Methods for Improving Bayesian Optimization for AutoML

Matthias Feurer FEURERM@CS.UNI-FREIBURG.DE
Aaron Klein KLEINAA@CS.UNI-FREIBURG.DE
Katharina Eggensperger EGGENSPK@CS.UNI-FREIBURG.DE
Jost Tobias Springenberg SPRINGJ@CS.UNI-FREIBURG.DE
Manuel Blum MBLUM@CS.UNI-FREIBURG.DE
Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg

Abstract

The success of machine learning relies heavily on selecting the right algorithm for a problem at hand, and on setting its hyperparameters. Recent work automates this task with the help of efficient Bayesian optimization methods. We substantially improve upon these methods by taking into account past performance on similar datasets, and by constructing ensembles from the models evaluated during Bayesian optimization. We empirically evaluate the benefit of these improvements and also introduce a robust new AutoML system based on scikit-learn (using 16 classifiers, 14 feature processing methods, and 3 data preprocessing methods, giving rise to a structured hypothesis space with 132 hyperparameters). This system, which we dub auto-sklearn, won the auto-track in the first phase of the ongoing *ChaLearn AutoML challenge*.

Keywords: Automated machine learning, Bayesian optimization, ensemble construction, Meta-learning

1. Introduction

Machine learning has recently made great strides in many application areas, fueling a growing demand for machine learning systems that can be used effectively by novices in machine learning. In order to push the current state-of-the-art to the next level, the *ChaLearn Automatic Machine Learning Challenge* (Guyon et al., 2015) provides a common base to evaluate and compare different AutoML approaches. An early version of the AutoML system we describe here won the auto-track in the first phase of that challenge.

Our approach to the AutoML problem is motivated by Auto-WEKA (Thornton et al., 2013), which combines the machine learning framework WEKA (Hall et al., 2009) with a Bayesian optimization (Brochu et al., 2010) method for automatically selecting a good instantiation of WEKA in a data-driven way. We extend this approach by reasoning about the performance of machine learning methods on previous datasets (also known as meta-learning (Brazdil et al., 2009)) and by automatically constructing ensembles of the models considered by the Bayesian optimization method. Importantly, the principles behind our approach apply to a wide range of machine learning frameworks. Based on our new AutoML methods (described in Section 2) and the popular machine learning framework scikit-learn (Pedregosa et al., 2011), we construct a new AutoML system we dub auto-sklearn (Section 3). We performed an extensive empirical analysis using a broad range of 140 datasets to demonstrate that auto-sklearn outperforms the previous state-of-the-art AutoML tool Auto-WEKA (Section 4) and to demonstrate that each of our contributions leads to substantial performance improvements (Section 5).

2. New Methods for Increasing Efficiency and Robustness of AutoML

In this section, we discuss new methods for constructing an AutoML system from a given machine learning (ML) framework. While these methods are defined for any kind of ML framework, we expect their effectiveness to be greater for flexible ML frameworks that offer many degrees of freedom (e.g., many algorithms, hyperparameters, and preprocessing methods). The general goals of our methods are *efficiency* in finding effective instantiations of the ML framework and *robustness* of the resulting AutoML system.

2.1. Meta-Learning for Finding Good Instantiations of Machine Learning Frameworks

Domain experts derive knowledge from previous tasks: They *learn about the performance of learning algorithms*. The area of meta-learning (Brazdil et al., 2009) mimics this strategy by reasoning about the performance of learning algorithms. In this work, we apply meta-learning to select instantiations of our given machine learning framework that are likely to perform well on a new dataset.

More precisely, our meta-learning approach was developed by Feurer et al. (2015) and works as follows: In an offline phase, for each machine learning dataset in a dataset repository (in our case 140 datasets from OpenML (Vanschoren et al., 2013)), we evaluated a set of meta-features (described below) and used Bayesian optimization to determine and store an instantiation of the given ML framework with strong empirical performance for that dataset. (In detail, we ran the random forest-based Bayesian optimization method SMAC (Hutter et al., 2011) for 24 hours with a 10-fold cross-validation on a 2/3 training set, selecting the best instantiation based on a 1/3 validation set.) Then, given a new dataset $\mathcal{D}$, we compute its meta-features, rank our 140 datasets by their distance to $\mathcal{D}$ (with respect to L_1 distance in metafeature space) and select the best ML framework instantiations belonging to the k nearest datasets. We evaluate these and use the results to warmstart SMAC.

