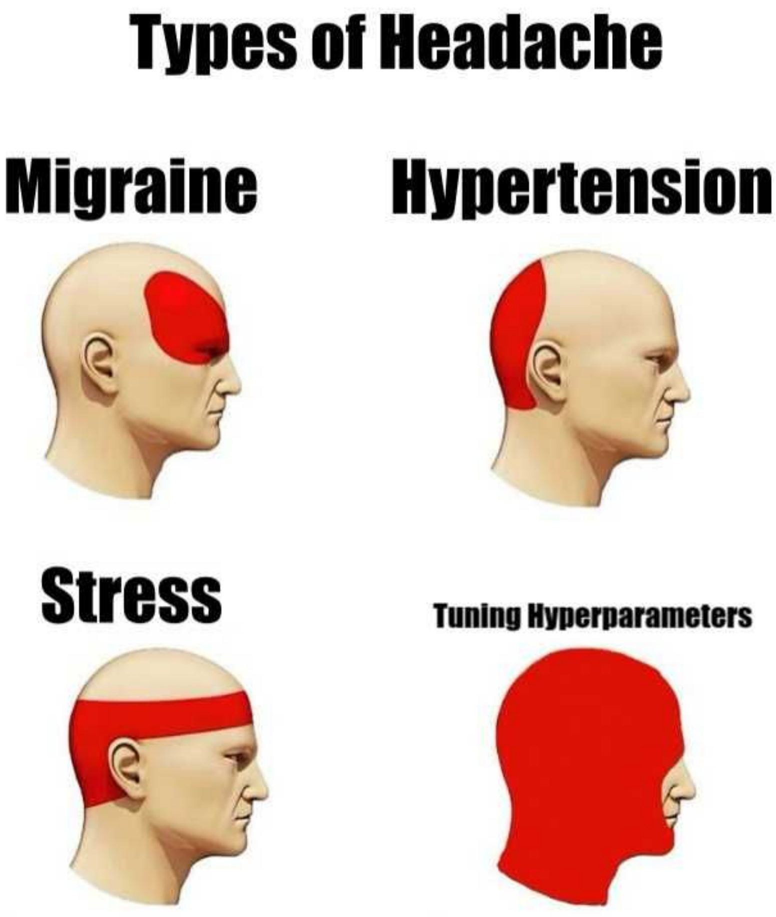
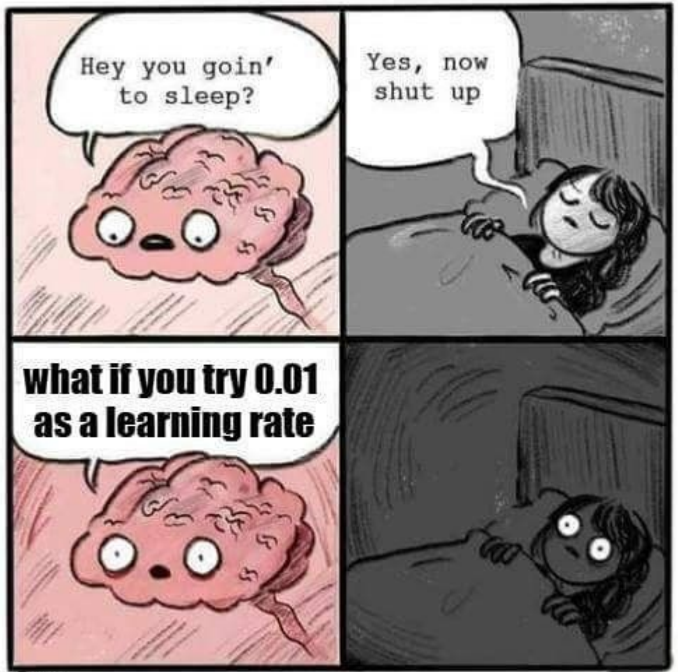




