Meta Learning for Defaults – Symbolic Defaults

Jan N. van Rijn
Data Science Institute
Columbia University

Florian Pfisterer
Department of Statistics
LMU Munich

Janek Thomas
Department of Statistics
LMU Munich

Andreas Müller
Data Science Institute
Columbia University

Bernd Bischl
Department of Statistics
LMU Munich

Joaquin Vanschoren
Mathematics and Computer Science
Eindhoven University of Technology

Abstract

In this work we propose to use meta-learning to learn sets of *symbolic* default hyperparameter configurations that work well across many data sets. A well known example for such a symbolic default is the logarithmic relation between the number of features of a dataset and the available features per split of a Random Forest, as observed by Breiman (2001). Symbolic functions allow for a more rich vocabulary to define defaults on. In the past, symbolic and static default values have been obtained either from hand-crafted heuristics or empirical evaluations of specific algorithms. We propose to automatically learn such symbolic configurations, i.e., *formulas* containing meta-features, from a large set of prior evaluations of numeric hyperparameters on multiple data sets via symbolic regression and optimization.

1 Introduction

The performance of most machine learning algorithms is greatly influenced by their hyperparameter settings. Various methods exist to automatically optimize hyperparameters, including random search (Bergstra and Bengio, 2012), Bayesian optimization (Snoek et al., 2012; Hutter et al., 2011), meta-learning (Brazdil et al., 2008) and bandit-based methods (Li et al., 2017). Depending on the algorithm, proper tuning of hyperparameters can yield considerable performance gains (Lavesson and Davidsson, 2006). Despite the acknowledged importance of tuning hyperparameters, the additional run time, code complexity and experimental design questions cause many practitioners to leave many hyperparameters to their default values, especially in real-world machine learning pipelines containing many hyperparameters. Moreover, it seems less than ideal to optimize all hyperparameters from scratch with every new dataset. If the optimal values of an hyperparameter are functionally dependent on properties of the data, we could learn this functional relationship and express them as symbolic default configurations that work well across many data sets. That way we can transfer information from previous optimization runs to obtain better data set dependent defaults and good starting configurations for further tuning.

Some of these functional relationships are reported in the literature, such as is the logarithmic relation between the number of features of a dataset and the available features per split of a Random Forest Breiman (2001). Other examples are the interaction between the RBF kernel bandwidth parameter (gamma) in SVM's and the number of features (Vanschoren et al., 2012) or the median distance between observations (Caputo et al., 2002). Some of these are also implemented in machine-learning workbenches such as `sklearn` (Pedregosa et al., 2011), `weka` (Hall et al., 2009) or `mlr` (Bischl et al., 2016). It is often not clear and rarely reported how such relationships were discovered, nor does there seem to be a clear consensus between workbenches on which symbolic defaults to implement. Also, they are typically limited to a single hyperparameter, and don't take into account how multiple hyperparameters may interact.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Meta-learning approaches have been proposed to learn *static* defaults (Pfisterer et al., 2018; Probst et al., 2018; Weerts et al., 2018; Wistuba et al., 2015), to find which hyperparameters are most important to optimize (van Rijn and Hutter, 2018; Probst et al., 2018; Weerts et al., 2018), or to build meta-models to select the kernel or kernel width in SVMs (Soares et al., 2004; Valerio and Vilalta, 2014; Strang et al., 2018).

This paper addresses a new meta-learning challenge: "Can we *learn* sets of *symbolic* configurations for hyperparameters of state-of-the-art machine learning algorithms?". Contrary to static defaults, symbolic defaults should be a function of the meta-features of the data set at hand. Ideally, these meta-features are easily computed, so that the symbolic default configurations can be easily implemented into software frameworks with little to no computational overhead. We show that such symbolic defaults outperform the best overall static defaults, and propose techniques to learn such symbolic defaults via symbolic regression and optimization.

