AutoML: Evaluation

Nested Resampling

Bernd Bischl[1] Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

[1] Some slides taken from Bernd Bischl's "Introduction to Machine Learning" lecture at LMU. [Bischl]

Motivation

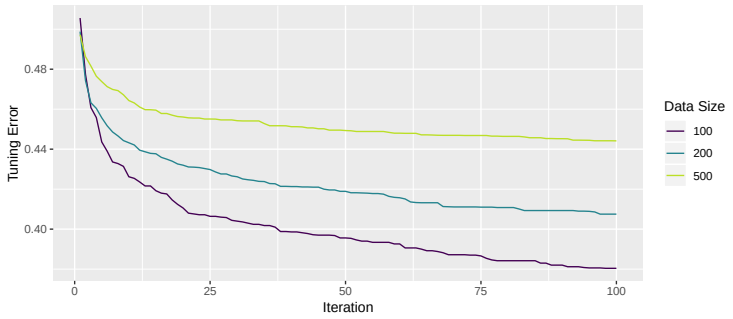
Selecting the best model from a set of potential candidates (e.g. different classes of learners, different hyperparameter settings, different feature sets, different preprocessing...) is an important part of most machine learning problems. However,

- cannot evaluate selected learner on the same resampling splits used to select it
- repeatedly evaluating learner on same test set or same CV splits "leaks" information about test set into evaluation
- danger of overfitting to the resampling splits or overtuning
- final performance estimate would be optimistically biased
- similar to multiple hypothesis testing

Motivating Example I

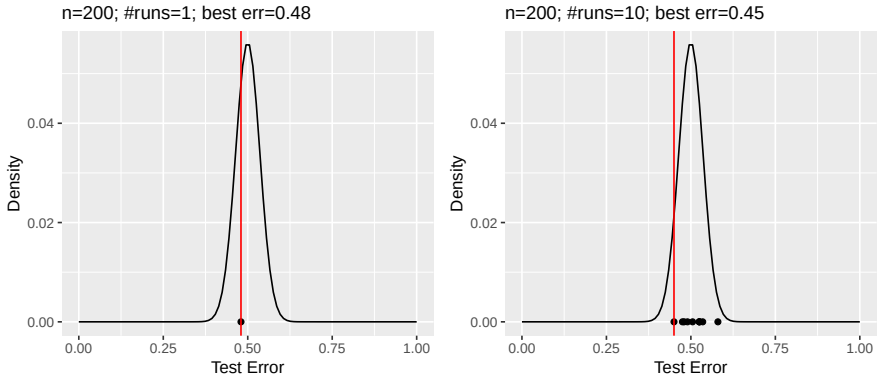
- binary classification problem with equal class sizes
- learner with hyperparameter $\boldsymbol{\lambda}$
- learner is (nonsense) feature-independent classifier that picks random labels with equal probability, $\boldsymbol{\lambda}$ has no effect
- true generalization error is 50%
- cross-validation of learner (with any fixed $\boldsymbol{\lambda}$) will easily show this (if the partitioned data set for CV is not too small)
- let's "tune" it by trying out 100 different $\boldsymbol{\lambda}$ values
- repeat this experiment 50 times and average results

Motivating Example II



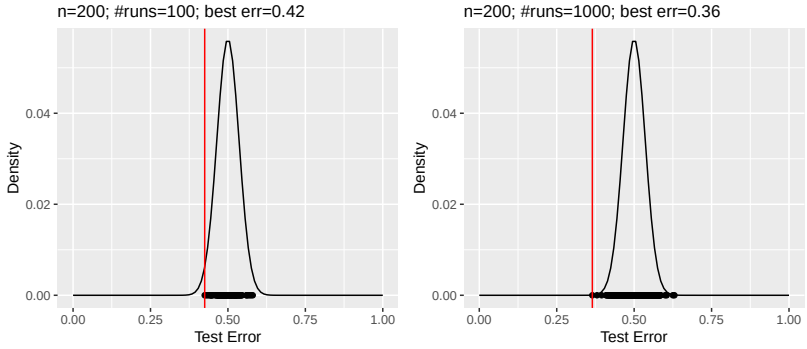
- shown is best "tuning error" (i.e. performance of model with fixed $\boldsymbol{\lambda}$ in cross-validation) after k tuning iterations
- evaluated for different data set sizes

