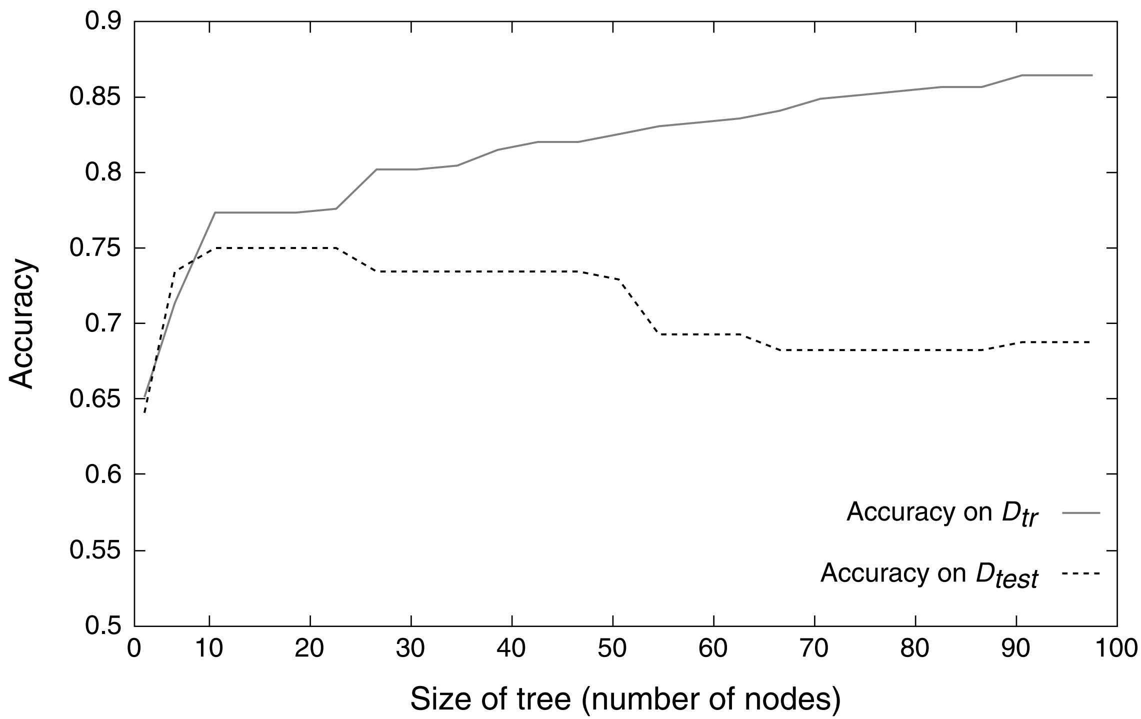
Chapter ML:VI

VI. Decision Trees

- Decision Trees Basics
- Impurity Functions
- Decision Tree Algorithms
- Decision Tree Pruning

