Chapter ML:VI

VI. Decision Trees

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

Decision Tree Algorithms

ID3 Algorithm [Quinlan 1986] [CART Algorithm]

Setting:

- X is a multiset of feature vectors.
- C is a set of classes.
- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

- Fit D using a decision tree T.

Decision Tree Algorithms

ID3 Algorithm [Quinlan 1986] [CART Algorithm]

Setting:

- X is a multiset of feature vectors.
- C is a set of classes.
- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

- Fit D using a decision tree T.

Characteristics of the ID3 algorithm:

1. Each splitting is based on one nominal feature and considers its complete domain. Splitting based on feature A with domain *dom*$(A) = \{a_1, \ldots, a_m\}$:

   $$X = \{\mathbf{x} \in X : \mathbf{x}|_A = a_1\} \cup \ldots \cup \{\mathbf{x} \in X : \mathbf{x}|_A = a_m\}$$

2. Splitting criterion is information gain.

Decision Tree Algorithms

ID3 Algorithm (continued) [Mitchell 1997 version] [algorithm template]

ID3(D, Features)

1. Create a node t for the tree.
2. Label t with the most common class in D.
3. If all examples in D have the same class, return the single-node tree t.
4. If Features is empty, return the single-node tree t.

Decision Tree Algorithms

ID3 Algorithm (continued) [Mitchell 1997 version] [algorithm template]

ID3(D, Features)

1. Create a node t for the tree.
2. Label t with the most common class in D.
3. If all examples in D have the same class, return the single-node tree t.
4. If Features is empty, return the single-node tree t.

   Otherwise:

   5. Let A* be the feature from Features that best classifies examples in D.

      Assign t the decision feature A*.

   6. For each possible value "a" in dom(A*) do:
      - Add a new tree branch below t, corresponding to the test A* = "a".
      - Let D_a be the subset of D that has value "a" for A*.
      - If D_a is empty:
Then add a leaf node with the label of the most common class in D.
Else add the subtree ID3(D_a, Features $\setminus$ {A*}).

7. Return t.

Decision Tree Algorithms

ID3 Algorithm (continued) [algorithm template]

ID3$(D,$ *Features*$)$

1. $t =$ *createNode*$()$
2. *label*$(t) =$ *mostCommonClass*(D)
3. `IF` $\forall(\mathbf{x}, c) \in D : c =$ *label*(t) `THEN` *return*(t) `ENDIF` // D is pure.
4. `IF` *Features* $= \emptyset$ `THEN` *return*(t) `ENDIF` // We are running out of features.
5.
6.
7.

Decision Tree Algorithms

ID3 Algorithm (continued) [algorithm template]

ID3$(D, \textit{Features})$

1. $t = \textit{createNode}()$
2. $\textit{label}(t) = \textit{mostCommonClass}(D)$
3. `IF` $\forall(\mathbf{x}, c) \in D : c = \textit{label}(t)$ `THEN` $\textit{return}(t)$ `ENDIF` `//` D `is pure.`
4. `IF` $\textit{Features} = \emptyset$ `THEN` $\textit{return}(t)$ `ENDIF` `// We are running out of features.`
5. $A^* = \text{argmax}_{A \in \textit{Features}}(\textit{informationGain}(D, A))$
6.
7.

Decision Tree Algorithms

ID3 Algorithm (continued) [algorithm template]

ID3$(D,$ *Features*$)$

1. $t =$ *createNode*$()$
2. *label*$(t) =$ *mostCommonClass*(D)

3. IF $\forall(\mathbf{x}, c) \in D : c =$ *label*(t) THEN *return*(t) ENDIF // D is pure.
4. IF *Features* $= \emptyset$ THEN *return*(t) ENDIF // We are running out of features.

5. $A^* = \text{argmax}_{A \in \textit{Features}}(\textit{informationGain}(D, A))$

6. **FOREACH** $a \in$ *dom*(A^*) **DO**

   $D_a = \{(\mathbf{x}, c) \in D : \mathbf{x}|_{A^*} = a\}$

   IF $D_a = \emptyset$ THEN

   ELSE

   createEdge$(t, a,$ *ID3*$(D_a,$ *Features*$\setminus\{A^*\}))$

   ENDIF

   ENDDO

7. *return*(t)

Decision Tree Algorithms

ID3 Algorithm (continued) [algorithm template]

ID3$(D,$ *Features*$)$

1. $t =$ *createNode*$()$
2. *label*$(t) =$ *mostCommonClass*(D)
3. IF $\forall(\mathbf{x}, c) \in D : c =$ *label*(t) THEN *return*(t) ENDIF // D is pure.
4. IF *Features* $= \emptyset$ THEN *return*(t) ENDIF // We are running out of features.
5. $A^* = \text{argmax}_{A \in \textit{Features}}(\textit{informationGain}(D, A))$
6. **FOREACH** $a \in$ *dom*(A^*) **DO**

   $D_a = \{(\mathbf{x}, c) \in D : \mathbf{x}|_{A^*} = a\}$

   IF $D_a = \emptyset$ THEN // We are running out of data.

   $t' =$ *createNode*$()$
   label$(t') =$ *label*(t)
   createEdge(t, a, t')

   ELSE

   createEdge$(t, a,$ *ID3*$(D_a,$ *Features*$\setminus\{A^*\}))$

   ENDIF

   ENDDO
7. *return*(t)

Remarks:

- Step 3 of of the ID3 algorithm checks the purity of D and, given this case, assigns the unique class to the respective node.
- The ID3 (Iterative Dichotomiser 3) was published by Ross Quinlan in 1986.

Decision Tree Algorithms

ID3 Algorithm: Example

Example set D for mushrooms, drawn from a set of feature vectors X over the three dimensions color, size, and points:

	Color	Size	Points	Edibility
1	red	small	yes	toxic
2	brown	small	no	edible
3	brown	large	yes	edible
4	green	small	no	edible
5	red	large	no	edible

Decision Tree Algorithms

ID3 Algorithm: Example (continued)

Top-level call of ID3. Analyze a splitting with regard to the feature "color" :

$D|_{\text{color}} =$

	toxic	edible
red	1	1
brown	0	2
green	0	1

$\leadsto \quad |D_{\text{red}}| = 2, \quad |D_{\text{brown}}| = 2, \quad |D_{\text{green}}| = 1$

Decision Tree Algorithms

ID3 Algorithm: Example (continued)

Top-level call of ID3. Analyze a splitting with regard to the feature "color" :

$D|_{\text{color}} =$

	toxic	edible
red	1	1
brown	0	2
green	0	1

$\rightsquigarrow \quad |D_{\text{red}}| = 2, \quad |D_{\text{brown}}| = 2, \quad |D_{\text{green}}| = 1$

Estimated prior probabilities:

$$\hat{P}(\textit{Color}{=}\text{red}) = \frac{2}{5} = 0.4, \quad \hat{P}(\textit{Color}{=}\text{brown}) = \frac{2}{5} = 0.4, \quad \hat{P}(\textit{Color}{=}\text{green}) = \frac{1}{5} = 0.2$$

Decision Tree Algorithms

ID3 Algorithm: Example (continued)

Top-level call of ID3. Analyze a splitting with regard to the feature “color” :

$D|_{\text{color}} =$

	toxic	edible
red	1	1
brown	0	2
green	0	1

$\rightsquigarrow \quad |D_{\text{red}}| = 2, \quad |D_{\text{brown}}| = 2, \quad |D_{\text{green}}| = 1$

Estimated prior probabilities:

$$\hat{P}(\mathit{Color}{=}\text{red}) = \frac{2}{5} = 0.4, \quad \hat{P}(\mathit{Color}{=}\text{brown}) = \frac{2}{5} = 0.4, \quad \hat{P}(\mathit{Color}{=}\text{green}) = \frac{1}{5} = 0.2$$

Conditional entropy:

$$\begin{aligned} H(\mathcal{A} \mid \mathcal{B}_1) &= H(\ \{A_1, A_2\} \mid \{B_{1,1}, B_{1,2}, B_{1,3}\}\ ) \\ &= H(\ \{C{=}\text{toxic}, C{=}\text{edible}\} \mid \{\mathit{Color}{=}\text{red}, \mathit{Color}{=}\text{brown}, \mathit{Color}{=}\text{green}\}\ ) \\ &= -(0.4{\cdot}(\tfrac{1}{2}{\cdot} \log_2 \tfrac{1}{2} + \tfrac{1}{2}{\cdot} \log_2 \tfrac{1}{2}) + 0.4{\cdot}(\tfrac{0}{2}{\cdot} \log_2 \tfrac{0}{2} + \tfrac{2}{2}{\cdot} \log_2 \tfrac{2}{2}) + 0.2{\cdot}(\tfrac{0}{1}{\cdot} \log_2 \tfrac{0}{1} + \tfrac{1}{1}{\cdot} \log_2 \tfrac{1}{1})) = \ 0.4 \end{aligned}$$

Decision Tree Algorithms

ID3 Algorithm: Example (continued)

Top-level call of ID3. Analyze a splitting with regard to the feature “color” :

$D|_{\text{color}}$ =

	toxic	edible
red	1	1
brown	0	2
green	0	1

$\rightsquigarrow \quad |D_{\text{red}}| = 2, \quad |D_{\text{brown}}| = 2, \quad |D_{\text{green}}| = 1$

Estimated prior probabilities:

$$\hat{P}(\textit{Color}{=}\text{red}) = \frac{2}{5} = 0.4, \quad \hat{P}(\textit{Color}{=}\text{brown}) = \frac{2}{5} = 0.4, \quad \hat{P}(\textit{Color}{=}\text{green}) = \frac{1}{5} = 0.2$$

Conditional entropy:

$$\begin{aligned} H(\mathcal{A} \mid \mathcal{B}_1) &= H(\ \{A_1, A_2\} \mid \{B_{1,1}, B_{1,2}, B_{1,3}\}\ ) \\ &= H(\ \{C{=}\text{toxic}, C{=}\text{edible}\} \mid \{\textit{Color}{=}\text{red}, \textit{Color}{=}\text{brown}, \textit{Color}{=}\text{green}\}\ ) \\ &= -(0.4{\cdot}(\tfrac{1}{2}{\cdot} \log_2 \tfrac{1}{2} + \tfrac{1}{2}{\cdot} \log_2 \tfrac{1}{2}) + 0.4{\cdot}(\tfrac{0}{2}{\cdot} \log_2 \tfrac{0}{2} + \tfrac{2}{2}{\cdot} \log_2 \tfrac{2}{2}) + 0.2{\cdot}(\tfrac{0}{1}{\cdot} \log_2 \tfrac{0}{1} + \tfrac{1}{1}{\cdot} \log_2 \tfrac{1}{1})) = \ 0.4 \end{aligned}$$

$$H(\mathcal{A} \mid \mathcal{B}_2) = H(\ \{C{=}\text{toxic}, C{=}\text{edible}\} \mid \{\textit{Size}{=}\text{small}, \textit{Size}{=}\text{large}\}\ ) = \ \ldots \ \approx \ 0.55$$

$$H(\mathcal{A} \mid \mathcal{B}_3) = H(\ \{C{=}\text{toxic}, C{=}\text{edible}\} \mid \{\textit{Points}{=}\text{yes}, \textit{Points}{=}\text{no}\}\ ) = \ \ldots \ = \ 0.4$$

Remarks:

- The smaller $H(\mathcal{A} \mid \mathcal{B})$ is, the larger becomes the information gain. Hence, the difference $H(\mathcal{A}) - H(\mathcal{A} \mid \mathcal{B})$ needs not to be computed since $H(\mathcal{A})$ is constant within each recursion step.
- In the example, the information gain in the first recursion step becomes maximum for the features "color" and "points".
- Notation. When used in the role of a random variable (here: in the argument of a probability P), features are written in italics and capitalized.
- Notation. The probabilities, denoted as $P(\,\cdot\,)$, are unknown and estimated by the relative frequencies, denoted as $\hat{P}(\,\cdot\,)$.

ID3 Algorithm: Example (continued)

Decision tree before the first recursion step:

feature: points

yes

Color	Size	Edibility
red	small	toxic
brown	large	edible

no

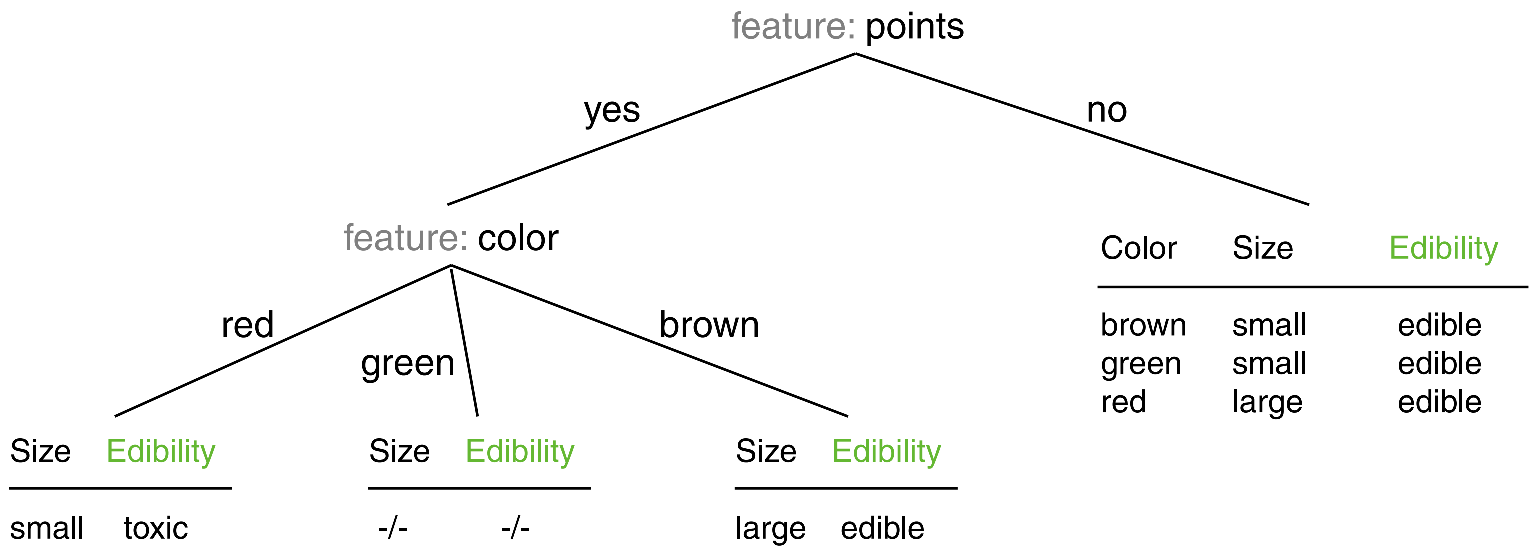
Color	Size	Edibility
brown	small	edible
green	small	edible
red	large	edible

Choosing the feature "points" in Step 5 of the ID3 algorithm.

Decision Tree Algorithms

ID3 Algorithm: Example (continued)

Decision tree before the second recursion step:

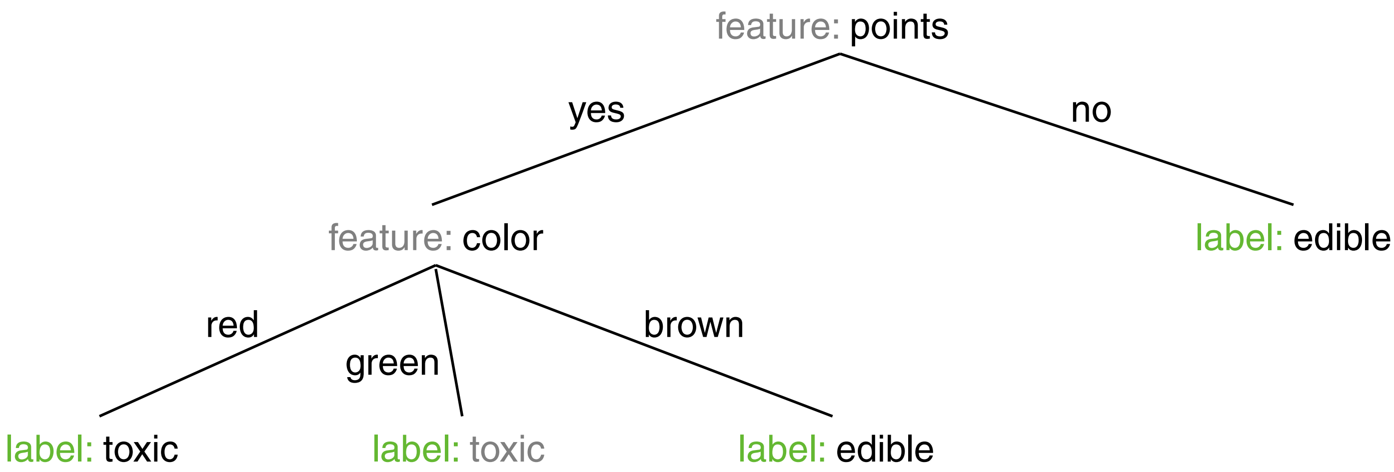


Choosing the feature “color” in Step 5 of the ID3 algorithm.

