Chapter ML:III

III. Decision Trees

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

Decision Trees Basics

Specification of Classification Problems [ML Introduction]

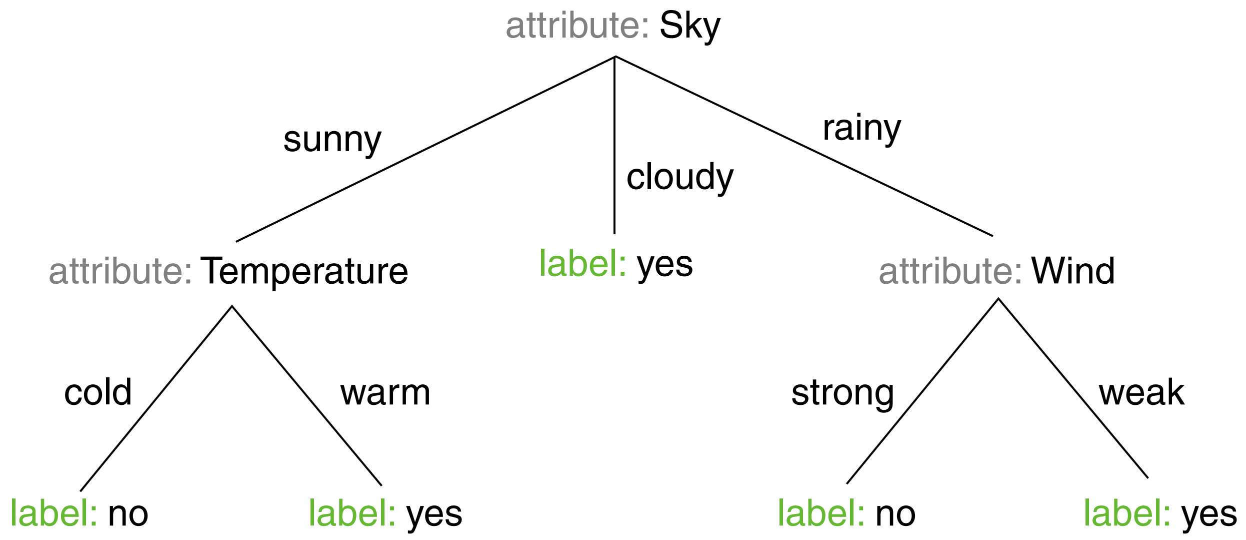
Characterization of the model (model world):

- 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.

Decision Trees Basics

Decision Tree for the Concept “EnjoySport”

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
1	sunny	warm	normal	strong	warm	same	yes
2	sunny	warm	high	strong	warm	same	yes
3	rainy	cold	high	strong	warm	change	no
4	sunny	warm	high	strong	cool	change	yes



Partitioning of X at the root node:

$$X = \{\mathbf{x} \in X : \mathbf{x}|_{\text{Sky}} = \text{sunny}\} \cup \{\mathbf{x} \in X : \mathbf{x}|_{\text{Sky}} = \text{cloudy}\} \cup \{\mathbf{x} \in X : \mathbf{x}|_{\text{Sky}} = \text{rainy}\}$$

Decision Trees Basics

Definition 1 (Splitting)

Let X be feature space and let D be a set of examples. A splitting of X is a partitioning of X into mutually exclusive subsets $X_1, \ldots, X_s$. I.e., $X = X_1 \cup \ldots \cup X_s$ with $X_j \neq \emptyset$ and $X_j \cap X_{j'} = \emptyset$, where $j, j' \in \{1, \ldots, s\}, j \neq j'$.

A splitting $X_1, \ldots, X_s$ of X induces a splitting $D_1, \ldots, D_s$ of D, where D_j, $j = 1, \ldots, s$, is defined as $\{(\mathbf{x}, c(\mathbf{x})) \in D \mid \mathbf{x} \in X_j\}$.

Decision Trees Basics

Definition 1 (Splitting)

Let X be feature space and let D be a set of examples. A splitting of X is a partitioning of X into mutually exclusive subsets $X_1, \ldots, X_s$. I.e., $X = X_1 \cup \ldots \cup X_s$ with $X_j \neq \emptyset$ and $X_j \cap X_{j'} = \emptyset$, where $j, j' \in \{1, \ldots, s\}, j \neq j'$.

