III. Decision Trees

- Decision Trees Basics
- Impurity Functions
- Decision Tree Algorithms
- Decision Tree Pruning
**Definition 10 (Overfitting)**

Let $D$ be a set of examples and let $H$ be a hypothesis space. The hypothesis $h \in H$ is considered to overfit $D$ if an $h' \in H$ with the following property exists:

$$\text{Err}(h, D) < \text{Err}(h', D) \quad \text{and} \quad \text{Err}^*(h) > \text{Err}^*(h'),$$

where $\text{Err}^*(h)$ denotes the true misclassification rate of $h$, while $\text{Err}(h, D)$ denotes the error of $h$ on the example set $D$. 


**Decision Tree Pruning**

**Overfitting**

**Definition 10 (Overfitting)**

Let \( D \) be a set of examples and let \( H \) be a hypothesis space. The hypothesis \( h \in H \) is considered to overfit \( D \) if an \( h' \in H \) with the following property exists:

\[
\text{Err}(h, D) < \text{Err}(h', D) \quad \text{and} \quad \text{Err}^*(h) > \text{Err}^*(h'),
\]

where \( \text{Err}^*(h) \) denotes the true misclassification rate of \( h \), while \( \text{Err}(h, D) \) denotes the error of \( h \) on the example set \( D \).

Reasons for overfitting are often rooted in the example set \( D \):

- \( D \) is noisy  and we “learn noise”
- \( D \) is biased  and hence non-representative
- \( D \) is too small  and hence pretends unrealistic data properties
Let $D_{tr} \subset D$ be the training set. Then $\text{Err}^*(h)$ can be estimated with a test set $D_{ts} \subset D$ where $D_{ts} \cap D_{tr} = \emptyset$ [holdout estimation]. The hypothesis $h \in H$ is considered to overfit $D$ if an $h' \in H$ with the following property exists:

$$\text{Err}(h, D_{tr}) < \text{Err}(h', D_{tr}) \quad \text{and} \quad \text{Err}(h, D_{ts}) > \text{Err}(h', D_{ts})$$
Let $D_{tr} \subset D$ be the training set. Then $Err^*(h)$ can be estimated with a test set $D_{ts} \subset D$ where $D_{ts} \cap D_{tr} = \emptyset$ [holdout estimation]. The hypothesis $h \in H$ is considered to overfit $D$ if an $h' \in H$ with the following property exists:

$$Err(h, D_{tr}) < Err(h', D_{tr}) \text{ and } Err(h, D_{ts}) > Err(h', D_{ts})$$
Remarks:

- Accuracy is the percentage of correctly classified examples.
- When does $\text{Err}(T, D_{tr})$ of a decision tree $T$ become zero?
- The training error $\text{Err}(T, D_{tr})$ of a decision tree $T$ is a monotonically decreasing function in the size of $T$. See the following Lemma.
Lemma 10

Let $t$ be a node in a decision tree $T$. Then, for each induced splitting $D(t_1), \ldots, D(t_s)$ of a set of examples $D(t)$ holds:

$$\text{Err}_{\text{cost}}(t, D(t)) \geq \sum_{i \in \{1, \ldots, s\}} \text{Err}_{\text{cost}}(t_i, D(t_i))$$

The equality is given in the case that all nodes $t, t_1, \ldots, t_s$ represent the same class.
**Proof (sketch)**

\[
\text{Err}_{\text{cost}}(t, D(t)) = \min_{c' \in C} \sum_{c \in C} p(c \mid t) \cdot p(t) \cdot \text{cost}(c' \mid c)
\]

\[
= \sum_{c \in C} p(c, t) \cdot \text{cost}(\text{label}(t) \mid c)
\]

\[
= \sum_{c \in C} (p(c, t_1) + \ldots + p(c, t_k)) \cdot \text{cost}(\text{label}(t) \mid c)
\]

\[
= \sum_{i \in \{1, \ldots, k_s\}} \sum_{c \in C} p(c, t_i) \cdot \text{cost}(\text{label}(t) \mid c)
\]

\[
\text{Err}_{\text{cost}}(t, D(t)) - \sum_{i \in \{1, \ldots, k_s\}} \text{Err}_{\text{cost}}(t_i, D(t_i)) = \left( \sum_{i \in \{1, \ldots, k_s\}} \left( \sum_{c \in C} p(c, t_i) \cdot \text{cost}(\text{label}(t) \mid c) - \min_{c' \in C} \sum_{c \in C} p(c, t_i) \cdot \text{cost}(c' \mid c) \right) \right)
\]

Observe that the summands on the right equation side are greater than or equal to zero.
Remarks:

- The lemma does also hold if the misclassification rate is used as performance measure.

- The algorithm template for the construction of decision trees, $DT$-construct, prefers larger trees, entailing a more fine-grained partitioning of $D$. A consequence of this behavior is a tendency to overfitting.
Decision Tree Pruning

Overfitting (continued)

Approaches to counter overfitting:

1. **Stopping** of the decision tree construction process during training.