Motivating Example III



- for one experiment, CV score is close to 0.5, as expected
- errors essentially sampled from (rescaled) binomial distribution
- scores from multiple experiments also arranged around expected mean of 0.5

Motivating Example IV



- tuning means we take the minimum of the scores
- not estimate of average performance, but best-case performance
- the more we sample, the more "biased" this value becomes $\rightarrow$ unrealistic generalization performance estimate

Untouched Test Set Principle I

Instead: simulate what actually happens when applying machine learning models

- all parts of model construction (including model selection, preprocessing) evaluated **on training data**
- test set only touched once, so no way of "cheating"
- test dataset is only used once *after* model is completely trained (including e.g. deciding hyperparameter values)
- performance estimates from test set now **unbiased estimates** of the true performance

Untouched Test Set Principle II

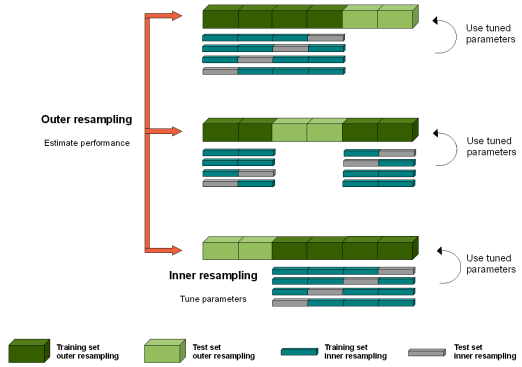
- for steps that themselves require resampling (e.g. hyperparameter tuning) this results in **nested resampling**, i.e. resampling strategies for both
  - inner evaluation to find what works best based on training data
  - outer evaluation on data not used in inner evaluation to get unbiased estimates of expected performance on new data

Nested Resampling I

- holdout can be generalized to resampling for more reliable generalization performance estimates
- resampling can be generalized to nested resampling
- nested resampling loops for inner and outer evaluation for hyperparameter tuning

Nested Resampling II

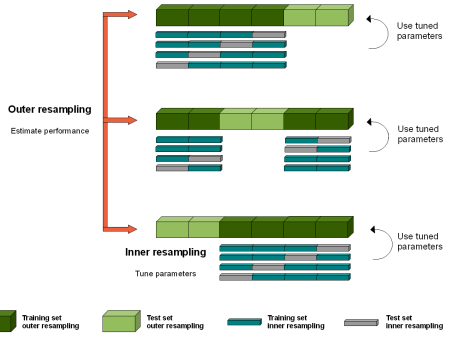
Example: four-fold CV for inner resampling, three-fold CV in outer resampling



Nested Resampling III

In each iteration of the outer loop:

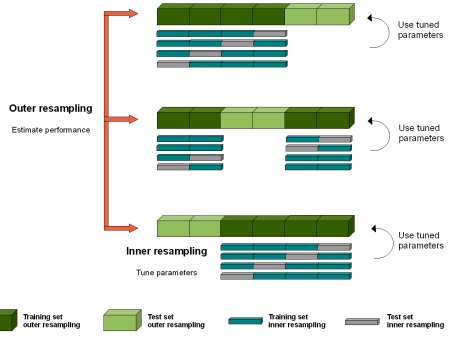
- split light green testing data
- run hyperparameter tuner on dark green part of data, i.e. evaluate each $\boldsymbol{\lambda}_i$ through four-fold CV on dark green part



Nested Resampling IV

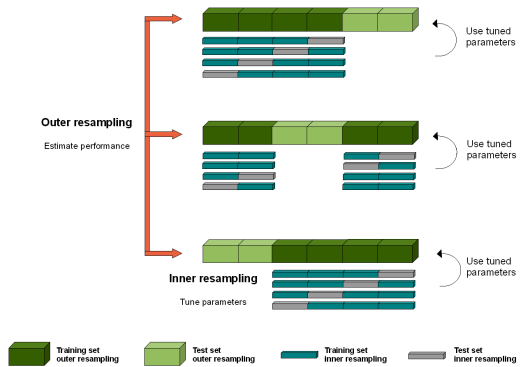
In each iteration of the outer loop:

- return winning $\hat{\boldsymbol{\lambda}}$ that performed best on the grey inner test sets
- re-train model on full outer dark green training set
- evaluate model on outer light green test set



Nested Resampling V

$\rightarrow$ error estimates on outer samples (light green) are unbiased because this data was not used process constructing the tested model



Nested Resampling Example

Revisited motivating example: expected performance estimate with nested resampling: