Evolutionary optimization of machine learning pipelines

Gabriela Suchopárová

Abstract

The subject of this work is the automated machine learning (AutoML), which is a field that aims to automatize the process of model selection for a given machine learning problem. We have developed a system that, for a given supervised learning task, finds a suitable pipeline — combination of machine learning, ensembles and preprocessing methods.

Workflows

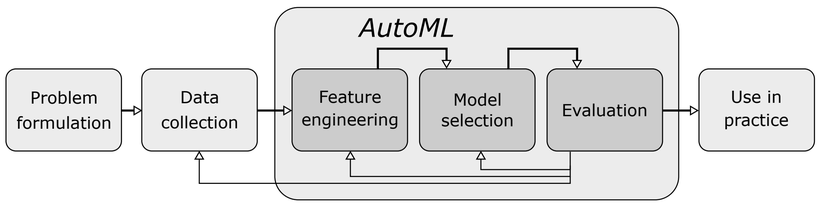

Figure 1: Schema of a general workflow

A machine learning workflow is the process of solving a given machine learning problem. As we can see in Figure 1, a general workflow can be divided into several steps. The first two steps are usually handled by an expert from an unrelated field, who provides data to machine learning experts. Then, the data is preprocessed and a suitable machine learning method needs to be chosen. This process is usually iterative, as there are many different methods with several hyperparameters. To obtain good results, it is necessary to carefully choose them, and so the procedure may be very time consuming.

AutoML

The AutoML systems aim to automatize the process of feature preprocessing and model selection. The idea is that with these systems, machine learning may be accessible even to non-experts, and on the other hand experts may use it for model recommendation.

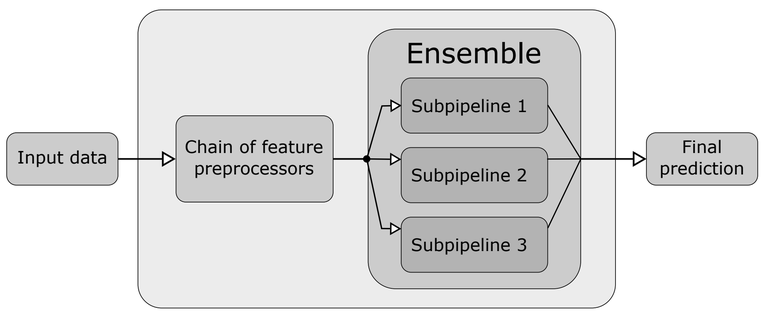

Figure 2: A more complex machine learning pipeline

One of the tasks targeted by AutoML systems is *pipeline optimization*. A machine learning pipeline is composed from a chain of feature preprocessing methods and one estimator. The estimator can be a simple method or an ensemble that contains one or more base-estimators. Because a base-estimator can be a pipeline as well, a general pipeline is a directed acyclic graph.

Our system

Existing methods for pipeline optimization operated only on a limited structure. Some of them focused only on hyperparameter optimization of simple models and they did not include any ensembles or feature preprocessing methods. Others allowed an arbitrary-sized preprocessor chain, however, no ensembles with pipelines as base-estimators were supported.
Our systems aims to search for a pipeline that performs well on the given task. The pipelines can be composed from both ensembles and feature preprocessing methods, but simple methods are tried out as well.

Developmental GP

For pipeline optimization, we used the genetic programming (GP), which is a subfield of evolutionary algorithms. An individual in GP is in fact a computer program. Usually, it is represented as an expression tree. The fitness of the tree is determined by running the program, or also evaluating all functions and their arguments.

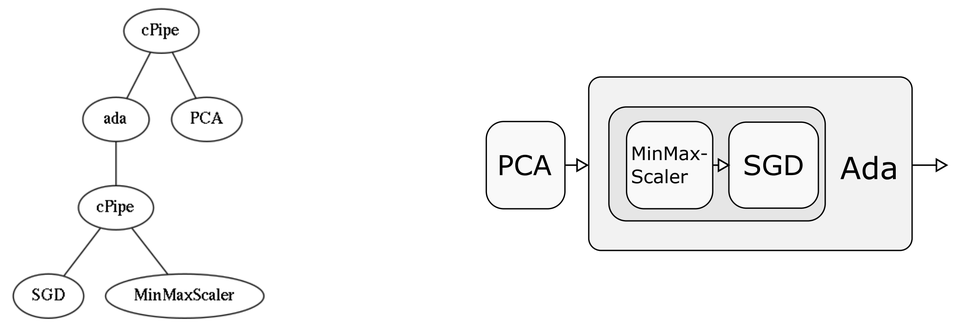

Figure 3: Example of an encoded pipeline

We created a specific encoding that enables to convert pipelines in the form of a DAG into a tree representation. Instead of directly encoding pipeline steps as nodes, we apply the developmental GP, where the nodes represent *operations* that create the pipeline.
An example of the encoding is shown in Figure 3. The tree individual contains instructions which construct the actual pipeline:

cPipe — create a pipeline with a preprocessor chain and an estimator
- **ada** — insert an AdaBoost ensemble with a base-estimator
  - **cPipe** — create a pipeline with a preprocessor chain
    - **SGD** — insert a SGD classifier
    - **MinMaxScaler** — insert a MinMaxScaler
- **PCA** — insert the PCA preprocessor

OpenML-CC18 benchmark

Here will be a nice description of the experiment. May be reduced to only one of the graphs for illustrative purpose only.