A splitting $X_1, \ldots, X_s$ of X induces a splitting $D_1, \ldots, D_s$ of D, where D_j, $j = 1, \ldots, s$, is defined as $\{(\mathbf{x}, c(\mathbf{x})) \in D \mid \mathbf{x} \in X_j\}$.

A splitting depends on the measurement scale of a feature:

1. m-ary splitting induced by a (nominal) feature A with finite domain:

2. Binary splitting induced by a (nominal) feature A:

3. Binary splitting induced by an ordinal feature A:

Decision Trees Basics

Definition 1 (Splitting)

Let X be feature space and let D be a set of examples. A splitting of X is a partitioning of X into mutually exclusive subsets $X_1, \ldots, X_s$. I.e., $X = X_1 \cup \ldots \cup X_s$ with $X_j \neq \emptyset$ and $X_j \cap X_{j'} = \emptyset$, where $j, j' \in \{1, \ldots, s\}, j \neq j'$.

A splitting $X_1, \ldots, X_s$ of X induces a splitting $D_1, \ldots, D_s$ of D, where D_j, $j = 1, \ldots, s$, is defined as $\{(\mathbf{x}, c(\mathbf{x})) \in D \mid \mathbf{x} \in X_j\}$.

A splitting depends on the measurement scale of a feature:

1. m-ary splitting induced by a (nominal) feature A with finite domain:

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

2. Binary splitting induced by a (nominal) feature A:

   $A' \subset A: \quad X = \{\mathbf{x} \in X : \mathbf{x}|_A \in A'\} \cup \{\mathbf{x} \in X : \mathbf{x}|_A \notin A'\}$

3. Binary splitting induced by an ordinal feature A:

   $v \in \mathit{dom}(A): \quad X = \{\mathbf{x} \in X : \mathbf{x}|_A \succeq v\} \cup \{\mathbf{x} \in X : \mathbf{x}|_A \prec v\}$

Remarks:

- ❑ The syntax $\mathbf{x}|_A$ denotes the projection operator, which returns that vector component (dimension) of $\mathbf{x} = x_1, \ldots, x_p$ that is associated with A. Without loss of generality this projection can be presumed being unique.
- ❑ A splitting of X into two disjoint, non-empty subsets is called a binary splitting.
- ❑ We consider only splittings of X that are induced by a splitting of a single feature A of X. Keyword: monothetic splitting

Decision Trees Basics

Definition 2 (Decision Tree)

Let X be feature space and let C be a set of classes. A decision tree T for X and C is a finite tree with a distinguished root node. A non-leaf node t of T has assigned (1) a set $X(t) \subseteq X$, (2) a splitting of $X(t)$, and (3) a one-to-one mapping of the subsets of the splitting to the successors of t.

$X(t) = X$ iff t is root node. A leaf node of T has assigned a class from C.

Decision Trees Basics

Definition 2 (Decision Tree)

Let X be feature space and let C be a set of classes. A decision tree T for X and C is a finite tree with a distinguished root node. A non-leaf node t of T has assigned (1) a set $X(t) \subseteq X$, (2) a splitting of $X(t)$, and (3) a one-to-one mapping of the subsets of the splitting to the successors of t.

$X(t) = X$ iff t is root node. A leaf node of T has assigned a class from C.

Classification of some $\mathbf{x} \in X$ given a decision tree T:

1. Find the root node of T.

2. If t is a non-leaf node, find among its successors that node whose subset of the splitting of $X(t)$ contains $\mathbf{x}$. Repeat this step.

3. If t is a leaf node, label $\mathbf{x}$ with the respective class.

➜ The set of possible decision trees forms the hypothesis space H.

