Chapter ML:III

III. Decision Trees

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

Decision Trees Basics

Classification Problems with Nominal Features

Setting:

- X is a set of feature vectors.
- C is a set of classes.
- $c : X \to C$ is the (unknown) 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.

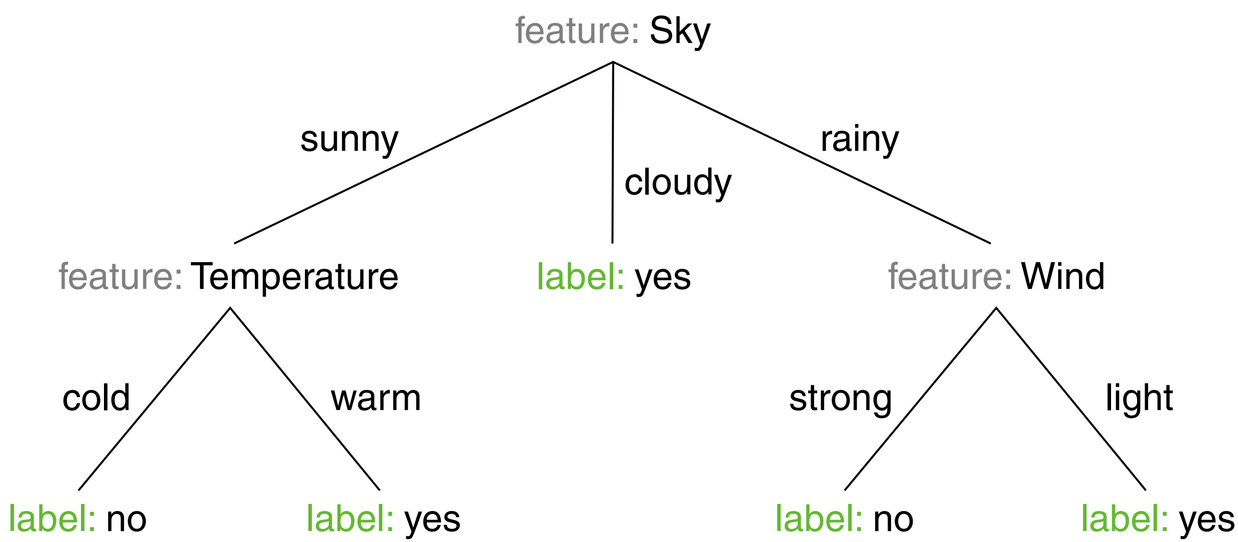
Todo:

- Approximate $c(\mathbf{x})$, which is implicitly given via D, with a decision tree.

Decision Trees Basics

Decision Tree for the Concept “EnjoySport”

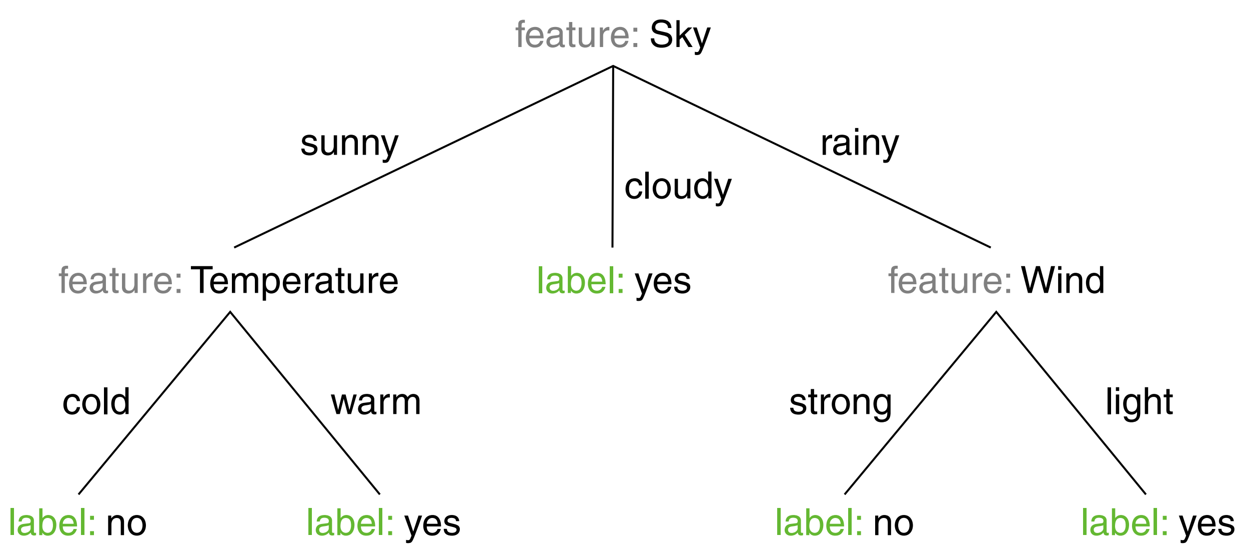
Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
1	sunny	warm	normal	strong	warm	same	yes
2	sunny	warm	high	strong	warm	same	yes
3	rainy	cold	high	strong	warm	change	no
. . .							



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
...							



Splitting 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 a set of feature vectors and D a set of examples. A splitting of X is a decomposition 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 $\{(\mathbf{x}, c(\mathbf{x})) \in D \mid \mathbf{x} \in X_j\}$.

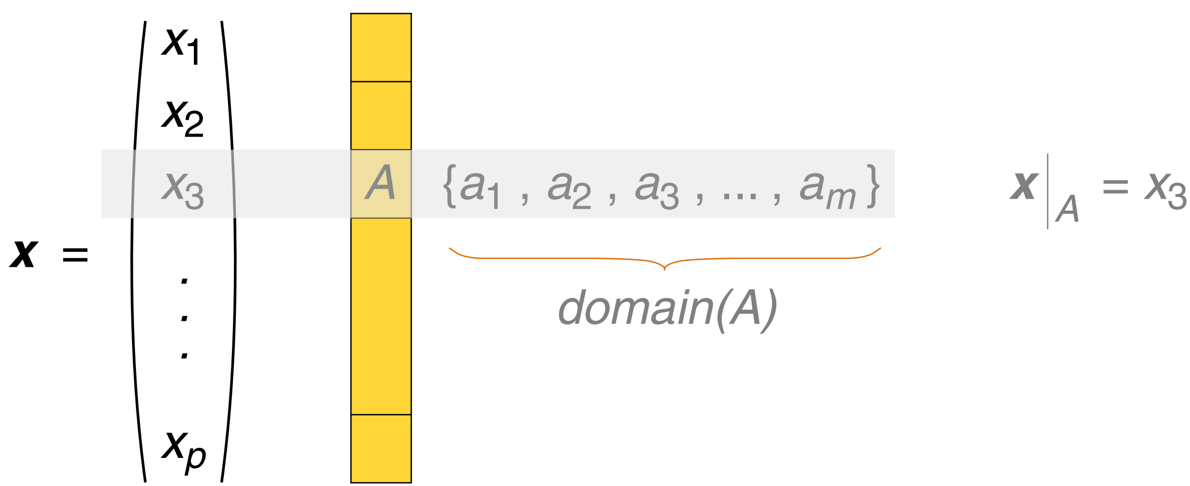
Decision Trees Basics

Definition 1 (Splitting)

Let X be a set of feature vectors and D a set of examples. A splitting of X is a decomposition 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:



Decision Trees Basics

Definition 1 (Splitting)

Let X be a set of feature vectors and D a set of examples. A splitting of X is a decomposition 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:

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

Decision Trees Basics

Definition 1 (Splitting)

Let X be a set of feature vectors and D a set of examples. A splitting of X is a decomposition 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 the feature 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.

  By contrast, a polythetic splitting considers several features at the same time.

Decision Trees Basics

Definition 2 (Decision Tree)

Let X be a set of features and C 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 its successors.

$X(t) = X$ 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 a set of features and C 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 its successors.

$X(t) = X$ 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 t of T.
2. If t is a non-leaf node, find among its successors that node t' whose subset of the splitting of $X(t)$ contains $\mathbf{x}$. Repeat this step with $t = t'$.
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 along with 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=light THEN EnjoySport=yes

  If all tests in T are of the kind shown in the example, namely, an equality test regarding a 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.