To characterize datasets, we implemented a total of 38 meta-features from the literature, including simple, information-theoretic and statistical metafeatures (Michie et al., 1994; Kalousis, 2002), such as statistics about the number of data points, features, and their ratio, the number of classes, data points or features with missing values, data skewness, and the entropy of the targets.

2.2. Automated Construction of Ensembles of Models Evaluated During Optimization

It is well known that ensembles often outperform individual models (Guyon et al., 2010; Lacoste et al., 2014), and that effective ensembles can be created from a library of models (Caruana et al., 2004, 2006). Ensembles perform particularly well if the models they are based on (1) are individually strong and (2) make uncorrelated errors (Breiman, 2001). Since this is much more likely when the individual models are very different in nature, this

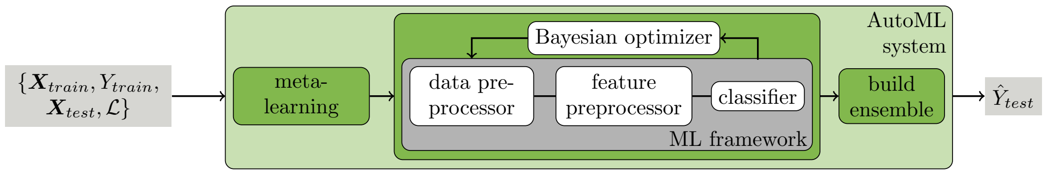

Figure 1: auto-sklearn workflow: our approach to AutoML. We add 2 components to Bayesian hyperparameter optimization of a ML framework: meta-learning for initializing Bayesian optimization and automated ensemble construction from configurations evaluated by Bayesian optimization.

approach is perfectly suited for combining strong instantiations of a flexible ML framework (as found by Bayesian optimization). After each evaluation of SMAC we saved the latest model's prediction on the validation dataset and constructed an ensemble with the previously seen models using the *ensemble selection* approach by Caruana et al. (2004). Figure 1 summarizes the overall workflow of an AutoML system including both of our improvements.

3. A Practical Automated Machine Learning System

Since our goal was to study the performance of our new AutoML system based on a state-of-the-art machine learning framework we implemented it based on scikit-learn (Pedregosa et al., 2011), one of the best known and most widely used machine learning libraries. scikit-learn offers a large range of well established and efficiently-implemented machine learning algorithms and is easy to use for both experts and non-experts. Due to its relationship to scikit-learn, we dub our resulting AutoML system auto-sklearn.

The 16 well-established classification algorithms in auto-sklearn are depicted in Table 1. They fall into different categories, such as general linear models (3 algorithms), support vector machines (2), discriminant analysis (2), nearest neighbors (1), naïve Bayes (3), decision trees (1) and ensemble methods (4). In contrast to Auto-WEKA (Thornton et al., 2013), we focused our configuration space on base classifiers and did not include meta-models (such as Boosting and Bagging with arbitrary base classifiers) or ensemble methods with several different arbitrary base classifiers (such as voting and stacking with up to 5 base classifiers). While these ensembles increased Auto-WEKA's number of hyperparameters by almost a factor of five (to 786), auto-sklearn "only" features 132 hyperparameters. We instead construct ensembles using our new method from Section 2.2. Compared to Auto-WEKA's solution, this is much more data-efficient: in Auto-WEKA, evaluating the performance of an ensemble with 5 components requires the construction and evaluation of 5 additional models; in contrast, in auto-sklearn, ensembles come for free, and it is possible to mix and match models evaluated at arbitrary times during the optimization. Moreover, the post-how ensemble construction allows us to directly optimize for arbitrary loss functions $\mathcal{L}$.