Remarks:

- The classification of an $\mathbf{x} \in X$ determines a unique path from the root node of T to some leaf node of T.
- At each non-leaf node a particular feature of $\mathbf{x}$ is evaluated in order to find the next node and hence a possible next feature to be analyzed.
- Each path from the root node to some leaf node corresponds to a conjunction of feature values, which are successively tested. This test can be formulated as a decision rule. Example:

  IF Sky=rainy AND Wind=weak THEN EnjoySport=yes

  If all tests in T are of the kind shown in the example, namely, a comparison with a single feature value, all feature domains must be finite.
- If in all non-leaf nodes of T only one feature is evaluated at a time, T is called a *monothetic* decision tree. Examples for *polythetic* decision trees are the so-called oblique decision trees.
- Decision trees became popular in 1986, with the introduction of the ID3 Algorithm by J. R. Quinlan.

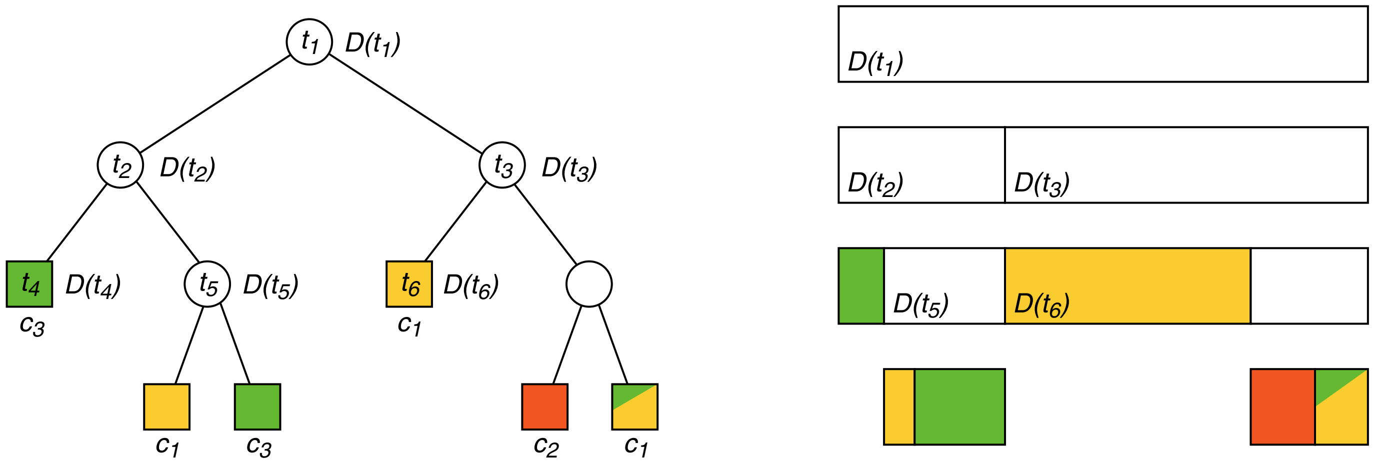
Decision Trees Basics

Notation

Let T be decision tree for X and C, let D be a set of examples, and let t be a node of T. Then we agree on the following notation:

- $X(t)$ denotes the subset of the feature space X that is represented by t (as used in Definition 2).
- $D(t)$ denotes the subset of the example set D that is represented by t, where $D(t) = \{(\mathbf{x}, c(\mathbf{x})) \in D \mid \mathbf{x} \in X(t)\}$

Illustration:



Remarks:

- The set $X(t)$ comprises those members $\mathbf{x}$ of X that are filtered by a path from the root node of T to the node t.
- *leaves*(T) denotes the set of all leaf nodes of T.
- A single node t of a decision tree T, and hence T itself, encode a piecewise constant function. This way, t as well as T can form complex non-linear classifiers. The functions encoded by t and T differ in the number of evaluated features of $\mathbf{x}$, which is one for t and the tree height for T.
- In the following we will use the symbols "t" and "T" to denote also the classifiers that are encoded by a node t and a tree T respectively:

$$t, T : X \to C \quad \text{(instead of } y_t, y_T : X \to C\text{)}$$

Decision Trees Basics

Algorithm Template: Construction

Algorithm:	*DT-construct*	Decision Tree Construction
Input:	D	(Sub)set of examples.
Output:	t	Root node of a decision (sub)tree.

```
DT-construct(D)

 1.  $t = newNode()$
     $label(t) = representativeClass(D)$

 2.  IF   $impure(D)$
     THEN $criterion = splitCriterion(D)$
     ELSE $return(t)$

 3.  $\{D_1, \ldots, D_s\} = decompose(D, criterion)$

 4.  FOREACH $D'$ IN $\{D_1, \ldots, D_s\}$ DO

        $addSuccessor(t, DT\text{-}construct(D'))$

     ENDDO

 5.  $return(t)$
```

[Illustration]

Decision Trees Basics

Algorithm Template: Classification

Algorithm:	*DT-classify*	Decision Tree Classification
Input:	$\mathbf{x}$	Feature vector.
	t	Root node of a decision (sub)tree.
Output:	$y(\mathbf{x})$	Class of feature vector $\mathbf{x}$ in the decision (sub)tree below t.

DT−classify$(\mathbf{x}, t)$

```
1.  IF   $isLeafNode(t)$
    THEN $return(label(t))$
    ELSE $return(DT{-}classify(\mathbf{x}, splitSuccessor(t, \mathbf{x})))$
```

Remarks:

- Since *DT-construct* assigns to each node of a decision tree T a class, each subtree of T as well as each pruned version of a subtree of T represents a valid decision tree on its own.
- Functions of *DT-construct*:
  - *representativeClass*(D)
    Returns a representative class for the example set D. Note that, due to pruning, each node may become a leaf node.
  - *impure*(D)
    Evaluates the (im)purity of a set D of examples.
  - *splitCriterion*(D)
    Returns a split criterion for $X(t)$ based on the examples in $D(t)$.
  - *decompose*$(D,$ *criterion*$)$
    Returns a splitting of D according to *criterion*.
  - *addSuccessor*(t, t')
    Inserts the successor t' for node t.
- Functions of *DT-classify*:
  - *isLeafNode*(t)
    Tests whether t is a leaf node.
  - *splitSuccessor*$(t, \mathbf{x})$
    Returns the (unique) successor t' of t for which $\mathbf{x} \in X(t')$ holds.

Decision Trees Basics

When to Use Decision Trees

Problem characteristics that may suggest a decision tree classifier:

- the objects can be described by feature-value combinations.
- the domain and range of the target function are discrete
- hypotheses take the form of disjunctions
- the training set contains noise

Selected application areas:

- medical diagnosis
- fault detection in technical systems
- risk analysis for credit approval
- basic scheduling tasks such as calendar management
- classification of design flaws in software engineering

Decision Trees Basics

On the Construction of Decision Trees

- How to exploit an example set both efficiently and effectively?
- According to what rationale should a node become a leaf node?
- How to assign a class for nodes of impure example sets?
- How to evaluate decision tree performance?

Decision Trees Basics

Performance of Decision Trees

1. Size

2. Classification error

Decision Trees Basics

Performance of Decision Trees

1. Size

   Among those theories that can explain an observation, the most simple one is to be preferred *(Ockham's Razor)* :

   Entia non sunt multiplicanda praeter necessitatem or *sine necessitate.*

   [Johannes Clauberg 1622-1665]

   Here: among all decision trees of minimum classification error we choose the one of smallest size.

2. Classification error

   Quantifies the rigor according to which a class label is assigned to $\mathbf{x}$ in a leaf node of T, based on the examples in D.

   If all leaf nodes of a decision tree T represent a single example of D, the classification error of T with respect to D is zero.

Decision Trees Basics

Performance of Decision Trees: Size

- ❑ Leaf node number
- ❑ Tree height
- ❑ External path length
- ❑ Weighted external path length

Decision Trees Basics

Performance of Decision Trees: Size

- Leaf node number

  The leaf node number corresponds to number of rules that are encoded in a decision tree.

- Tree height

  The tree height corresponds to the maximum rule length and bounds the number of premises to be evaluated to reach a class decision.

- External path length

  The external path length totals the lengths of all paths from the root of a tree to its leaf nodes. It corresponds to the space to store all rules that are encoded in a decision tree.