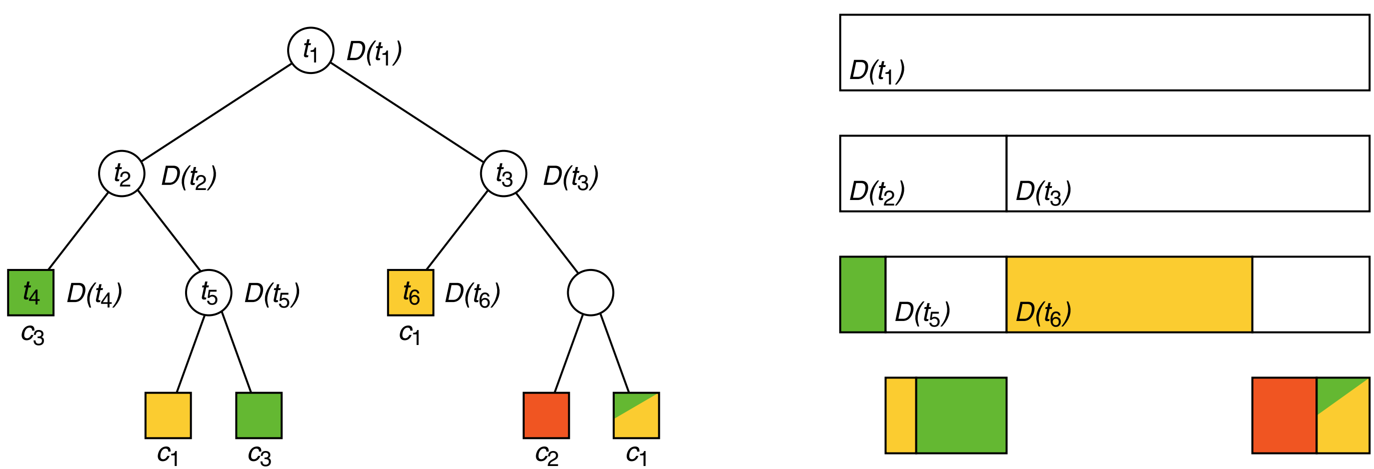
Decision Trees Basics

Notation

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

- $X(t)$ denotes the subset of X that is represented by t.
- $D(t)$ denotes the subset of the example set D that is represented by t, where $D(t) = \{(\mathbf{x}, c(\mathbf{x})) \in D \mid \mathbf{x} \in X(t)\}$. (see the splitting definition)

Illustration:



Remarks:

- The set $X(t)$ is comprised of 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−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 can be represented in disjunctive normal form
- 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

Evaluation of Decision Trees

1. Size

2. Classification error

Decision Trees Basics

Evaluation 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 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. [Illustration]

   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

Evaluation of Decision Trees: Size

- Leaf node number

- Tree height

- External path length

- Weighted external path length

Decision Trees Basics

Evaluation 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 with each length value weighted by the number of examples in D that are classified by this path.

Decision Trees Basics

Evaluation of Decision Trees: Size (continued)

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

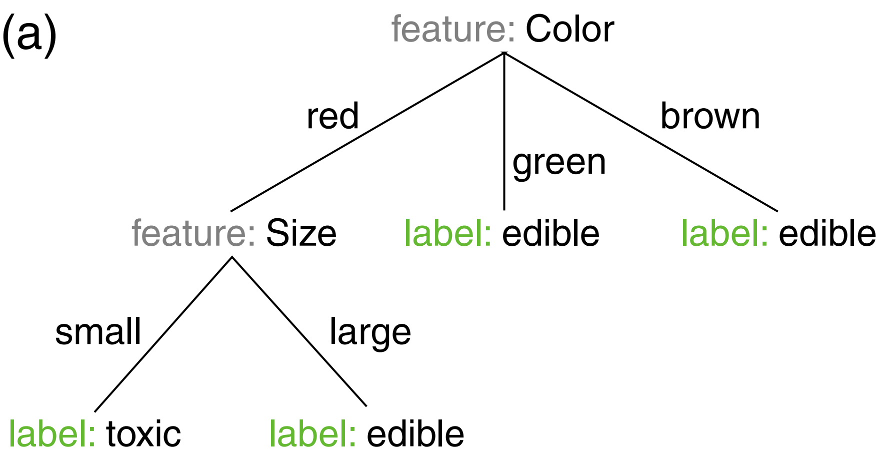
	Color	Size	Points	Edibility
1	red	small	yes	toxic
2	brown	small	no	edible
3	brown	large	yes	edible
4	green	small	no	edible
5	red	large	no	edible

Decision Trees Basics

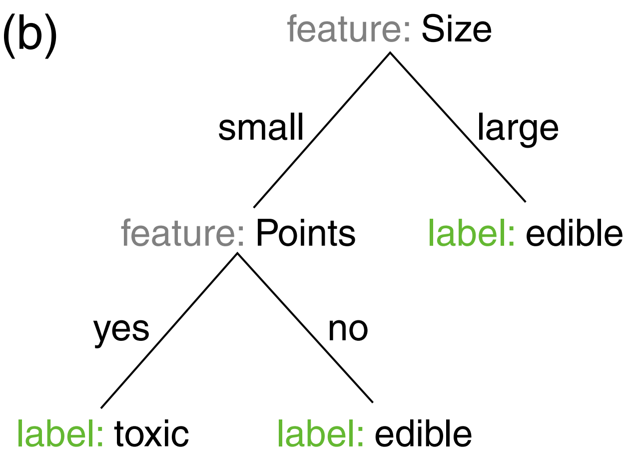
Evaluation of Decision Trees: Size (continued)

The following trees correctly classify all examples in D :

(a)



(b)

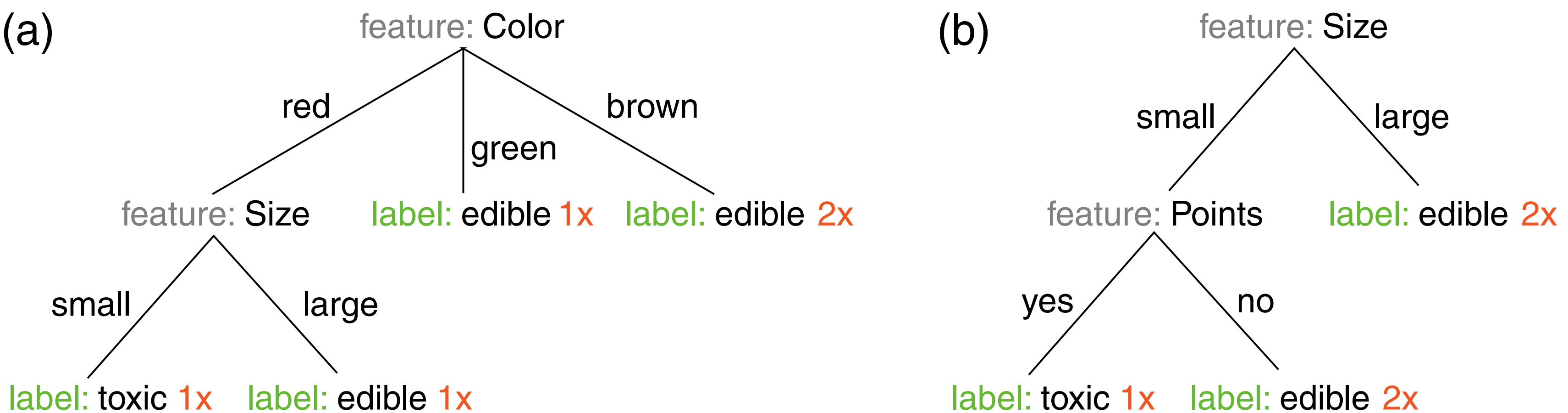


Criterion	**(a)**	**(b)**
Leaf node number	4	3
Tree height	2	2
External path length	6	5

Decision Trees Basics

Evaluation of Decision Trees: Size (continued)

The following trees correctly classify all examples in D :



Criterion	**(a)**	**(b)**
Leaf node number	4	3
Tree height	2	2
External path length	6	5
Weighted external path length	7	8

Decision Trees Basics

Evaluation 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

Evaluation 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 [Illustration]:

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

Decision Trees Basics

Evaluation 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 [Illustration]:

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

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

Evaluation 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 [Illustration]:

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

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 :

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

Remarks:

- Observe the difference between max(f) and argmax(f). Both expressions maximize f, but the former returns the maximum f-value (the image) while the latter returns the argument (the preimage) for which f becomes maximum:
  - $\max_{c \in C}(f(c)) = \max\{f(c) \mid c \in C\}$
  - $\operatorname{argmax}_{c \in C}(f(c)) = c^* \quad \Rightarrow \; f(c^*) = \max_{c \in C}(f(c))$
- The classifiers t and T may not have been constructed using $D(t)$ as training data. I.e., 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 that the distribution of C over the training set D corresponds to the distribution of C over X .

Decision Trees Basics

Evaluation of Decision Trees: Misclassification Costs

Given a decision tree T, a set of examples D, and a node t of T that represents the example subset $D(t) \subseteq D$. In addition, there is a cost measure for misclassification. 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

Evaluation of Decision Trees: Misclassification Costs

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

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

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

$$\textit{Err}_{\textit{cost}}(t, D(t)) = \frac{1}{|D_t|} \cdot \sum_{(\mathbf{x}, c(\mathbf{x})) \in D(t)} \textit{cost}(\textit{label}(t) \mid c(\mathbf{x})) = \min_{c' \in C} \sum_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|} \cdot \textit{cost}(c' \mid c)$$

Decision Trees Basics

Evaluation of Decision Trees: Misclassification Costs

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

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

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

$$\mathit{Err}_{\mathit{cost}}(t, D(t)) = \frac{1}{|D_t|} \cdot \sum_{(\mathbf{x}, c(\mathbf{x})) \in D(t)} \mathit{cost}(\mathit{label}(t) \mid c(\mathbf{x})) = \min_{c' \in C} \sum_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|} \cdot \mathit{cost}(c' \mid c)$$

Misclassification costs of decision tree classifier T wrt. D :

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

Remarks:

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

Chapter ML:III

III. Decision Trees

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

Impurity Functions

Splitting

Let t be a leaf node of an incomplete decision tree, and let $D(t)$ be the subset of the example set D that is represented by t. [illustration]

Possible criteria for a splitting of $X(t)$:

1. Size of $D(t)$.

2. Purity of $D(t)$.

3. Impurity reduction of $D(t)$.

Impurity Functions

Splitting

Let t be a leaf node of an incomplete decision tree, and let $D(t)$ be the subset of the example set D that is represented by t. [illustration]

Possible criteria for a splitting of $X(t)$:

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

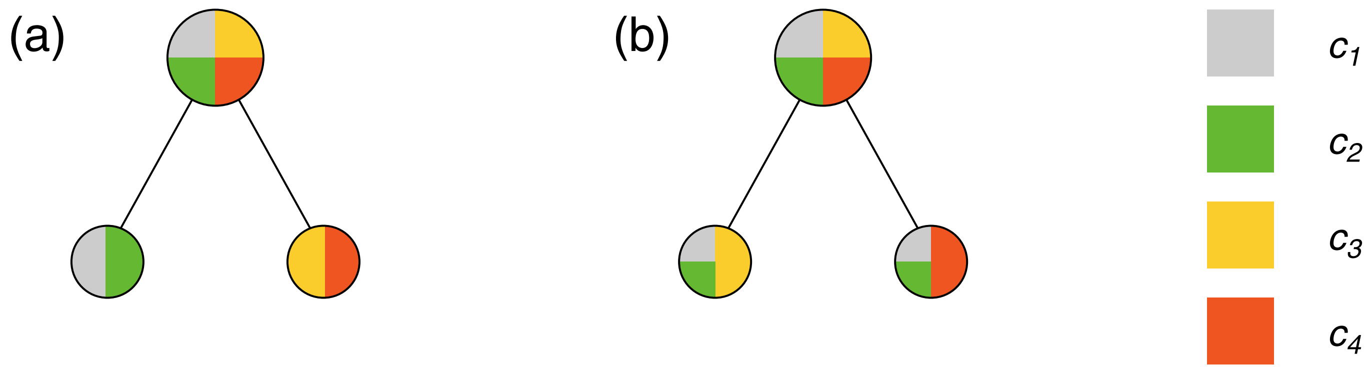
2. Purity of $D(t)$.
   $D(t)$ is not split if all examples in $D(t)$ are members of the same class.

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

Impurity Functions

Splitting (continued)

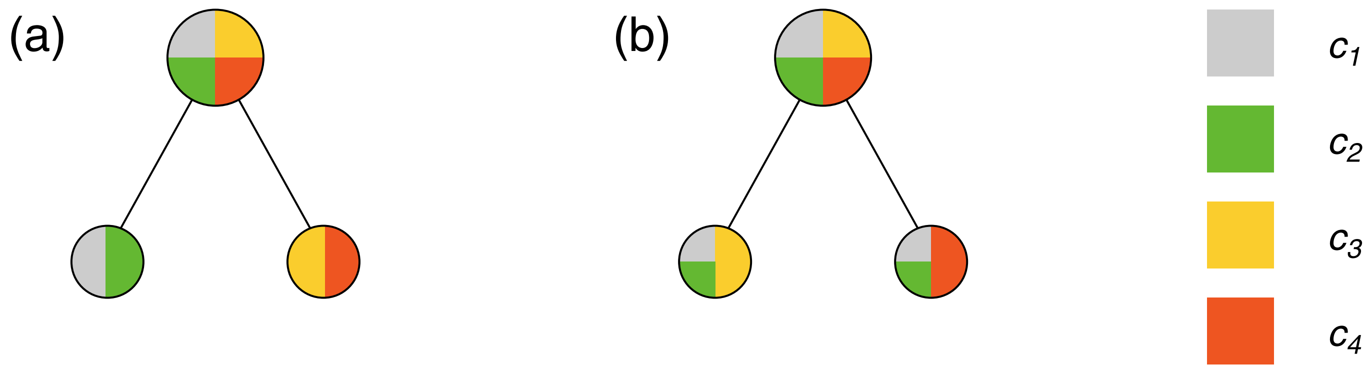
Let X be a set of feature vectors, $D \subseteq X$ a set of examples, and $C = \{c_1, c_2, c_3, c_4\}$ a set of classes. Distribution of D for two possible splittings of X :



Impurity Functions

Splitting (continued)

Let X be a set of feature vectors, $D \subseteq X$ a set of examples, and $C = \{c_1, c_2, c_3, c_4\}$ a set of classes. Distribution of D for two possible splittings of X :



- Splitting (a) minimizes the *impurity* of the subsets of D in the leaf nodes and should be preferred over splitting (b). This argument presumes that the misclassification costs are independent of the classes.
- The impurity is a function defined on $\mathcal{P}(D)$, the set of all subsets of an example set D.

Impurity Functions

Definition 4 **(Impurity Function ι)**

Let $k \in \mathbf{N}$. An impurity function $\iota : [0;1]^k \rightarrow \mathbf{R}$ is a function defined on the standard $k{-}1$-simplex, denoted Δ^{k-1}, for which the following properties hold:

(a) ι becomes minimum at points $(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, \ldots, 0, 1)$.

(b) ι is symmetric with regard to its arguments, $p_1, \ldots, p_k$.

(c) ι becomes maximum at point $(1/k, \ldots, 1/k)$.

Impurity Functions

Definition 5 (Impurity of an Example Set $\iota(D)$)

Let X be a set of feature vectors, $C = \{c_1, \ldots, c_k\}$ a set of classes, $c : X \to C$ the ideal classifier for X, and $D \subseteq X \times C$ a set of examples. Moreover, let $\iota : [0;1]^k \to \mathbf{R}$ be an impurity function. Then, the impurity of D, denoted as $\iota(D)$, is defined as follows:

$$\iota(D) = \iota\left(\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|}, \ldots, \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_k\}|}{|D|}\right)$$

Impurity Functions

Definition 5 (Impurity of an Example Set $\iota(D)$)

Let X be a set of feature vectors, $C = \{c_1, \ldots, c_k\}$ a set of classes, $c : X \to C$ the ideal classifier for X, and $D \subseteq X \times C$ a set of examples. Moreover, let $\iota : [0;1]^k \to \mathbf{R}$ be an impurity function. Then, the impurity of D, denoted as $\iota(D)$, is defined as follows:

$$\iota(D) = \iota\left(\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|}, \ldots, \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_k\}|}{|D|}\right)$$

Definition 6 (Impurity Reduction $\Delta\iota$)

Let $D_1, \ldots, D_s$ be a splitting of an example set D, which is induced by a splitting of X. Then, the resulting impurity reduction, denoted as $\Delta\iota(D, \{D_1, \ldots, D_s\})$, is defined as follows:

$$\Delta\iota\big(D,\ \{D_1, \ldots, D_s\}\big) \ = \ \iota(D) - \sum_{j=1}^{s} \frac{|D_j|}{|D|} \cdot \iota(D_j)$$

Remarks:

- The standard $k-1$-simplex comprises all k-tuples with non-negative elements that sum to 1: $\Delta^{k-1} = \left\{(p_1, \ldots, p_k) \in \mathbf{R}^k : \sum_{i=1}^{k} p_i = 1 \text{ and } p_i \geq 0 \text{ for all } i\right\}$
- Observe the different domains of the impurity function ι in the definitions for ι and $\iota(D)$, namely, $[0;1]^k$ and D. The domains correspond to each other: the set of examples, D, defines via its class portions an element from $[0;1]^k$ and vice versa.
- The properties in the definition of the impurity function ι suggest to minimize the external path length of T with respect to D in order to minimize the overall impurity characteristics of T.
- Within the *DT-construct* algorithm usually a greedy strategy (local optimization) is employed to minimize the overall impurity characteristics of a decision tree T.

Impurity Functions

Impurity Functions Based on the Misclassification Rate

Definition for two classes [impurity function] :

$$\iota_{\mathit{misclass}}(p_1, p_2) = 1 - \max\{p_1, p_2\} = \begin{cases} p_1 & \text{if } 0 \leq p_1 \leq 0.5 \\ 1 - p_1 & \text{otherwise} \end{cases}$$

Impurity Functions

Impurity Functions Based on the Misclassification Rate

Definition for two classes [impurity function] :

$$\iota_{misclass}(p_1, p_2) = 1 - \max\{p_1, p_2\} = \begin{cases} p_1 & \text{if } 0 \leq p_1 \leq 0.5 \\ 1 - p_1 & \text{otherwise} \end{cases}$$

$$\iota_{misclass}(D) = 1 - \max\left\{ \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|}, \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_2\}|}{|D|} \right\}$$

Impurity Functions

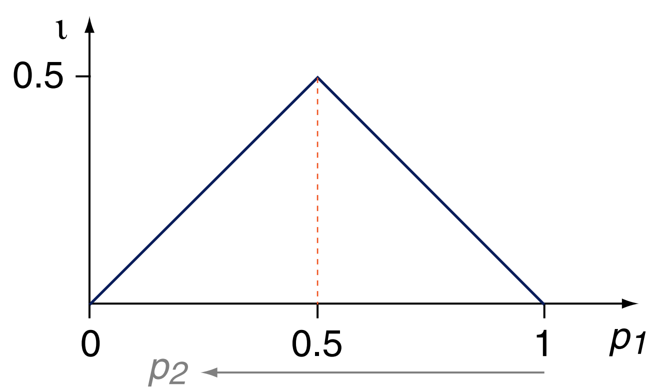
Impurity Functions Based on the Misclassification Rate

Definition for two classes [impurity function] :

$$\iota_{misclass}(p_1, p_2) = 1 - \max\{p_1, p_2\} = \begin{cases} p_1 & \text{if } 0 \leq p_1 \leq 0.5 \\ 1 - p_1 & \text{otherwise} \end{cases}$$

$$\iota_{misclass}(D) = 1 - \max\left\{\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|}, \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_2\}|}{|D|}\right\}$$

Graph of the function $\iota_{misclass}(p_1, 1 - p_1)$:



[Graphs: misclassification, entropy, gini]

Impurity Functions

Impurity Functions Based on the Misclassification Rate (continued)

Definition for k classes:

$$\iota_{\textit{misclass}}(p_1, \ldots, p_k) = 1 - \max_{i=1,\ldots,k} p_i$$

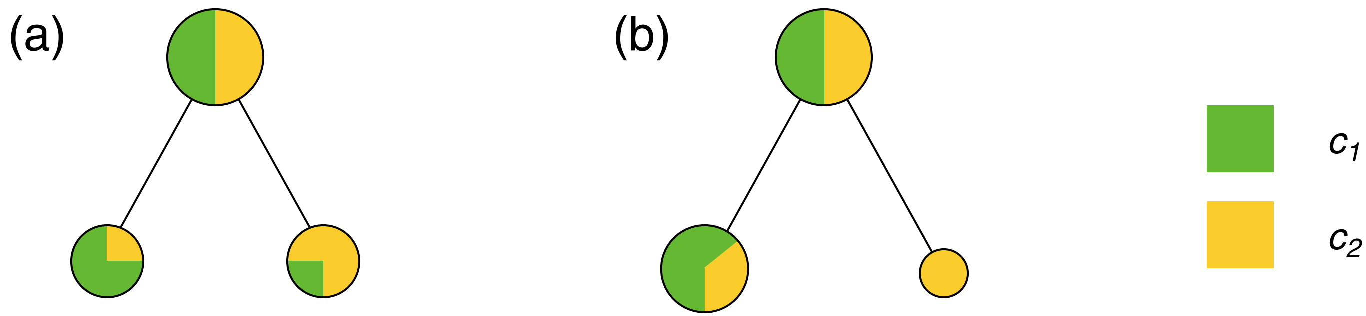
$$\iota_{\textit{misclass}}(D) = 1 - \max_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c\}|}{|D|}$$

Impurity Functions

Impurity Functions Based on the Misclassification Rate (continued)

Problems:

- $\Delta\iota_{misclass} = 0$ may hold for all possible splittings.
- The impurity function that is induced by the misclassification rate underestimates pure nodes, as illustrated in splitting (b):

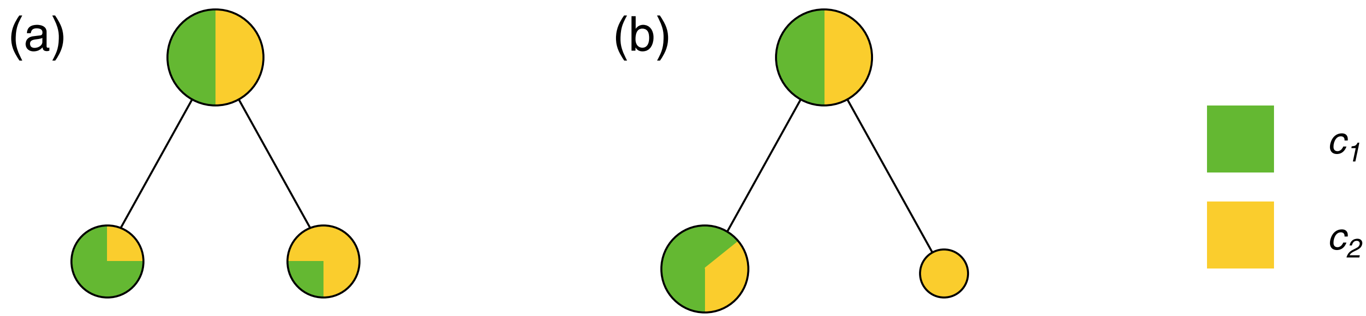


Impurity Functions

Impurity Functions Based on the Misclassification Rate (continued)

Problems:

- $\Delta\iota_{misclass} = 0$ may hold for all possible splittings.
- The impurity function that is induced by the misclassification rate underestimates pure nodes, as illustrated in splitting (b):



$$\Delta\iota_{misclass} = \iota_{misclass}(D) - \left(\frac{|D_1|}{|D|} \cdot \iota_{misclass}(D_1) + \frac{|D_2|}{|D|} \cdot \iota_{misclass}(D_2)\right)$$

left splitting: $\Delta\iota_{misclass} = \frac{1}{2} - (\frac{1}{2} \cdot \frac{1}{4} + \frac{1}{2} \cdot \frac{1}{4}) = \frac{1}{4}$

right splitting: $\Delta\iota_{misclass} = \frac{1}{2} - (\frac{3}{4} \cdot \frac{1}{3} + \frac{1}{4} \cdot 0) = \frac{1}{4}$

Impurity Functions

Definition 7 (Strict Impurity Function)

Let $\iota : [0;1]^k \to \mathbf{R}$ be an impurity function and let $\mathbf{p}$, $\mathbf{p}' \in \Delta^{k-1}$. Then ι is called strict, if it is strictly concave:

(c) $\to$ (c') $\quad \iota\big(\,\lambda\mathbf{p} + (1-\lambda)\mathbf{p}'\,\big) \;>\; \lambda\,\iota(\mathbf{p}) \;+\; (1-\lambda)\,\iota(\mathbf{p}'), \quad 0 < \lambda < 1,\; \mathbf{p} \neq \mathbf{p}'$

Impurity Functions

Definition 7 (Strict Impurity Function)

Let $\iota : [0;1]^k \to \mathbf{R}$ be an impurity function and let $\mathbf{p}$, $\mathbf{p}' \in \Delta^{k-1}$. Then ι is called strict, if it is strictly concave:

(c) $\to$ (c') $\quad \iota\big(\,\lambda\mathbf{p} + (1-\lambda)\mathbf{p}'\,\big) \;>\; \lambda\,\iota(\mathbf{p}) \;+\; (1-\lambda)\,\iota(\mathbf{p}'), \quad 0 < \lambda < 1,\ \mathbf{p} \neq \mathbf{p}'$

Lemma 8

Let ι be a *strict* impurity function and let $D_1, \ldots, D_s$ be a splitting of an example set D, which is induced by a splitting of X. Then the following inequality holds:

$$\Delta\iota(D, \{D_1, \ldots, D_s\}) \geq 0$$

The equality is given iff for all $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, s\}$ holds:

$$\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_i\}|}{|D|} = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D_j : c(\mathbf{x}) = c_i\}|}{|D_j|}$$

Remarks:

- Equality means that the splitting of D resembles exactly the class distribution of D.
- Strict concavity entails Property (c) of the impurity function definition.
- For two classes, strict concavity means $\iota(p_1, 1-p_1) > 0$, where $0 < p_1 < 1$.
- If ι is a twice differentiable function, strict concavity is equivalent with a negative definite Hessian of ι.
- With properly chosen coefficients, polynomials of second degree fulfill the Properties (a) and (b) of the impurity function definition as well as strict concavity. See impurity functions based on the Gini index in this regard.
- The impurity function that is induced by the misclassification rate is concave, but it is not strictly concave.
- The proof of Lemma 8 exploits the strict concavity property of ι.

Impurity Functions

Impurity Functions Based on Entropy

Definition 9 (Entropy)

Let A denote an event and let $P(A)$ denote the occurrence probability of A. Then the entropy (self-information, information content) of A is defined as $-\log_2(P(A))$.

Let $\mathcal{A}$ be an experiment with the exclusive outcomes (events) $A_1, \ldots, A_k$. Then the mean information content of $\mathcal{A}$, denoted as $H(\mathcal{A})$, is called Shannon entropy or entropy of experiment $\mathcal{A}$ and is defined as follows:

$$H(\mathcal{A}) = -\sum_{i=1}^{k} P(A_i) \cdot \log_2(P(A_i))$$

Remarks:

- The smaller the occurrence probability of an event, the larger is its entropy. An event that is certain has zero entropy.
- The Shannon entropy combines the entropies of an experiment's outcomes, using the outcome probabilities as weights.
- In the entropy definition we stipulate the identity $0 \cdot \log_2(0) = 0$.
- Related. Entropy encoding methods such as Huffman coding. [Wikipedia]
- Related. The perplexity of a discrete probability distribution p is defined as $2^{H(p)}$. [Wikipedia]

Impurity Functions

Impurity Functions Based on Entropy (continued)

Definition 10 (Conditional Entropy, Information Gain)

Let $\mathcal{A}$ be an experiment with the exclusive outcomes (events) $A_1, \ldots, A_k$, and let $\mathcal{B}$ be another experiment with the outcomes $B_1, \ldots, B_s$. Then the conditional entropy of the combined experiment $(\mathcal{A} \mid \mathcal{B})$ is defined as follows:

$$H(\mathcal{A} \mid \mathcal{B}) = \sum_{j=1}^{s} P(B_j) \cdot H(\mathcal{A} \mid B_j),$$

$$\text{where} \quad H(\mathcal{A} \mid B_j) = -\sum_{i=1}^{k} P(A_i \mid B_j) \cdot \log_2(P(A_i \mid B_j))$$

Definition 10 (Conditional Entropy, Information Gain)

Let $\mathcal{A}$ be an experiment with the exclusive outcomes (events) $A_1, \ldots, A_k$, and let $\mathcal{B}$ be another experiment with the outcomes $B_1, \ldots, B_s$. Then the conditional entropy of the combined experiment $(\mathcal{A} \mid \mathcal{B})$ is defined as follows:

$$H(\mathcal{A} \mid \mathcal{B}) = \sum_{j=1}^{s} P(B_j) \cdot H(\mathcal{A} \mid B_j),$$

where $$H(\mathcal{A} \mid B_j) = -\sum_{i=1}^{k} P(A_i \mid B_j) \cdot \log_2(P(A_i \mid B_j))$$

Impurity Functions

Impurity Functions Based on Entropy (continued)

Definition 10 (Conditional Entropy, Information Gain)

Let $\mathcal{A}$ be an experiment with the exclusive outcomes (events) $A_1, \ldots, A_k$, and let $\mathcal{B}$ be another experiment with the outcomes $B_1, \ldots, B_s$. Then the conditional entropy of the combined experiment $(\mathcal{A} \mid \mathcal{B})$ is defined as follows:

$$H(\mathcal{A} \mid \mathcal{B}) = \sum_{j=1}^{s} P(B_j) \cdot H(\mathcal{A} \mid B_j),$$

$$\text{where} \quad H(\mathcal{A} \mid B_j) = -\sum_{i=1}^{k} P(A_i \mid B_j) \cdot \log_2(P(A_i \mid B_j))$$

The information gain due to experiment $\mathcal{B}$ is defined as follows:

$$H(\mathcal{A}) - H(\mathcal{A} \mid \mathcal{B}) = H(\mathcal{A}) - \sum_{j=1}^{s} P(B_j) \cdot H(\mathcal{A} \mid B_j)$$

Remarks [Bayes for classification] :

- Information gain is defined as reduction in entropy.

- In the context of decision trees, experiment $\mathcal{A}$ corresponds to classifying feature vector $\mathbf{x}$ with regard to the target concept. A possible question, whose answer will inform us about which event $A_i \in \mathcal{A}$ occurred, is the following: "Does $\mathbf{x}$ belong to class c_i?"
  Likewise, experiment $\mathcal{B}$ corresponds to evaluating feature B of feature vector $\mathbf{x}$. A possible question, whose answer will inform us about which event $B_j \in \mathcal{B}$ occurred, is the following: "Does $\mathbf{x}$ have value b_j for feature B?"

- Rationale: Typically, the events "target concept class" and "feature value" are statistically dependent. Hence, the entropy of the event "$\mathbf{x}$ belongs to class c_i" will become smaller if we learn about the value of some feature of $\mathbf{x}$ (recall that the class of $\mathbf{x}$ is unknown).
  We experience an information gain with regard to the outcome of experiment $\mathcal{A}$, which is rooted in our information about the outcome of experiment $\mathcal{B}$. Under no circumstances the information gain will be negative; the information gain is zero if the involved events are *conditionally independent*:

$$P(A_i) = P(A_i \mid B_j), \quad i \in \{1, \ldots, k\},\ j \in \{1, \ldots, s\},$$

  which leads to a split as specified as the special case in Lemma 8.

Remarks (continued) :

- Since $H(\mathcal{A})$ is constant, the feature that provides the maximum information gain (= the maximally informative feature) is given by the minimization of $H(\mathcal{A} \mid \mathcal{B})$.

- The expanded form of $H(\mathcal{A} \mid \mathcal{B})$ reads as follows:

$$H(\mathcal{A} \mid \mathcal{B}) = -\sum_{j=1}^{s} P(B_j) \cdot \sum_{i=1}^{k} P(A_i \mid B_j) \cdot \log_2(P(A_i \mid B_j))$$

Impurity Functions

Impurity Functions Based on Entropy (continued)

Definition for two classes [impurity function] :

$$\iota_{\mathit{entropy}}(p_1, p_2) = -(p_1 \cdot \log_2(p_1) + p_2 \cdot \log_2(p_2))$$

Impurity Functions

Impurity Functions Based on Entropy (continued)

Definition for two classes [impurity function] :

$$\iota_{entropy}(p_1, p_2) = -(p_1 \cdot \log_2(p_1) + p_2 \cdot \log_2(p_2))$$

$$\iota_{entropy}(D) = -\left(\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|} \cdot \log_2 \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|} + \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_2\}|}{|D|} \cdot \log_2 \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_2\}|}{|D|}\right)$$

Impurity Functions

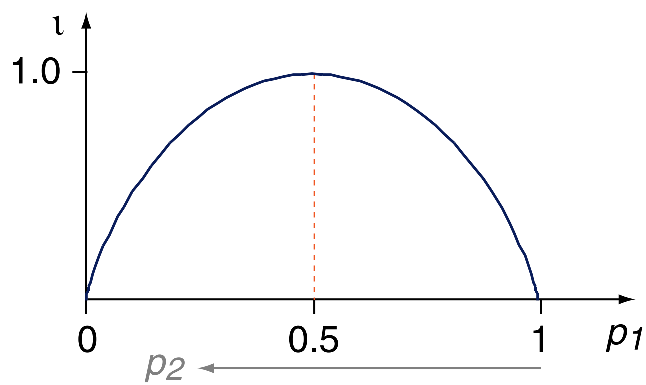
Impurity Functions Based on Entropy (continued)

Definition for two classes [impurity function] :

$$\iota_{entropy}(p_1, p_2) = -(p_1 \cdot \log_2(p_1) + p_2 \cdot \log_2(p_2))$$

$$\iota_{entropy}(D) = -\left(\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|} \cdot \log_2 \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|} + \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_2\}|}{|D|} \cdot \log_2 \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_2\}|}{|D|}\right)$$

Graph of the function $\iota_{entropy}(p_1, 1 - p_1)$:

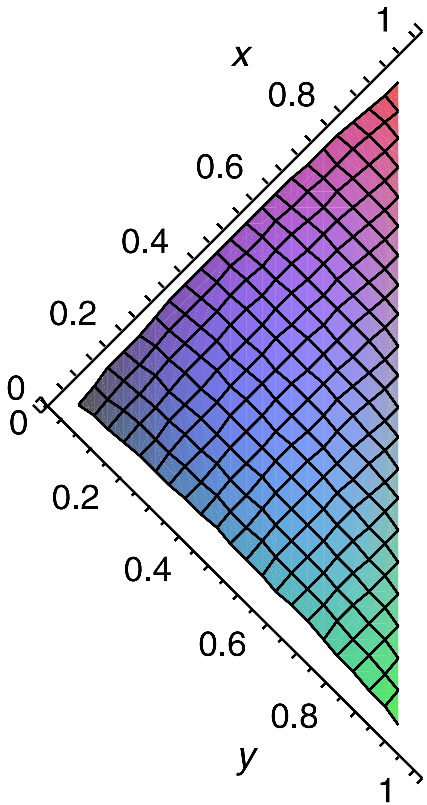


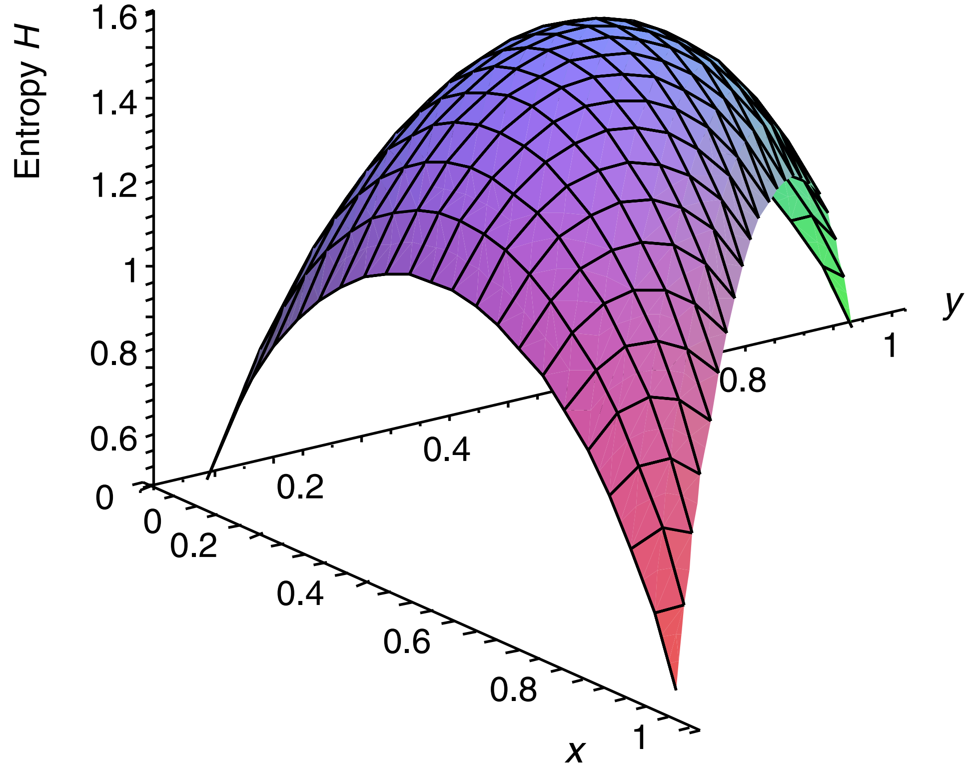
[Graphs: misclassification, Entropy, gini]

