Chapter ML:III

III. Decision Trees

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

Decision Tree Algorithms

ID3 Algorithm [Quinlan 1986] [CART Algorithm]

Characterization of the model (model world) [ML Introduction] :

- X is a set of feature vectors, also called feature space.
- C is a set of classes.
- $c : X \to C$ is the ideal classifier for X.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.

Task: Based on D, construction of a decision tree T to approximate c.

Decision Tree Algorithms

ID3 Algorithm [Quinlan 1986] [CART Algorithm]

Characterization of the model (model world) [ML Introduction] :

- X is a set of feature vectors, also called feature space.
- C is a set of classes.
- $c : X \to C$ is the ideal classifier for X.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.

Task: Based on D, construction of a decision tree T to approximate c.

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 $\{a_1, \ldots, a_k\}$:

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

2. Splitting criterion is the information gain.

Decision Tree Algorithms

ID3 Algorithm [Mitchell 1997] [algorithm template]

ID3(D, Attributes, Target)

1. Create a node t for the tree.
2. Label t with the most common value of Target in D.
3. If all examples in D are positive, return the single-node tree t, with label “+”.

   If all examples in D are negative, return the single-node tree t, with label “–”.
4. If Attributes is empty, return the single-node tree t.

- Otherwise:

  5. Let A* be the attribute from Attributes that best classifies examples in D.

     Assign t the decision attribute A*.
  6. For each possible value “a” in 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 label of the most common value of Target in D.
       Else add the subtree ID3(D_a, Attributes \ {A*}, Target).

7. Return t.

Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

ID3$(D,$ *Attributes*, *Target*$)$

1. $t =$ *createNode*$()$
2. *label*$(t) =$ *mostCommonClass*$(D,$ *Target*$)$
3. **IF** $\forall \langle \mathbf{x}, c(\mathbf{x}) \rangle \in D : c(\mathbf{x}) = c$ **THEN** *return*(t) **ENDIF**
4. **IF** *Attributes* $= \emptyset$ **THEN** *return*(t) **ENDIF**
5.
6.
7.

Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

ID3$(D,$ *Attributes*, *Target*$)$

1. $t =$ *createNode*$()$
2. *label*$(t) =$ *mostCommonClass*$(D,$ *Target*$)$
3. **IF** $\forall \langle \mathbf{x}, c(\mathbf{x}) \rangle \in D : c(\mathbf{x}) = c$ **THEN** *return*(t) **ENDIF**
4. **IF** *Attributes* $= \emptyset$ **THEN** *return*(t) **ENDIF**
5. $A^* = \text{argmax}_{A \in \textit{Attributes}}(\textit{informationGain}(D, A))$
6.
7.

Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

ID3$(D,$ *Attributes*, *Target*$)$

1. $t =$ *createNode*$()$
2. *label*$(t) =$ *mostCommonClass*$(D,$ *Target*$)$
3. **IF** $\forall \langle \mathbf{x}, c(\mathbf{x}) \rangle \in D : c(\mathbf{x}) = c$ **THEN** *return*(t) **ENDIF**
4. **IF** *Attributes* $= \emptyset$ **THEN** *return*(t) **ENDIF**
5. $A^* = \text{argmax}_{A \in \textit{Attributes}}(\textit{informationGain}(D, A))$
6. **FOREACH** $a \in A^*$ **DO**

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

    ELSE
      createEdge$(t, a,$ *ID3*$(D_a,$ *Attributes* $\setminus \{A^*\},$ *Target*$))$
    ENDIF

   ENDDO
7. *return*(t)

Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

ID3$(D,$ *Attributes*, *Target*$)$

1. $t =$ *createNode*$()$
2. *label*$(t) =$ *mostCommonClass*$(D,$ *Target*$)$
3. **IF** $\forall \langle \mathbf{x}, c(\mathbf{x}) \rangle \in D : c(\mathbf{x}) = c$ **THEN** *return*(t) **ENDIF**
4. **IF** *Attributes* $= \emptyset$ **THEN** *return*(t) **ENDIF**
5. $A^* = \text{argmax}_{A \in \textit{Attributes}}(\textit{informationGain}(D, A))$
6. **FOREACH** $a \in A^*$ **DO**

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

   IF $D_a = \emptyset$ **THEN**

   $t' =$ *createNode*$()$

   label$(t') =$ *mostCommonClass*$(D,$ *Target*$)$

   createEdge(t, a, t')

   ELSE

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

   ENDIF

   ENDDO
7. *return*(t)

Remarks:

- "*Target*" designates the feature (= attribute) that is comprised of the labels according to which an example can be classified. Within Mitchell's algorithm the respective class labels are '+' and '–', modeling the binary classification situation. In the pseudo code version, *Target* may contain multiple (more than two) classes.