Alicja Gosiewska
15 IV 2019

Journal of Machine Learning Research 18 (2018) 1-32 Submitted 7/18; Revised 2/19; Published 3/19

Tunability: Importance of Hyperparameters of Machine Learning Algorithms

Philipp Probst PROBST@IBE.MED.UNI-MUENCHEN.DE
Institute for Medical Information Processing, Biometry and Epidemiology, LMU Munich
Marchioninistr. 15, 81377 München, Germany

Anne-Laure Boulesteix BOULESTEIX@IBE.MED.UNI-MUENCHEN.DE
Institute for Medical Information Processing, Biometry and Epidemiology, LMU Munich
Marchioninistr. 15, 81377 München, Germany

Bernd Bischl BERND.BISCHL@STAT.UNI-MUENCHEN
Department of Statistics, LMU Munich
Ludwigstraße 33, 80539 München, Germany

Editor: Ryan Adams

Abstract

Modern supervised machine learning algorithms involve hyperparameters that have to be set before running them. Options for setting hyperparameters are default values from the software package, manual configuration by the user or configuring them for optimal predictive performance by a tuning procedure. The goal of this paper is two-fold. Firstly, we formalize the problem of tuning from a statistical point of view, define data-based defaults and suggest general measures quantifying the tunability of hyperparameters of algorithms. Secondly, we conduct a large-scale benchmarking study based on 38 datasets from the OpenML platform and six common machine learning algorithms. We apply our measures to assess the tunability of their parameters. Our results yield default values for hyperparameters and enable users to decide whether it is worth conducting a possibly time consuming tuning strategy, to focus on the most important hyperparameters and to choose adequate hyperparameter spaces for tuning.

Problem:

- hyperparameters have to be set before running them,
    - default values
    - manual configuration
    - tuning procedure

1. *ML users—Which hyperparameters should be tuned and in which ranges?*
2. *Designers of ML algorithms—How do I define robust defaults?*

Problem:

- hyperparameters have to be set before running them,
    - default values
    - manual configuration
    - tuning procedure

Solution (goal of paper):

- yield default values for hyperparameters
    - formalization of the problem of tuning
        - define data-based defaults
        - suggest general measures quantifying the tunability of algorithm and hyperparameters
    - conduct a "*large-scale*" benchmark study

Problem:

- hyperparameters have to be set before running them,
  - default values
  - manual configuration
  - tuning procedure

Solution (goal of paper):

- yield default values for hyperparameters
  - formalization of the problem of tuning
    - define data-based defaults
    - suggest general measures quantifying the tunability of algorithm and hyperparameters
  - conduct a "*large-scale*" benchmark study

How:

- surrogate models (empirical performance models), which estimate the performance of arbitrary hyperparameter configurations based on a limited number of prior experiments

Experimental setup

OpenML Benchmarking Suites and the OpenML100

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael G. Mantovani, Jan N. van Rijn, Joaquin Vanschoren

https://arxiv.org/abs/1708.03731

Experimental setup

OpenML Benchmarking Suites and the OpenML100

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael G. Mantovani, Jan N. van Rijn, Joaquin Vanschoren

https://arxiv.org/abs/1708.03731

We provide a new standard benchmark suite of 100 high-quality datasets carefully curated from the many thousands available on OpenML.

Experimental setup

OpenML Benchmarking Suites and the OpenML100
Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael G. Mantovani, Jan N. van Rijn, Joaquin Vanschoren
https://arxiv.org/abs/1708.03731

A new standard benchmark suite of 100 high-quality datasets carefully curated from the many thousands available on OpenML.

a) the number of observations are between 500 and 100 000 to focus on medium-sized datasets,
b) the number of features does not exceed 5000 features to keep the runtime of algorithms low,
c) the target attribute has at least two classes,
d) the ratio of the minority class and the majority class is above 0.05 (to eliminate highly imbalanced datasets).

