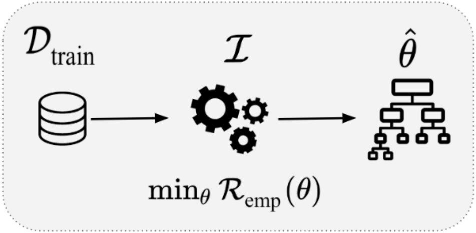
Introduction to Machine Learning

Hyperparameter Tuning - Introduction

MOTIVATING EXAMPLE

- Given a data set, we want to train a classification tree.
- We feel that a maximum tree depth of 4 has worked out well for us previously, so we decide to set this hyperparameter to 4.
- The learner ("inducer") $\mathcal{I}$ takes the input data, internally performs **empirical risk minimization**, and returns a fitted tree model $\hat{f}(\mathbf{x}) = f(\mathbf{x}, \hat{\theta})$ of at most depth $\lambda = 4$ that minimizes the empirical risk.

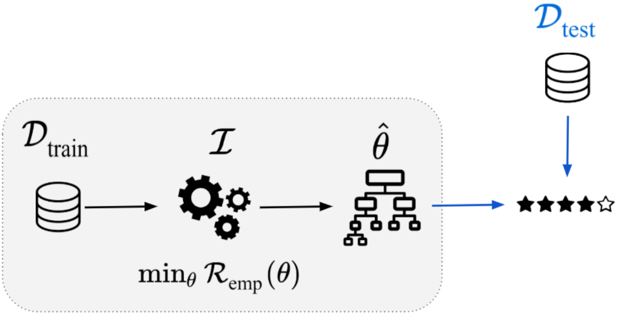


$\mathcal{D}_{\text{train}}$
$\mathcal{I}$
$\hat{\theta}$
$\min_\theta \mathcal{R}_{\text{emp}}(\theta)$

MOTIVATING EXAMPLE

- We are **actually** interested in the **generalization performance** $GE\left(\hat{f}\right)$ of the estimated model on new, previously unseen data.
- We estimate the generalization performance by evaluating the model $\hat{f}$ on a test set $\mathcal{D}_{\text{test}}$:

$$\widehat{GE}_{\mathcal{D}_{\text{test}}}\left(\hat{f}\right) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{test}}} L\left(y, \hat{f}(\mathbf{x})\right)$$



$\mathcal{D}_{\text{test}}$
$\mathcal{D}_{\text{train}}$
$\mathcal{I}$
$\hat{\theta}$
★★★★☆
$\min_\theta \mathcal{R}_{\text{emp}}(\theta)$

MOTIVATING EXAMPLE

- But many ML algorithms are sensitive w.r.t. a good setting of their hyperparameters, and generalization performance might be bad if we have chosen a suboptimal configuration:
    - The data may be too complex to be modeled by a tree of depth 4
    - The data may be much simpler than we thought, and a tree of depth 4 overfits

$\Longrightarrow$ Algorithmically try out different values for the tree depth. For each maximum depth λ, we have to train the model **to completion** and evaluate its performance on the test set.

- We choose the tree depth λ that is **optimal** w.r.t. the generalization error of the model.

MODEL PARAMETERS VS. HYPERPARAMETERS

It is critical to understand the difference between model parameters and hyperparameters.

Model parameters are optimized during training, typically via loss minimization. They are an **output** of the training. Examples:

- The splits and terminal node constants of a tree learner
- Coefficients $\boldsymbol{\theta}$ of a linear model $f(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x}$

MODEL PARAMETERS VS. HYPERPARAMETERS

In contrast, **hyperparameters** (HPs) are not decided during training. They must be specified before the training, they are an **input** of the training. Hyperparameters often control the complexity of a model, i.e., how flexible the model is. But they can in principle influence any structural property of a model or computational part of the training process.

Examples:

- The maximum depth of a tree
- k and which distance measure to use for k-NN
- The number and maximal order of interactions to be included in a linear regression model

TYPES OF HYPERPARAMETERS

We summarize all hyperparameters we want to tune over in a vector $\boldsymbol{\lambda} \in \Lambda$ of (possibly) mixed type. HPs can have different types:

- Real-valued parameters, e.g.:
  - Minimal error improvement in a tree to accept a split
  - Bandwidths of the kernel density estimates for Naive Bayes
- Integer parameters, e.g.:
  - Neighborhood size *k* for *k*-NN
  - *mtry* in a random forest
- Categorical parameters, e.g.:
  - Which split criterion for classification trees?
  - Which distance measure for *k*-NN?

Hyperparameters are often **hierarchically dependent** on each other, e.g., *if* we use a kernel-density estimate for Naive Bayes, what is its width?