Decision Tree Algorithms

ID3 Algorithm: Example (continued)

Final decision tree after second recursion step:

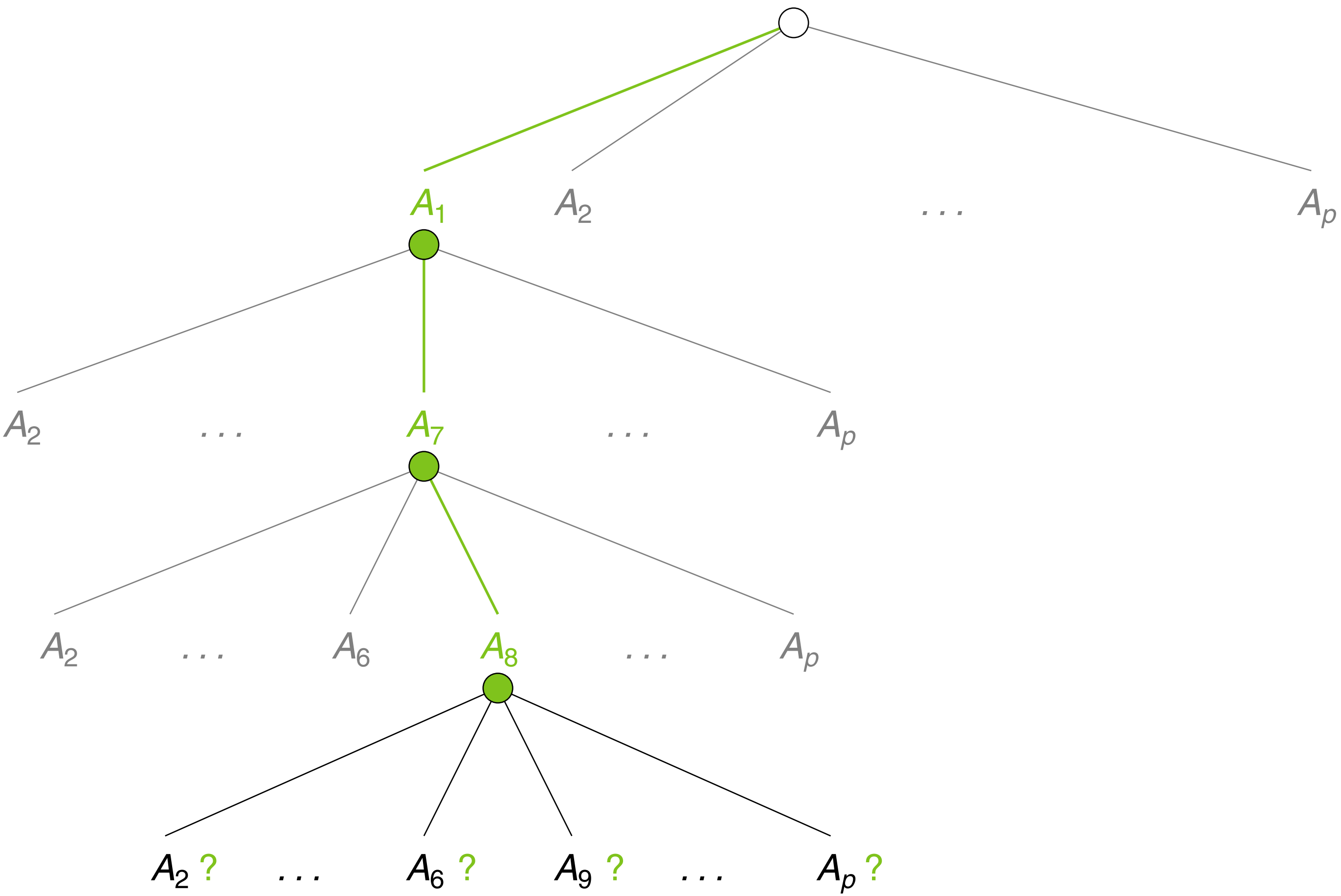


Break of a tie: choosing the class "toxic" for D_{green} in Step 6 of the ID3 algorithm.

Decision Tree Algorithms

ID3 Algorithm: Search Space

Features $= \{A_1, A_2, \ldots, A_p\}$



Remarks (search space versus hypothesis space) :

- The underlying search space of an algorithm that samples without replacement a single feature in each step (= monothetic splitting) consists of all permutations of the features in the feature set. In particular, if the number of features (= dimensionality of a feature vector $\mathbf{x}$) is p, then the search space contains $p!$ elements.

- The set of possible decision trees over D forms the hypothesis space H. The maximum size of H, i.e., the maximum number of decision trees for a data set D in a binary classification setting, is $2^{|D|}$: If the feature vectors are pairwise distinct, every subset of D can form a class while the complement of the subset will form the other class. The set of possible subsets of D is $\mathcal{P}(D)$, where $|\mathcal{P}(D)| = 2^{|D|}$.

- Observe that either $p! < 2^{|D|}$ or $p! > 2^{|D|}$ can hold. I.e., the search space due to feature ordering can be smaller or larger than its underlying hypothesis space. The former characterizes the typical situation; also note that both the search space and the hypothesis space grow exponentially in the number of features and examples respectively.

