Efficient and Robust Automated Machine Learning

Matthias Feurer Aaron Klein Katharina Eggensperger Jost Tobias Springenberg Manuel Blum Frank Hutter
{ feurerm, kleinaa, eggenspk, springj, mblum, fh }@cs.uni-freiburg.de
Department of Computer Science, University of Freiburg, Germany

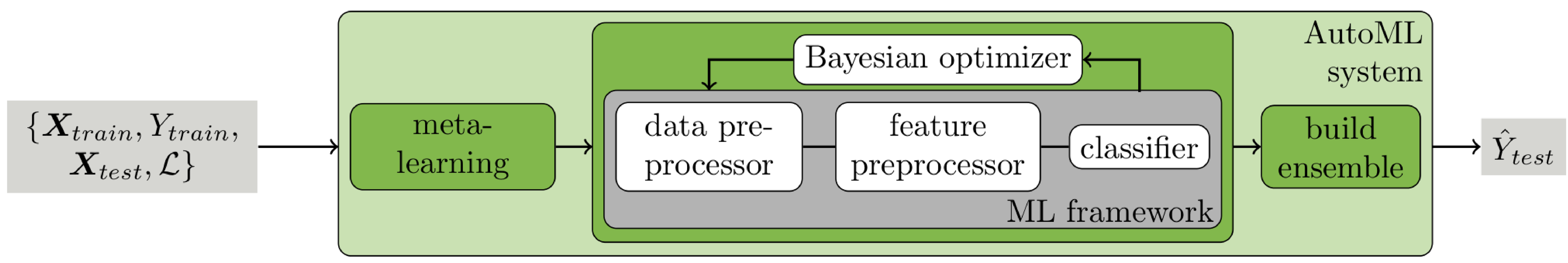
UNI FREIBURG

Basics

In 30 seconds

- **AutoML**: automatically choosing an algorithm and setting its hyperparameters for a new dataset without human intervention
- We combine scikit-learn and Bayesian optimization: **auto-sklearn**
- Two new components: **meta-learning** to speed up convergence and **post-hoc ensemble construction** to improve robustness
- We improve upon previous results obtained by Auto-WEKA
- auto-sklearn **won several prizes** in the ongoing *ChaLearn AutoML challenge*

The AutoML workflow



Software

Easy-to-use drop-in replacement for scikit-learn:

```
import autosklearn.classification as cls
automl = cls.AutoSklearnClassifier()
automl.fit(X_train, y_train)
y_hat = automl.predict(X_test)
```

Available on `github.com`, see link or QR code at the bottom of the poster.

auto-sklearn

Machine learning pipeline

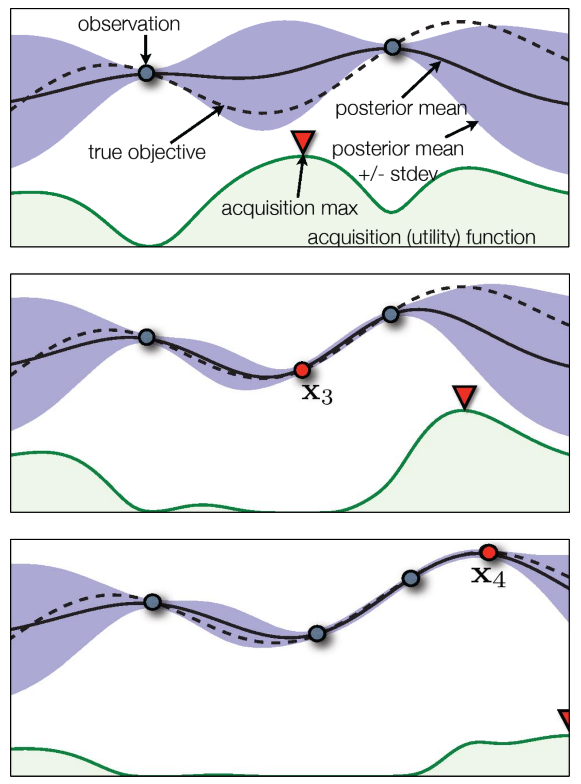
- A configurable machine learning pipeline built around **scikit-learn**
- We use 15 classifiers, 14 feature preprocessing methods and 4 data preprocessing methods; yielding a **C**ombined **A**lgorithm **S**election and **H**yperparameter Optimization (CASH) problem with **110 hyperparameters**
- We use the **Bayesian optimization** toolkit **SMAC** for optimization