Impurity Functions

Impurity Functions Based on Entropy (continued)

Graph of the function $\iota_{entropy}(p_1, p_2, 1 - p_1 - p_2)$:





Impurity Functions

Impurity Functions Based on Entropy (continued)

Definition for k classes:

$$\iota_{\mathit{entropy}}(p_1, \ldots, p_k) = -\sum_{i=1}^{k} p_i \cdot \log_2(p_i)$$

$$\iota_{\mathit{entropy}}(D) = -\sum_{i=1}^{k} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_i\}|}{|D|} \cdot \log_2 \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_i\}|}{|D|}$$

Impurity Functions

Impurity Functions Based on Entropy (continued)

$\Delta\iota_{\mathit{entropy}}$ corresponds to the information gain $H(\mathcal{A}) - H(\mathcal{A} \mid \mathcal{B})$:

$$\Delta\iota_{\mathit{entropy}} = \underbrace{\iota_{\mathit{entropy}}(D)}_{H(\mathcal{A})} \quad - \quad \underbrace{\sum_{j=1}^{s} \frac{|D_j|}{|D|} \cdot \iota_{\mathit{entropy}}(D_j)}_{H(\mathcal{A} \mid \mathcal{B})}$$

Impurity Functions

Impurity Functions Based on Entropy (continued)

$\Delta\iota_{entropy}$ corresponds to the information gain $H(\mathcal{A}) - H(\mathcal{A} \mid \mathcal{B})$:

$$\Delta\iota_{entropy} = \underbrace{\iota_{entropy}(D)}_{H(\mathcal{A})} \quad - \quad \underbrace{\sum_{j=1}^{s} \frac{|D_j|}{|D|} \cdot \iota_{entropy}(D_j)}_{H(\mathcal{A} \mid \mathcal{B})}$$

Derivation:

- $A_i,\ i = 1, \ldots, k$, denotes the event that $\mathbf{x} \in X(t)$ belongs to class c_i.
  The experiment $\mathcal{A}$ corresponds to the classification $c : X(t) \to C$.

- $B_j,\ j = 1, \ldots, s$, denotes the event that $\mathbf{x} \in X(t)$ has value b_j for feature B.
  The experiment $\mathcal{B}$ corresponds to evaluating feature B and entails the following splitting:

  $$X(t) = X(t_1) \cup \ldots \cup X(t_s) = \{\mathbf{x} \in X(t) : \mathbf{x}|_B = b_1\} \ \cup \ldots \cup \ \{\mathbf{x} \in X(t) : \mathbf{x}|_B = b_s\}$$

- $\iota_{entropy}(D) \ = \ \iota_{entropy}(P(A_1), \ldots, P(A_k)) \ = \ -\sum_{i=1}^{k} P(A_i) \cdot \log_2(P(A_i)) \ = \ H(\mathcal{A})$

- $\frac{|D_j|}{|D|} \cdot \iota_{entropy}(D_j) \ = \ P(B_j) \cdot \iota_{entropy}(P(A_1 \mid B_j), \ldots, P(A_k \mid B_j)),\ j = 1, \ldots, s$

- $P(A_i), P(B_j), P(A_i \mid B_j)$ are estimated as relative frequencies based on D.

Impurity Functions

Impurity Functions Based on the Gini Index

Definition for two classes [impurity function] :

$$\iota_{Gini}(p_1, p_2) = 1 - ({p_1}^2 + {p_2}^2) = 2 \cdot p_1 \cdot p_2$$

Impurity Functions

Impurity Functions Based on the Gini Index

Definition for two classes [impurity function] :

$$\iota_{Gini}(p_1, p_2) = 1 - ({p_1}^2 + {p_2}^2) = 2 \cdot p_1 \cdot p_2$$

$$\iota_{Gini}(D) = 2 \cdot \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|} \cdot \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_2\}|}{|D|}$$

Impurity Functions

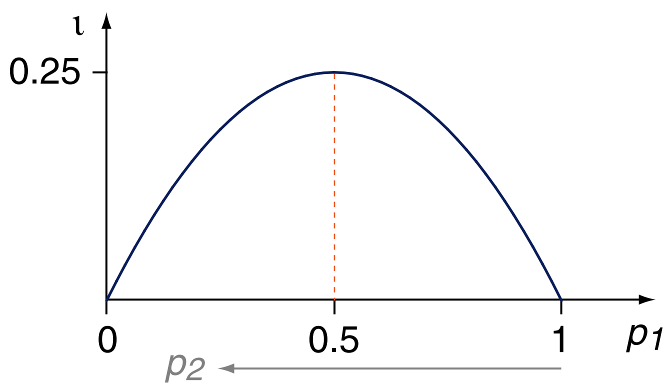
Impurity Functions Based on the Gini Index

Definition for two classes [impurity function] :

$$\iota_{Gini}(p_1, p_2) = 1 - ({p_1}^2 + {p_2}^2) = 2 \cdot p_1 \cdot p_2$$

$$\iota_{Gini}(D) = 2 \cdot \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|} \cdot \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_2\}|}{|D|}$$

Graph of the function $\iota_{Gini}(p_1, 1 - p_1)$:



[Graphs: misclassification, entropy, Gini]

Impurity Functions

Impurity Functions Based on the Gini Index (continued)

Definition for k classes:

$$\iota_{Gini}(p_1, \ldots, p_k) = 1 - \sum_{i=1}^{k} (p_i)^2$$

$$\iota_{Gini}(D) = \left(\sum_{i=1}^{k} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_i\}|}{|D|} \right)^2 - \sum_{i=1}^{k} \left(\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_i\}|}{|D|} \right)^2$$

$$= 1 - \sum_{i=1}^{k} \left(\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_i\}|}{|D|} \right)^2$$

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]

Setting:

- X is a set of feature vectors.
- 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.

Todo:

- Approximate $c(\mathbf{x})$, which is implicitly given via D, with a decision tree.

Decision Tree Algorithms

ID3 Algorithm [Quinlan 1986] [CART Algorithm]

Setting:

- X is a set of feature vectors.
- 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.

Todo:

- Approximate $c(\mathbf{x})$, which is implicitly given via D, with a decision tree.

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 information gain.

Decision Tree Algorithms

ID3 Algorithm [Mitchell 1997] [algorithm template]

ID3(D, Features, 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 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 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, Features \ {A*}, Target).

7. Return t.

Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

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

```
t = createNode()
label(t) = mostCommonClass(D, Target)
IF ∀⟨x, c(x)⟩ ∈ D : c(x) = c THEN return(t) ENDIF
IF Features = ∅ THEN return(t) ENDIF
```

Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

ID3$(D$, *Features*, *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** *Features* $= \emptyset$ **THEN** *return*(t) **ENDIF**
5. $A^* = \text{argmax}_{A \in \textit{Features}}(\textit{informationGain}(D, A))$
6.
7.

Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

ID3$(D,$ *Features*, *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** *Features* $= \emptyset$ **THEN** *return*(t) **ENDIF**
5. $A^* = \text{argmax}_{A \in \textit{Features}}(\textit{informationGain}(D, A))$
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*$(D_a,$ *Features* $\setminus \{A^*\},$ *Target*$))$

   ENDIF

   ENDDO
7. *return*(t)

Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

ID3$(D,$ *Features*, *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** *Features* $= \emptyset$ **THEN** *return*(t) **ENDIF**
5. $A^* = \text{argmax}_{A \in \textit{Features}}(\textit{informationGain}(D, A))$
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*$(D_a,$ *Features* $\setminus \{A^*\},$ *Target*$))$

   ENDIF

   ENDDO
7. *return*(t)

Remarks:

- "*Target*" designates the class label 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) class labels.

