Meta-Learning for Pipeline Optimization

Pavel Brazdil, Jan N. van Rijn, Felix Mohr, Henry Gouk

Universidad de La Sabana - Colombia

Motivation

Like in the algorithm selection setting, we are still in model selection.

Main difference:

- Algorithm Selection: candidate *set*,
- HPO/Pipeline Optimization: candidate *sequence*

Motivation

Like in the algorithm selection setting, we are still in model selection.

Main difference:

- Algorithm Selection: candidate *set*,
- HPO/Pipeline Optimization: candidate *sequence*

Every candidate is evaluated, and we check whether it is the new best one.

Meta-learning can exploit knowledge for two important tasks in this process:

1. *influence* the sequence in which candidates are generated.
2. *accelerate* the evaluation of concrete candidates.

Overview

Candidate Generating Procedures
- Hyperparameter Optimization
- Pipeline Optimization

Meta-Learning for HPO and Automated Pipeline Design
- Warm-Starting
- Early Discarding
- Budget Picking

Overview

Candidate Generating Procedures
- Hyperparameter Optimization
- Pipeline Optimization

Meta-Learning for HPO and Automated Pipeline Design

Hyperparameter Optimization

In hyperparameter optimization (HPO), one assumes that we optimize *one* machine learning algorithm, e.g., a support vector machine.

The behavior of this algorithm is controlled by k hyperparameters, with domains $\Theta_1, .., \Theta_k$.

Let $\Theta := \Theta_1 \times .. \times \Theta_k$ be the space of all combinations of hyperparameter values. Let $m : \Theta \to \mathbb{R}$ be a performance metric. Then the goal is to find

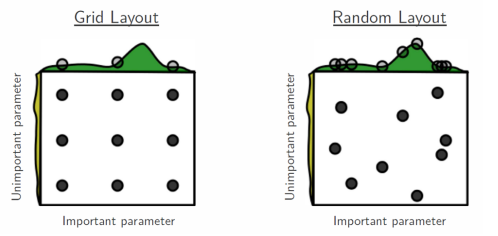
$$\arg\max_{\theta \in \Theta} m(\theta)$$

Grid Search and Random Search

A grid defines a finite set of candidates: Reduction to algorithm selection.

Common objection: few samples for important hyperparameter.

This can be addressed with a random layout [Bergstra and Bengio, 2012]:



Both approaches ignore the observations.

Bayesian Optimization

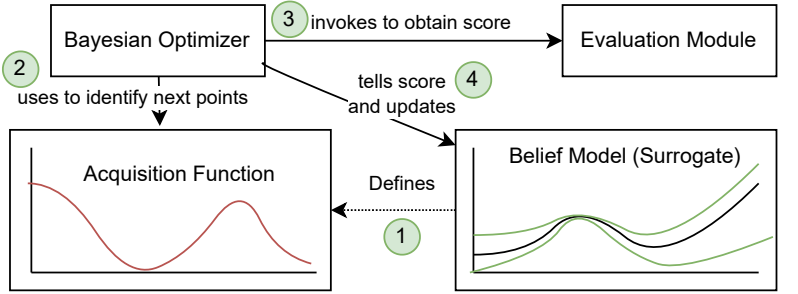
Since the metric m is a black box, we can see the performance $Y_\theta = m(\theta)$ of each configuration θ as a random variable.

We could hold an explicit belief model over the distributions of $\mathbb{P}(Y_\theta \mid \mathcal{H})$, for each θ, where $\mathcal{H} = \{(\theta_1, m(\theta_1)), .., (\theta_n, m(\theta_n))\}$ is a history of observations.

This is what Bayesian Optimization (BO) does:

1. an acquisition function defines a score for each θ based on $\mathbb{P}(Y_\theta \mid \mathcal{H})$,
2. the optimizer identifies the maximizer $\tilde{\theta}$ of the acquisition function
3. the optimizer queries the candidate performance $m(\tilde{\theta})$
4. the optimizer extends $\mathcal{H}$ by $(\tilde{\theta}, m(\tilde{\theta}))$ and asks the belief model to update

Bayesian Optimization



Natural next questions:

- how to model the beliefs over the space, i.e. $\mathbb{P}(Y_\theta \mid \mathcal{H})$?
- what should be the acquisition function?
- how can we optimize the acquisition function?