- Step 3 of of the ID3 algorithm checks the purity of D and, given this case, assigns the unique class c, $c \in$ *dom*$($*Target*$)$, as label to the respective node.

Decision Tree Algorithms

ID3 Algorithm: Example

Example set D for mushrooms, implicitly defining a feature space X over the three dimensions color, size, and points:

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

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	eatable
red	1	1
brown	0	2
green	0	1

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

Estimated a-priori probabilities:

$$p_{\text{red}} = \frac{2}{5} = 0.4, \quad p_{\text{brown}} = \frac{2}{5} = 0.4, \quad p_{\text{green}} = \frac{1}{5} = 0.2$$

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	eatable
red	1	1
brown	0	2
green	0	1

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

Estimated a-priori probabilities:

$$p_{\text{red}} = \frac{2}{5} = 0.4, \quad p_{\text{brown}} = \frac{2}{5} = 0.4, \quad p_{\text{green}} = \frac{1}{5} = 0.2$$

Conditional entropy values for all attributes:

$$\begin{aligned} H(C \mid \text{color}) &= -(\, 0.4 \cdot (\tfrac{1}{2} \cdot \log_2 \tfrac{1}{2} + \tfrac{1}{2} \cdot \log_2 \tfrac{1}{2}) + \\ &\qquad 0.4 \cdot (\tfrac{0}{2} \cdot \log_2 \tfrac{0}{2} + \tfrac{2}{2} \cdot \log_2 \tfrac{2}{2}) + \\ &\qquad 0.2 \cdot (\tfrac{0}{1} \cdot \log_2 \tfrac{0}{1} + \tfrac{1}{1} \cdot \log_2 \tfrac{1}{1})\ ) \ = \ 0.4 \end{aligned}$$

$$\begin{aligned} H(C \mid \text{size}) &\approx 0.55 \\ H(C \mid \text{points}) &= 0.4 \end{aligned}$$

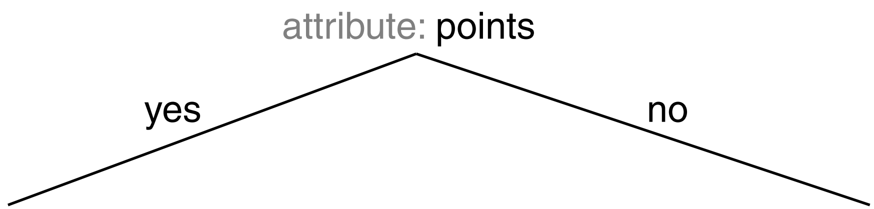
Remarks:

- The smaller $H(C \mid \textit{feature})$ is, the larger becomes the information gain. Hence, the difference $H(C) - H(C \mid \textit{feature})$ needs not to be computed since $H(C)$ is constant within each recursion step.
- In the example, the information gain in the first recursion step becomes maximum for the two features "color" and "points".

Decision Tree Algorithms

ID3 Algorithm: Example (continued)

Decision tree before the first recursion step:



color	size	eatability
red	small	toxic
brown	large	eatable

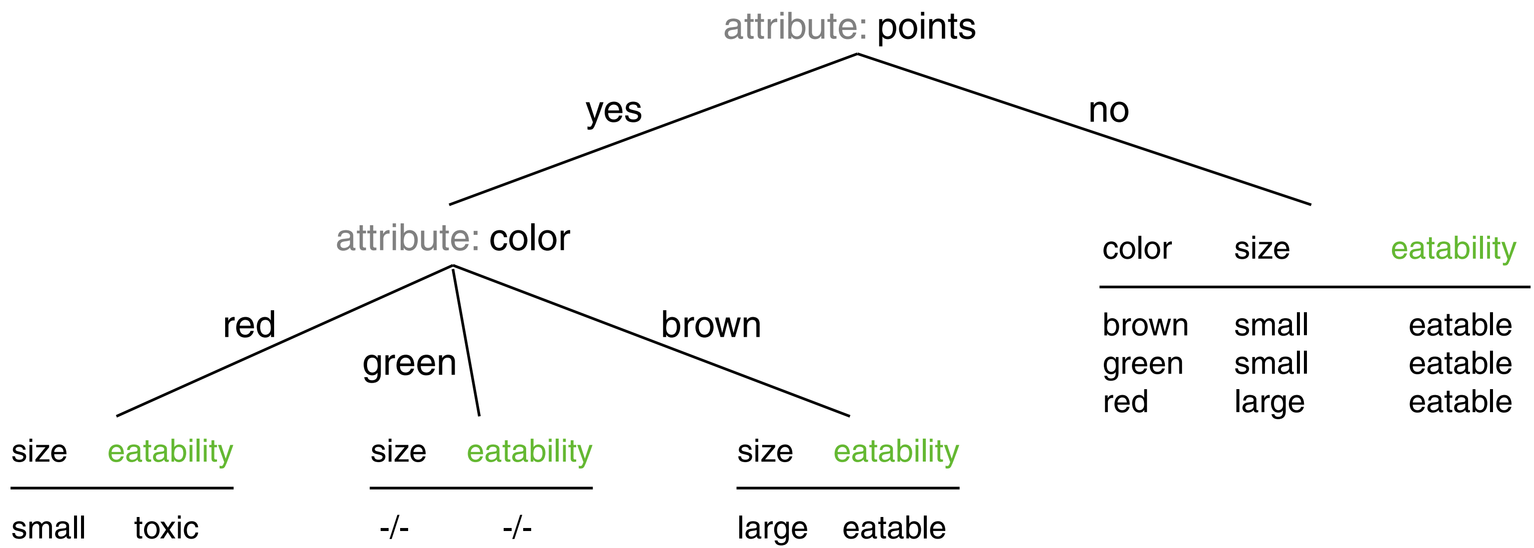
color	size	eatability
brown	small	eatable
green	small	eatable
red	large	eatable

The feature “points” was chosen in Step 5 of the ID3 algorithm.

Decision Tree Algorithms

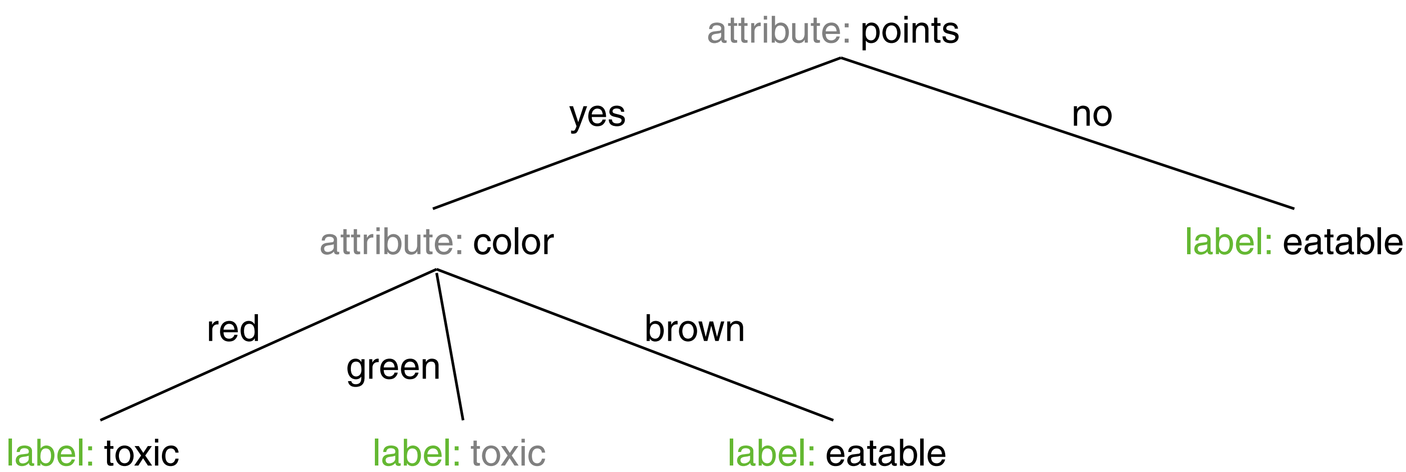
ID3 Algorithm: Example (continued)

Decision tree before the second recursion step:



The feature “color” was chosen in Step 5 of the ID3 algorithm.