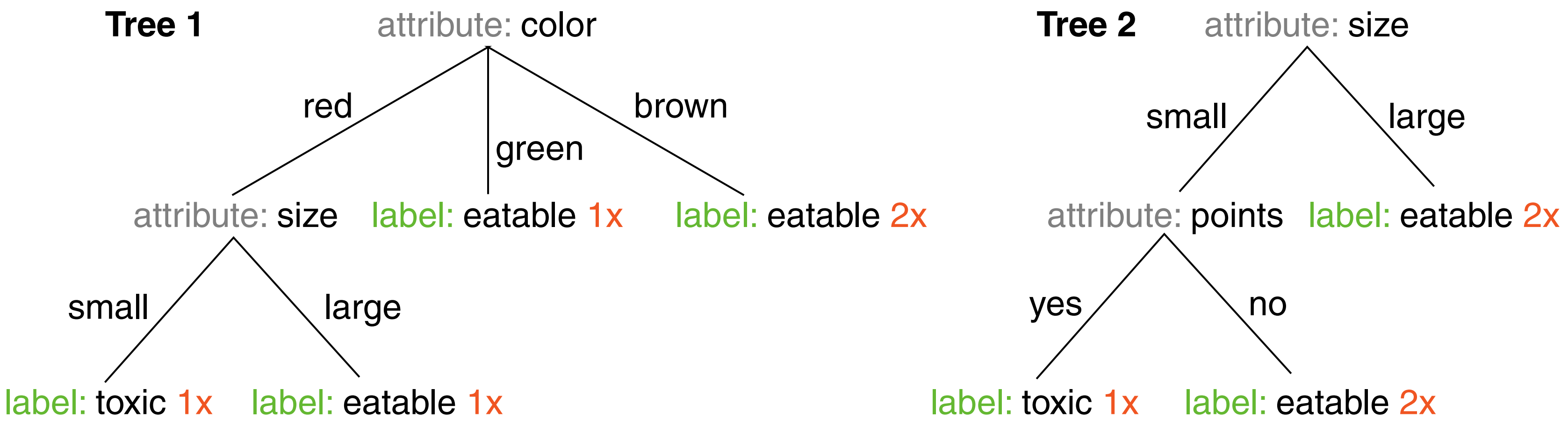
- Weighted external path length

  The weighted external path length is defined as the external path length where each length value is weighted by the number of examples in D that are classified by this path.

Decision Trees Basics

Performance of Decision Trees: Size (continued)

Both trees below correctly classify all examples in D :



Criterion	Tree 1	Tree 2
Leaf node number	4	3
Tree height	2	2
External path length	6	5
Weighted external path length	7	8

Decision Trees Basics

Performance of Decision Trees: Size (continued)

Theorem 3 (External Path Length Bound)

The problem to decide for a set of examples D whether or not a decision tree exists whose external path length is bounded by b, is NP-complete.

Decision Trees Basics

Performance of Decision Trees: Classification Error

Given a decision tree T, a set of examples D, and a node t of T that represents the example subset $D(t) \subseteq D$. Then, the class that is assigned to t, *label*(t), is defined as follows:

$$\textit{label}(t) = \underset{c \in C}{\text{argmax}} \; \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Decision Trees Basics

Performance of Decision Trees: Classification Error

Given a decision tree T, a set of examples D, and a node t of T that represents the example subset $D(t) \subseteq D$. Then, the class that is assigned to t, *label*(t), is defined as follows:

$$\textit{label}(t) = \underset{c \in C}{\text{argmax}} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Misclassification rate of node classifier t wrt. $D(t)$:

$$\textit{Err}(t, D(t)) = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) \neq \textit{label}(t)\}|}{|D(t)|} = 1 - \max_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Decision Trees Basics