Experimental setup

OpenML Benchmarking Suites and the OpenML100

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael G. Mantovani, Jan N. van Rijn, Joaquin Vanschoren

https://arxiv.org/abs/1708.03731

A new standard benchmark suite of 100 high-quality datasets carefully curated from the many thousands available on OpenML.

a) the number of observations are between 500 and 100 000 to focus on medium-sized datasets,
b) the number of features does not exceed 5000 features to keep the runtime of algorithms low,
c) the target attribute has at least two classes,
d) the ratio of the minority class and the majority class is above 0.05 (to eliminate highly imbalanced datasets).

Tunability: Importance of Hyperparameters of Machine Learning Algorithms

Only use the 38 binary classification tasks that do not contain any missing values.

Experimental setup

38 datasets (binary classification)

Experimental setup

38 datasets (binary classification)
6 models

Models:

- elastic net (`glmnet`)
- decision tree (`rpart`)
- k-nearest neighbours (`kknn`)
- support vector machines (`svm`)
- random forest (`ranger`)
- gradient boosting (`xgboost`)

Experimental setup

38 datasets (binary classification)
6 models
hyperparameters

Models:

- elastic net (`glmnet`)
- decision tree (`rpart`)
- k-nearest neighbours (`kknn`)
- support vector machines (`svm`)
- random forest (`ranger`)
- gradient boosting (`xgboost`)

Algorithm	Hyperparameter	Type	Lower	Upper	Trafo
glmnet					
(Elastic net)	alpha	numeric	0	1	-
	lambda	numeric	-10	10	2^x
rpart					
(Decision tree)	cp	numeric	0	1	-
	maxdepth	integer	1	30	-
	minbucket	integer	1	60	-
	minsplit	integer	1	60	-
kknn	-	-			
(k-nearest neighbor)	k	integer	1	30	-
svm					
(Support vector machine)	kernel	discrete	-	-	-
	cost	numeric	-10	10	2^x
	gamma	numeric	-10	10	2^x
	degree	integer	2	5	-
ranger					
(Random forest)	num.trees	integer	1	2000	-
	replace	logical	-	-	-
	sample.fraction	numeric	0.1	1	-
	mtry	numeric	0	1	$x \cdot p$
	respect.unordered.factors	logical	-	-	-
	min.node.size	numeric	0	1	n^x
xgboost					
(Gradient boosting)	nrounds	integer	1	5000	-
	eta	numeric	-10	0	2^x
	subsample	numeric	0.1	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

Experimental setup

38 datasets (binary classification)
6 models
hyperparameters
3 performance measures (separate CVs)

Considered performance measures:

- **AUC**
- Accuracy
- Brier Score

Experimental setup

38 datasets (binary classification)
6 models
hyperparameters
3 performance measures (separate CVs)

Considered performance measures:

- **AUC**
- Accuracy
- Brier Score

Brier Score - the accuracy of probabilistic predictions

$$BS = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2$$

in which f_t is the probability that was forecast, o_t the actual outcome of the event at instance t (0 if it does not happen and 1 if it does happen) and N is the number of forecasting instances.

Random Bot (OpenML bot)

Automatic Exploration of Machine Learning Experiments on OpenML
Daniel Kühn, Philipp Probst, Janek Thomas, Bernd Bischl
https://arxiv.org/abs/1806.10961

Random Bot (OpenML bot)

Automatic Exploration of Machine Learning Experiments on OpenML
Daniel Kühn, Philipp Probst, Janek Thomas, Bernd Bischl
https://arxiv.org/abs/1806.10961

Each iteration:
- random dataset (38 data sets from OpenML)
- random classification algorithm (6 algorithms)
- random hyperparameters configuration (uniform distribution)
- 10-fold cross-validation

Random Bot (OpenML bot)

Automatic Exploration of Machine Learning Experiments on OpenML
Daniel Kühn, Philipp Probst, Janek Thomas, Bernd Bischl
https://arxiv.org/abs/1806.10961

Each iteration:
- random dataset (38 data sets from OpenML)
- random classification algorithm (6 algorithms)
- random hyperparameters configuration (uniform distribution)
- 10-fold cross-validation

A subset of 500000 experiments for each algorithm and all datasets are used for our analysis here (except of kknn for which only 1140 results are available).