- 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	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

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

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	edible
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 features:

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

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

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:

feature: Points

yes

Color	Size	Edibility
red	small	toxic
brown	large	edible

no

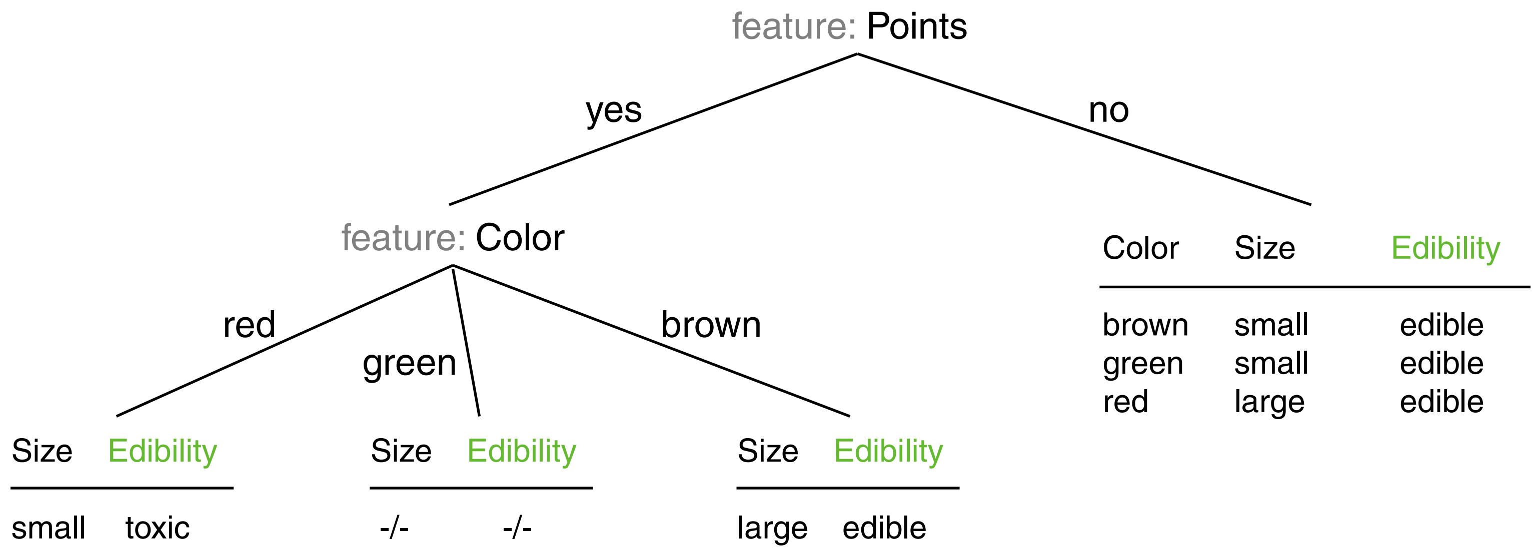
Color	Size	Edibility
brown	small	edible
green	small	edible
red	large	edible

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

Decision Tree Algorithms

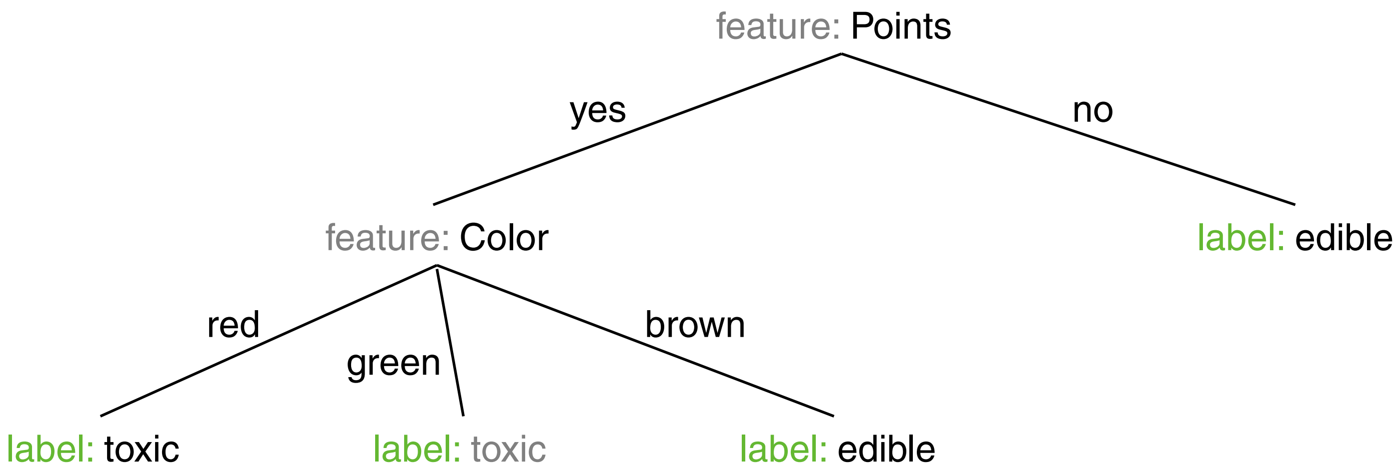
ID3 Algorithm: Example (continued)

Decision tree before the second recursion step:



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

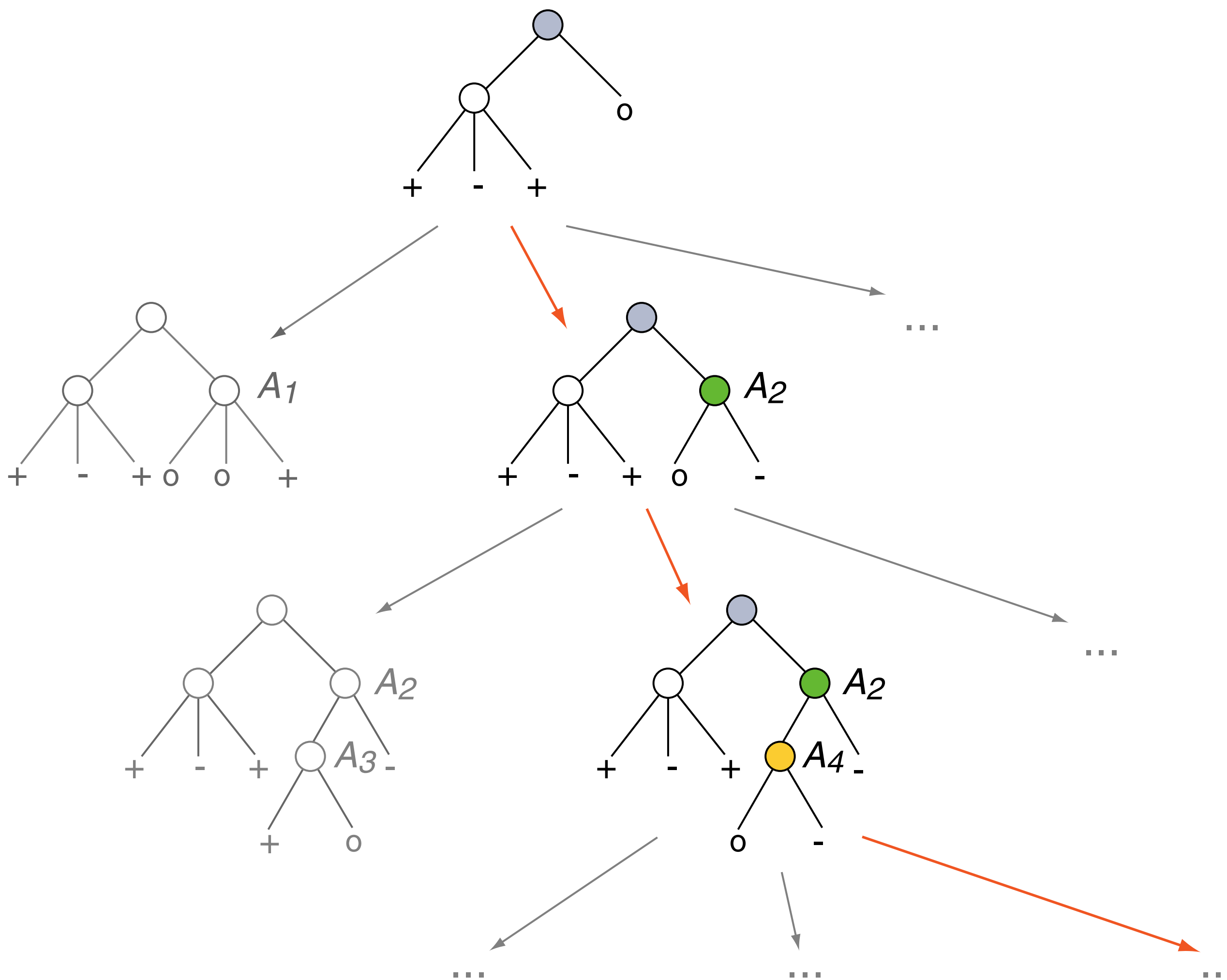
Final decision tree after second recursion step:



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
  - ➜ 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:

- 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 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. 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]

Setting:

- X is a set of feature vectors. No restrictions are presumed for the features' measurement scales.
- C is a set of classes.
- $c : X \rightarrow C$ 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.

Todo:

- Approximate $c(\mathbf{x})$, which is implicitly given via D, with a decision tree.

Decision Tree Algorithms

CART Algorithm [Breiman 1984] [ID3 Algorithm]

Setting:

- X is a set of feature vectors. 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.

Todo:

- Approximate $c(\mathbf{x})$, which is implicitly given via D, with a decision tree.

