Hyperparameter Importance Across Datasets

Jan N. van Rijn, Frank Hutter
Presentation: Katarzyna Woźnica

May 2020

Intro i

Hyperparameter Importance Across Datasets (2017/2018)

Tunability: Importance of Hyperparameters of Machine Learning Algorithms (2018)

Intro ii

Abstract: (. . .) The conducted experiments confirm that the hyperparameters selected by the proposed method are indeed the most important ones and that the obtained **priors** also lead to **statistically significant improvements in hyperparameter** optimization.

Intro iii

algorithm A has n hyperparameters with domain $\Theta_1, \Theta_2, \ldots, \Theta_N$

configuration space: $\boldsymbol{\Theta} = \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N$

instantiation - one configuration: $\boldsymbol{\theta} = \langle \theta_1, \theta_2, \ldots, \theta_N \rangle$

partial instantiation of A: $\boldsymbol{\theta}_U = \langle \theta_i, \ldots, \theta_j \rangle$ with a subset $U \subseteq N$ of the hyperparameters fixed, and the values for other hyperparameters unspecified.

Decomposition of variance (functional ANOVA)

y - predictive measure e.g. accuracy

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

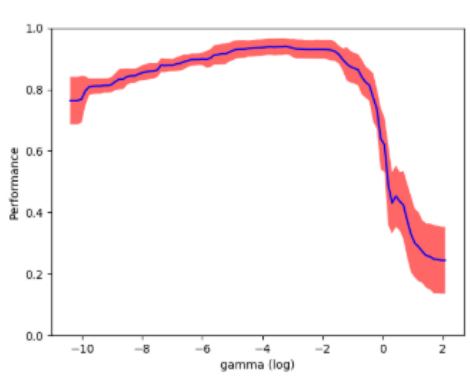
where $f(\boldsymbol{\theta}_U)$ capture the effect of varying hyperparameters

$$f(\boldsymbol{\theta}_U) = \begin{cases} f_\emptyset, & U = \emptyset \\ a_U(\boldsymbol{\theta}_U) - \sum_{W \subsetneq U} f(\boldsymbol{\theta}_W), & U \neq \emptyset \end{cases}$$

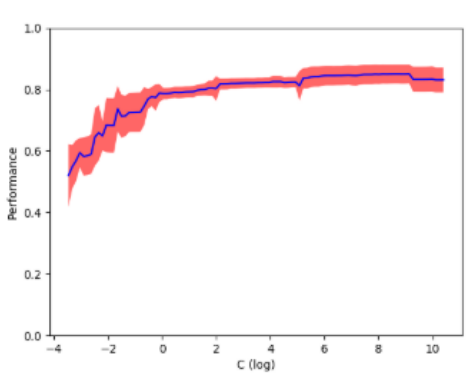
Overall variance $\mathbb{V}$ of y is decompose into $\mathbb{V} = \sum_{U \subset N} \mathbb{V}_U$ with $\mathbb{V}_U = \frac{1}{||\Theta_U||} \int f(\boldsymbol{\theta}_U)^2 d\boldsymbol{\theta}_U$

Efficient Marginal Performance Predictions

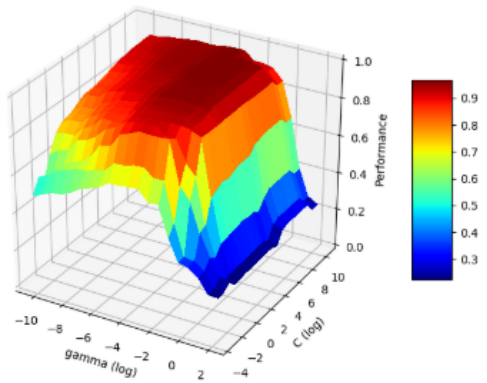
The marginal performance $a_U(\theta_U)$ is defined as the average performance of all complete instantiations θ that agree with θ_U in the instantiations of hyperparameters U.



(a) Gamma



(b) Complexity



(c) Gamma vs. Complexity

Figure 1: Marginal predictions for a SVM with RBF kernel on the letter dataset. The hyperparameter values are on a log scale.