Bayesian Optimization

Belief Model - Gaussian Processes

Suppose that all hyperparmaters are numeric, i.e., $\Theta_i \subseteq \mathbb{R}$.

Then we can model a belief of how m behaves via a Gaussian Process (GP).

In an, GP the posterior $\mathbb{P}(Y_\theta \mid \mathcal{H})$ for the performance Y_θ of θ is a Gaussian itself, which can be computed in *closed form*:

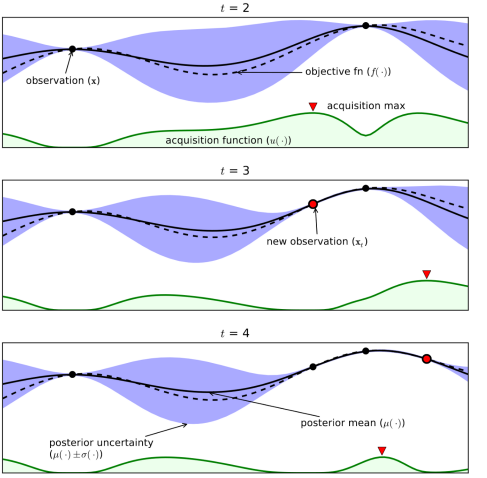
$$\mathbb{P}(Y_\theta \mid \mathcal{H}) \sim \mathcal{N}(\mu_{\theta,\mathcal{H}}, \sigma^2_{\theta,\mathcal{H}}),$$

where $\mu_{\theta,\mathcal{H}}$ and $\sigma^2_{\theta,\mathcal{H}}$ are computed from means and covariances in $\mathcal{H}$.

Updating the belief model hence only means to add items to $\mathcal{H}$.

Bayesian Optimization

Belief Model - Gaussian Processes



Bayesian Optimization

Belief Model - Random Forests

One shortcoming of GPs: can only treat numerical domains Θ_i.

Alternative: Train a regression Random Forest based on the data $\mathcal{H}$.

For a new θ, every regression tree in the forest predicts some score. This yields:

1. μ_θ as the average score across all trees
2. σ_θ^2 as the variance in the predictions across trees

Again, assuming a normal distribution, this leads to a belief of

$$\mathbb{P}(Y_\theta \mid \mathcal{H}) \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$$

Bayesian Optimization

Acquisition Functions

A common acquisition function is the expected improvement:

$$EI_\tau(\theta) := \int_{-\infty}^{\infty} \max\{0, y_\theta - \tau\} \mathbb{P}(y_\theta | \mathcal{H}) dy_\theta,$$

where τ is a threshold; typically the currently best known score.

Other acquisition functions are:

1. Probability of Improvement (where can I improve?)
2. Entropy Search (where is the minimum?)

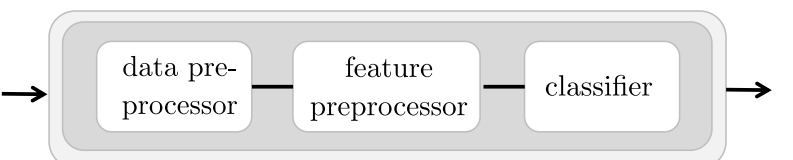
Finding the maximizer of an acquisition function is a hard problem.

The Pipeline Optimization Problem

In pipeline optimization, one decides about

- structure
- algorithms
- hyperparameters

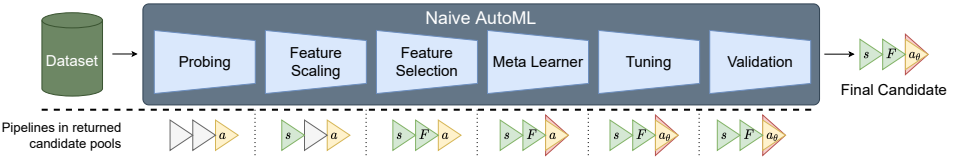
Many approaches optimize a pre-defined pipeline template, in which slots are defined for specific algorithm types (scalers, feature selectors, classifiers):



Pipeline Optimization

Naive AutoML

Hypothesis: Optimizing pipeline slots locally leads to a globally optimal solution.



