Contest Winner: Winning the AutoML Challenge with Auto-sklearn

Tags: Automated Data Science, Competition, scikit-learn, Weka

This post is the first place prize recipient in the recent KDnuggets blog contest. Auto-sklearn is an open-source Python tool that automatically determines effective machine learning pipelines for classification and regression datasets. It is built around the successful scikit-learn library and won the recent AutoML challenge.

By Matthias Feurer, Aaron Klein and Frank Hutter, University of Freiburg.

Editor's note: This blog post was an entrant in the recent KDnuggets Automated Data Science and Machine Learning blog contest, where it was selected as the first prize recipient.

Despite the great success of machine learning (ML) in many fields (e.g., computer vision, speech recognition or machine translation), it is very hard for novice users to effectively apply ML: they need to decide between dozens of available ML algorithms and preprocessing methods, and tune the hyperparameters of the selected approaches for the dataset at hand. Doing this right often makes the difference between poor and state-of-the-art performance, but the need for these tedious manual tasks constitutes a substantial burden for real-world applications of machine learning.

With ever-more industrial uses of machine learning, there is now a strong demand for robust machine learning systems that autonomously perform inference on a given dataset. Recent automated machine learning (AutoML) systems find the right algorithm and hyperparameters in a data-driven way without any human intervention. In this blog post we describe how this is done in our AutoML system Auto-sklearn that won the recent AutoML challenge.

Algorithm and hyperparameter selection

A prominent method for optimizing machine learning hyperparameters is Bayesian optimization, which iterates the following steps:

1. Build a probabilistic model to capture the relationship between hyperparameter settings and their performance
2. Use the model to select useful hyperparameter settings to try next by trading off exploration (searching in parts of the space where the model is uncertain) and exploitation (focussing on parts of the space predicted to perform well)
3. Run the machine learning algorithm with those hyperparameter settings.

This process can be generalized to jointly select algorithms, preprocessing methods, and their hyperparameters as follows: the choices of classifier / regressor and preprocessing methods are top-level, categorical hyperparameters, and based on their settings the hyperparameters of the selected methods become active. The combined space can then be searched with Bayesian optimization methods that handle such high-dimensional,
conditional spaces; we use the random-forest-based SMAC, which has been shown to work best for such cases.

This AutoML approach of using Bayesian optimization to automatically customize very general ML frameworks for given datasets was first introduced in Auto-WEKA. (As the name suggests, Auto-WEKA is based on WEKA; now in version 2.0, it is integrated with WEKA’s package manager, has a GUI, an API, and a command line interface, and is very popular with the WEKA user base, with an average of 250 downloads per week). Auto-sklearn serves a very similar purpose in Python.

Auto-sklearn

Auto-sklearn is our implementation of the above idea. It is open source, implemented in python and built around the scikit-learn library. It contains a machine learning pipeline which takes care of missing values, categorical features, sparse and dense data, and rescaling the data. Next, the pipeline applies a preprocessing algorithm and an ML algorithm.

Auto-sklearn includes 15 ML algorithms, 14 preprocessing methods, and all their respective hyperparameters, yielding a total of 110 hyperparameters; the resulting space is illustrated in Figure 1, where only the hyperparameters in grey boxes are active.

![Figure 1: The hyperparameter space of Auto-sklearn.](image)

Optimizing performance in Auto-sklearn’s space of 110 hyperparameters can of course be slow, and to jumpstart this process we use meta-learning to start from good hyperparameter settings for previous similar datasets. Specifically, Auto-sklearn comes with a database of previous optimization runs on 140 diverse datasets from OpenML. For a new dataset, it first identifies the most similar datasets and starts from the saved best settings for those.

A second improvement was to automatically construct ensembles: instead of returning a single hyperparameter setting (as standard Bayesian optimization would), we automatically construct ensembles from the models trained during the Bayesian optimization. Specifically, we used Ensemble Selection to create small, powerful
ensembles with increased predictive power and robustness.

In our NIPS 2015 paper we benchmarked Auto-sklearn with and without these new components on a total of 140 datasets (using leave-one-dataset-out validation) and found both of them to help substantially: Figure 2 shows that meta-learning improves performance directly from the start and ensembling helps more for longer optimization runs.

![Figure 2: The average rank of each method over 140 datasets (lower is better).](image)

**Auto-Net**

Deep neural networks gained a lot of attraction in recent years due to their great performance for many large datasets. However, due to their sensitivity to many hyperparameters they are even harder to use than other algorithms, and there is an obvious need for automatically-tuned neural networks. Neither Auto-WEKA nor Auto-sklearn support deep neural networks, and our recent Auto-Net extension to Auto-sklearn closes this gap. So far it is restricted to simple feed forward neural networks, but in the future we aim to extend it to convolutional and recurrent neural networks.

Auto-Net contains layer-independent network hyperparameters (such as the learning rate or the number of layers) as well as hyperparameters which are set for each layer individually (such as dropout rate or weight initialization). We use a maximum of 6 layers, with a total of 63 hyperparameters. To make use of Auto-sklearn's preprocessing methods and ensembling, Auto-Net is simply implemented as a 16th ML algorithm for Auto-sklearn.

**Winning the AutoML challenge**

The ChaLearn AutoML challenge was a machine learning competition run during the last 1.5 years. It consisted of five phases with two tracks each. In the auto track competing systems were run autonomously for 100 minutes
to process 5 previously unseen datasets per phase. The Kaggle-like tweakathon track featured a public Leaderboard, up to 150 participating teams, and 3 months time. We ran the same software for both tracks, just with very different computational resources (100 minutes on one machine vs. 2 days on a cluster of 25 machines). Auto-sklearn placed in the top three for nine out of ten phases, and won six of them. Particularly in the last two phases, Auto-sklearn won both the auto track and the tweakathon. During the last two phases of the tweakathon we also combined Auto-sklearn with Auto-Net for several datasets to further boost performance.

Applying Auto-sklearn to your own problem

Auto-sklearn is a drop-in replacement for a scikit-learn estimator. You can use the following four lines of python code to obtain a machine learning pipeline:

```python
import autosklearn.classification

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

Matthias Feurer is a PhD student at the University of Freiburg, Germany. His research focuses on leveraging meta-information in the process of automated machine learning. He is lead developer of Auto-sklearn.

Aaron Klein is a PhD student at the University of Freiburg, Germany. His research aims to make hyperparameter optimization for expensive machine learning models like deep neural networks feasible.

Frank Hutter is an Emmy Noether research group lead (eq. Assistant professor) at the University of Freiburg, Germany. His group on Machine Learning for Automated Algorithm Design works on machine learning and optimization, with a recent focus on Bayesian optimization, automated machine learning, and deep learning.

Related:

- Automating Data Science
- Automated Data Science and Machine Learning in Digital Advertising
- Data Science Automation: Debunking Misconceptions