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 splitting 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 splitting 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 $\Delta\iota$:

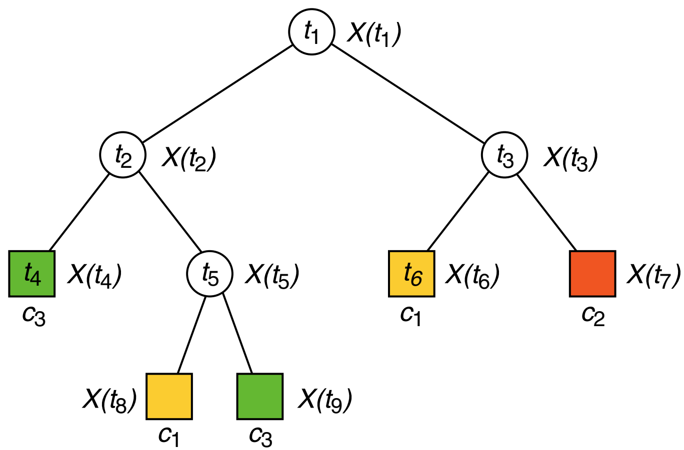
$$\Delta\iota\big(D(t),\ \{D(t_L), D(t_R)\}\big) \ = \ \iota(t) - \frac{|D_L|}{|D|} \cdot \iota(t_L) - \frac{|D_R|}{|D|} \cdot \iota(t_R),$$

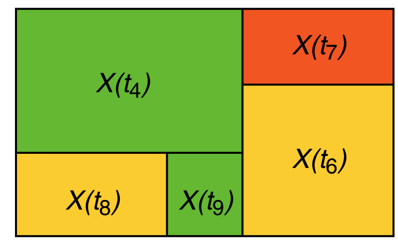
   where t_L and t_R denote the left and right successor of t.

Decision Tree Algorithms

CART Algorithm (continued)

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





$X = X(t_1)$

By a sequence of splittings the feature space $\mathbf{X}$ is split into rectangles that are parallel to the two axes.

Chapter ML:III

III. Decision Trees

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

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:

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

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

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:

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

where $\mathit{Err}^*(h)$ denotes the true misclassification rate of h, while $\mathit{Err}(h, D)$ 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 not representative
- D is too small and hence pretends unrealistic data properties

Decision Tree Pruning

Overfitting (continued)

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:

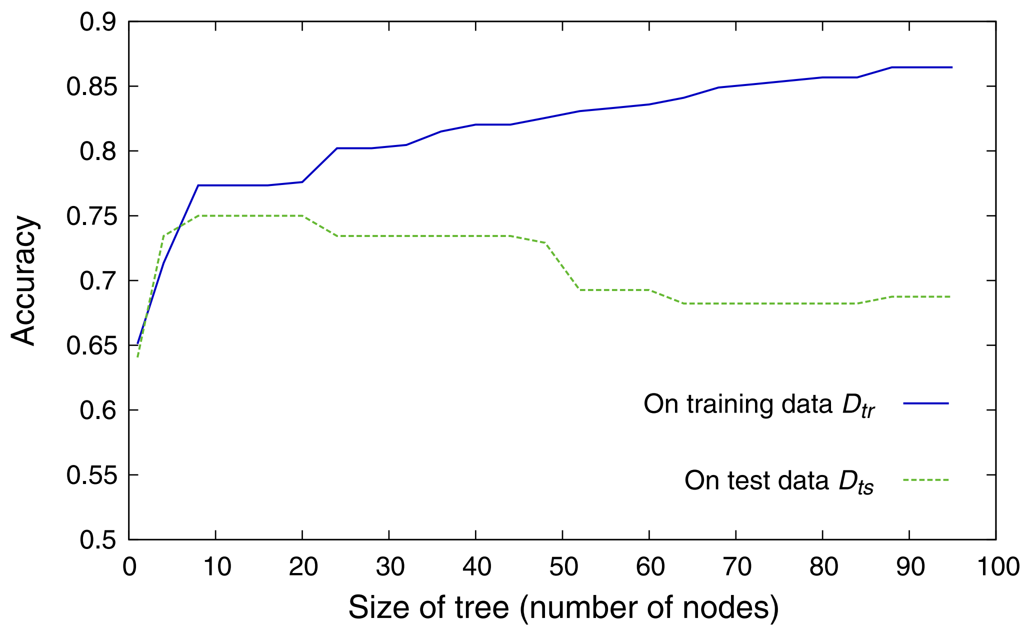
$$\mathit{Err}(h, D_{tr}) < \mathit{Err}(h', D_{tr}) \quad \text{and} \quad \mathit{Err}(h, D_{ts}) > \mathit{Err}(h', D_{ts})$$

Decision Tree Pruning

Overfitting (continued)

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:

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



[Mitchell 1997]

Remarks:

- Accuracy is the percentage of correctly classified examples.
- When does *Err*(T, D_{tr}) of a decision tree T become zero?
- The training error *Err*(T, D_{tr}) 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 11

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:

$$Err_{cost}(t, D(t)) \;\geq \sum_{i \in \{1,\ldots,s\}} Err_{cost}(t_i, D(t_i))$$

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

Decision Tree Pruning

Overfitting (continued)

Proof (sketch)

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

$$
\mathit{Err}_{\mathit{cost}}(t, D(t)) - \textstyle\sum_{i \in \{1,\ldots,k_s\}} \mathit{Err}_{\mathit{cost}}(t_i, D(t_i)) =
$$

$$
\sum_{i \in \{1,\ldots,k_s\}} \left(\sum_{c \in C} p(c, t_i) \cdot \mathit{cost}(\mathit{label}(t) \mid c) - \min_{c' \in C} \sum_{c \in C} p(c, t_i) \cdot \mathit{cost}(c' \mid c) \right)
$$

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

Remarks:

- The lemma does also hold if the misclassification rate is used to evaluate 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.

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 criteria] :

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.

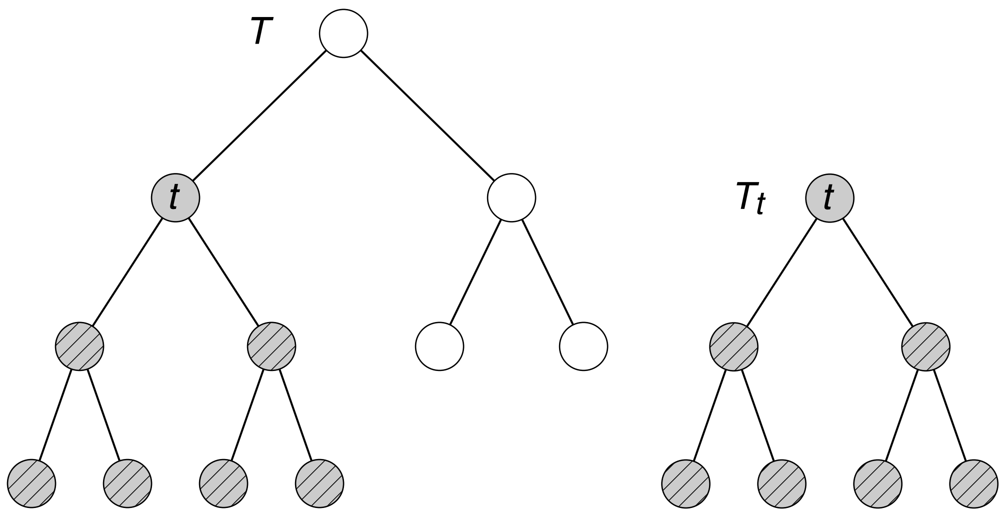
Decision Tree Pruning

(b) Pruning (continued)

Definition 12 (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 13 (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 13 (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]

Decision Tree Pruning

(b) Pruning (continued)

Definition 13 (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]

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

(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*$(T', D_{vd}) \leq$ *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

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

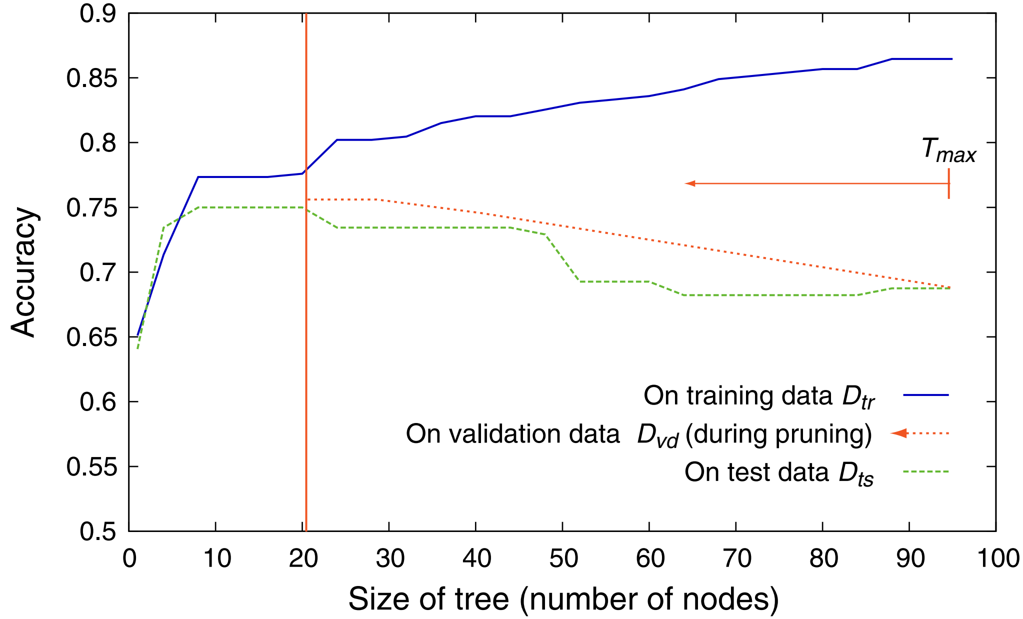
Problem:

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

(b) Pruning: Reduced Error Pruning (continued)



[Mitchell 1997]

Decision Tree Pruning

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