2 Problem definition

2.1 Preliminaries

Consider a target variable y, a feature vector X, and an unknown joint distribution P on (X, y), from which we have sampled a dataset $\mathcal{D}$ containing $|\mathcal{D}|$ observations. A machine learning (ML) algorithm tries to approximate the functional relationship between X and y by producing a prediction model $\hat{f}_\theta(X)$, controlled by a multi-dimensional hyperparameter configuration $\theta \in \Theta$ of length p: $\theta = \{\phi_1, \ldots, \phi_p\}$. In order to measure prediction performance pointwise between a true label y and its prediction $\hat{f}(X)$, we define a loss function $L(y, \hat{f}(X))$.
We are naturally interested in estimating the expected risk of the inducing algorithm, w.r.t. θ on new data, also sampled from $\mathcal{P}$: $R_\mathcal{P}(\theta) = E(L(y, \hat{f}(X))|\mathcal{P})$. Thus, $R_\mathcal{P}(\theta)$ quantifies the expected predictive performance associated with a hyperparameter configuration θ for a given data distribution, learning algorithm and performance measure. Given a data distribution, a learning algorithm and a performance measure, this mapping encodes the numerical quality for any hyperparameter configuration θ.

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

$$R_k(\theta) = E(L(y, \hat{f}(X, \theta))|\mathcal{P}_k), \qquad k = 1, ..., K.$$

2.2 Meta Data

Evaluations To learn symbolic defaults, we first gather meta-data that evaluates $R_k(\theta)$ on all K datasets. For a given fixed algorithm with hyperparameter space Θ and a performance measure, e.g., area under the ROC curve (AUC), a large number of experiments is run on datasets $P_1, \ldots, P_K$. These experiments can be generated by a simple random search, i.e., by sampling random hyperparameter configurations from Θ, and evaluating them via cross-validation.

Surrogate Models In principle, it is possible to estimate $R_k(\theta)$ empirically using cross-validation for every $\theta \in \Theta$. However, since each cross-validation involves training many models, this is costly if we want to obtaining results for a large number of configurations. Therefore, we propose to employ *surrogate models* that predict the outcome of a given performance measure and algorithm for a given hyperparameter configuration. We train one model for each dataset (and each algorithm) on a sufficiently large random sample of evaluations (Eggensperger et al., 2015). For this, we can also reuse evaluations shared on OpenML (Vanschoren et al., 2014). These surrogate models provides us with a fast approximate way to evaluate the performance of any given configuration, without the requirement of costly training and evaluating models for every possible configuration.

Considering the fact that performances on different datasets are usually not commensurable (Demšar, 2006), an appropriate scaling is required before training surrogate models to enable a comparison between datasets. This is done in literature by resorting to ranking (Bardenet et al., 2013), or scaling (Yogatama and Mann, 2014) to standard deviations from the mean. We mitigate the problem of lacking commensurability between datasets by normalizing performance results on a per-dataset

Table 1: Simple transformation functions, parameterized by a constant α and a meta-feature value x.

transformation	function
linear	$x \cdot \alpha$
square root	$\sqrt{x} \cdot \alpha$
logaritmic	$\log(x) \cdot \alpha$
inverse	α / x
exponentiation	x^{α}

basis. A drawback to this is that some information regarding the absolute performance of the algorithm and the spread across different configurations is lost.

Data set characteristics In addition to the performance of random hyperparameter-configurations, OpenML contains a range of dataset characteristics, i.e, meta-features. A full list of available characteristics is described by van Rijn (2016). These characteristics include (among many others) the number of observations, the number of features and information regarding class balance. We denote the set of characteristics $\{c_1, c_2, ..., c_L\}$ with C.

2.3 Hypothesis space

Finding an optimal default now corresponds to finding a configuration θ that minimizes the risk $R_k(\theta)$ across K datasets. We define the risk over K datasets $R(\theta) = \frac{1}{K} \sum_1^K R_k(\theta)$, i.e., aggregate over datasets using the mean.

We allow our configurations to be symbolic, i.e., contain formulas instead of static values. For this reason we define a set of transformations T that are functions of the data set's meta-features, and map from the values of these meta-features to a real value for a given numeric hyperparameter θ_i, thus $t(\mathbf{x}) : \mathbb{R} \rightarrow \mathbb{R}$. Table 1 shows a list of simple transformations from a single meta-feature x. Note that although these symbolic function have a parameter (denoted by α), the optimal value for this will be determined by the search procedure. Of course, many more complex transformations can be considered as well.

Note that not all possible combinations will map the input to sensible output ranges. For example, the exponential function may generate unreasonable high values for high values for α. We can either add additional constraints on the transformed values to map them back into a reasonable range, or constrain the search method at these functions to not consider them. In this work, we opted for the latter.