Classifier	#λ	cat	(cond)	cont	(cond)
AdaBoost	4	1	-	3	(-)
Bernoulli Naïve Bayes	2	1	(-)	1	(-)
Decision Tree	4	1	(-)	3	(-)
Extremely rand. Trees	5	2	(-)	3	(-)
Gaussian Naïve Bayes	-	-	(-)	-	(-)
Gradient Boosting	6	-	(-)	6	(-)
kNN	3	2	(-)	1	(-)
LDA	4	1	(-)	3	(1)
Linear SVM	4	2	(-)	2	(-)
Kernel SVM	7	2	(-)	5	2
Multinomial Naïve Bayes	2	1	(-)	1	(-)
Passive Aggressive	3	1	(-)	2	(-)
QDA	2	-	(-)	2	(-)
Random Forest	5	2	(-)	3	(-)
SGD	10	4	(-)	6	(-)

Feature Preprocessor	#λ	cat	(cond)	cont	(cond)
Extremely rand. Trees	5	2	(-)	3	(-)
Fast ICA	4	3	(-)	1	(-)
Feature Agglomeration	4	3	(-)	1	(-)
Kernel PCA	5	1	(-)	4	(3)
Random Kitchen Sinks	2	-	(-)	2	(-)
Linear SVM	3	1	(-)	2	(-)
No Preprocessing	-	-	(-)	0	(-)
Nystroem Sampler	5	1	(-)	4	-3
PCA	2	1	(-)	1	(-)
Random Trees Embedding	4	-	(-)	4	(-)
Select Percentile	2	1	(-)	1	(-)
Select Rates	3	2	(-)	1	(-)
Data preprocessor					
Imputation	1	-	(-)	1	(-)
Balancing	1	-	(-)	1	(-)
Rescaling	1	-	(-)	1	(-)
One Hot Encoding	2	1	(-)	1	(-)

Bayesian optimization

Sequential model-based Bayesian optimization, a popular approach to optimize expensive blackbox functions



Meta-learning

- Standard Bayesian optimization has to explore a very large configuration space from scratch
- We use **meta-learning to initialize Bayesian optimization**
- For a new dataset, we start Bayesian optimization with configurations that worked best on the most similar datasets
- Similarity based on the L_1-distance of their meta-features
- We used a total of **37 meta-features**

Example Metafeatures for the Iris and MNIST dataset

	Iris	MNIST
# training examples	150	60000
# classes	3	10
# features	4	784
# numerical features	4	784
# categorical features	0	0

Ensemble learning

- Standard Bayesian optimization trains many models, but only returns the single best model
- Ensembles almost always outperform single models
- **Build ensemble to make use of all models trained**
- Use ensemble selection by Caruana et al. (2004) to build an ensemble based on the models' prediction for the validation set

Procedure 1: EnsembleSelection(M, S)

Input : Models M, Ensemble size S , $n = |M|$
Output : Ensemble E

1 $E \leftarrow \emptyset$
2 **for** $i = 0 \ldots S$ **do**
3 $\quad b \leftarrow \mathrm{argmax}_{j=0 \ldots n}\, performance(E \cup M[j])$
4 $\quad E \leftarrow E \cup M[b]$
5 **return** E

