I2ML :: CHEAT SHEET

The **I2ML**: Introduction to Machine Learning course offers an introductory and applied overview of "supervised" Machine Learning. It is organized as a digital lecture.

Hyperparameters

Hyperparameters $\boldsymbol{\lambda}$ are parameters that are *inputs* to the training problem, in which a learner $\mathcal{I}$ minimizes the empirical risk on a training data set in order to find optimal **model parameters $\boldsymbol{\theta}$** which define the fitted model $\hat{f}$.

- not decided during training rather must be specified before the training
- an **input** of the training
- often control the complexity of a model
- can influence any structural property of a model or computational part of the training process

Types of hyperparameters:

- Real-valued parameters: Minimal error improvement in a tree to accept a split
- Integer parameters: Neighborhood size k for k-NN
- Categorical parameters: Which distance measure for k-NN

Hyperparameter Tuning

(Hyperparameter) Tuning is the process of finding good model hyperparameters

A bi-level optimization problem:

The well-known risk minimization problem to find $\hat{f}$ is **nested** within the outer hyperparameter optimization (also called second-level problem). **Nested hyperparameter tuning problem:**

$$\min_{\boldsymbol{\lambda}\in\Lambda} \widehat{GE}_{\mathcal{D}_{\text{test}}}\left(\mathcal{I}(\mathcal{D}_{\text{train}}, \boldsymbol{\lambda})\right)$$

Components of a tuning problem:

The dataset, the learner(tuned), the learner's hyperparameters and their respective regions-of-interest over which optimization is done, the performance measure, a (resampling) procedure for estimating the predictive performance.

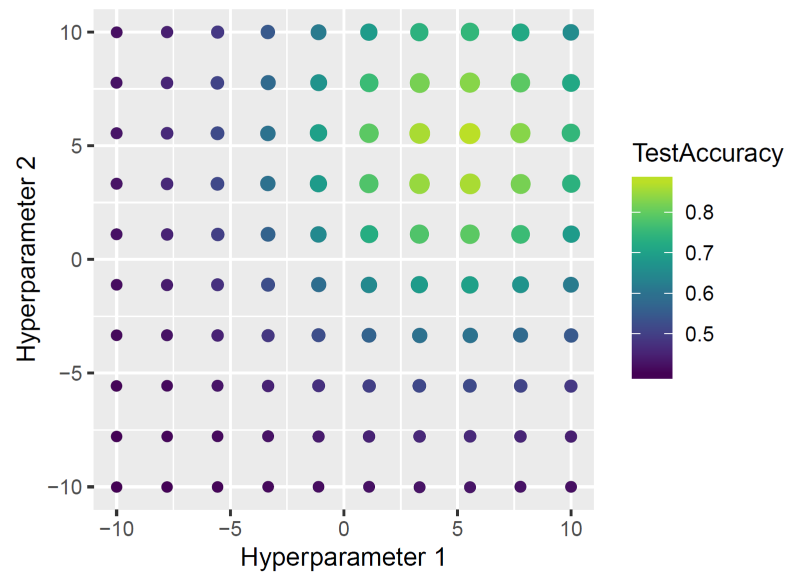
Tuning is hard:

Because, tuning is derivative-free which is a black-box problem. Every evaluation is **expensive** and the answer we get from that evaluation is **not exact, but stochastic** in most settings. The space of hyperparameters we optimize over has a non-metric, complicated structure

Basic Techniques

Grid Search:

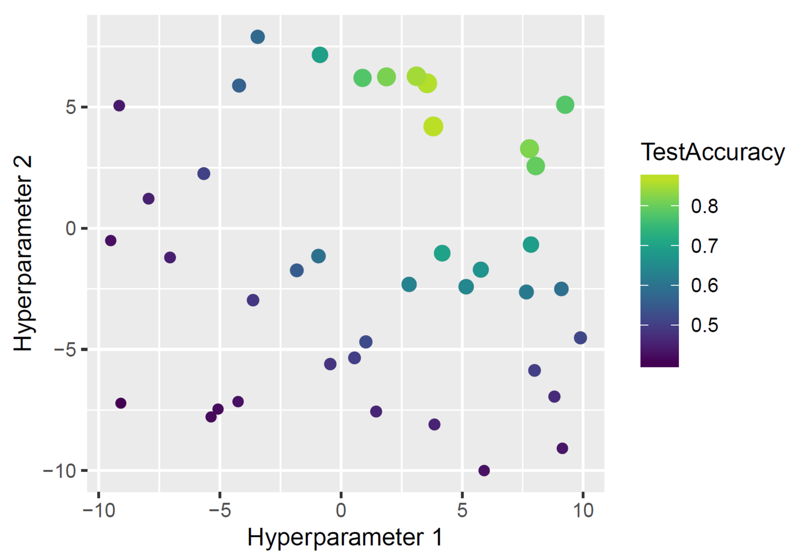
- Tries all hyperparameter combinations
- For each hyperparameter a finite set of candidates is predefined and searches all possible combinations in arbitrary order
- Grid Search over 10x10 points:



Note: It is very easy to implement, all parameter types possible and parallelizing computation is trivial. However, it scales badly and is inefficient.

Random Search:

- Small variation of Grid Search.
- Uniformly sample from the region-of-interest
- Random Search over 10x10 points:



Note: Very easy to implement, all parameter types possible, trivial parallelization and an anytime algorithm and no discretization. But, it is also inefficient and scales badly.

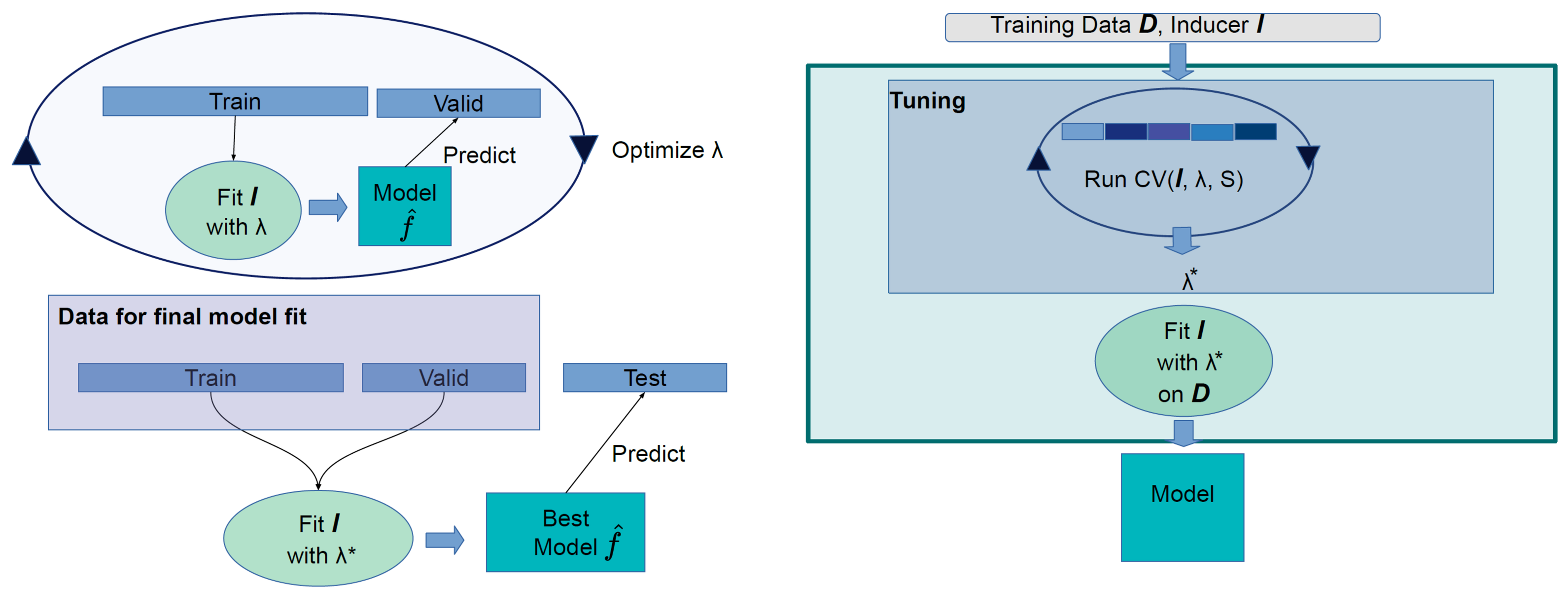
Tuning

Problem of Tuning

Need to **select an optimal learner** without compromising the **accuracy of the performance estimate** for that learner. For this **untouched test set** is needed.

