Introduction to Machine Learning

Nested Resampling

Bernd Bischl, Christoph Molnar, Daniel Schalk, Fabian Scheipl

Department of Statistics – LMU Munich

MOTIVATION

In model selection, we are interested in selecting the best model from a set of potential candidate models (e.g., different model classes, different hyperparameter settings, different feature sets).

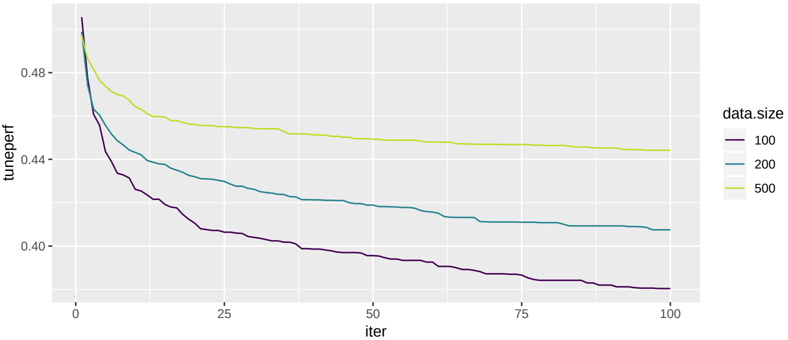
Problem

- We cannot evaluate our finally selected leaner 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.

INSTRUCTIVE AND PROBLEMATIC EXAMPLE

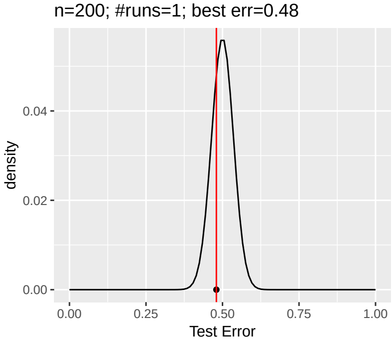
- Assume a binary classification problem with equal class sizes.
- Assume a learner with hyperparameter λ.
- Here, the learner is a (nonsense) feature-independent classifier, where λ has no effect. The learner simply predicts random labels with equal probability.
- Of course, it's true generalization error is 50%.
- A cross-validation of the learner (with any fixed λ) will easily show this (given that the partitioned data set for CV is not too small).
- Now lets "tune" it, by trying out 100 different λ values.
- We repeat this experiment 50 times and average results.

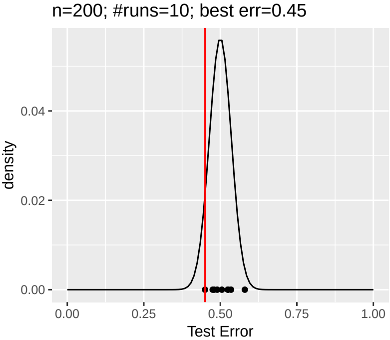
INSTRUCTIVE AND PROBLEMATIC EXAMPLE



- Plotted is the best “tuning error” after k tuning iterations
- We have performed the experiment for different sizes of learning data that where cross-validated.

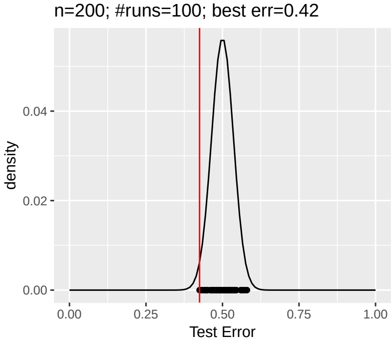
INSTRUCTIVE AND PROBLEMATIC EXAMPLE

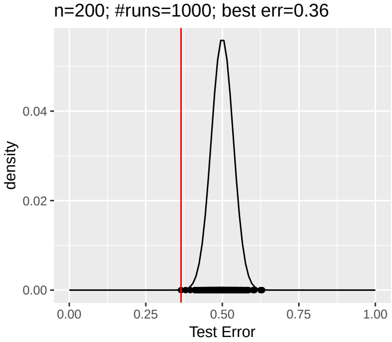




- For 1 experiment, the CV score will be nearly 0.5, as expected
- We basically sample from a (rescaled) binomial distribution when we calculate error rates
- And multiple experiment scores are also nicely arranged around the expected mean 0.5