Preprocessing methods in auto-sklearn, depicted in Table 1, comprise *data* preprocessors (which change the feature values and which are always used when they apply) and *feature* preprocessors (which change the actual set of features, and only one of which [or none] is used). Data preprocessing includes rescaling of the inputs, imputation of missing values, and balancing of the target classes. The 11 possible feature preprocessing methods can be categorized into feature selection (2), kernel approximation (2), matrix decomposition (3), embeddings (1), feature clustering (1), and methods that use a classifier for feature selection (2). For example, L_1-regularized linear SVMs are used for feature selection by fitting the SVM to the data and choosing features corresponding to non-zero model coefficients.

name	#λ	cat (cond)	cont (cond)
AdaBoost (AB)	3	-	3 (-)
Bernoulli naïve Bayes	2	1 (-)	1 (-)
decision tree (DT)	3	1 (-)	2 (-)
extreml. rand. trees	5	2 (-)	3 (-)
Gaussian naïve Bayes	-	-	-
gradient boosting (GB)	6	-	6 (-)
kNN	3	2 (-)	1 (-)
LDA	2	-	2 (-)
linear SVM	5	3 (-)	2 (-)
kernel SVM	8	3 (-)	5 (2)
multinomial naïve Bayes	2	1 (-)	1 (-)
passive aggressive	3	1 (-)	2 (-)
QDA	2	-	2 (-)
random forest (RF)	5	2 (-)	3 (-)
ridge regression (RR)	2	-	2 (-)
SGD	9	3 (-)	6 (3)

name	#λ	cat (cond)	cont (cond)
extreml. rand. trees prepr.	5	2 (-)	3 (-)
fast ICA	4	3 (-)	1 (-)
feature agglomeration	3	2 (-)	1 (-)
kernel PCA	5	1 (-)	4 (3)
rand. kitchen sinks	2	-	2 (-)
linear SVM prepr.	5	3 (-)	2 (-)
no preprocessing	-	-	-
nystroem sampler	5	1 (-)	4 (3)
PCA	2	1 (-)	1 (-)
random trees embed.	4	-	4 (-)
select percentile	2	1 (-)	1 (-)
select rates	3	2 (-)	1 (-)
imputation	1	1 (-)	-
balancing	1	1 (-)	-
rescaling	1	1 (-)	-

Table 1: Number of hyperparameters for each possible classifier (left) and feature preprocessing method (right) for a **binary classification** dataset in **dense** representation. We distinguish between categorical (cat) hyperparameters with discrete values and continuous (cont) numerical hyperparameters. Numbers in brackets are conditional hyperparameters, which are only relevant when another hyperparameter has a certain value.

4. Comparing auto-sklearn to Auto-WEKA

As a baseline experiment, we compared the performance of vanilla auto-sklearn (auto-sklearn without meta-learning and ensemble construction) and Auto-WEKA, using a similar setup as in the paper introducing Auto-WEKA (Thornton et al., 2013): 21 datasets with their original train/test split[1], a walltime limit of 30 hours, 10-fold cross-validation (where the evaluation of each fold was allowed to take 150 minutes), and 10 independent optimization runs with SMAC on each dataset. Table 2 shows that vanilla auto-sklearn works statistically significantly better than Auto-WEKA in 8/21 cases, ties in 9 cases, and looses in 4.

	Abalone	Amazon	Car	Cifar10	Cifar-10 Small	Convex	Dexter	Dorothea	German Credit	Gisette	KDD09 Appetency	KR-vs-KP	Madelon	MNIST Basic	MRBI	Secom	Semeion	Shuttle	Waveform	Wine Quality	Yeast
AW	**73.50**	30.00	**0.00**	61.47	56.19	21.49	**5.56**	**5.22**	28.00	2.24	**1.74**	**0.31**	19.62	**2.84**	59.85	**7.87**	**4.82**	**0.01**	14.20	**33.22**	**37.08**
AS	80.20	**13.99**	0.19	**51.93**	**52.28**	**14.95**	7.78	5.51	**26.00**	**1.29**	**1.74**	0.42	**12.82**	2.87	**47.84**	**7.87**	5.03	**0.01**	**14.07**	35.16	38.65

Table 2: Test set performance of Auto-WEKA (AW) and vanilla auto-sklearn (AS) as in the original evaluation of Auto-WEKA (Thornton et al., 2013). The results are median percent error rates across 100 000 bootstrap samples (out of 10 runs) simulating 4 parallel runs. Bold numbers indicate the best result. Underlined results are not statistically significantly different according to a bootstrap test with $p = 0.05$.

