Learning Decision Trees

Decision trees provide a very popular and efficient hypothesis space.

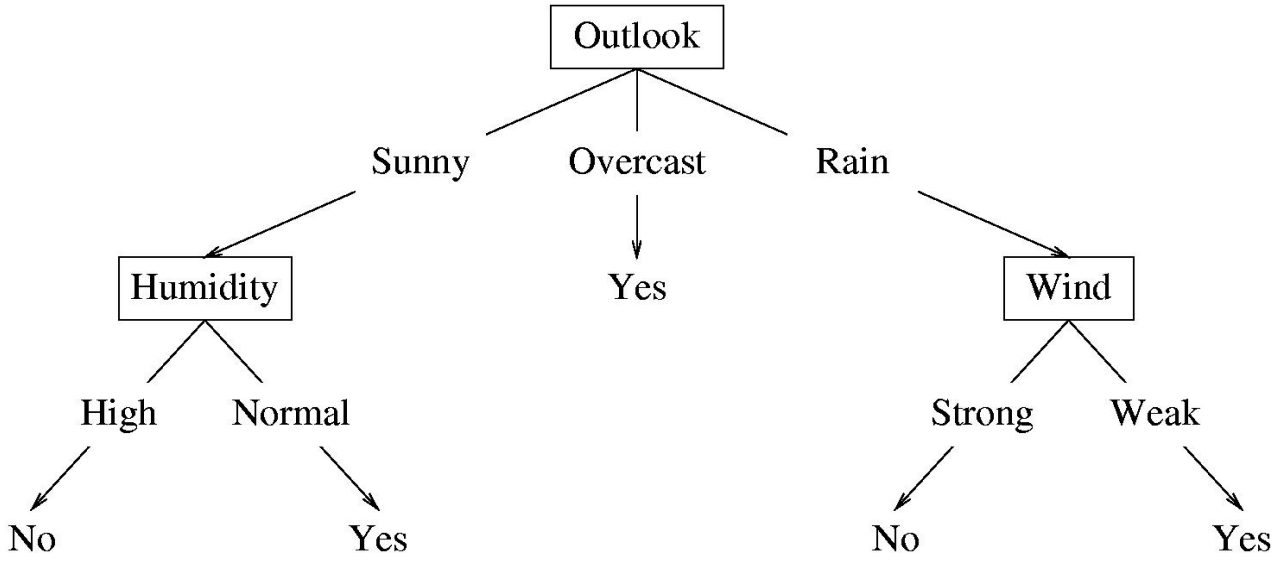
- **Variable Size**. Any boolean function can be represented.
- **Deterministic**.
- **Discrete and Continuous Parameters**.

Learning algorithms for decision trees can be described as

- **Constructive Search**. The tree is built by adding nodes.
- **Eager.**
- **Batch** (although online algorithms do exist).

Decision Tree Hypothesis Space

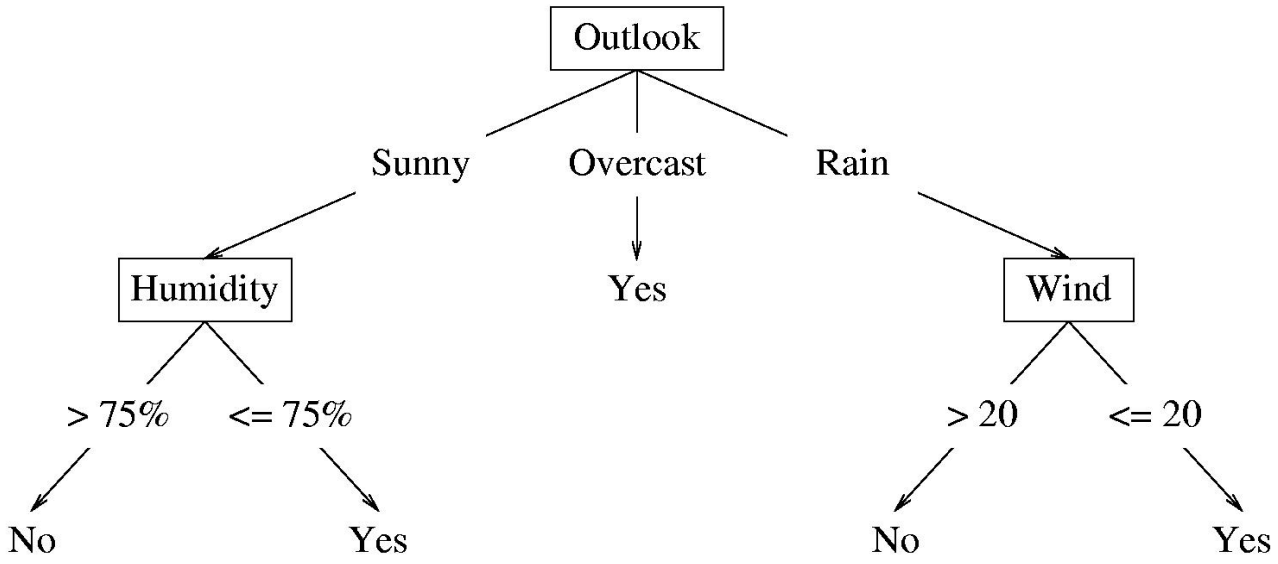
- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



Suppose the features are **Outlook** (x_1), **Temperature** (x_2), **Humidity** (x_3), and **Wind** (x_4). Then the feature vector $\mathbf{x} = (Sunny, Hot, High, Strong)$ will be classified as **No**. The **Temperature** feature is irrelevant.

