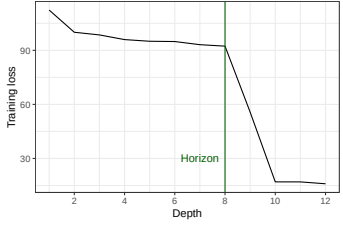
Introduction to Machine Learning

CART: Stopping Criteria & Pruning

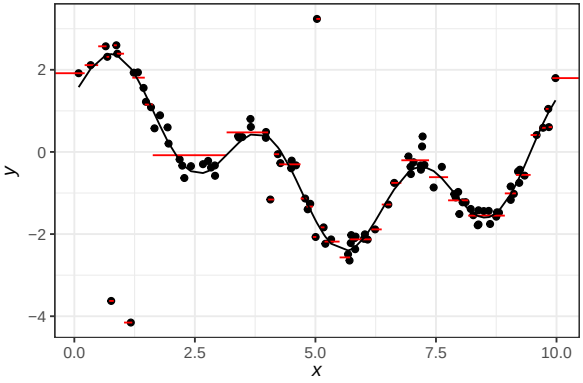


Learning goals

- Understand which problems arise when growing the tree until the end
- Know different stopping criteria
- Understand the idea of pruning

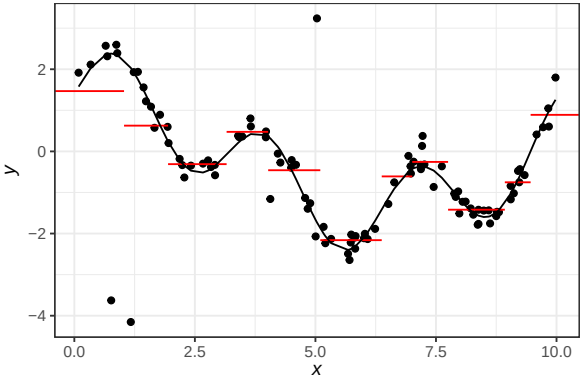
OVERFITTING TREES

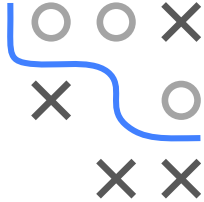
The recursive partitioning procedure used to grow a CART could run until every leaf only contains a single observation. Problem: Very complex trees will *overfit the training data.* At some point we should stop splitting nodes into ever smaller child nodes:



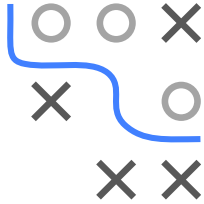
OVERFITTING TREES

We can reduce overfitting to some extent with a less deep tree:





STOPPING CRITERIA



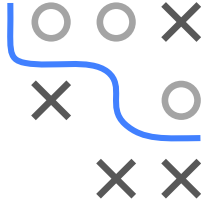
We can define different **stopping criteria**, e.g.: Don't split a node if

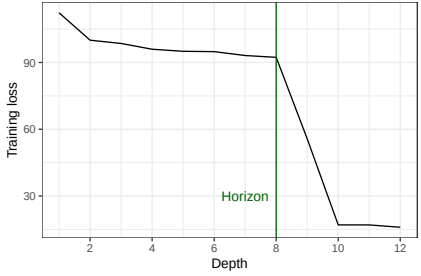
- a certain number of leaves if reached,
- it contains too few observations,
- splitting results in children with too few observations,
- splitting does not achieve a certain minimal improvement of the risk in the children, compared to the risk in the parent node,
- it already has the same target value (**pure node**) or identical feature values for all observations.

Selection of a stopping criterion and its concrete values are hyperparameters of CART.

HORIZON EFFECT

It is hard to tell where we should stop while we're growing the tree: Before we have actually tried all possible additional splits further down a branch, we can't know whether any one of them will be able to reduce the risk by a lot (*horizon effect*).

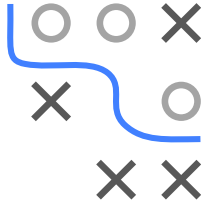




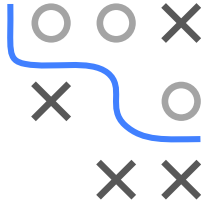
PRUNING

We try to tackle the horizon effect by **pruning**, a method to select the optimal size of a tree:

- Finding a combination of suitable strict stopping criteria (“pre-pruning”) is a hard problem (see chapter on **tuning**).
- Alternative: Grow a large tree, then remove branches so that the resulting smaller tree has lower risk (“post-pruning”).
- Often, post-pruning is meant when referring to pruning.



POST-PRUNING: CCP



- Prominent pruning method: Cost-complexity pruning (CCP)
- Idea: Grow a large tree and remove the least informative leaves
- CCP is steered with a regularization parameter α that penalizes the number of leaves in a sub tree

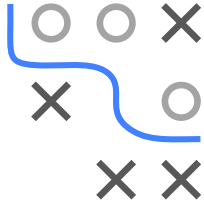
$$\mathcal{R}_{\text{reg}}(T) = \sum_{m=1}^{|T|} \sum_{i:x^{(i)} \in Q_m} L(y^{(i)}, c_m) + \alpha |T|,$$

where $|T|$ is the number of leaves of sub tree T, Q_m is the subset of the feature space related to the m-th terminal node, with its prediction c_m, and T_0 is the complete tree.