Random Bot (OpenML bot)

Automatic Exploration of Machine Learning Experiments on OpenML
Daniel Kühn, Philipp Probst, Janek Thomas, Bernd Bischl
https://arxiv.org/abs/1806.10961

Each iteration:

- random dataset (38 data sets from OpenML)
- random classification algorithm (6 algorithms)
- random hyperparameters configuration (uniform distribution)
- 10-fold cross-validation

A subset of 500000 experiments for each algorithm and all datasets are used for our analysis here (except of kknn for which only 1140 results are available).

The results of the bot are stored in a figshare repository.

https://figshare.com/articles/OpenML_R_Bot_Benchmark_Data_final_subset_/5882230/2

$\hat{f}(X, \theta)$ - prediction model controlled by the hyperparameter configuration $\theta = (\theta_1, ..., \theta_k)$ from the hyperparameter search space $\Theta = \Theta_1 \times ... \times \Theta_k$

$\hat{f}(X, \theta)$ - prediction model controlled by the hyperparameter configuration $\theta = (\theta_1, ..., \theta_k)$ from the hyperparameter search space $\Theta = \Theta_1 \times ... \times \Theta_k$

Given m different datasets (or data distributions) $\mathcal{P}_1, ..., \mathcal{P}_m$, we arrive at m hyperparameter risk mappings

$$R^{(j)}(\theta) := E(L(Y, \hat{f}(X, \theta))|\mathcal{P}_j), \qquad j = 1, ..., m$$

$\hat{f}(X,\theta)$ - prediction model controlled by the hyperparameter configuration $\theta = (\theta_1, ..., \theta_k)$ from the hyperparameter search space $\Theta = \Theta_1 \times ... \times \Theta_k$

Given m different datasets (or data distributions) $\mathcal{P}_1, ..., \mathcal{P}_m$, we arrive at m hyperparameter risk mappings

$$R^{(j)}(\theta) := E(L(Y, \hat{f}(X,\theta))|\mathcal{P}_j), \qquad j = 1, ..., m$$

we estimate $R^{(j)}(\theta)$ by using surrogate models $\hat{R}^{(j)}(\theta)$

The surrogate regression model map a hyperparameter configuration to estimated performance.

$\hat{f}(X, \theta)$ - prediction model controlled by the hyperparameter configuration $\theta = (\theta_1, ..., \theta_k)$ from the hyperparameter search space $\Theta = \Theta_1 \times ... \times \Theta_k$

Given m different datasets (or data distributions) $\mathcal{P}_1, ..., \mathcal{P}_m$, we arrive at m hyperparameter risk mappings

$$R^{(j)}(\theta) := E(L(Y, \hat{f}(X, \theta))|\mathcal{P}_j), \qquad j = 1, ..., m$$

we estimate $R^{(j)}(\theta)$ by using surrogate models $\hat{R}^{(j)}(\theta)$

The surrogate regression model map a hyperparameter configuration to estimated performance.

Choice: Random forest

Optimal Default hyperparameters

We define the best hyperparameter configuration for dataset *j* as

$$\theta^{(j)\star} := \underset{\theta \in \Theta}{\arg\min}\, R^{(j)}(\theta).$$

Optimal Default hyperparameters

We define the best hyperparameter configuration for dataset *j* as

$$\theta^{(j)\star} := \arg\min_{\theta \in \Theta} R^{(j)}(\theta).$$

An optimal default configuration, based on empirical experiments on m different benchmark datasets:

$$\theta^{\star} := \arg\min_{\theta \in \Theta} g(R^{(1)}(\theta), ..., R^{(m)}(\theta)).$$

Here, g is a summary function that has to be specified. Selecting the mean (or median) would imply minimizing the average (or median) risk over all datasets.