Plan of experiment

For every dataset:

surrogate model:
random forest

$y(\boldsymbol{\theta})$ - true values
$a_U(\boldsymbol{\theta}_U)$ - computionaly complex

$\hat{y}(\boldsymbol{\theta})$ - predicted values
$\hat{a}_U(\boldsymbol{\theta}_U)$ - linear complexity for random forest
$\hat{f}_U(\boldsymbol{\theta}_U)$
$\mathbb{V}_U$ and scaled $\mathbb{V}_U/\mathbb{V}$

Then across data sets $\mathbb{V}_U/\mathbb{V}$ can be compared.

Experiment settings

Table 1: SVM Hyperparameters.

hyperparameter	values	description
complexity	$[2^{-5}, 2^{15}]$ (log-scale)	Soft-margin constant, controlling the trade-off between model simplicity and model fit.
coef0	$[-1, 1]$	Additional coefficient used by the kernel (sigmoid kernel only).
gamma	$[2^{-15}, 2^{3}]$ (log-scale)	Length-scale of the kernel function, determining its locality.
imputation	{mean, median, mode}	Strategy for imputing missing numeric variables.
shrinking	{true, false}	Determines whether to use the shrinking heuristic (introduced in [24]).
tolerance	$[10^{-5}, 10^{-1}]$ (log-scale)	Determines the tolerance for the stopping criterion.

Table 2: Random Forest Hyperparameters.

hyperparameter	values	description
bootstrap	{true, false}	Whether to train on bootstrap samples or on the full train set.
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 in order to create a leaf.
min. samples split	[2, 20]	The minimal number of data points required to split an internal node.
imputation	{mean, median, mode}	Strategy for imputing missing numeric variables.
split criterion	{entropy, gini}	Function to determine the quality of a possible split.

Table 3: Adaboost Hyperparameters.

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

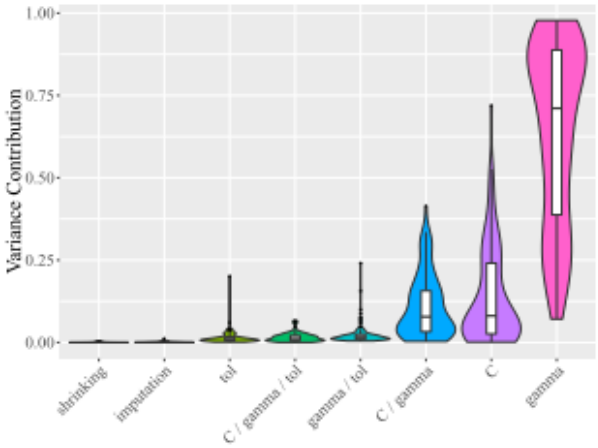
100 OpenML datasets, 3 algorithms

Verifying method

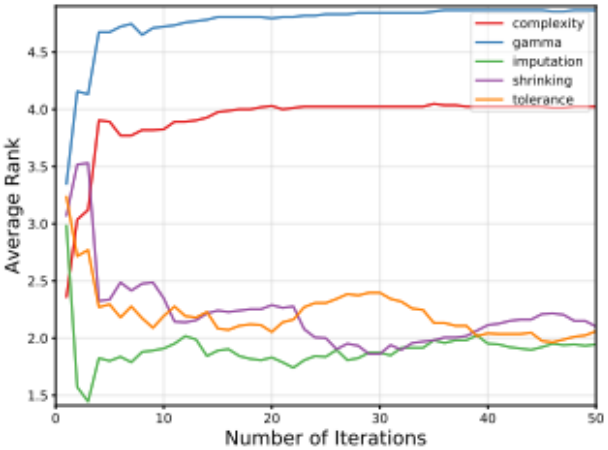
For each hyperparameter θ_j we measure $y^*_{j,f}$ - result of a random search for maximizing accuracy, fixing θ_j to a given value $f \in F_j$.

We then compute $y^*_j = \frac{1}{||F_j||} \sum y^*_{j,f}$, representing the score when not optimizing hyperparameter y^*_j , averaged over fixing $y^*_{j,f}$ to various values it can take.