1. Obtained from the Auto-WEKA website www.cs.ubc.ca/labs/beta/Projects/autoweka/

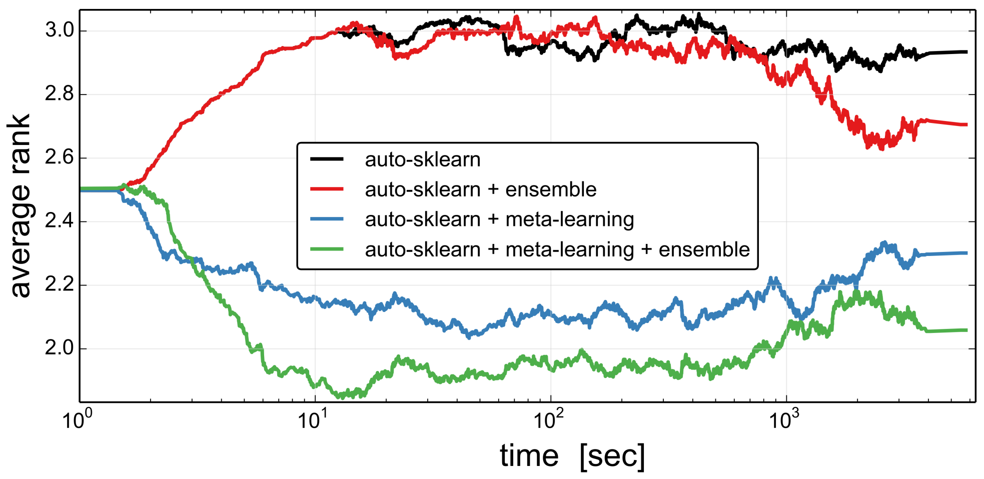

Figure 2: Average rank (test performance) of all four auto-sklearn versions across 140 datasets (lower is better).

5. Evaluation of our New AutoML Methods

In order to evaluate the robustness and general applicability of our new AutoML methods on a broad range of problems, we gathered 140 binary and multiclass classification datasets from the OpenML repository (Vanschoren et al., 2013), only selecting datasets with at least 1000 data points to allow robust performance evaluations. These datasets cover a diverse range of applications, such as text classification, digit and letter recognition, gene sequence and RNA classification, advertisement, particle classification for telescope data, and cancer detection in tissue samples. Since many of these datasets were quite imbalanced, we minimized balanced classification error (Guyon et al., 2015).

To study the performance of our new AutoML methods, we performed 10 runs of auto-sklearn with and without meta-learning, and with and without ensemble selection on each of these 140 datasets. To study their performance under rigid time constraints, and also due to computational resource constraints, we limited the CPU time for each auto-sklearn run to 1 hour; we also limited the runtime for each single model to a tenth of this (6 minutes).

To avoid testing on datasets already used to perform meta-learning, we performed a leave-one-dataset-out validation: when testing on a new dataset, we only used meta-information on the 139 other datasets. In order to avoid duplicating runs (and thus doubling resource requirements), we did not perform new SMAC configuration runs for auto-sklearn with ensembles, but rather built ensembles in an offline step after SMAC had finished.[2] We add the time necessary to build the ensemble to the time at which we obtained the predictions necessary for it, enabling us to plot performance over time.

Figure 2 shows the average ranks (test performance) of the four different versions of auto-sklearn. We observe that both of our additions yielded substantial improvements over vanilla auto-sklearn. The most striking result is that meta-learning yielded drastic improvements starting with the first configuration it selected and lasting until the end of the exper-

2. Our software is also able to build the ensembles in a second, parallel process.

iment. Moreover, both of our methods complement each other: our automated ensemble construction improved both vanilla auto-sklearn and auto-sklearn with meta-learning. Interestingly, the ensemble's influence on the performance started earlier for the meta-learning version. We believe that this is because meta-learning produces better machine learning models earlier, which can be more usefully combined into an ensemble; but when run longer, vanilla auto-sklearn also benefits substantially from automated ensemble construction.