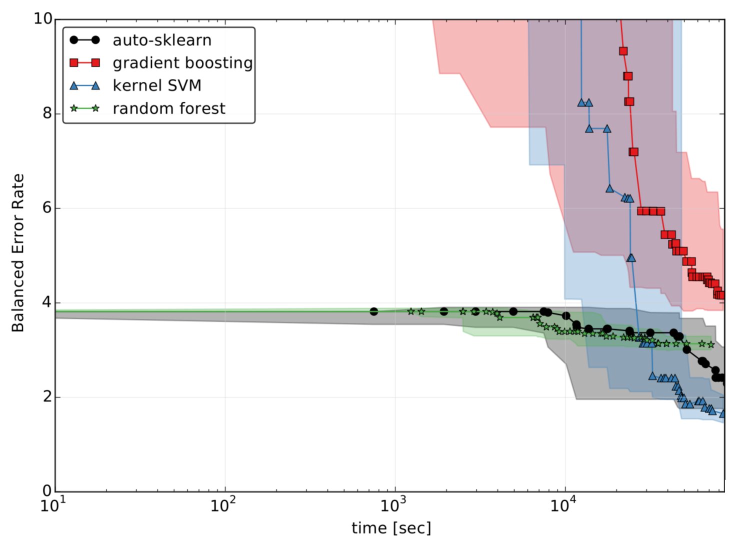
Experiments

auto-sklearn vs. Auto-WEKA & Hyperopt-Sklearn

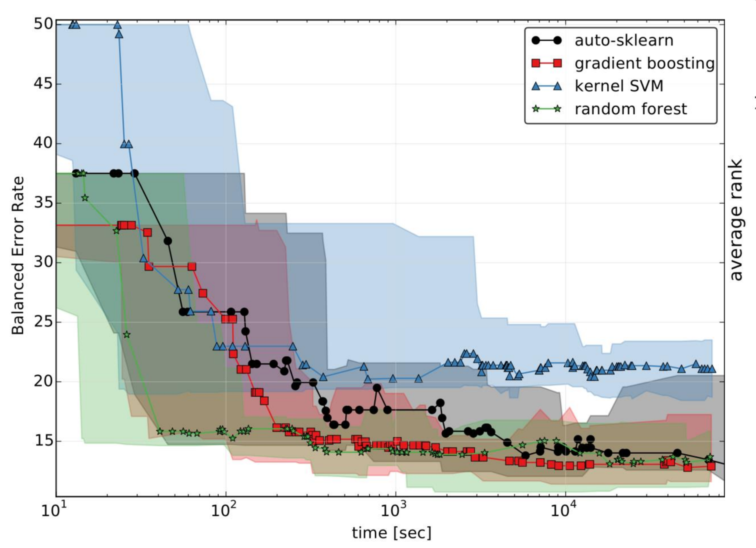
- Baseline comparison using the **original Auto-WEKA setup**:
  - Test error of the best configuration found with 10-fold cross-validation
  - Used 30 hours and 3GiB RAM to search for the best configuration
  - Used vanilla version of auto-sklearn (without meta-learning an ensembles)
- auto-sklearn performed significantly better than Auto-WEKA in 6/21 cases, tied in 12/21 and lost in 3/21
- auto-sklearn performed significantly better than Hyperopt-Sklearn in 7/21 cases and tied in 9 cases. Hyperopt-Sklearn was not able to construct models in 5 cases due to missing values, sparse feature representation or too much memory consumption.

	auto-sklearn	Auto-WEKA	Hyperopt-sklearn
Abalone	**73.50**	**73.50**	76.21
Amazon	**16.00**	30.00	16.22
Car	0.39	**0.00**	0.39
Cifar10	**51.70**	56.95	-
Cifar10 Small	**54.81**	56.20	57.95
Convex	**17.53**	21.80	19.18
Dexter	**5.56**	8.33	-
Dorothea	**5.51**	6.38	-
German Credit	**27.00**	28.33	27.67
Gisette	**1.62**	2.29	2.29
KDD09 Appetency	**1.74**	**1.74**	-
KR-vs-KP	0.42	**0.31**	0.42
Madelon	**12.44**	18.21	14.74
MNIST Basic	2.84	2.84	**2.82**
MRBI	**46.92**	60.34	55.79
Secom	**7.87**	8.09	-
Semeion	**46.92**	60.34	55.79
Shuttle	**0.01**	**0.01**	0.05
Waveform	14.93	14.13	**14.07**
Wine Quality	33.76	**33.36**	34.72
Yeast	40.67	**37.75**	38.45