CCP

CCP performs a greedy backward search:

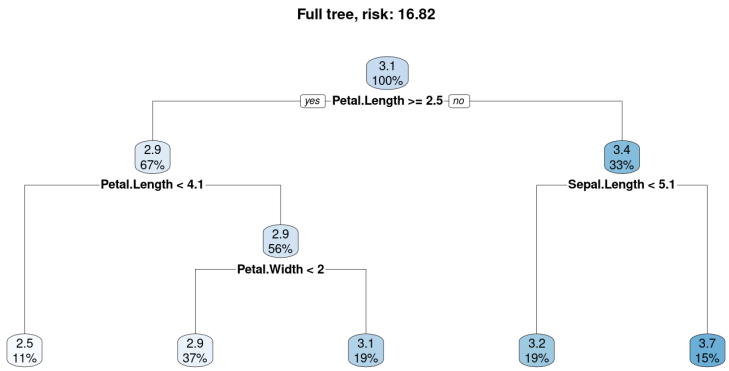
- Computes $\mathcal{R}_{\text{reg}}(T)$ with a fixed α for all possible sub trees that can be created by replacing one internal node with a leaf.
- By replacing a node we also eliminate all subsequent nodes.
- We select the sub tree with lowest risk and repeat the procedure.
- We stop if pruning does not further reduce the risk.

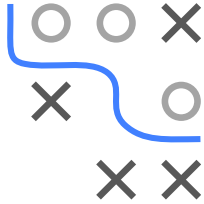


- This is proven to result in the pruned tree with the lowest risk.
- For $\alpha = 0$, we would obviously select T_0.
- Hyperparameter α is typically selected via cross-validation.
- Other prominent post-pruning methods include, e.g., reduced error pruning (REP) or pessimistic error pruning (PEP).

CCP

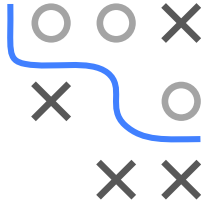
We run the CCP algorithm step-by-step with $\alpha = 1.2$:

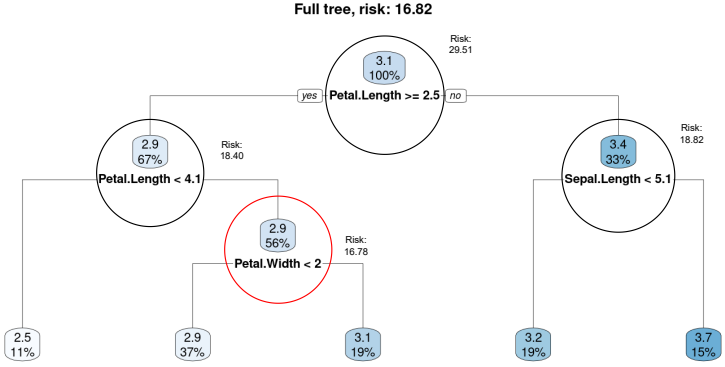




CCP

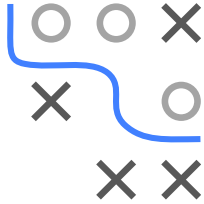
There are four possible nodes that we can eliminate to prune the tree.
We take the one replacement that results in the lowest risk (red).

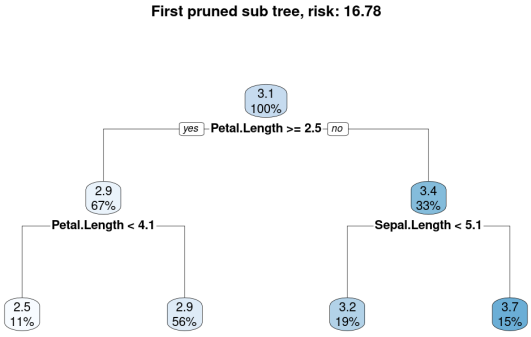




CCP

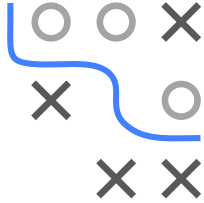
The first pruned sub tree has a lower risk than the full tree. Thus, we prefer it over the full tree.

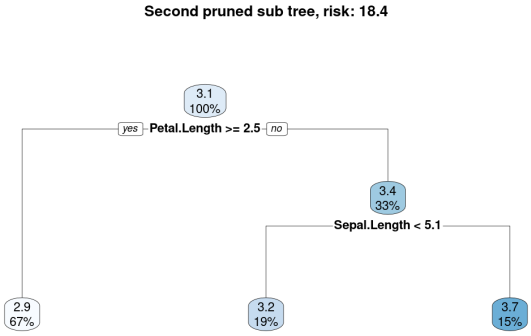




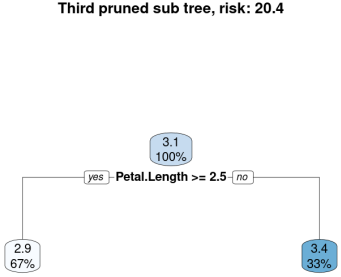
CCP

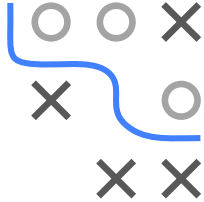
From here on, the risk increases.





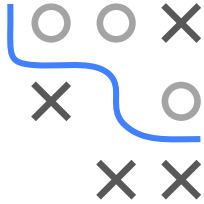
CCP


Third pruned sub tree, risk: 20.4
3.1
100%
yes
Petal.Length >= 2.5
no
2.9
67%
3.4
33%



CCP

We select the first sub tree as it results in the lowest risk in the complete sequence of sub trees.



Fully pruned sub tree, risk: 29.51