Hyperparameter Optimization and Using Learning Curves

AutoML – Assignment 2

Leiden University

The objective of this assignment is to provide a comprehensive understanding of surrogate models used for **H**yper**p**arameter **O**ptimization (HPO) and implement various learning curve-based methods. You will be provided with a dataset of empirical learning curves collected with an updated version of **L**earning **C**urve **D**ata**b**ase (LCDB) [3]. Using this collected data, you will be tasked with training a surrogate model that predicts the performance of a given configuration. You will be tasked by creating two hyperparameter optimisation techniques that use learning curves.

1 Surrogate model

Hyperparameter optimisation can be an expensive operation because of the individual models that need to be trained. Surrogate models are often employed to make the development and implementation of hyperparameter optimisation methods less expensive. These are models that are trained on prior performance data of configurations.

To simulate the HPO in a resource-constraint environment, you will utilise a surrogate model trained on LCDB learning curve data to predict the performance of the sampled configurations rather than training a model to get the true performance.

Please note that for your set of experiments, pairs of machine learning algorithms and datasets require independent surrogate models. For each dataset on which you optimise a model, you should train a uniquely/independently trained surrogate model. For each configuration, we have recorded the performance across various anchors (training set sizes). Methods trained on learning curves typically also require an anchor size to be added to the surrogate model. This way, the surrogate model can make predictions for every possible configuration at every possible anchor.

The best-known package for defining a configuration space is the `ConfigSpace` package (available on PyPI), which allows the definition of a configuration space. We have also provided a configuration space (which you don't need for building the surrogate model). Familiarise yourself with the content of this package, as it will be important during the course.[1]

We have provided you with three important files:

1. `surrogate_model.py`: the main class handling the surrogate model. Two functions need to be implemented to make it work. To get a feeling of how this would work, you can already have a look at `example_run_experiment.py` (which interacts with this file).

2. `vertical_model_evaluator.py`: in this file, we have provided an abstract base class that, in principle, can be used to bootstrap every vertical model selection method.

3. `lccv.py`: This is where we will implement the logic of LCCV.

[1] Documentation: `https://automl.github.io/ConfigSpace/latest/`

4. `example_run_experiment.py`: an example of how to interact with the API of the surrogate model and LCCV. Once the surrogate model and LCCV have been implemented, this should work out of the box as well.

Complete the functions in `surrogate_model.py`, and ensure that you understand the workings of the framework. If you are satisfied with your code of assignment 1, you can reuse many of this implementation.

2 Hyperparameter Optimization using learning curves

We will implement simplified versions of two learning curve extrapolation methods.

2.1 LCCV implementation

One way of utilising learning curves to speed up HPO is by using LCCV [2]. While the LCCV framework proposes several contributions, in this work, we will implement the one that is closest related to learning curves, i.e., the optimistic extrapolation. We will evaluate configurations on a fixed set of increasing anchor sizes (e.g., $[128, 256, \ldots, s_T]$, where s_T is the size of the dataset), and after every anchor, we will do an optimistic extrapolation. Since we are working with a simplified surrogate model, we can not measure the infimum and supremum of the distribution. Just calculate the slope between the points and use the following formula for extrapolation:

$$C_t + (s_T - s_t)\left(\frac{C_{t-1} - C_t}{s_{t-1} - s_t}\right), \tag{1}$$

See [2] for the description of the individual elements of the formula. Fill in the functions, and ensure that the method is extensively tested.

2.2 IPL-extrapolation implementation

Another way of utilising learning curves to speed up HPO is by using a parametric model [1]. While Domhan et al. also use many complicated contributions (such as the utilisation of a mixture model), we will just use a simple inverse power law. Determine a *fixed* learning curve schedule for every configuration (e.g., $[16, 32, 64, 128, 256]$) and fit an IPL to the learning curve data. In case the IPL prediction does not improve over the best-seen performance so far, the configuration can be discarded. If it performs better, it can be evaluated on the full dataset. Determine a Python class that inherits from the `VerticalModelEvaluator` (from `vertical_model_evaluator.py`).

3 General remarks and advises

1. Read the (important parts) of the aforementioned papers, so you have a grasp of the ideas presented by the methods.

2. Note that we are not reimplementing these methods, just simple ideas that are being used in the methods

3. There are sometimes various ways of doing it. Use your scientific creativity in a serious way.

4. Make sure that the report is written in a serious way. That means an introduction (with a problem statement), a method section, and a section introducing the experiments, conclusions and references. Be precise and concise (e.g., you can put much information in a single plot). Refer back to literature where needed.

5. Some methods can be unit tested. Create unit tests and (optionally) hand them in as well.

4 Report and Submission

Report: Write a 4-page (maximum) report (excluding references) using the provided Latex template. Refrain from making changes to the template formatting. Demonstrate the working of both the surrogate model (e.g., using a holdout set and reporting the Spearman correlation), the SMBO algorithm optimising *KNN* algorithm on several datasets using the surrogate model (for which we have provided learning curve data). Consider appropriate methods to plot the results and compare LCCV with the IPL-extrapolation. Determine what a proper way of measuring their effectiveness against each other is (your scientific creativity is appreciated, but keep it serious!)

Submission: Please provide the pdf (generated by your latex report) and the adjusted Python scripts. Be sure to hand in each Python script that produced a plot so we can reproduce the plots of your report. Do not use packages that are hard to install (e.g., anything beyond Scikit-learn, scipy, numpy, pandas and ConfigSpace is typically not needed.) It is important that we can reproduce all plots that you created! **DO NOT** submit the pdf file in zip format since they need to be checked by our plagiarism tools, including *TurnItIn*.

References

[1] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 3460–3468. AAAI Press, 2015.

[2] Felix Mohr and Jan Nicolaas van Rijn. Fast and informative model selection using learning curve cross-validation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 45(8):9669–9680, 2023.

[3] Felix Mohr, Tom J Viering, Marco Loog, and Jan N van Rijn. Lcdb 1.0: An extensive learning curves database for classification tasks. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, volume 13717 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2022.