INSTRUCTIVE AND PROBLEMATIC EXAMPLE





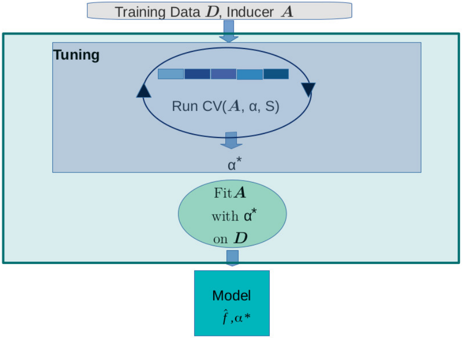
- But in tuning we take the minimum of those!
- The more we sample, the more "biased" this value becomes

UNTOUCHED TEST SET PRINCIPLE

- Again, simply simulate what happens in model application.
- 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 we should only touch once, so we have no way of “cheating”. The test dataset is only used once a model is completely trained, after deciding for example on specific hyper-parameters. Performances 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 two **nested resampling** loops, i.e. a resampling strategy for both tuning and outer evaluation.

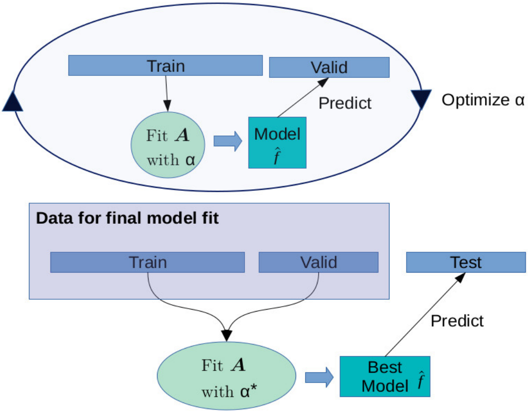
TUNING AS PART OF MODEL BUILDING

- It conceptually helps to see the tuning step as now effectively 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"



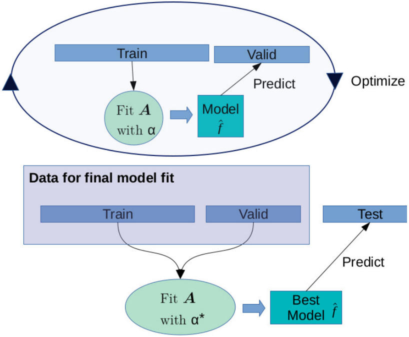
TRAIN VALIDATION TEST

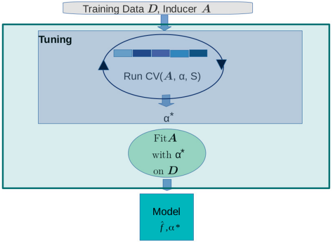
- Simple 3-way split; during tuning, a learner is trained on the training set, evaluated on the validation set
- After the final model is selected, we fit on joint (training+validation) set and evaluate a final time on the test set



TRAIN VALIDATION TEST

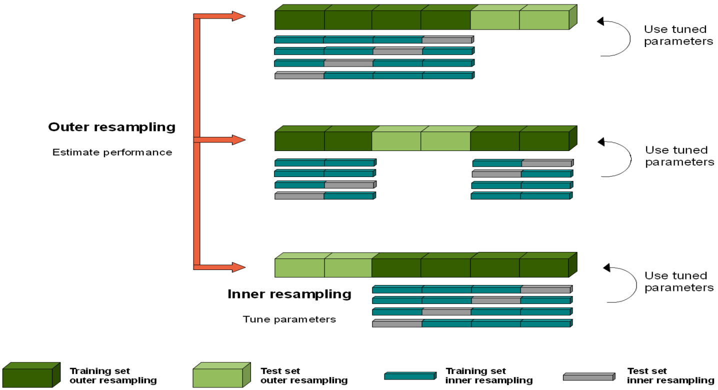
More precisely: the joint train+valid set is actually the training test for the "self-tuning" endowed algorithm