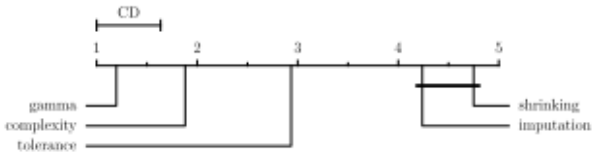
Hyperparameter importance - SVM



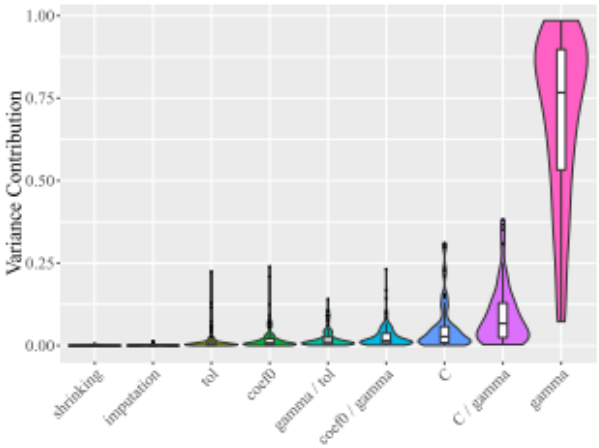
(a) Marginal contribution per dataset



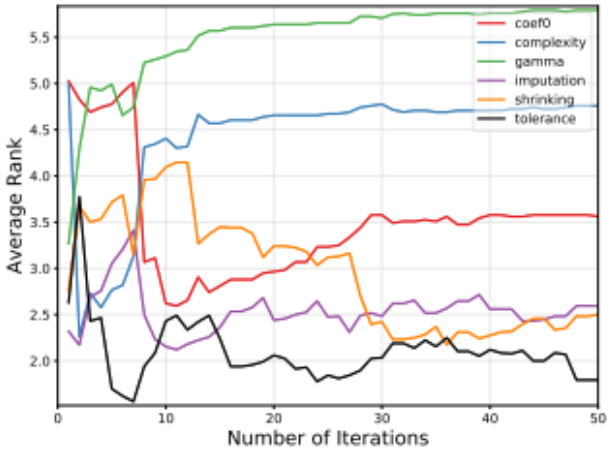
(b) Random Search, excluding one parameter at a time



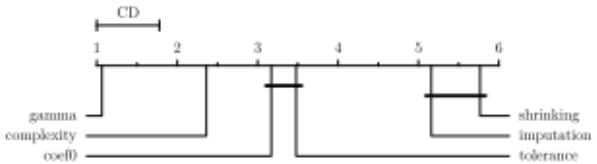
(c) Ranked hyperparameter importance, $\alpha = 0.05$.



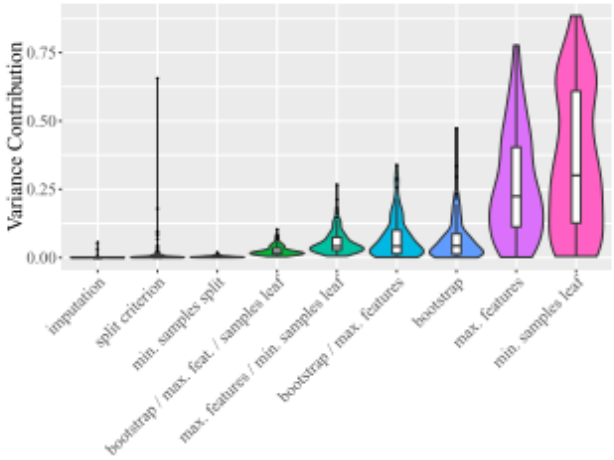
(a) Marginal contribution per dataset



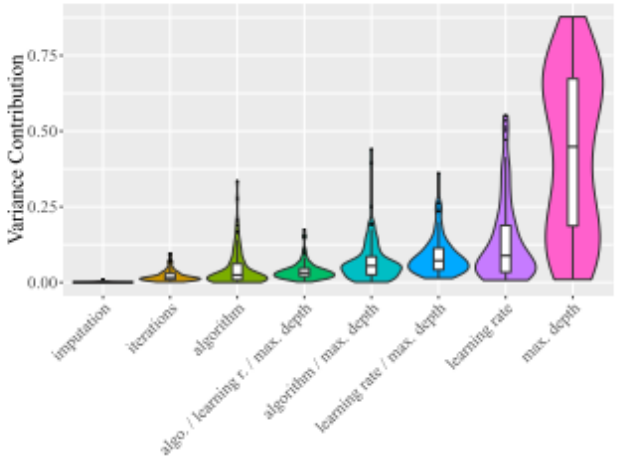
(b) Random Search, excluding one parameter at a time