Optimal Default hyperparameters

We define the best hyperparameter configuration for dataset *j* as

$$\theta^{(j)\star} := \underset{\theta \in \Theta}{\arg\min}\ R^{(j)}(\theta).$$

An optimal default configuration, based on empirical experiments on m different benchmark datasets:

$$\theta^{\star} := \underset{\theta \in \Theta}{\arg\min}\ g(R^{(1)}(\theta), \ldots, R^{(m)}(\theta)).$$

Here, g is a summary function that has to be specified. Selecting the mean (or median) would imply minimizing the average (or median) risk over all datasets.

For the estimation of the defaults for each algorithm we randomly sample 100000 points in the hyperparameter space and determine the configuration with the minimal average risk.

Parameter	Def.P	Def.O	Tun.P	Tun.O	$q_{0.05}$	$q_{0.95}$
glmnet			0.069	0.024		
alpha	1	0.403	0.038	0.006	0.009	0.981
lambda	0	0.004	0.034	0.021	0.001	0.147
rpart			0.038	0.012		
cp	0.01	0	0.025	0.002	0	0.008
maxdepth	30	21	0.004	0.002	12.1	27
minbucket	7	12	0.005	0.006	3.85	41.6
minsplit	20	24	0.004	0.004	5	49.15
kknn			0.031	0.006		
k	7	30	0.031	0.006	9.95	30
svm			0.056	0.042		
kernel	radial	radial	0.030	0.024		
cost	1	682.478	0.016	0.006	0.002	920.582
gamma	$1/p$	0.005	0.030	0.022	0.003	18.195
degree	3	3	0.008	0.014	2	4
ranger			0.010	0.006		
num.trees	500	983	0.001	0.001	206.35	1740.15
replace	TRUE	FALSE	0.002	0.001		
sample.fraction	1	0.703	0.004	0.002	0.323	0.974
mtry	$\sqrt{p}$	$p \cdot 0.257$	0.006	0.003	0.035	0.692
respect.unordered.factors	TRUE	FALSE	0.000	0.000		
min.node.size	1	1	0.001	0.001	0.007	0.513
xgboost			0.043	0.014		
nrounds	500	4168	0.004	0.002	920.7	4550.95
eta	0.3	0.018	0.006	0.005	0.002	0.355
subsample	1	0.839	0.004	0.002	0.545	0.958
booster	gbtree	gbtree	0.015	0.008		
max_depth	6	13	0.001	0.001	5.6	14
min_child_weight	1	2.06	0.008	0.002	1.295	6.984
colsample_bytree	1	0.752	0.006	0.001	0.419	0.864
colsample_bylevel	1	0.585	0.008	0.001	0.335	0.886
lambda	1	0.982	0.003	0.002	0.008	29.755
alpha	1	1.113	0.003	0.002	0.002	6.105

Parameter	Def.P	Def.O	Tun.P	Tun.O	$q_{0.05}$	$q_{0.95}$
glmnet			0.069	0.024		
alpha	1	0.403	0.038	0.006	0.009	0.981
lambda	0	0.004	0.034	0.021	0.001	0.147
rpart			0.038	0.012		
cp	0.01	0	0.025	0.002	0	0.008
maxdepth	30	21	0.004	0.002	12.1	27
minbucket	7	12	0.005	0.006	3.85	41.6
minsplit	20	24	0.004	0.004	5	49.15
kknn			0.031	0.006		
k	7	30	0.031	0.006	9.95	30
svm			0.056	0.042		
kernel	radial	radial	0.030	0.024		
cost	1	682.478	0.016	0.006	0.002	920.582
gamma	$1/p$	0.005	0.030	0.022	0.003	18.195
degree	3	3	0.008	0.014	2	4
ranger			0.010	0.006		
num.trees	500	983	0.001	0.001	206.35	1740.15
replace	TRUE	FALSE	0.002	0.001		
sample.fraction	1	0.703	0.004	0.002	0.323	0.974
mtry	$\sqrt{p}$	$p \cdot 0.257$	0.006	0.003	0.035	0.692
respect.unordered.factors	TRUE	FALSE	0.000	0.000		
min.node.size	1	1	0.001	0.001	0.007	0.513
xgboost			0.043	0.014		
nrounds	500	4168	0.004	0.002	920.7	4550.95
eta	0.3	0.018	0.006	0.005	0.002	0.355
subsample	1	0.839	0.004	0.002	0.545	0.958
booster	gbtree	gbtree	0.015	0.008		
max_depth	6	13	0.001	0.001	5.6	14
min_child_weight	1	2.06	0.008	0.002	1.295	6.984
colsample_bytree	1	0.752	0.006	0.001	0.419	0.864
colsample_bylevel	1	0.585	0.008	0.001	0.335	0.886
lambda	1	0.982	0.003	0.002	0.008	29.755
alpha	1	1.113	0.003	0.002	0.002	6.105