- The difference between search space size and hypothesis space size results from Step 6 of the ID3 algorithm: the same feature selection order will lead to different decision trees when given different data sets. However, since the splitting operation in Step 6 is deterministic it has no effect on the search space.

- The runtime of the ID3 algorithm is in $O(p^2 \cdot n)$, i.e., significantly below $p!$ since only a small part of the search space is explored. At each split, the algorithm greedily (in fact, irrevocably) selects the most informative feature by applying information gain as a heuristic for feature selection.

Decision Tree Algorithms

ID3 Algorithm: Inductive Bias

Inductive bias is the rigidity in applying the (little bit of) knowledge learned from a training set for the classification of unseen feature vectors.

Observations:

- Decision tree search happens in the space of all hypotheses.

- To generate a decision tree, the ID3 algorithm needs per branch at most as many decisions as features are given.

ID3 Algorithm: Inductive Bias

Inductive bias is the rigidity in applying the (little bit of) knowledge learned from a training set for the classification of unseen feature vectors.

Observations:

- Decision tree search happens in the space of all hypotheses.
  - ➜ The target concept is a member of the hypothesis space.
- To generate a decision tree, the ID3 algorithm needs per branch at most as many decisions as features are given.
  - ➜ no backtracking takes place
  - ➜ the decision tree is a result of *local optimization*

Decision Tree Algorithms

ID3 Algorithm: Inductive Bias

Inductive bias is the rigidity in applying the (little bit of) knowledge learned from a training set for the classification of unseen feature vectors.

Observations:

- Decision tree search happens in the space of all hypotheses.
  - ➜ The target concept is a member of the hypothesis space.
- To generate a decision tree, the ID3 algorithm needs per branch at most as many decisions as features are given.
  - ➜ no backtracking takes place
  - ➜ the decision tree is a result of *local optimization*

Where the inductive bias of the ID3 algorithm becomes manifest:

1. Small decision trees are preferred.
2. Highly discriminative features tend to be closer to the root.

Is this justified?

Remarks (inductive bias) :

- The inductive bias of the ID3 algorithm is of a different kind than the inductive bias of the candidate elimination algorithm:

  1. The underlying hypothesis space H of the candidate elimination algorithm is incomplete. H corresponds to a coarsened view onto the space of all hypotheses since H contains only conjunctions of feature-value pairs as hypotheses.

     However, this restricted hypothesis space is searched completely by the candidate elimination algorithm. Keyword: restriction bias

  2. The underlying hypothesis space H of the ID3 algorithm is complete since it contains all decision trees that can be constructed over D.

     However, this complete hypothesis space is searched incompletely, but following a preference. Keyword: preference bias or search bias

- The inductive bias of the ID3 algorithm renders the algorithm robust wrt. noise.

Decision Tree Algorithms

CART Algorithm [Breiman 1984] [ID3 Algorithm]

Setting:

- X is a multiset of feature vectors. No restrictions are presumed for the features' measurement scales.
- C is a set of classes.
- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

- Fit D using a decision tree T.

Decision Tree Algorithms

CART Algorithm [Breiman 1984] [ID3 Algorithm]

Setting:

- X is a multiset of feature vectors. No restrictions are presumed for the features' measurement scales.
- C is a set of classes.
- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

- Fit D using a decision tree T.

Characteristics of the CART algorithm:

1. Each splitting is binary and considers one feature at a time.
2. Splitting criterion is the information gain or the Gini index.

Decision Tree Algorithms

CART Algorithm (continued)

Let A be a feature with domain *dom*(A). Apply (probably multiple times) the respective rule to induce a finite number of binary splittings of X :

R1: If A is nominal, choose $B \subset$ *dom*(A) such that $0 < |B| \leq |$*dom*$(A) \setminus B|$.

R2: If A is ordinal, choose $a \in$ *dom*(A) such that $x_{\min} < a < x_{\max}$, where $x_{\min}$, $x_{\max}$ are the minimum and maximum values of feature A in D.

R3: If A is numeric, choose $a \in$ *dom*(A) such that $a = 0.5 \cdot (x_{l_1} + x_{l_2})$, where x_{l_1}, x_{l_2} are consecutive elements in the ordered value list of feature A in D.

Decision Tree Algorithms

CART Algorithm (continued)

Let A be a feature with domain *dom*(A). Apply (probably multiple times) the respective rule to induce a finite number of binary splittings of X :

R1: If A is nominal, choose $B \subset$ *dom*(A) such that $0 < |B| \leq |$*dom*$(A) \setminus B|$.

R2: If A is ordinal, choose $a \in$ *dom*(A) such that $x_{\min} < a < x_{\max}$, where $x_{\min}, x_{\max}$ are the minimum and maximum values of feature A in D.

R3: If A is numeric, choose $a \in$ *dom*(A) such that $a = 0.5 \cdot (x_{l_1} + x_{l_2})$, where x_{l_1}, x_{l_2} are consecutive elements in the ordered value list of feature A in D.

Adapt Step 5 and 6 to turn the ID3 algorithm into the CART algorithm:

→ 5. For all $A \in$ *Features* : Generate with the above rules all splittings of $D(t)$.
Choose a splitting that maximizes the impurity reduction $\Delta\iota$:

$$\Delta\iota\big(D(t),\ \{D(t_L), D(t_R)\}\big) \ = \ \iota(D(t)) - \frac{|D(t_L)|}{|D|} \cdot \iota(D(t_L)) - \frac{|D(t_R)|}{|D|} \cdot \iota(D(t_R))$$

→ 6. Recursively call CART to process $D(t_L)$ and $D(t_R)$.