This is a very simple and fast approach and is hardly ever significantly outperformed by sophisticated techniques such as BO [Mohr and Wever, 2022].

Pipeline Optimization

Auto-WEKA/auto-sklearn

Idea: See the pipeline optimization problem as an HPO problem (algorithms = HPs) with dependencies among hyperparameters, and solve it with BO.

Use one "meta-hyperparameter" for

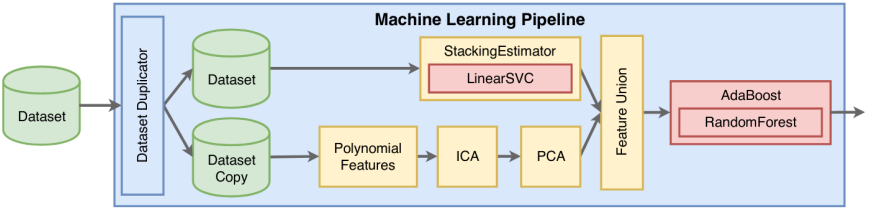
- every algorithm that might occur in the pipeline
- every hyperparameter of every algorithm

The optimizer knows the dependencies among these parameters and only proposes valid configurations.

Pipeline Optimization

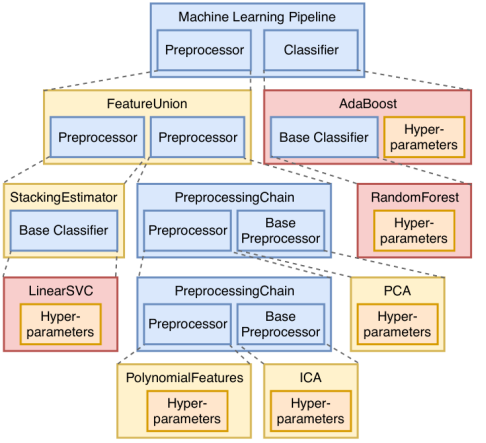
Complex Pipeline Structures

Pipelines can have all sorts of structures not covered by the above approaches:



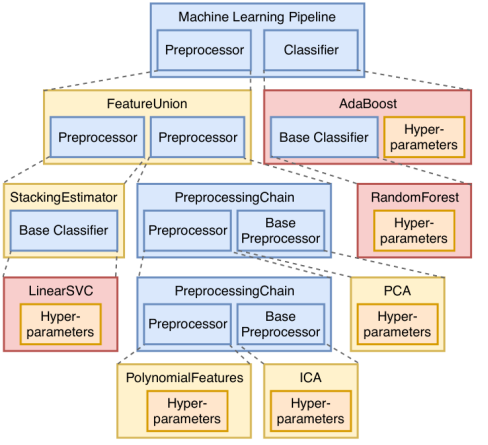
ML-Plan

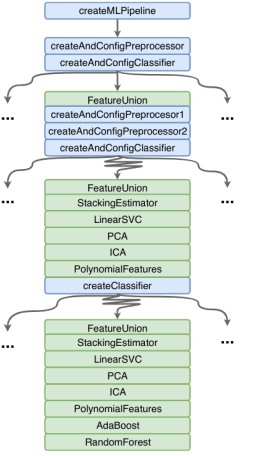
Definition of the Planning Tree



ML-Plan

Definition of the Planning Tree





ML-Plan

Traversing the Planning Tree

Two problems arise when searching the tree spanned by the HTN semantics:

- the tree is *huge* and cannot be traversed completely,
- we have no classical heuristic values in inner nodes

Typical approaches:

- Use an MCTS-like scheme that produces roll-outs under each inner node and trades off exploration vs. exploitation [Rakotoarison et al., 2019]
- Be greedy, i.e., limit exploration to some constant scope. [Mohr et al., 2018]

Overview

Candidate Generating Procedures

Meta-Learning for HPO and Automated Pipeline Design

- Warm-Starting
- Early Discarding
- Budget Picking

Meta-Learning for HPO and Automated Pipeline Design

Accumulated knowledge is used to ...

- define an *initial order* on candidates (Warm-Starting)
- use learning curves to early detect sub-optimality (Early Discarding)
- define the sufficient budget to evaluate learner (Budget Picking)

Warm-Starting

Assume that a set of pipeline optimization problem instances is available.