Decision Tree Pruning

Overfitting

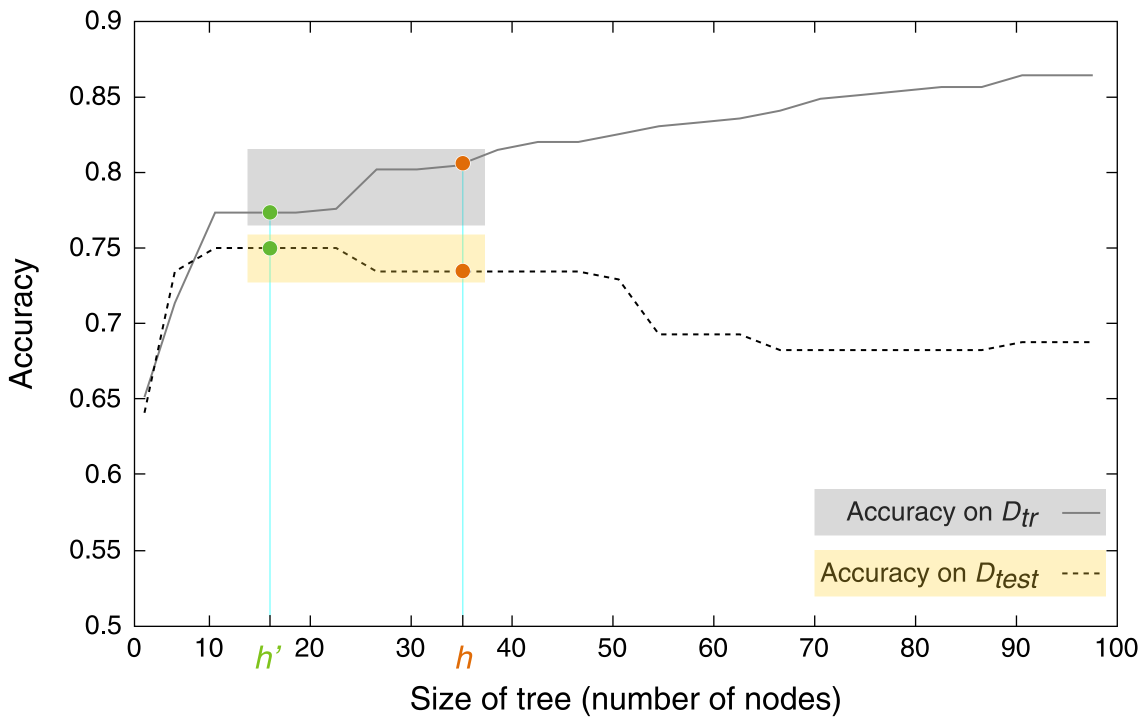


[Mitchell 1997]

Recall overfitting from section Overfitting in part Linear Models.

Decision Tree Pruning

Overfitting (continued)

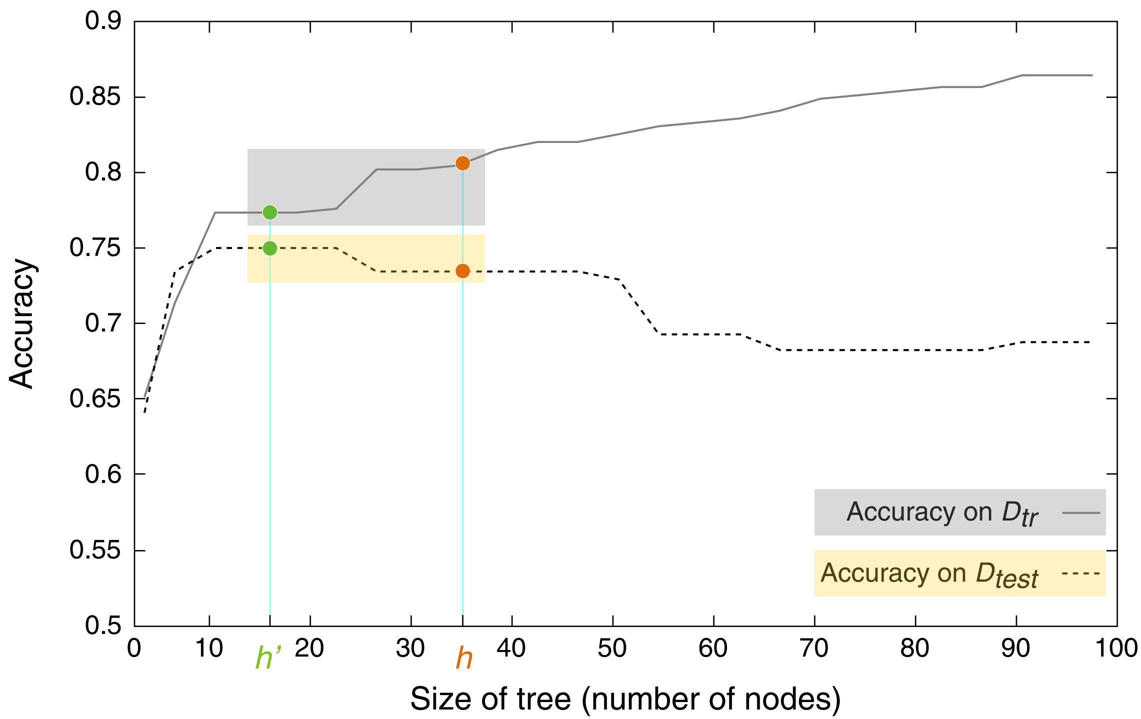


[Mitchell 1997]