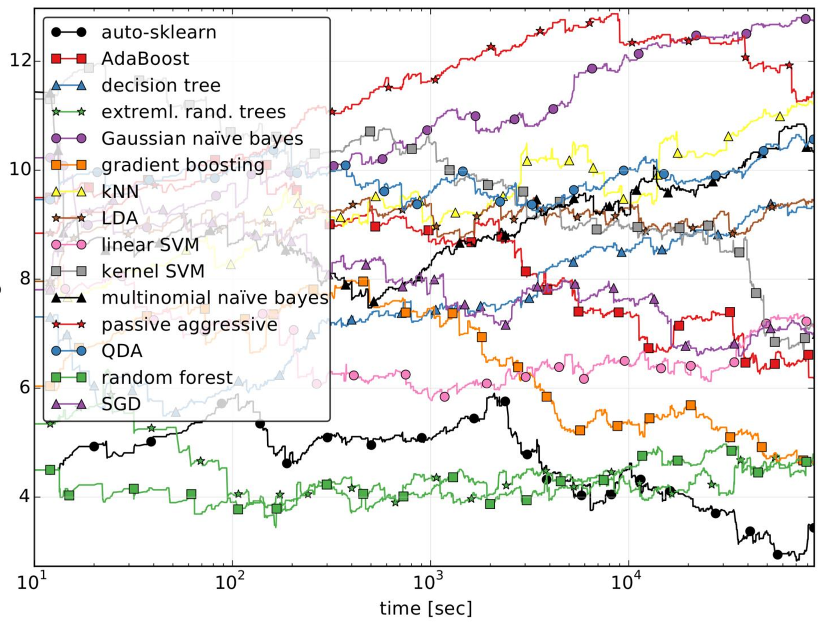
Analysis of classification algorithms



Left: Performance of a subset of classifiers on two example datasets compared to auto-sklearn over time. Top: MNIST, Bottom: Promise PC4.

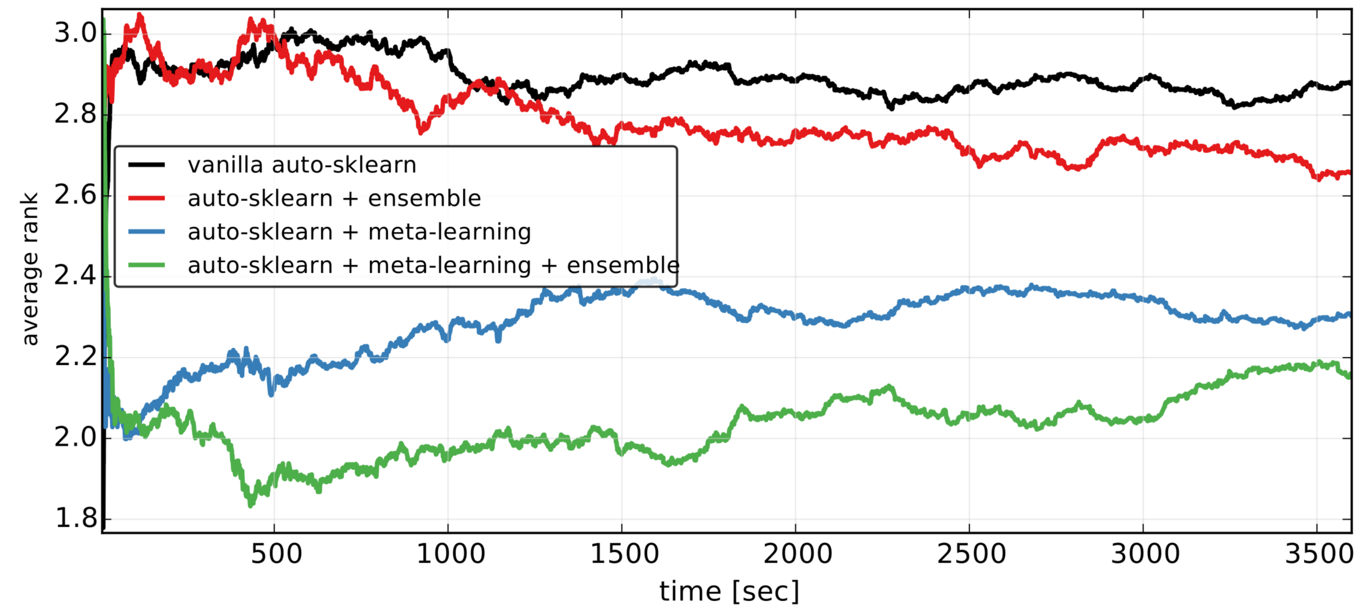


Bottom: Rank of test error over time for auto-sklearn vs. all individual classifiers. Each line shows the average across 13 datasets.



Evaluation of our extensions to AutoML

- We ran auto-sklearn for 1 hour to simulate the AutoML challenge setting:
  - 4 different versions of auto-sklearn
  - **140 datasets** from `OpenML.org`, each with at least 1000 samples
  - **Leave-one-dataset-out**: ran auto-sklearn on one dataset and assumed knowledge of all other 139.
- **Both meta-learning and ensemble building improve auto-sklearn**; auto-sklearn is further improved when both methods are combined.



ML4AAD

Code available: https://github.com/automl/auto-sklearn