Simplest idea:

- Solve all of them pseudo-optimally, and
- assign a score to each algorithm based on its occurrence in best solutions.

This creates a ranking reflecting how well algorithms work *on average* that can be used by any pipeline optimizer:

- Naive AutoML: sort candidates in each slot based on recommendation
- auto-sklearn: evaluate recommended pipelines prior to starting BO
- ML-Plan: enforce roll-outs over paths that lead to recommended solutions.

Warm-Starting

Instance-Specific Algorithm Configuration: Optimal Solutions for Regions

We can warm-start a bit more instance-specific by using dataset meta-features.

One approach based on clustering is ISAC [Kadioglu et al., 2010].

Offline Phase:

- create a clustering on the dataset meta-features (flexible k, e.g., g-means).
- find θ^* (or ranking) for each cluster using exhaustive optimization.

Online Phase:

- for a new problem instance, identify closest cluster.
- evaluate candidates in the order suggested for this cluster.

Warm-Starting

Ranking Problems by Proximity

A special case of ISAC is where we have a (pseudo-)optimal solution for *every* previous instance available.

In that case, we can

- associate the dataset meta-features with the best (k) pipeline(s) for a specific problem instance
- for a new instance,
  - compute the dataset meta-features,
  - sort all the known cases by their distance in terms of dataset meta-features,
  - use this distance to create a ranking of candidates by the best solutions to those instances

Warm-Starting

πBO: Placing a Prior in a Belief Model

πBO [Hvarfner et al., 2022] is a simple extension of the BO framework.

In each iteration, one picks

$$\arg\max_{\theta} \alpha(\theta, \mathcal{H})\pi(\theta),$$

where

- α is some acquisition function and
- π is simply a weight function over the configuration space.

π can be encoded data-driven or by an expert.

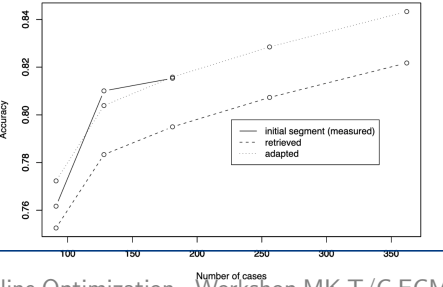
Early Discarding

Using Learning Curves to Determine Similarity

Idea: Instead of evaluating on the full budget, evaluate on some cheap budgets.

Now leverage knowledge of rankings for these budgets ...

- ... for same learners on other datasets [Leite and Brazdil, 2008]
- ... for other learners on same dataset [Chandrashekaran and Lane, 2017]

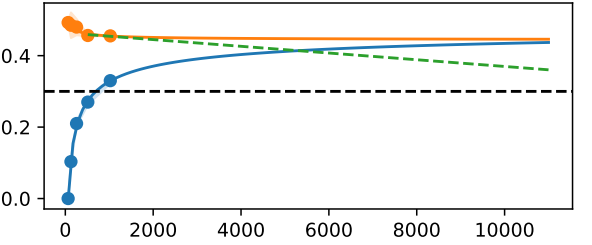


Early Discarding

Learning Curve Based Cross Validation (LCCV)

Another common approach is to compare the *extrapolation* of a single learning curve with some baseline.

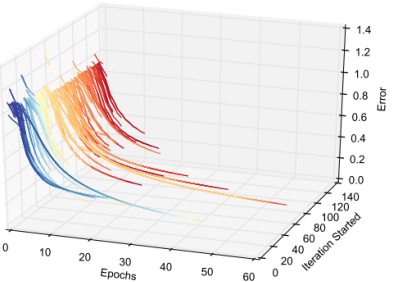
This is done in the learning curve based cross validation (LCCV) [Mohr and van Rijn, 2021].



Early Discarding

Freeze-Thaw BO

Model learning curves with a GP, and only keep training candidates whose curve has a competitive (extrapolated) limit value [Swersky et al., 2014].

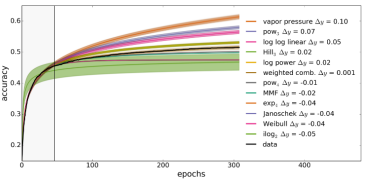


This requires that learners can be trained incrementally.