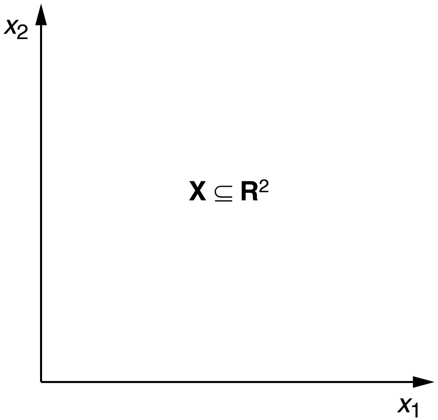
Remarks:

- t_L and t_R denote the left and right successor of t in the decision tree. These nodes are returned by the calls of the CART algorithm and connected to t via *createEdge*$()$.
- Since the CART algorithm creates binary splittings only, the feature A^* chosen in Step 5 can be chosen again later on. Hence, a call of CART to process $D(t_L)$ (or $D(t_R)$) in Step 6 passes the complete set of features as second parameter (and not: *Features*$\setminus\{A^*\}$).

Decision Tree Algorithms

CART Algorithm (continued)

Illustration for two numeric features; i.e., the feature space $\mathbf{X}$ underlying X corresponds to a two-dimensional plane such as the $\mathbf{R}^2$:



Decision Tree Algorithms

CART Algorithm (continued)

Illustration for two numeric features; i.e., the feature space $\mathbf{X}$ underlying X corresponds to a two-dimensional plane such as the $\mathbf{R}^2$:

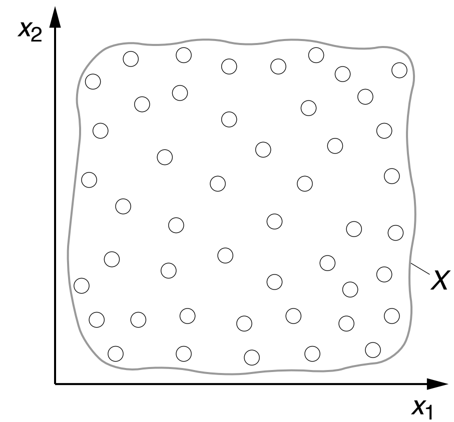
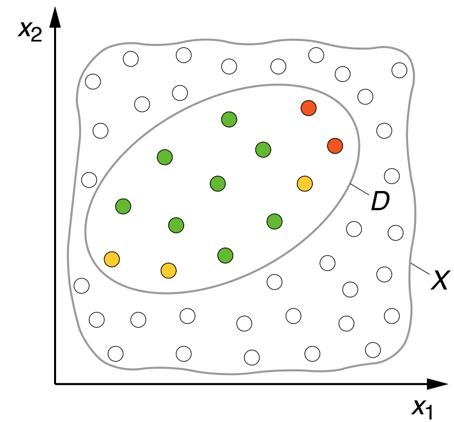


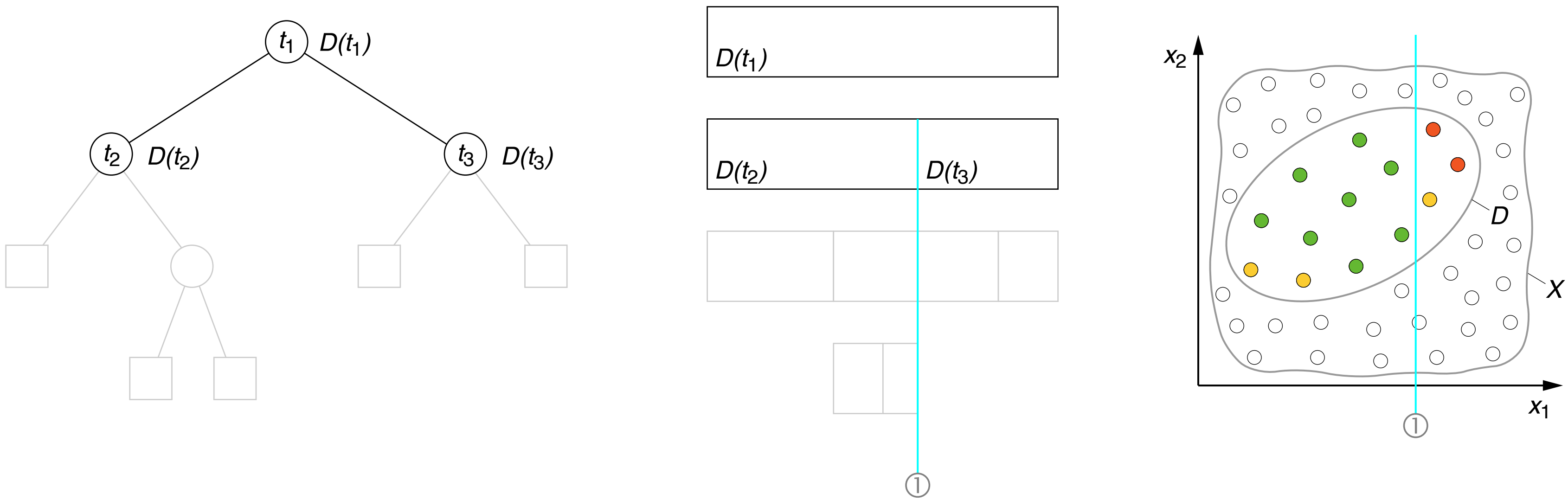
Illustration for two numeric features; i.e., the feature space $\mathbf{X}$ underlying X corresponds to a two-dimensional plane such as the $\mathbf{R}^2$:



Decision Tree Algorithms

CART Algorithm (continued)

