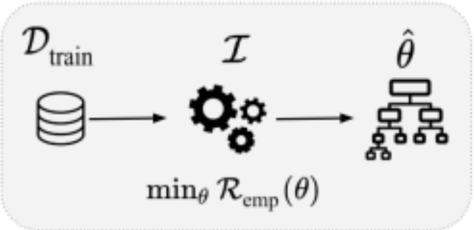
MOTIVATING EXAMPLE

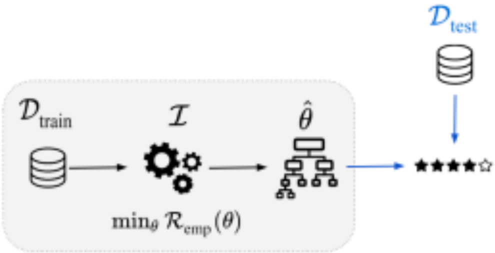
- Given a data set, we want to train a classification tree.
- We feel that a maximum tree depth of 4 has worked out well for us previously, so we decide to set this hyperparameter to 4.
- The learner ("inducer") $\mathcal{I}$ takes the input data, internally performs **empirical risk minimization**,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



MOTIVATING EXAMPLE / 2

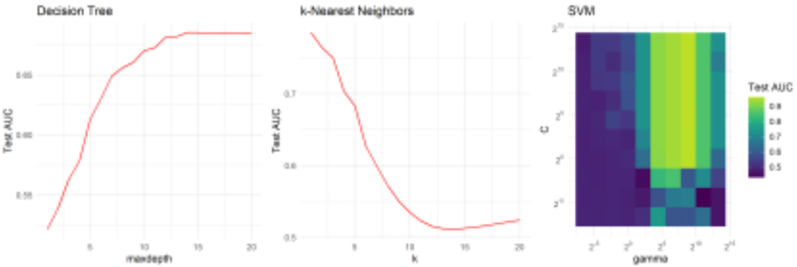
- We are **actually** interested in the **generalization performance** $\mathrm{GE}\left(\hat{f}\right)$ of the estimated model on new, previously unseen data.
- We estimate the generalization performance by evaluating the model $\hat{f} = \mathcal{I}(\mathcal{D}_{\text{train}}, \boldsymbol{\lambda})$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

$$\widehat{\mathrm{GE}}_{\mathcal{D}_{\text{train}},\mathcal{D}_{\text{test}}}(\mathcal{I}, \boldsymbol{\lambda}, n_{\text{train}}, \rho) = \rho\left(\mathbf{y}_{\mathcal{D}_{\text{test}}}, \boldsymbol{F}_{\mathcal{D}_{\text{test}},\hat{f}}\right)$$



MOTIVATING EXAMPLE / 3

- 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.
- Consider a simulation example of 3 ML algorithms below, where we use the dataset *mlbench.spiral* and 10,000 testing points. As can be seen, variating hyperparameters can lead to big difference in model's generalization performance.



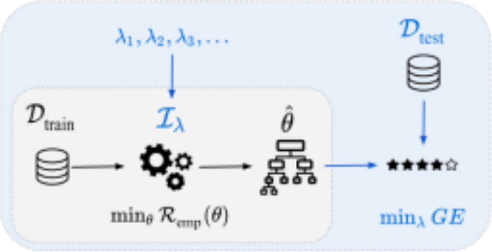
MOTIVATING EXAMPLE / 4

For our example this could mean:

- Data too complex to be modeled by a tree of depth 4
- Data much simpler than we thought, 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 θ are optimized during training. They are an **output** of the training.

Examples:

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

MODEL PARAMETERS VS. HYPERPARAMETERS / 2

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

The process of finding the best hyperparameters is called **tuning**.

Examples:

- Maximum depth of a tree
- k and which distance measure to use for k-NN
- Number and maximal order of interactions to be included in a linear regression model
- Number of optimization steps if the empirical risk minimization is done via gradient descent

MODEL PARAMETERS VS. HYPERPARAMETERS / 3

- Number of optimization steps if the empirical risk minimization is done via gradient descent.

We summarize all hyperparameters we want to tune in a vector $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ 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?