FICO data set, 5-fold CV

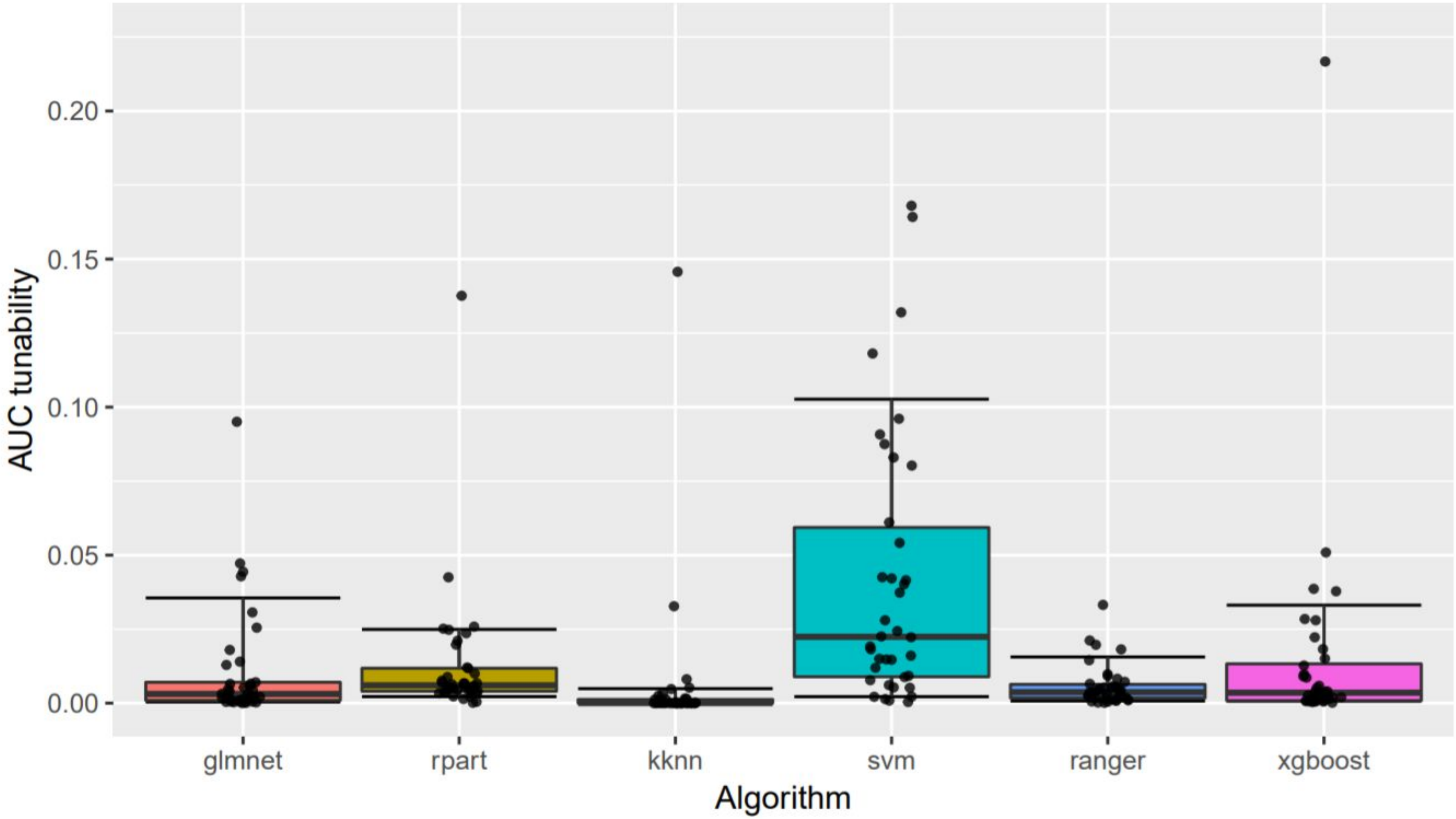
algorithm	package defaults	optimal defaults
glmnet	0.778	**0.780**
rpart	0.707	**0.740**
kknn	0.716	**0.744**
ranger	**0.793**	0.792
xgboost	0.767	**0.778**