Recall overfitting from section Overfitting in part Linear Models.

Decision Tree Pruning

Overfitting (continued)



[Mitchell 1997]

Recall overfitting from section Overfitting in part Linear Models. The hypothesis $h \in H$ is considered to overfit D if an $h' \in H$ with the following property exists:

- *Err*$(h, D) <$ *Err*(h', D) and *Err*$^*(h) >$ *Err*$^*(h')$ or, similarly:
- *Acc*$(h, D) >$ *Acc*(h', D) and *Acc*$^*(h) <$ *Acc*$^*(h')$

Remarks:

- The accuracy, *Acc*, is the percentage of correctly classified examples, i.e., $\mathit{Acc} = 1 - \mathit{Err}$.
- The holdout error of a hypothesis h, $\mathit{Err}(h, D_{\mathit{test}})$, is used as a proxy for the true error $\mathit{Err}^*(h)$.
- The training error $\mathit{Err}_{\mathit{tr}}(T)$ of a decision tree T is a monotonically decreasing function in the size of T. See the following Lemma.

Decision Tree Pruning

Overfitting (continued)

Lemma 10

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

$$\mathit{Err}(t, D(t)) \geq \sum_{i \in \{1,\ldots,m\}} \mathit{Err}(t_i, D(t_i))$$

The equality is given in the case that all nodes $t, t_1, \ldots, t_m$ represent the same class.

Decision Tree Pruning

Overfitting (continued)

Proof (sketch)

$$\begin{aligned}
\textit{Err}(t, D(t)) &= \min_{c' \in C} \sum_{c \in C} p(c \mid t) \cdot p(t) \cdot I_{\neq}(c', c) \\
&= \sum_{c \in C} p(c, t) \cdot I_{\neq}(\textit{label}(t), c) \\
&= \sum_{c \in C} (p(c, t_1) + \ldots + p(c, t_{k_m})) \cdot I_{\neq}(\textit{label}(t), c) \\
&= \sum_{i \in \{1,\ldots,k_m\}} \sum_{c \in C} (p(c, t_i) \cdot I_{\neq}(\textit{label}(t), c))
\end{aligned}$$

$$\textit{Err}(t, D(t)) - \sum_{i \in \{1,\ldots,k_m\}} \textit{Err}(t_i, D(t_i)) =$$

$$\sum_{i \in \{1,\ldots,k_m\}} \left(\sum_{c \in C} p(c, t_i) \cdot I_{\neq}(\textit{label}(t), c) \; - \min_{c' \in C} \sum_{c \in C} p(c, t_i) \cdot I_{\neq}(c', c) \right)$$

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

Remarks:

- The lemma does also hold if a function for misclassification cost is used to assess effectiveness.
- The algorithm template for the construction of decision trees, *DT-construct*, prefers larger trees, entailing a more fine-grained splitting of D. A consequence of this behavior is a tendency to overfitting.
- $I_{\neq}$ is an indicator function that returns 1 if its arguments are *unequal* (and 0 if its arguments are equal).

Decision Tree Pruning

Overfitting (continued)

Approaches to counter overfitting:

(a) Stopping of the decision tree construction process during training.