3 Exhaustive search results

To demonstrate the utility of symbolic defaults, we first perform an exhaustive search on the simplified hypothesis space shown in Table 1. Given these transformation functions, a set of meta-features and a set of constant values (for parameter α), we enumerate all possible symbolic functions for a given hyperparameter, evaluate them on a wide set of datasets, and select the optimal one. Also, instead of jointly learning symbolic defaults for all hyperparameters, we only allow one symbolic default at a time, and set all other hyperparameters to a static default such that for the configuration the risk across datasets is minimized.

Setup The experiment is based on datasets from the OpenML100 (Bischl et al., 2017) benchmark suite. A rbf-SVM is used and only the hyperparameters γ and C are optimized. We generate candidate transformations according to Table 1, using a numerical constant (α) geometrically increasing with 10 steps from 0.1 to 2, and 80+ meta-features available from OpenML. This allows for $4{,}000$ symbolic expressions per hyperparameter. The search procedure should select the best among these.

The evaluation is based on a leave-one-dataset-out strategy, where the (symbolic) defaults are computed based on all but one dataset, and compared to the best *vanilla* default values. The vanilla defaults were computed by doing a full grid search over the hyperparameter space (with 8 values per hyperparameter), using the corresponding surrogate model to predict the performance on a specific

Table 2: Comparison between vanilla defaults and symbolic defaults, on 98 datasets from the OpenML100 (Bischl et al., 2017). Full results are displayed in Table 3 in the appendix.

strategy	wins	symbolc default configuration
symbolic	59	$\gamma = 0.189824/\mathrm{NumberOfFeatures}$, $C = 86.13$
vanilla	36	$\gamma = 0.001078$, $C = 4522.35$

dataset. The best overall configuration across all datasets is the best vanilla default value. The meta-data used to train the surrogate models is the same as used by van Rijn and Hutter (2018).

Results The results of the experiment can be seen in Table 2, in the appendix. The OpenML100 consists of 100 datasets; for 2 datasets the meta-data was incomplete. Out of the 98 datasets on which the defaults were evaluated, the symbolic defaults outperform the vanilla defaults in 59 cases, lose in 36 and draws in 3 cases. In all cases, the found default configuration was consistent across all leave-one-out cross-validation folds, for both symbolic and vanilla defaults. The latter indicates that (i) the set of datasets is large enough to learn meaningful defaults on, and (ii) the learned defaults generalize over tasks. Moreover. since these results were obtained from a simplified search space, it is quite possible that even better symbolic defaults can be discovered, as well as configurations in which multiple hyperparameters have symbolic defaults.

Note that these findings are in line what was reported by Vanschoren et al. (2012), who stated that they could not find a direct correlation, but that high gamma values are predominantly performing well on datasets with a low number of features.

4 Outlook

The method detailed in the previous sections demonstrated the feasibility of learning a set of *simple* data dependent defaults. In future work we first of all plan to extend the search space: we want to find formulas not for a single hyperparameter, but instead for all sensible hyperparameters of an algorithm. The current experiment additionally introduces prior assumptions with regards to the kind of functions we are able to learn, and we currently limit our approach to transformations that contain a single meta-feature. In future work, we want to introduce fewer restrictions to the space of possible transformations. As such, we plan to include combinations of meta-features, as well as introduce a host of significantly more complex transformations. Allowing for more complex formulas thus reduces the amount of prior assumptions we have to introduce. This comes with a cost: it is no longer sensible, or depending on the search space impossible, to exhaustively search through the space of possible formulas, even when using a surrogate model. One possible approach to solve this is to represent the space of functions as a grammar in Backus-Naur form and represent generated formulas as integer vectors where each entry represents which element of the right side of the grammar rule to follow (O'Neill and Ryan (2001), Noorian et al. (2016)). Using this representation, more advanced techniques like genetic algorithms can also be used to search larger and more complex sets of transformation functions in a much more efficient way.

Multiple challenges with this approach still exist. A search across the space of all possible functions may result in invalid values, or values out of the valid range of the hyperparameter for specific datasets. This does not necessarily pose a problem for genetic algorithms, as a few valid formulas already suffice, but hampers the efficiency of the search procedure. Additionally, a concurrent search for optimal formulas of all hyperparameters of an algorithm is difficult, because obtaining a bad value for only a single hyperparameter ϕ out of the full configuration θ can result in a bad performance overall. We propose to solve this using a round-robin approach, where we repeatedly iterate over all parameters and only learn a formula for one hyperparameter at a time.

On the other hand, we hope to gain several insights that do not only advance the state of research, but also improve the performance and robustness of many widely used machine learning algorithms, and thus widely influence the quality of learned models for users who are not able to tune all model hyperparameters.