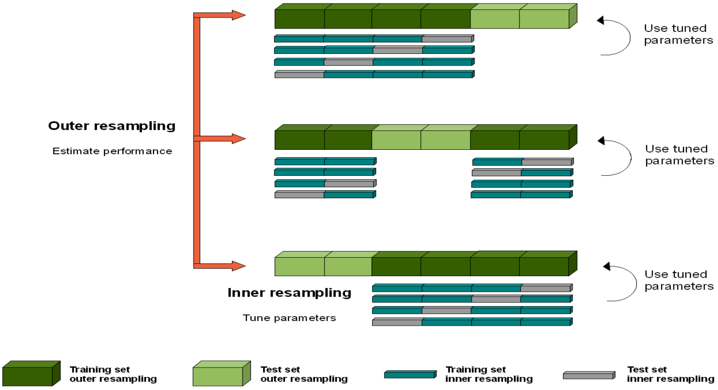
NESTED RESAMPLING

As we can generalize holdout splitting to resampling, we can generalize the train+valid+test approach to nested resampling. This results in in two nested resampling loops, i.e. a resampling strategy for both tuning and outer evaluation.



NESTED RESAMPLING

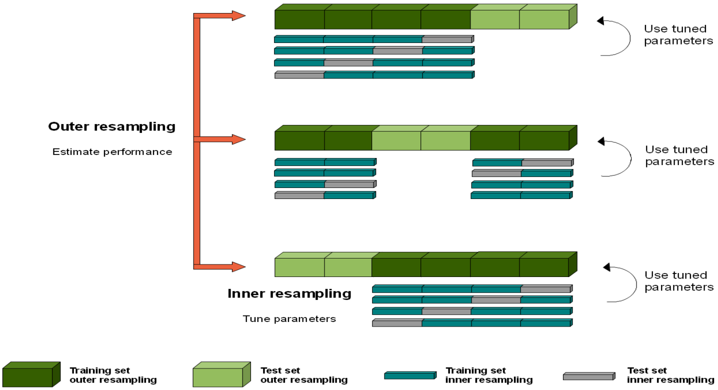
Assume we want to tune over a set of candidate HP configurations $\lambda_i; i = 1, \dots$ with 4-fold CV in the inner resampling and 3-fold CV in the outer loop. The outer loop is visualized as the lightgreen and darkgreen parts.



NESTED RESAMPLING

In each outer loop we do:

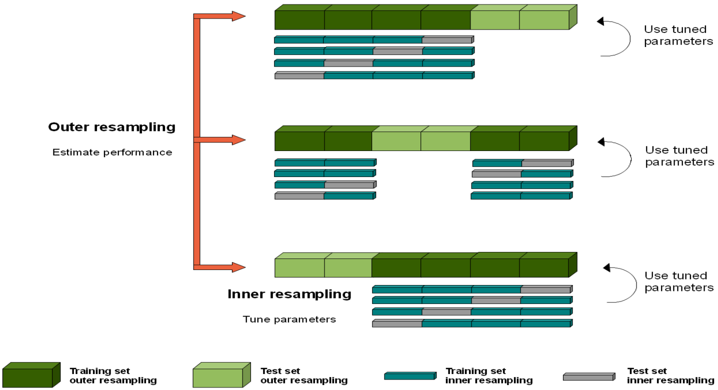
- Split off lightgreen testing data
- Run the tuner on the darkgeen part, e.g., evaluate each λ_i through 4CV on the darkgreen part



NESTED RESAMPLING

In each outer loop we do:

- Return the winning λ^*
- Re-train the model on the full outer darkgreen train set;
- Predict on the outer lightgreen test set



NESTED RESAMPLING

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