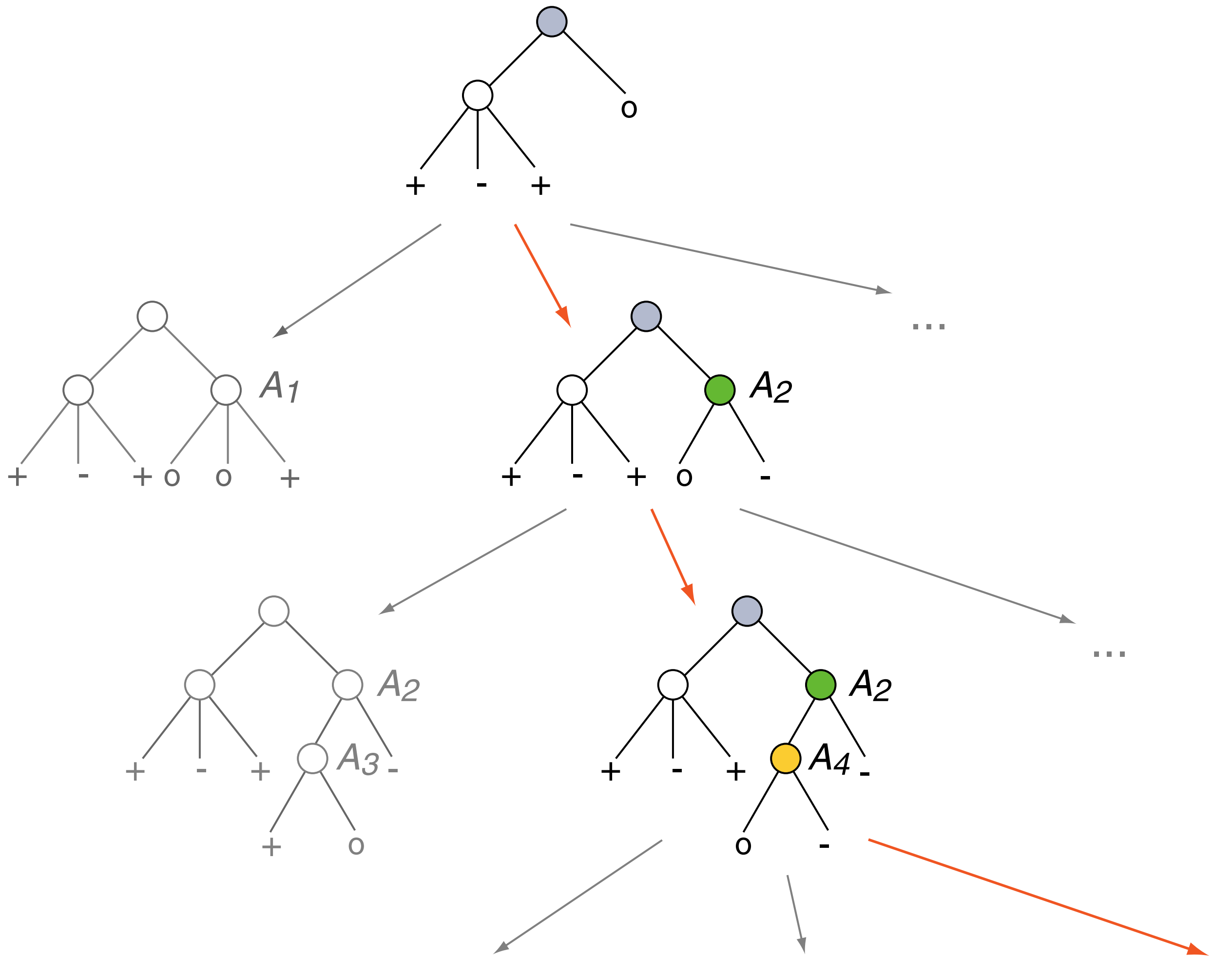
Final decision tree after second recursion step:



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

Decision Tree Algorithms

ID3 Algorithm: Hypothesis Space



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.

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
  - ➜ *local* optimization of decision trees

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
  - ➜ *local* optimization of decision trees

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:

- Let $\mathbf{A}_j$ be the finite domain (the possible values) of feature $A_j,\ j = 1, \ldots, p$, and let C be a set of classes. Then, a hypothesis space H that is comprised of all decision trees corresponds to the set of all functions $h,\ h : \mathbf{A}_1 \times \ldots \times \mathbf{A}_p \to C$. Typically, $C = \{0, 1\}$.

- The inductive bias of the ID3 algorithm is of a different kind than the inductive bias of the candidate elimination algorithm (version space 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 attribute-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. H corresponds to the set of all discrete functions (from the Cartesian product of the feature domains onto the set of classes) that can be represented in the form of a decision tree.

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

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

Decision Tree Algorithms

CART Algorithm [Breiman 1984] [ID3 Algorithm]

Characterization of the model (model world) [ML Introduction] :

- X is a set of feature vectors, also called feature space.
  No restrictions are presumed for the features' measurement scales.
- C is a set of classes.
- $c : X \to C$ is the ideal classifier for X.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.

