AutoML: Hyperparameter Optimization

Nested Resampling

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Nested Crossvalidation I

Selecting the best model from a set of potential candidates (e.g., different classes of learners, different hyperparameter settings, different feature sets, different preprocessing, ...) is an important part of most machine learning problems.

Problem

- We cannot evaluate our finally selected learner on the same resampling splits that we have used to perform model selection for it, e.g., to tune its hyperparameters.
- By repeatedly evaluating the learner on the same test set, or the same CV splits, information about the test set leaks into our evaluation.
- Danger of overfitting to the resampling splits / overtuning!
- The final performance estimate will be optimistically biased.
- One could also see this as a problem similar to multiple testing.

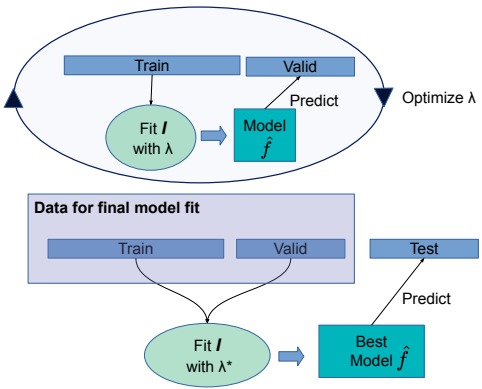
Nested Crossvalidation II

- All parts of the model building (including model selection, preprocessing) should be embedded in the model-finding process **on the training data**.
- The test set should only be touched once, so we have no way of *cheating*. The test dataset is only used once *after* a model is completely trained, after deciding for example on specific hyper-parameters.
  Only if we do this, the performance estimates we obtained from the test set are **unbiased estimates** of the true performance.
- For steps that themselves require resampling (e.g., hyperparameter tuning) this results in **nested resampling**, i.e., resampling strategies for both.
  - tuning: an inner resampling loop to find what works best based on training data
  - evaluation: an outer resampling loop to evaluate on data not used for tuning to get honest estimates of the expected performance on new data

Nested Crossvalidation III

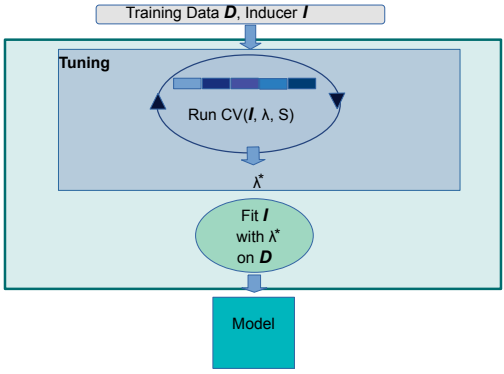
Simplest method to achieve this: a 3-way split

- During tuning, a learner is trained on the **training set**, evaluated on the **validation set**.
- After the best model configuration $\hat{\boldsymbol{\lambda}}$ is selected, we re-train on the joint (training+validation) set and evaluate the model's performance on the **test set**.



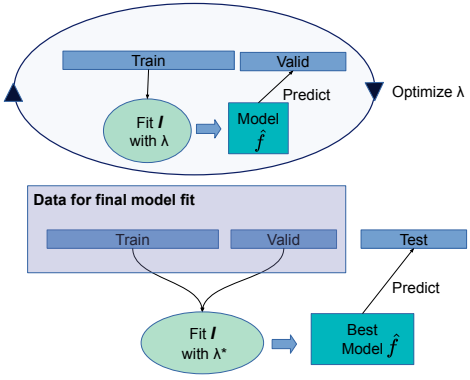
Nested Crossvalidation IV

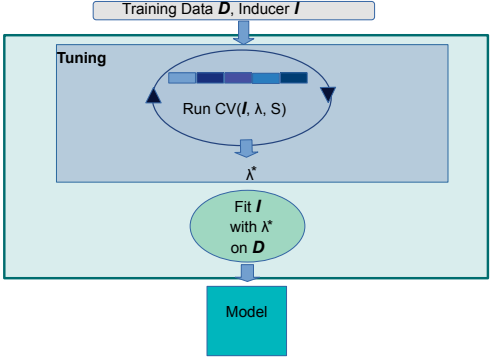
- Effectively, the tuning step is now simply part of a more complex training procedure.
- We could see this as removing the hyperparameters from the inputs of the algorithm and making it *self-tuning*.



Nested Crossvalidation V

More precisely: the combined training & validation set is actually the training set for the *self-tuning* endowed algorithm.





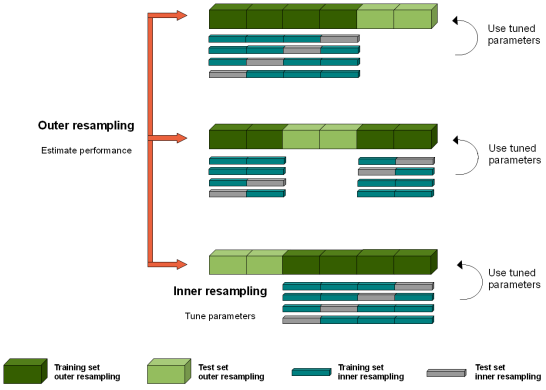
Nested Crossvalidation VI

Just like we can generalize holdout splitting to resampling to get more reliable estimates of the predictive performance, we can generalize the training/validation/test approach to **nested resampling**.

This results in two resampling loops, whereas the resampling for the tuning is nested in the resampling for the outer evaluation, i.e., one resampling strategy for tuning and another for outer evaluation.

Nested Crossvalidation VII

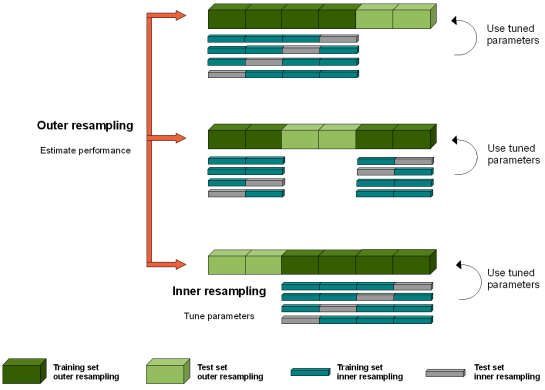
Assume we want to tune over a set of candidate HP configurations $\boldsymbol{\lambda}_i; i = 1, \ldots$ with 4-fold CV in the inner resampling and 3-fold CV in the outer loop. The outer loop is visualized as the light green and dark green parts.



Nested Crossvalidation VIII

In each iteration of the outer loop we:

- Split off the light green testing data
- Run the tuner on the dark green part of the data, e.g., evaluate each $\boldsymbol{\lambda}_i$ through fourfold CV on the dark green part
- Return the winning $\hat{\boldsymbol{\lambda}}$ that performed best on the grey inner test sets
- Re-train the model on the full outer dark green train set
- Evaluate it on the outer light green test set



The error estimates on the outer samples (light green) are unbiased because this data was strictly excluded from the model-building process of the model that was tested on.