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant No. 1740305 and by DARPA under Grant No. DARPA-BAA-16-51.

References

Bardenet, R., Brendel, M., Kégl, B., and Sebag, M. (2013). Collaborative hyperparameter tuning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages II–199–II–207. JMLR.org.

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

Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., and Vanschoren, J. (2017). OpenML Benchmarking Suites and the OpenML100. *arXiv preprint arXiv:1708.03731*.

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G., and Jones, Z. M. (2016). mlr: Machine learning in R. *JMLR*, 17(170):1–5.

Brazdil, P., Giraud-Carrier, C., Soares, C., and Vilalta, R. (2008). *Metalearning: Applications to Data Mining*. Springer Publishing Company, Incorporated, 1 edition.

Breiman, L. (2001). Random forests. *Mach. Learn.*, 45(1):5–32.

Caputo, B., Sim, K., Furesjo, F., and Smola, A. (2002). Appearance-based object recognition using svms: which kernel should i use? In *Procceedings of NIPS workshop on Statitsical methods for computational experiments in visual processing and computer vision, Whistler*, pages 1–10.

Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, 7:1–30.

Eggensperger, K., Hutter, F., Hoos, H., and Leyton-Brown, K. (2015). Efficient benchmarking of hyperparameter optimizers via surrogates. In *Proc. of AAAI 2015*, pages 1114–1120.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer.

Lavesson, N. and Davidsson, P. (2006). Quantifying the impact of learning algorithm parameter tuning. In *AAAI*, volume 6, pages 395–400.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2017). Hyperband: Bandit-Based Configuration Evaluation for Hyperparameter Optimization. In *Proc. of ICLR 2017*.

Noorian, F., de Silva, A. M., Leong, P. H., et al. (2016). gramevol: Grammatical evolution in r. *Journal of Statistical Software*, 71(i01).

O'Neill, M. and Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pfisterer, F., van Rijn, J. N., Probst, P., Müller, A. M., and Bischl, B. (2018). Learning Multiple Defaults for Machine Learning Algorithms. *arXiv preprint arXiv:1811.09409*.

Probst, P., Bischl, B., and Boulesteix, A. (2018). Tunability: Importance of hyperparameters of machine learning algorithms. *arXiv preprint arXiv:1802.09596*.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2951–2959, USA. Curran Associates Inc.

Soares, C., Brazdil, P., and Kuba, P. (2004). A meta-learning method to select the kernel width in support vector regression. *Mach. Learn.*, 54:195–209.

Strang, B., van der Putten, P., van Rijn, J. N., and Hutter, F. (2018). Don't rule out simple models prematurely: A large scale benchmark comparing linear and non-linear classifiers in openml. In *International Symposium on Intelligent Data Analysis*, pages 303–315. Springer.

Valerio, R. and Vilalta, R. (2014). Kernel selection in support vector machines using gram-matrix properties. In *NIPS Workshop on Modern Nonparametrics: Automating the Learning Pipeline*, volume 14.

van Rijn, J. N. (2016). *Massively Collaborative Machine Learning*. PhD thesis, Leiden University.

van Rijn, J. N. and Hutter, F. (2018). Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2367–2376. ACM.

Vanschoren, J., Blockeel, H., Pfahringer, B., and Holmes, G. (2012). Experiment databases. *Machine Learning*, 87(2):127–158.

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2014). OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60.

Weerts, H., Meuller, M., and Vanschoren, J. (2018). Importance of tuning hyperparameters of machine learning algorithms. Technical report, TU Eindhoven.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2015). Learning hyperparameter optimization initializations. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE.

Yogatama, D. and Mann, G. (2014). Efficient Transfer Learning Method for Automatic Hyperparameter Tuning. In Kaski, S. and Corander, J., editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 1077–1085, Reykjavik, Iceland. PMLR.