(b) Pruning of a decision tree after training:

- Splitting of D into three sets for training, validation, and test:
  - reduced error pruning
  - minimal cost complexity pruning
  - rule post pruning
- statistical tests such as χ^2 to assess generalization capability
- heuristic pruning

Decision Tree Pruning

(a) Stopping

Possible criteria for stopping [splitting criria] :

1. Size of $D(t)$.
   $D(t)$ is not split if $|D(t)|$ is below a threshold.

2. Purity of $D(t)$.
   $D(t)$ is not split if all examples in $D(t)$ are members of the same class.

3. Impurity reduction of $D(t)$.
   $D(t)$ is not split if the resulting impurity reduction, $\Delta\iota$, is below a threshold.

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) Perfect purity cannot be expected with noisy data.

ad 3) $\Delta\iota$ cannot be extrapolated with regard to the tree height.

Decision Tree Pruning

(b) Pruning

The pruning principle:

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

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

- $D(t)$ is sufficiently small. Typically, $|D(t)| \leq 5$.
- $D(t)$ is pure.
- $D(t)$ is comprised of examples with identical feature vectors.

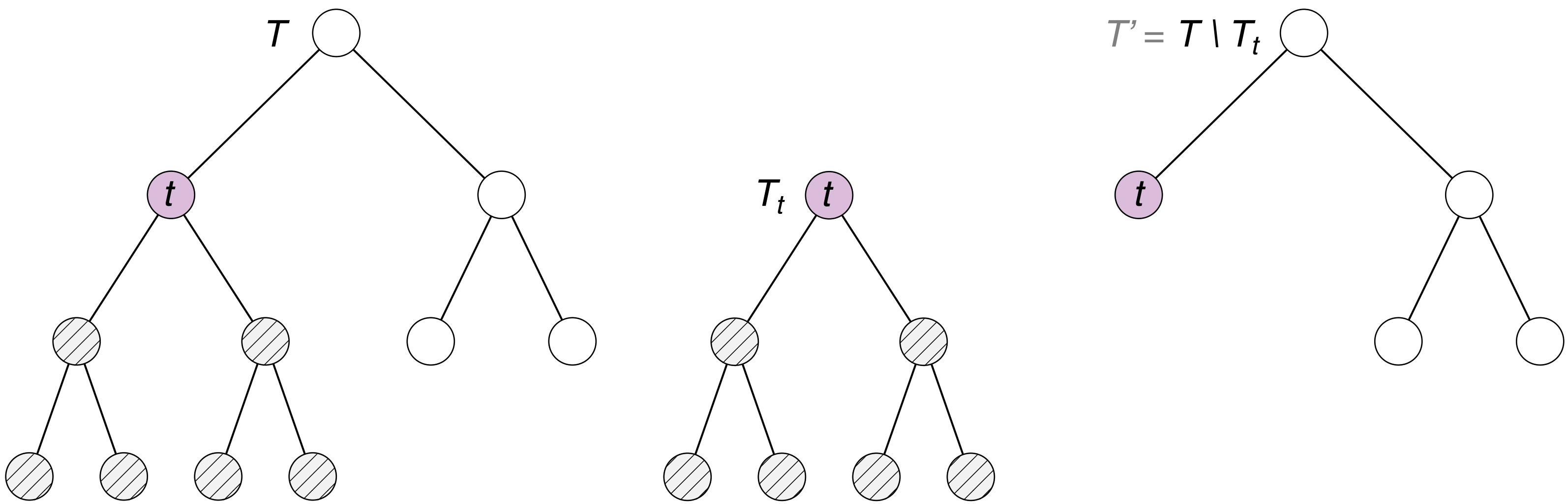
Decision Tree Pruning

(b) 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:



Decision Tree Pruning

(b) 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.

Decision Tree Pruning

(b) 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 $Err_{tr}(T)$ as an estimator for $Err^*(T)$. [Lemma]

Decision Tree Pruning

(b) 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 $Err_{tr}(T)$ as an estimator for $Err^*(T)$. [Lemma]

Control pruning with a validation set D_{val} :

1. $D_{test} \subset D$, test set for decision tree assessment after pruning.
2. $D_{val} \subset (D \setminus D_{test})$, validation set for overfitting analysis during pruning.
3. $D_{tr} = D \setminus (D_{test} \cup D_{val})$, training set for decision tree construction.