Performance of Decision Trees: Classification Error

Given a decision tree T, a set of examples D, and a node t of T that represents the example subset $D(t) \subseteq D$. Then, the class that is assigned to t, *label*(t), is defined as follows:

$$\textit{label}(t) = \underset{c \in C}{\text{argmax}} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Misclassification rate of node classifier t wrt. $D(t)$:

$$\textit{Err}(t, D(t)) = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) \neq \textit{label}(t)\}|}{|D(t)|} = 1 - \max_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Misclassification rate of decision tree classifier T wrt. $D(t)$:

$$\textit{Err}(T, D) = \sum_{t \in \textit{leaves}(T)} \frac{|D(t)|}{|D|} \cdot \textit{Err}(t, D(t))$$

Remarks:

- Observe the difference between max(f) and argmax(f). Both expressions maximize f, whereas the former returns the maximum f-value (the image) while the latter returns the argument (the preimage) for which f becomes maximum:
  - $\max\limits_{c \in C}(f(c)) = \max\{f(c) \mid c \in C\}$
  - $\operatorname*{argmax}\limits_{c \in C}(f(c)) = c^* \quad \Rightarrow \; f(c^*) = \max\limits_{c \in C}(f(c))$

- The classifiers t and T may not have been constructed using $D(t)$ as training data. Stated another way, the example set $D(t)$ is in the role of a holdout test set.

- The true misclassification rate *Err*$^*(T)$ is based on a probability measure P on $X \times C$ (and not on relative frequencies). For a node t of T this probability becomes minimum iff:

$$\textit{label}(t) = \operatorname*{argmax}_{c \in C} \; P(c \mid X(t))$$

- If D has been used as training set, a reliable interpretation of the (training) error *Err*(T, D) in terms of *Err*$^*(T)$ requires the Inductive Learning Hypothesis to hold. This implies, among others, that the distribution of C over the feature space X corresponds to the distribution of C over the training set D.

Decision Trees Basics

Performance of Decision Trees: Classification Error (continued)

Given a decision tree T, a set of examples D, and a node t of T that represents the example subset $D(t) \subseteq D$. Then, the class that is assigned to t, *label*(t), is defined as follows:

$$\textit{label}(t) = \underset{c' \in C}{\operatorname{argmin}} \sum_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|} \cdot \textit{cost}(c' \mid c)$$

Decision Trees Basics

Performance of Decision Trees: Classification Error (continued)

Given a decision tree T, a set of examples D, and a node t of T that represents the example subset $D(t) \subseteq D$. Then, the class that is assigned to t, *label*(t), is defined as follows:

$$\textit{label}(t) = \underset{c' \in C}{\operatorname{argmin}} \sum_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|} \cdot \textit{cost}(c' \mid c)$$

Misclassification costs of node classifier t wrt. $D(t)$:

$$\textit{Err}_{\textit{cost}}(t, D(t)) = \sum_{(\mathbf{x}, c(\mathbf{x})) \in D(t)} \textit{cost}(\textit{label}(t) \mid c(\mathbf{x})) = \min_{c' \in C} \sum_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|} \cdot \textit{cost}(c' \mid c)$$

Decision Trees Basics

Performance of Decision Trees: Classification Error (continued)

Given a decision tree T, a set of examples D, and a node t of T that represents the example subset $D(t) \subseteq D$. Then, the class that is assigned to t, *label*(t), is defined as follows:

$$\textit{label}(t) = \underset{c' \in C}{\operatorname{argmin}} \sum_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|} \cdot \textit{cost}(c' \mid c)$$

Misclassification costs of node classifier t wrt. $D(t)$:

$$\textit{Err}_{\textit{cost}}(t, D(t)) = \sum_{(\mathbf{x},c(\mathbf{x})) \in D(t)} \textit{cost}(\textit{label}(t) \mid c(\mathbf{x})) = \min_{c' \in C} \sum_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|} \cdot \textit{cost}(c' \mid c)$$

Misclassification costs of decision tree classifier T wrt. $D(t)$:

$$\textit{Err}_{\textit{cost}}(T, D) = \sum_{t \in \textit{leaves}(T)} \frac{|D(t)|}{|D|} \cdot \textit{Err}_{\textit{cost}}(t, D(t))$$

Remarks:

- Again, observe the difference between min(f) and argmin(f). Both expressions minimize f, whereas the former returns the minimum f-value (the image) while the latter returns the argument (the preimage) for which f becomes minimum.