A Results per dataset

Table 3: The difference between symbolic defaults and vanilla defaults, for 98 datasets from the OpenML100 (Bischl et al., 2017). The best found symbolic default was '$\gamma = 0.189824/\mathrm{NumberOfFeatures},\ C = 86.13$', consistent across all tasks. The best found vanilla default was '$\gamma = 0.001078,\ C = 4522.35$', consistent across all tasks.

dataset	symbolic	vanilla
kr-vs-kp	**0.993986**	0.993256
letter	**0.962879**	0.913438
balance-scale	**0.948774**	0.915973
mfeat-factors	0.997986	**0.998242**
mfeat-fourier	**0.994684**	0.991689
breast-w	0.878953	**0.936559**
mfeat-karhunen	**0.993776**	0.989547
mfeat-morphological	0.974419	**0.976140**
mfeat-pixel	**0.993166**	0.982842
car	**0.972400**	0.971796
mfeat-zernike	0.968154	**0.976384**
cmc	0.740027	**0.864494**
mushroom	0.999673	**0.999885**
optdigits	**0.998058**	0.993982
credit-approval	0.905408	**0.951653**
credit-g	**0.771748**	0.682432
pendigits	**0.995317**	0.989628
segment	0.976898	**0.991487**
diabetes	0.835865	**0.874568**
soybean	**0.991540**	0.987747
spambase	**0.993783**	0.982050
splice	**0.993554**	0.976631
tic-tac-toe	0.980773	**0.989053**
vehicle	**0.952675**	0.921425
waveform-5000	**0.959832**	0.922108
electricity	**0.499086**	0.440656
satimage	**0.959691**	0.958328
eucalyptus	0.865020	**0.950900**
sick	**0.968849**	0.927118
vowel	**0.997565**	0.967072
isolet	0.996529	**0.999196**
scene	**0.945410**	0.923344
monks-problems-1	**0.999999**	0.978106
monks-problems-2	**0.997830**	0.888671
monks-problems-3	0.993750	**0.997656**
JapaneseVowels	0.956405	**0.958125**
synthetic_control	**0.988586**	0.986362
irish	0.999619	**1.000000**
analcatdata_authorship	0.997976	**0.998362**
analcatdata_dmft	**0.689489**	0.681528
profb	0.631895	**0.931988**
collins	1.000000	1.000000
mnist_784	0.869533	**0.995322**
sylva_agnostic	**0.974975**	0.958614
gina_agnostic	0.949198	**0.994346**
ada_agnostic	**0.939905**	0.915434
mozilla4	**0.799870**	0.696705
pc4	**0.973807**	0.939981
pc3	**0.987253**	0.987021
jm1	**0.519439**	0.411916
kc2	**0.717854**	0.453129
kc1	**0.800003**	0.477086
pc1	**0.411465**	0.367177
KDDCup09_upselling	0.988281	0.988281
MagicTelescope	**0.947935**	0.839991
adult	0.879703	**0.960864**
wilt	**0.998123**	0.988539
wdbc	**0.986231**	0.959469
micro-mass	**0.960787**	0.837078
phoneme	**0.636225**	0.422257
one-hundred-plants-margin	**0.986408**	0.984546
one-hundred-plants-shape	0.935152	**0.937328**
one-hundred-plants-texture	**0.988585**	0.982143
qsar-biodeg	**0.953944**	0.949305
wall-robot-navigation	**0.965796**	0.918964
semeion	0.986686	**0.989221**
steel-plates-fault	0.998367	**0.999982**
tamilnadu-electricity	1.000000	1.000000
hill-valley	0.442016	**0.766961**
ilpd	**0.907537**	0.898711
madelon	0.859188	**0.908491**
nomao	**0.998656**	0.990892
ozone-level-8hr	**0.920647**	0.832482
cardiotocography	0.998393	**0.999983**
climate-model-simulation-crashes	**0.917094**	0.901476
cnae-9	**0.989397**	0.930784
eeg-eye-state	0.341995	**0.376861**
first-order-theorem-proving	0.635427	**0.683760**
gas-drift	**0.998310**	0.997685
banknote-authentication	**1.000000**	0.978950
blood-transfusion-service-center	**0.801340**	0.734690
artificial-characters	**0.562517**	0.473162
bank-marketing	0.767096	**0.902828**
Bioresponse	**0.936898**	0.882554
cjs	0.970188	**0.994433**
cylinder-bands	**0.943682**	0.914568
GesturePhaseSegmentationProcessed	**0.748805**	0.652932
har	0.996902	**0.999590**
PhishingWebsites	**0.937853**	0.914617
MiceProtein	0.788341	**0.858647**
Amazon_employee_access	**0.550310**	0.334129
dresses-sales	0.779151	**0.788013**
LED-display-domain-7digit	0.946904	**0.967014**
texture	0.998820	**0.999320**
Australian	0.927873	**0.953403**
connect-4	**0.950426**	0.940032
higgs	**0.954954**	0.844238
SpeedDating	**0.736372**	0.724337