6. Winning Entry to the ChaLearn AutoML Challenge

Since our AutoML system won the first place in the auto-track of the first phase of the *ChaLearn AutoML challenge*, we made several improvements. In this section we describe the differences of that system compared to what we describe in this paper.

First of all, we used a different setting to generate meta-data. We generated meta-data only for binary datasets, but including datasets from the LibSVM repository. These were a total of 96 datasets. We removed all categorical attributes from these datasets to be closer to the datasets in phase one of the challenge.

Second, the configuration space of auto-sklearn was smaller. We used only extremely randomized trees, gradient boosting, kNN, linear SVM, kernel SVM, random forests and SGD as classifiers. For preprocessing we only used random kitchen sinks, select percentile and PCA.

The third and final difference was the ensemble construction method. To obtain the weights for a linear combination of the predictions, we used the state-of-the-art blackbox optimization algorithm CMA-ES (Hansen, 2006). CMA-ES produces a non-sparse weight vector, assigning non-zero weights to all models. We removed all models which performed worse than random guessing in order to reduce the dimensionality of the optimization problem. However, in several experiments we noted overfitting on the training data as well as a long ensemble construction time.

7. Discussion and Conclusion

We have demonstrated that our new auto-sklearn system performs favorably against the current state of the art in AutoML, and that meta-learning and ensemble construction enhance its efficiency and robustness further. This finding is backed by the fact that auto-sklearn won the first place in the auto-track of the ongoing *ChaLearn AutoML challenge*. Although not described here, we also evaluated the use of auto-sklearn for interactive machine learning with an expert in the loop using weeks of CPU power. This effort led to third place in the human track of the same challenge. As such, we believe that auto-sklearn is a very useful system for both machine learning novices and experts. We released the source code at https://github.com/automl/auto-sklearn.

For future work, we would like to add support for regression and semi-supervised problems. Most importantly, though, the focus on scikit-learn implied a focus on small to medium-sized datasets, and an obvious direction for future work will be to apply our new AutoML methods to modern deep learning systems that yield state-of-the-art performance on large datasets; we expect that in that domain especially automated ensemble construction will lead to tangible performance improvements over Bayesian optimization.

Acknowledgments

This work was supported by the German Research Foundation (DFG), under Priority Programme Autonomous Learning (SPP 1527, grant HU 1900/3-1) and under the BrainLinks-BrainTools Cluster of Excellence (grant number EXC 1086).

References

P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining.* Springer, 2009.

L. Breiman. Random forests. *MLJ*, 45:5–32, 2001.

E. Brochu, V. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010.

R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proc. of ICML'04*, page 18, 2004.

R. Caruana, A. Munson, and A. Niculescu-Mizil. Getting the most out of ensemble selection. In *Proc. of ICDM'06*, pages 828–833, 2006.

M. Feurer, J. Springenberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proc. of AAAI'15*, pages 1128–1135, 2015.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Model selection: Beyond the Bayesian/Frequentist divide. *JMLR*, 11:61–87, 2010.

I. Guyon, K. Bennett, G. Cawley, H. Escalante, S. Escalera, T. Ho, N.Macià, B. Ray, M. Saeed, A. Statnikov, and E. Viegas. Design of the 2015 ChaLearn AutoML Challenge. In *Proc. of IJCNN'15*, 2015. To appear.

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

N. Hansen. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION'11*, pages 507–523, 2011.

A. Kalousis. *Algorithm Selection via Meta-Learning.* PhD thesis, University of Geneve, 2002.

A. Lacoste, M. Marchand, F. Laviolette, and H. Larochelle. Agnostic Bayesian learning of ensembles. In *Proc. of ICML'14*, pages 611–619, 2014.

D. Michie, D. Spiegelhalter, C. Taylor, and J. Campbell. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

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

C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In *Proc. of KDD'13*, pages 847–855, 2013.

J. Vanschoren, J. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.