Early Discarding

Probabilistic Forecast

GPs are not good at extrapolation, so [Domhan et al., 2015] propose to use established learning curve models instead:



Reference name	Formula
vapor pressure	$\exp(a + \frac{b}{x} + c\log(x))$
pow_3	$c - ax^{-\alpha}$
log log linear	$\log(a\log(x) + b)$
Hill_3	$\frac{y_{\max} x^\eta}{\kappa^\eta + x^\eta}$
log power	$\frac{a}{1+\left(\frac{x}{e^b}\right)^c}$
pow_4	$c - (ax + b)^{-\alpha}$
MMF	$\alpha - \frac{\alpha-\beta}{1+(\kappa x)^\delta}$
exp_4	$c - e^{-ax^\alpha + b}$
Janoschek	$\alpha - (\alpha - \beta)e^{-\kappa x^\delta}$
Weibull	$\alpha - (\alpha - \beta)e^{-(\kappa x)^\delta}$
ilog_2	$c - \frac{a}{\log x}$

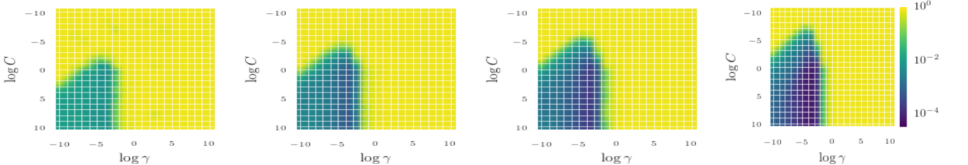
Approach

- Estimate the model parameters are estimated with MCMC.
- Quantify the probability that a partially observed curve will eventually exceed the best performance observed so far.
- Discard the candidate if the probability is too low.

Budget Picking

FaBOLAS

For non-incremental learners, it seems desirable to *pick* the smallest budget that still gives sufficiently reliable results. FaBOLAS does this [Klein et al., 2017].



(a) $s = 1/128$ (b) $s = 1/16$ (c) $s = 1/4$ (d) $s = 1$

The evaluation budget s (portion in $[0, 1]$) becomes subject of optimization itself, and a model is learned on this. No offline knowledge is used.

Summary

Meta-Learning can be used for HPO and pipeline optimization.

This is done mainly with two goals:

- Influencing the order in which candidates are generated.
  This is achieved by warm-starting (often contextualized).
- Increasing efficiency of candidate evaluations.
  This is achieved using learning curves for
  - Early Discarding
  - Budget Selection

References

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

A. Chandrashekaran and I. R. Lane. Speeding up hyper-parameter optimization by extrapolation of learning curves using previous builds. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017*, volume 10534 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2017.

T. Domhan, J. T. Springenberg, and F. 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.

C. Hvarfner, D. Stoll, A. Souza, M. Lindauer, F. Hutter, and L. Nardi. bo: Augmenting acquisition functions with user beliefs for bayesian optimization, 2022. URL `https://arxiv.org/abs/2204.11051`.

S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. Isac–instance-specific algorithm configuration. In *ECAI 2010*, pages 751–756. IOS Press, 2010.

A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536. PMLR, 2017.

R. Leite and P. Brazdil. Selecting classifiers using metalearning with sampling landmarks and data characterization. In *Proceedings of the 2nd Planning to Learn Workshop (PlanLearn) at ICML/COLT/UAI*, pages 35–41, 2008.

F. Mohr and J. N. van Rijn. Fast and informative model selection using learning curve cross-validation. *arXiv preprint arXiv:2111.13914*, 2021.

F. Mohr and M. Wever. Naive automated machine learning. *Machine Learning Journal*, 2022.

F. Mohr, M. Wever, and E. Hüllermeier. Ml-plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107(8): 1495–1515, 2018.

H. Rakotoarison, M. Schoenauer, and M. Sebag. Automated machine learning with monte-carlo tree search. *arXiv preprint arXiv:1906.00170*, 2019.

K. Swersky, J. Snoek, and R. P. Adams. Freeze-thaw bayesian optimization. *CoRR*, abs/1406.3896, 2014.