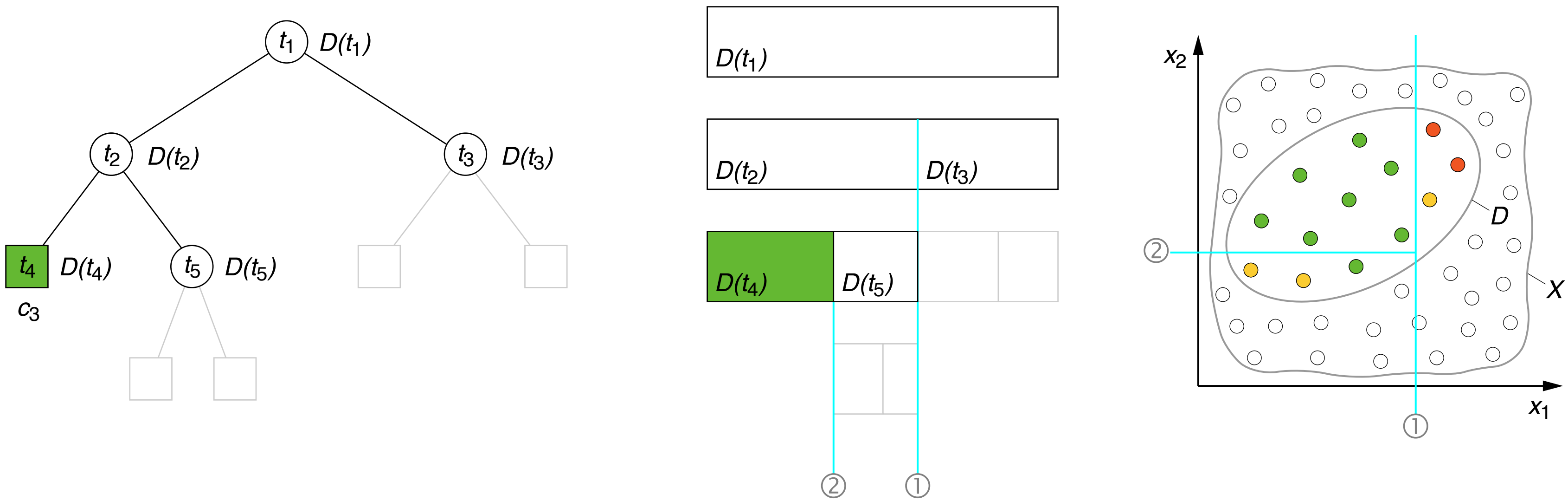
Illustration for two numeric features; i.e., the feature space $\mathbf{X}$ underlying X corresponds to a two-dimensional plane such as the $\mathbf{R}^2$:



Decision Tree Algorithms

CART Algorithm (continued)

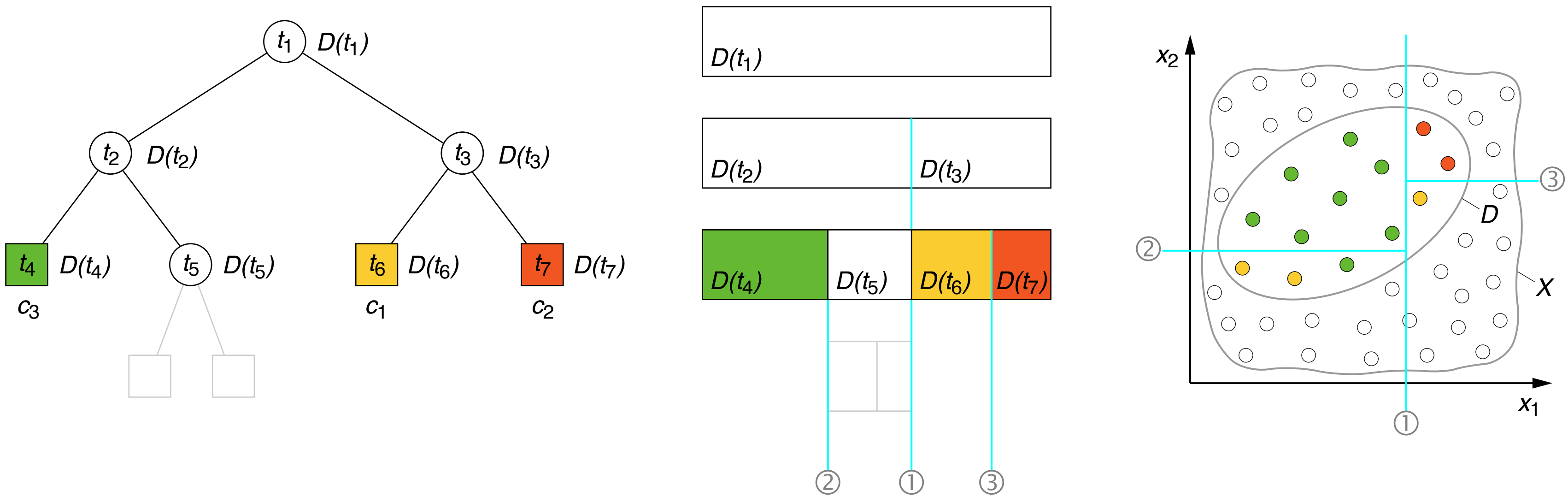
Illustration for two numeric features; i.e., the feature space $\mathbf{X}$ underlying X corresponds to a two-dimensional plane such as the $\mathbf{R}^2$:



Decision Tree Algorithms

CART Algorithm (continued)