Estimation of the tunability of an algorithm

A general measure of the tunability of an algorithm per dataset can then be computed based on the difference between the risk of an overall reference configuration and the risk of the best possible configuration on that dataset:

$$d^{(j)} := \underbrace{R^{(j)}(\theta^\star)}_{\text{risk of an overall reference configuration}} - \underbrace{R^{(j)}(\theta^{(j)\star})}_{\text{risk of the best possible configuration on } j\text{-th dataset}}, \text{ for } j = 1, ..., m$$

The strategy 100000 random points is used to obtain the best hyperparameter setting on each dataset that is needed for the estimation of the tunability of an algorithm.


AUC tunability
0.00
0.05
0.10
0.15
0.20
glmnet
rpart
kknn
svm
ranger
xgboost
Algorithm

Measuring Tunability of a Specific Hyperparameter

The best hyperparameter value for one parameter i on dataset j, when all other parameters are set to defaults from $\theta^\star := (\theta_1^\star, ..., \theta_k^\star)$, is denoted by

$$\theta_i^{(j)\star} := \underset{\theta \in \Theta, \theta_l = \theta_l^\star \forall l \neq i}{\arg\min} R^{(j)}(\theta).$$

Measuring Tunability of a Specific Hyperparameter

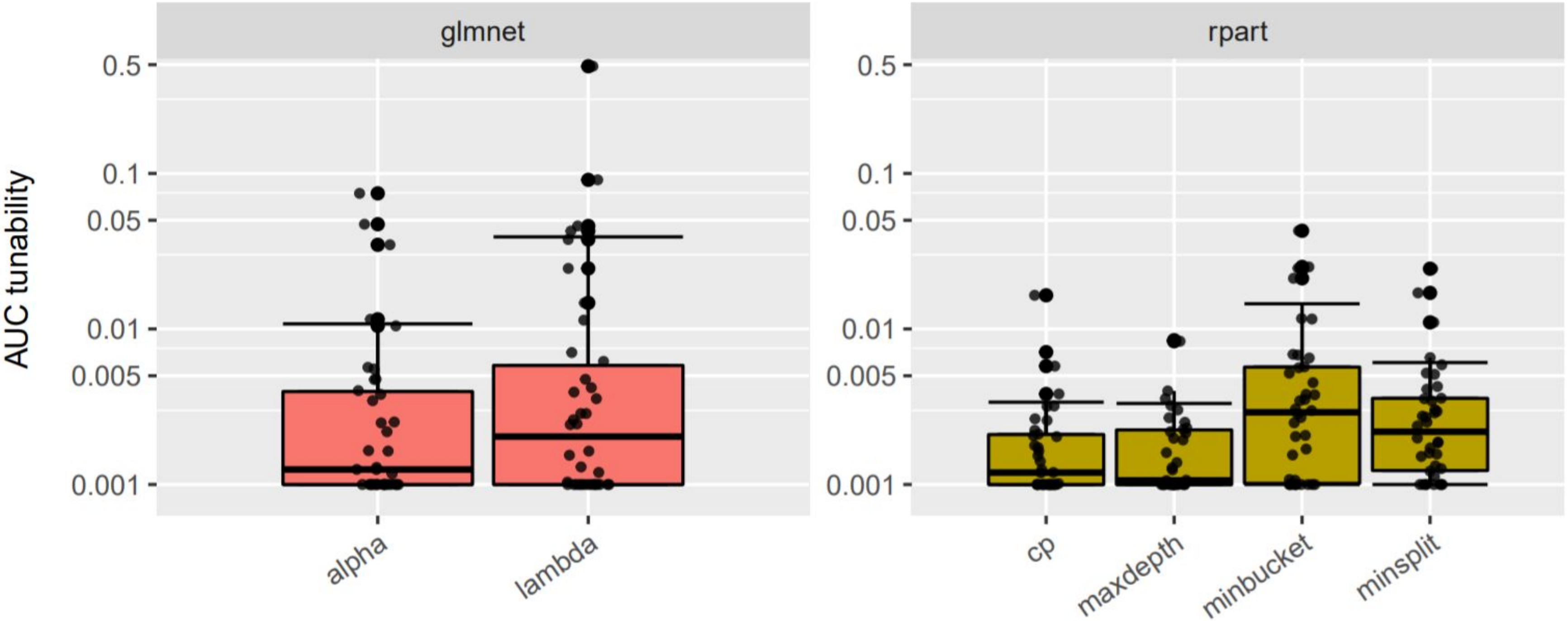
The best hyperparameter value for one parameter i on dataset j, when all other parameters are set to defaults from $\theta^\star := (\theta_1^\star, ..., \theta_k^\star)$, is denoted by

$$\theta_i^{(j)\star} := \underset{\theta \in \Theta, \theta_l = \theta_l^\star \forall l \neq i}{\arg\min} R^{(j)}(\theta).$$