Task: Based on D, construction of a decision tree T to approximate c.

Decision Tree Algorithms

CART Algorithm [Breiman 1984] [ID3 Algorithm]

Characterization of the model (model world) [ML Introduction] :

- X is a set of feature vectors, also called feature space.
  No restrictions are presumed for the features' measurement scales.
- C is a set of classes.
- $c : X \to C$ is the ideal classifier for X.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.

Task: Based on D, construction of a decision tree T to approximate c.

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)

1. Let A be a feature with domain $\mathbf{A}$. Ensure a finite number of binary splittings for X by applying the following domain partitioning rules:

   – If A is nominal, choose $\mathbf{A}' \subset \mathbf{A}$ such that $0 < |\mathbf{A}'| \leq |\mathbf{A} \setminus \mathbf{A}'|$.

   – If A is ordinal, choose $a \in \mathbf{A}$ such that $x_{\text{min}} < a < x_{\text{max}}$, where x_{min}, x_{max} are the minimum and maximum values of feature A in D.

   – If A is numeric, choose $a \in \mathbf{A}$ such that $a = (x_k + x_l)/2$, where x_k, x_l are consecutive elements in the ordered value list of feature A in D.

Decision Tree Algorithms

CART Algorithm (continued)

1. Let A be a feature with domain $\mathbf{A}$. Ensure a finite number of binary splittings for X by applying the following domain partitioning rules:

   – If A is nominal, choose $\mathbf{A}' \subset \mathbf{A}$ such that $0 < |\mathbf{A}'| \leq |\mathbf{A} \setminus \mathbf{A}'|$.

   – If A is ordinal, choose $a \in \mathbf{A}$ such that $x_{\text{min}} < a < x_{\text{max}}$, where x_{min}, x_{max} are the minimum and maximum values of feature A in D.

   – If A is numeric, choose $a \in \mathbf{A}$ such that $a = (x_k + x_l)/2$, where x_k, x_l are consecutive elements in the ordered value list of feature A in D.

2. For node t of a decision tree generate all splittings of the above type.

3. Choose a splitting from the set of splittings that maximizes the impurity reduction $\Delta\iota$:

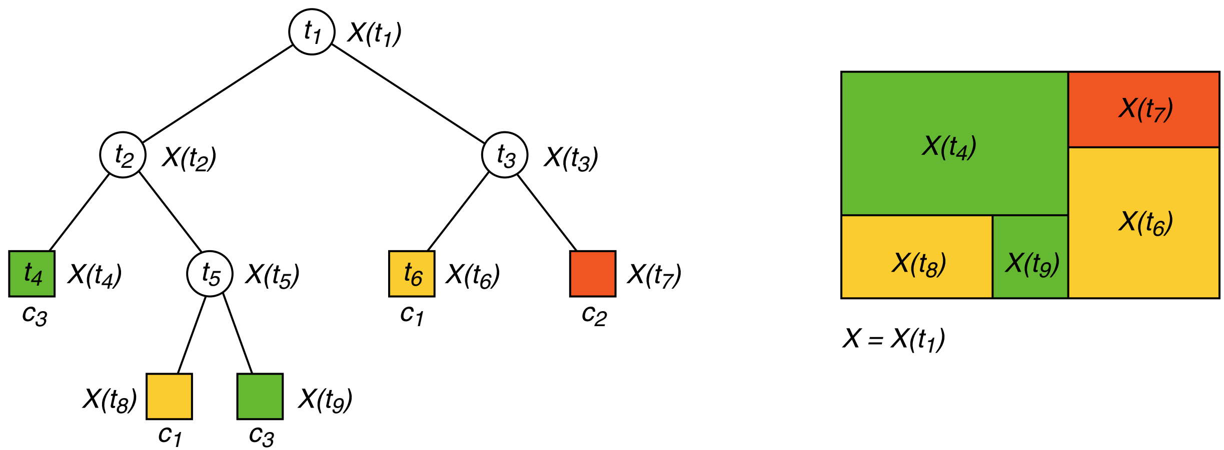
$$\Delta\iota\big(D(t),\ \{D(t_L), D(t_R)\}\big) \;=\; \iota(t) - \frac{|D_L|}{|D|} \cdot \iota(t_L) - \frac{|D_R|}{|D|} \cdot \iota(t_R),$$

where t_L and t_R denote the left and right successor of t.

Decision Tree Algorithms

CART Algorithm (continued)

Illustration for two numeric features, i.e., the feature space X corresponds to a two-dimensional plane:



By a sequence of splittings the feature space X is partitioned into rectangles that are parallel to the two axes.