Decision Tree Pruning

(b) Pruning: Reduced Error Pruning

Steps 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 *Err*$(T', D_{\mathit{val}}) \leq$ *Err*(T, D_{val}) then accept pruning: $T = T'$.
5. Continue with Step 2 until all inner nodes of T are tested.

Decision Tree Pruning

(b) Pruning: Reduced Error Pruning (continued)

Steps 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 *Err*$(T', D_{\textit{val}}) \leq$ *Err*$(T, D_{\textit{val}})$ then accept pruning: $T = T'$.
5. Continue with Step 2 until all inner nodes of T are tested.

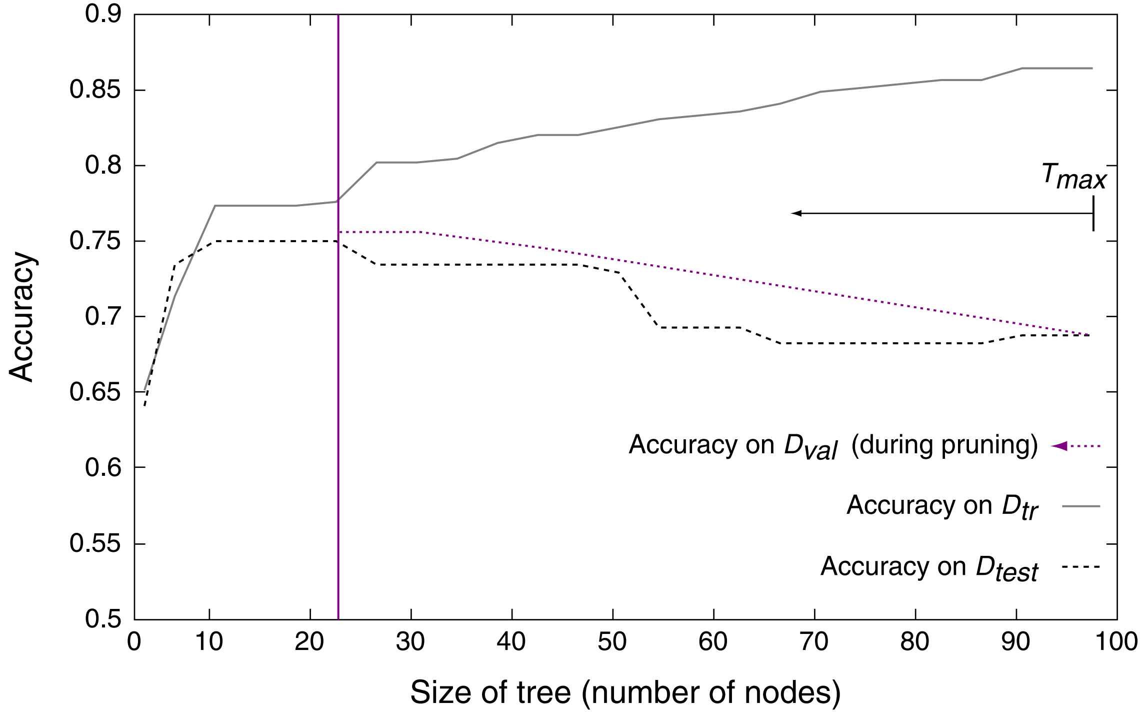
Problem:

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

(b) Pruning: Reduced Error Pruning (continued)



[Mitchell 1997]

Remarks (pruning extensions) :

- pruning considering misclassification cost
- weakest link pruning

Remarks (splitting extensions) :

- splitting considering misclassification cost
- “surrogate splittings” for insufficiently covered feature domains
- splittings based on (linear) combinations of features

Remarks (generic extensions) :

- discrete features with many values
- features of different importance
- features with missing values
- regression trees