2. **Pruning** of a decision tree after training:
   - Partitioning of $D$ into three sets for training, validation, and test:
     - reduced error pruning
     - minimal cost complexity pruning
     - rule post pruning
   - statistical tests such as $\chi^2$ to assess generalization capability
   - heuristic pruning
Decision Tree Pruning

Stopping

Possible criteria for stopping [splitting criteria] :

1. Size of $D(t)$.
   $D(t)$ will not be partitioned further if the number of examples, $|D(t)|$, is below a certain threshold.

2. Purity of $D(t)$.
   $D(t)$ will not be partitioned further if all induced splittings yield no significant impurity reduction $\Delta \iota$.

Problems:

ad 1) A threshold that is too small results in oversized decision trees.

ad 1) A threshold that is too large omits useful splittings.

ad 2) $\Delta \iota$ cannot be extrapolated with regard to the tree height.
Decision Tree Pruning

The pruning principle:

1. Construct a sufficiently large decision tree $T_{\text{max}}$.
2. Prune $T_{\text{max}}$, starting from the leaf nodes upwards the tree root.

Each leaf node $t$ of $T_{\text{max}}$ fulfills one or more of the following conditions:

- $D(t)$ is sufficiently small. Typically, $|D(t)| \leq 5$.
- $D(t)$ is comprised of examples of only one class.
- $D(t)$ is comprised of examples with identical feature vectors.
Decision Tree Pruning
Pruning (continued)

**Definition 11 (Decision Tree Pruning)**
Given a decision tree $T$ and an inner (non-root, non-leaf) node $t$. Then pruning of $T$ with regard to $t$ is the deletion of all successor nodes of $t$ in $T$. The pruned tree is denoted as $T \setminus T_t$. The node $t$ becomes a leaf node in $T \setminus T_t$.

Illustration:
Definition 12 (Pruning-Induced Ordering)
Let $T'$ and $T$ be two decision trees. Then $T' \preceq T$ denotes the fact that $T'$ is the result of a (possibly repeated) pruning applied to $T$. The relation $\preceq$ forms a partial ordering on the set of all trees.
Decision Tree Pruning

Pruning (continued)

Definition 12 (Pruning-Induced Ordering)
Let $T'$ and $T$ be two decision trees. Then $T' \preceq T$ denotes the fact that $T'$ is the result of a (possibly repeated) pruning applied to $T$. The relation $\preceq$ forms a partial ordering on the set of all trees.

Problems when assessing pruning candidates:

- Pruned decision trees may not stand in the $\preceq$-relation.
- Locally optimum pruning decisions may not result in the best candidates.
- Its monotonicity disqualifies $Err(T, D_{tr})$ as an estimator for $Err^*(T)$. [Lemma]
Pruning (continued)

**Definition 12 (Pruning-Induced Ordering)**

Let $T'$ and $T$ be two decision trees. Then $T' \preceq T$ denotes the fact that $T'$ is the result of a (possibly repeated) pruning applied to $T$. The relation $\preceq$ forms a partial ordering on the set of all trees.

Problems when assessing pruning candidates:

- Pruned decision trees may not stand in the $\preceq$-relation.
- Locally optimum pruning decisions may not result in the best candidates.
- Its monotonicity disqualifies $\text{Err}(T, D_{tr})$ as an estimator for $\text{Err}^*(T)$. [Lemma]

Control pruning with **validation set** $D_{vd}$, where $D_{vd} \cap D_{tr} = \emptyset$, $D_{vd} \cap D_{ts} = \emptyset$:

1. $D_{tr} \subset D$ for decision tree construction.
2. $D_{vd} \subset D$ for overfitting analysis during pruning.
3. $D_{ts} \subset D$ for decision tree evaluation after pruning.
Decision Tree Pruning
Pruning: Reduced Error Pruning

Basic principle of reduced error pruning:

1. \( T = T_{\text{max}} \)
2. Choose an inner node \( t \) in \( T \).
3. Perform a tentative pruning of \( T \) with regard to \( t \) : \( T' = T \setminus T_t \).
   Based on \( D(t) \) assign class to \( t \). [DT-construct]
4. If \( \text{Err}(T', D_{vd}) \leq \text{Err}(T, D_{vd}) \) then accept pruning: \( T = T' \).
5. Continue with Step 2 until all inner nodes of \( T \) are tested.
Decision Tree Pruning
Pruning: Reduced Error Pruning

Basic principle of reduced error pruning:

1. $T = T_{\text{max}}$

2. Choose an inner node $t$ in $T$.

3. Perform a tentative pruning of $T$ with regard to $t$: $T' = T \setminus T_t$.
   Based on $D(t)$ assign class to $t$. [DT-construct]

4. If $\text{Err}(T', D_{vd}) \leq \text{Err}(T, D_{vd})$ then accept pruning: $T = T'$.

5. Continue with Step 2 until all inner nodes of $T$ are tested.

Problem:

If $D$ is small, its partitioning into three sets for training, validation, and test will discard valuable information for decision tree construction.

Improvement: rule post pruning
Decision Tree Pruning

Pruning: Reduced Error Pruning (continued)

On training data $D_{tr}$
On validation data $D_{vd}$ (during pruning)
On test data $D_{ts}$

$T_{max}$

Accuracy vs. Size of tree (number of nodes) [Mitchell 1997]
Extensions

- consideration of the misclassification cost introduced by a splitting
- “surrogate splittings” for insufficiently covered feature domains
- splittings based on (linear) combinations of features
- regression trees