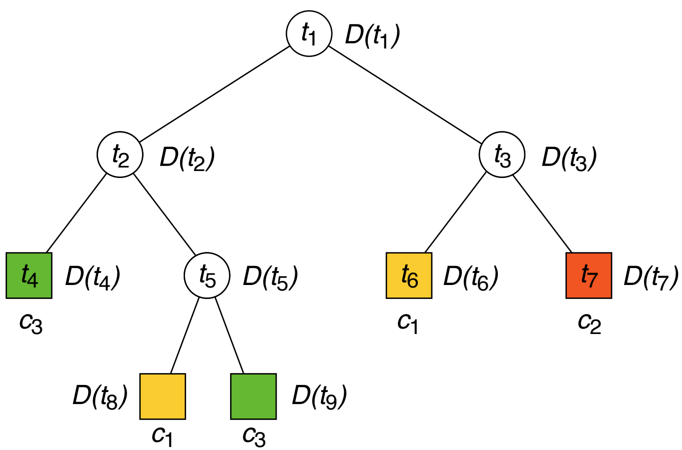
Illustration for two numeric features; i.e., the feature space $\mathbf{X}$ underlying X corresponds to a two-dimensional plane such as the $\mathbf{R}^2$:

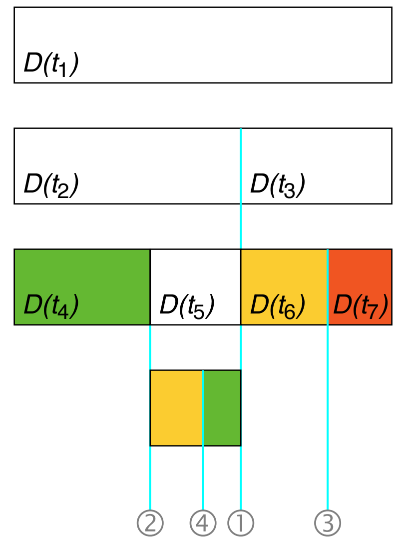


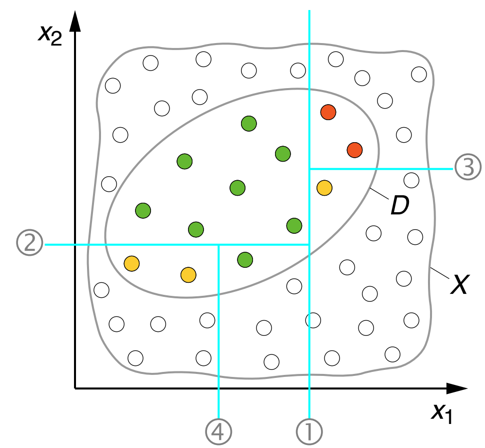
Decision Tree Algorithms

CART Algorithm (continued)

Illustration for two numeric features; i.e., the feature space $\mathbf{X}$ underlying X corresponds to a two-dimensional plane such as the $\mathbf{R}^2$:



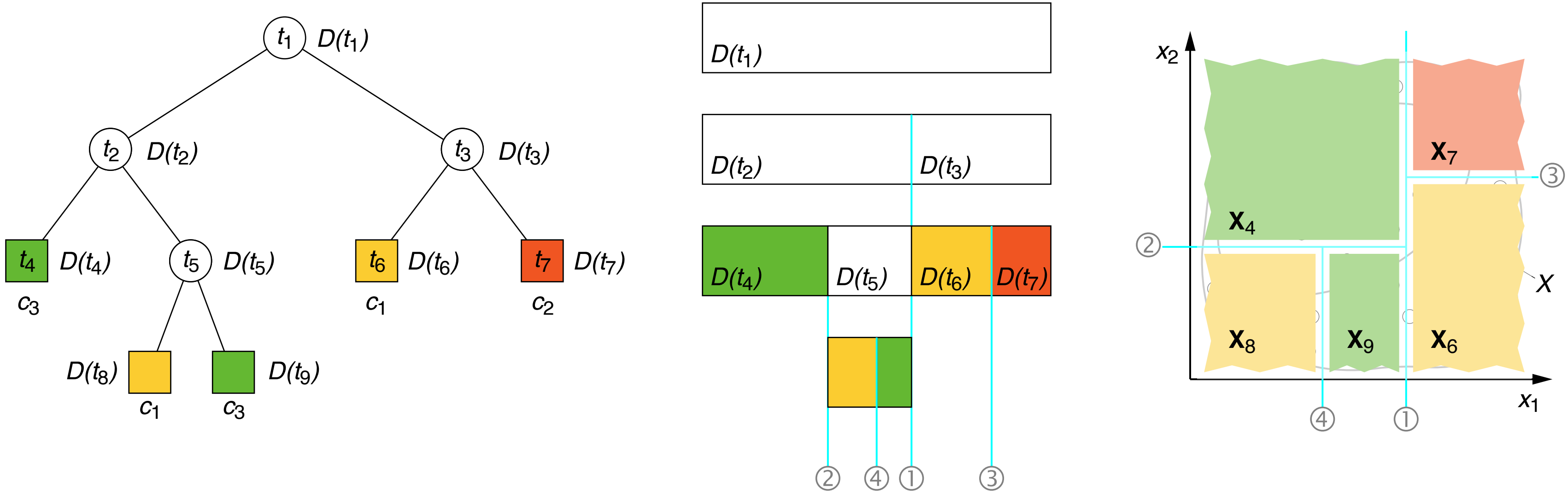




Decision Tree Algorithms

CART Algorithm (continued)

Illustration for two numeric features; i.e., the feature space $\mathbf{X}$ underlying X corresponds to a two-dimensional plane such as the $\mathbf{R}^2$:



By the sequence of (here: four) splittings of D the feature space $\mathbf{X}$ is cut into rectangular areas that are parallel to the two axes. Keyword: guillotine cuts