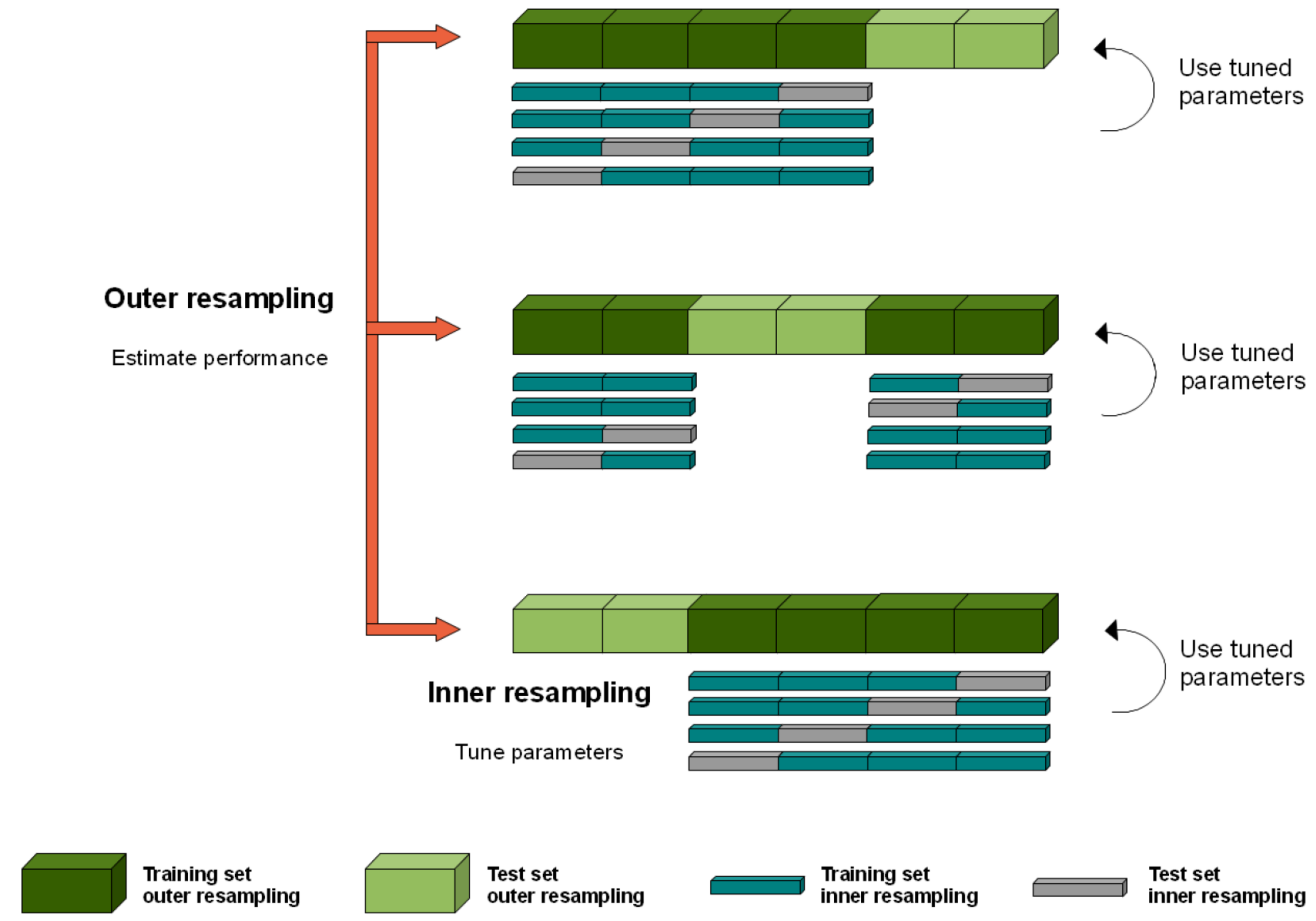
Train-validation-test

A 3-way split is the simplest method: During tuning, a learner is trained on the **training set**, evaluated on the **validation set** and after the best model configuration λ^* is selected, we re-train on the joint (training+validation) set and evaluate the model's performance on the **test set**.



If we want to tune over a set of candidate HP configurations $\lambda_i; i = 1, \ldots$ 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 Resampling



The outer loop is visualized as the light green and dark green parts. This is with 4-fold CV in the inner resampling and 3-fold CV in the outer loop.