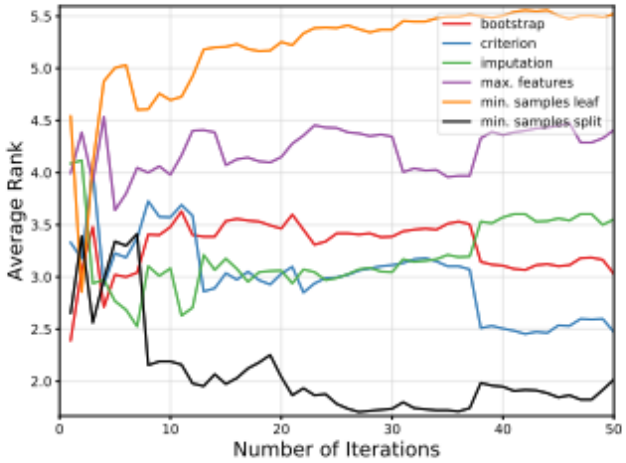
Hyperparameter importance - Random Forest and Adaboost



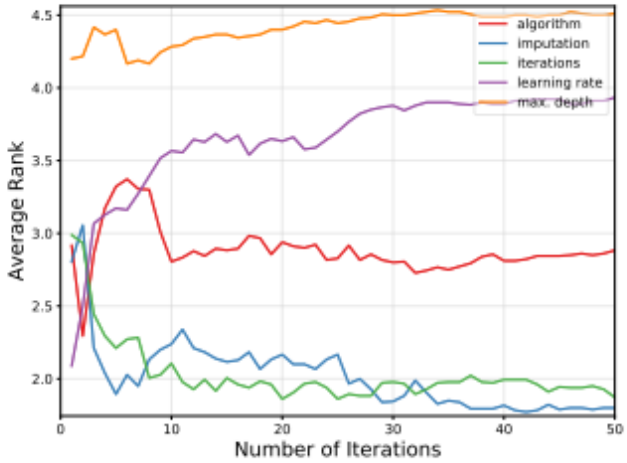
(a) Marginal contribution per dataset



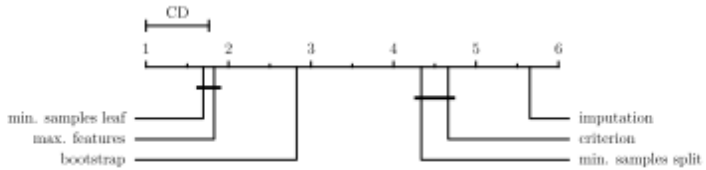
(a) Marginal contribution per dataset

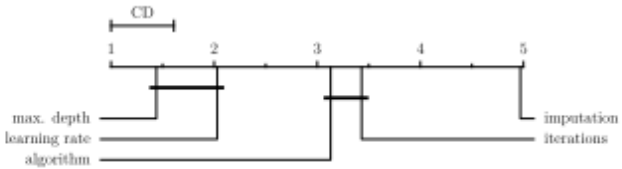


(b) Random Search, excluding one parameter at a time



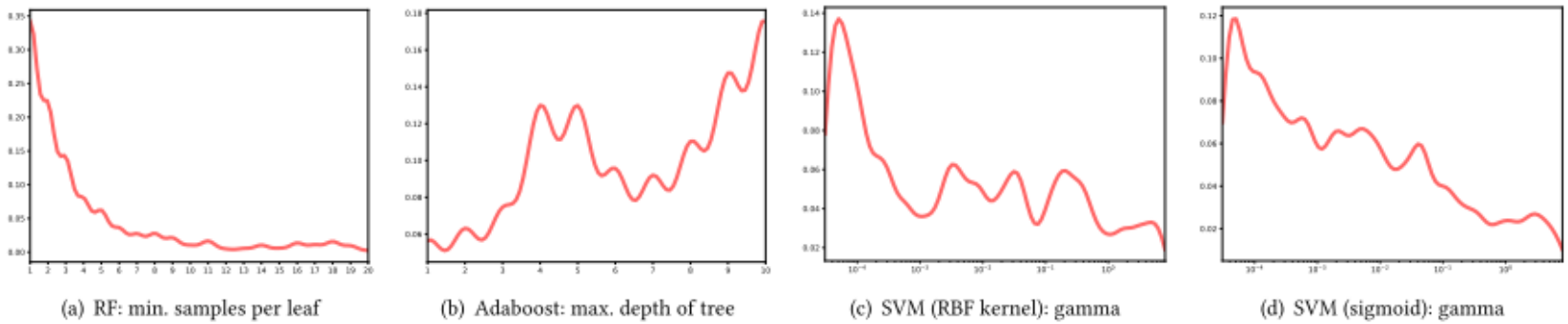
(b) Random Search, excluding one parameter at a time





Prior distribution

(...) proposed to fit kernel density estimators (...) we used the top n configurations observed for each of the datasets; in our experiments, we set n = 10.



(a) RF: min. samples per leaf (b) Adaboost: max. depth of tree (c) SVM (RBF kernel): gamma (d) SVM (sigmoid): gamma

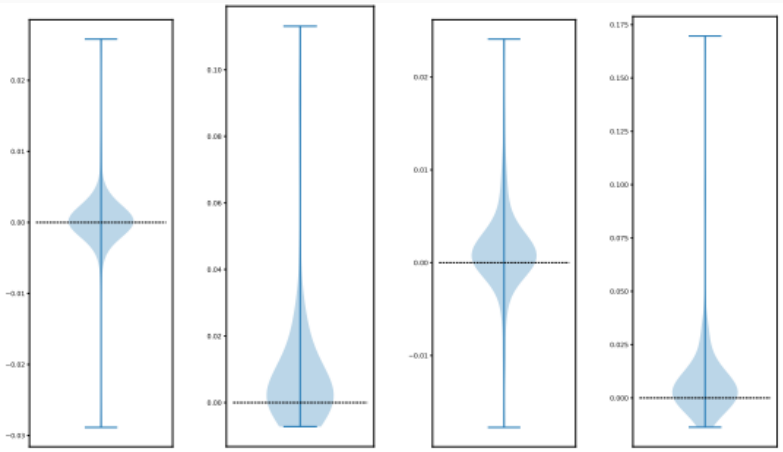
Figure 6: Obtained priors for the hyperparameter found to be most important for each classifier. The x-axis represents the value, the y-axis represents the probability that this value will be sampled (integer parameters will be rounded).

Verifying the usefulness of priors

Hyperband is based on the procedure of successive halving, which evaluates a large number of randomly-chosen configurations using only a small budget, and iteratively increases this budget, at each step only retaining a fraction of configurations that are best so far. For each dataset, we propose to run two versions of this optimization

procedure: one sampling uniformly from the hyperparameter space and one sampling from the obtained priors

Verifying the usefulness of priors



(a) SVM (RBF) (b) Sigmoid (c) Adaboost (d) RF

Figure 7: Difference in performance between two instances of Hyperband, one sampling based on the obtained priors and one using uniform sampling. Values bigger than zero indicate superior performance for the procedure sampling based on the priors, and vice-versa.

Table 4: Results of Nemenyi test ($\alpha = 0.05, CD \approx 0.20$). We report ranks across M datasets (max. 100), boldface the better approach (lower rank) and show whether the improvement is significant.

Classifier	M	Uniform	Priors	Sig.
random forest	100	1.72	**1.28**	yes
Adaboost	92	1.71	**1.29**	yes
SVM (sigmoid)	86	1.73	**1.27**	yes
SVM (RBF)	89	1.60	**1.40**	yes