A natural measure for tunability of the i-th parameter on dataset j is then the difference in risk between the above and our default reference configuration:

$$d_i^{(j)} := R^{(j)}(\theta^\star) - R^{(j)}(\theta_i^{(j)\star}), \text{ for } j = 1, ..., m, i = 1, ..., k.$$

More



glmnet
rpart
AUC tunability
0.5
0.1
0.05
0.01
0.005
0.001
alpha
lambda
cp
maxdepth
minbucket
minsplit

What I’ve liked about the article:

- released code **and data**
- shiny app **:)**
  https://philipppro.shinyapps.io/tunability/

What I've liked about the article:

- released code **and data**
- shiny app **:)**
  https://philipppro.shinyapps.io/tunability/

Possible improvements:

- application to multiclass classification, regression, survival analysis
- diversity of datasets in benchmark
- data-based hyperparameters
- better sampling (problem for high dimensional spaces)
- explanations of surrogate model

What I’ve liked about the article:

- released code **and data**
- shiny app **:)**
  https://philipppro.shinyapps.io/tunability/

Possible improvements:

- application to multiclass classification, regression, survival analysis
- diversity of datasets in benchmark
- data-based hyperparameters
- better sampling (problem for high dimensional spaces)
- explanations of surrogate model

Hyperparameter Importance Across Datasets
J. N. van Rijn, F. Hutter
https://arxiv.org/abs/1710.04725

Meta learning for defaults: symbolic defaults
Jan N. van Rijn, Florian Pfisterer, Janek Thomas, Andreas Muller, Bernd Bischl, J. Vanschoren
https://research.tue.nl/en/publications/meta-learning-for-defaults-symbolic-defaults