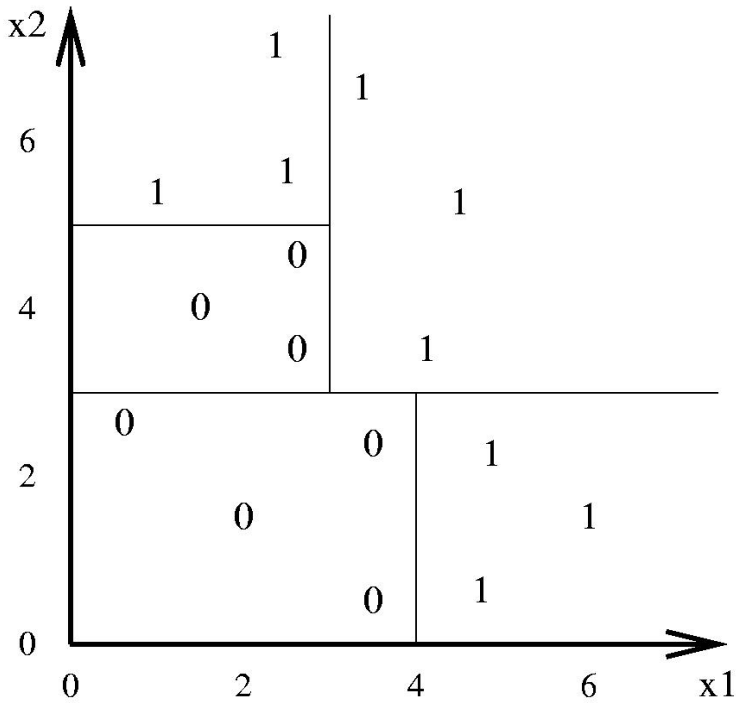
Decision Tree Hypothesis Space

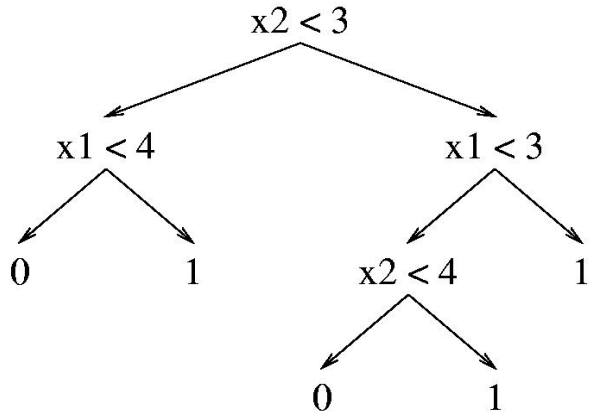
If the features are continuous, internal nodes may test the value of a feature against a threshold.



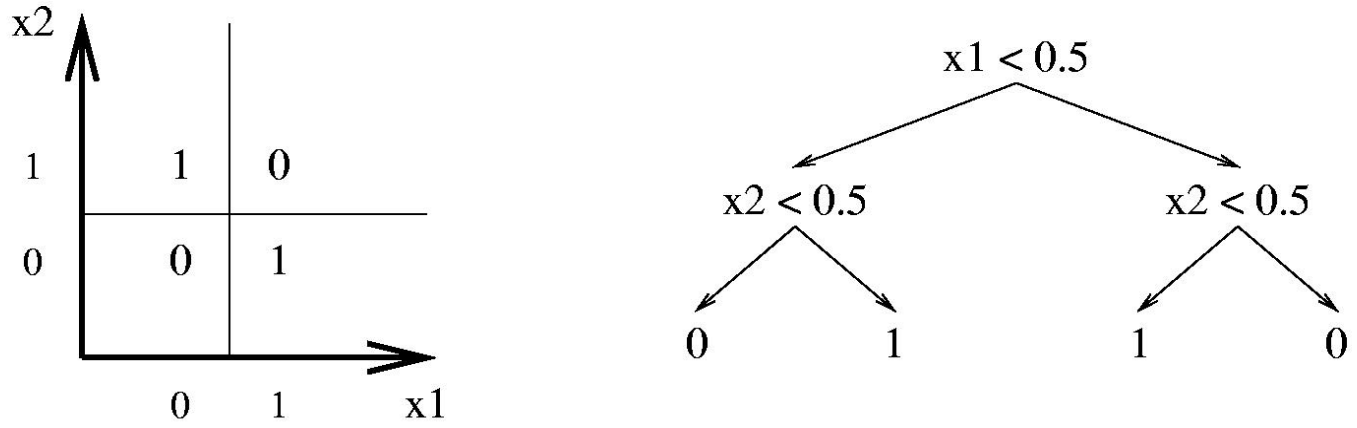
Decision Tree Decision Boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





Decision Trees Can Represent Any Boolean Function



The tree will in the worst case require exponentially many nodes, however.

Decision Trees Provide Variable-Size Hypothesis Space

As the number of nodes (or depth) of tree increases, the hypothesis space grows

- **depth 1** ("decision stump") can represent any boolean function of one feature.
- **depth 2** Any boolean function of two features; some boolean functions involving three features (e.g., $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3)$
- **etc.**

Decision Trees Basics

Specification of Classification Problems [ML Introduction]

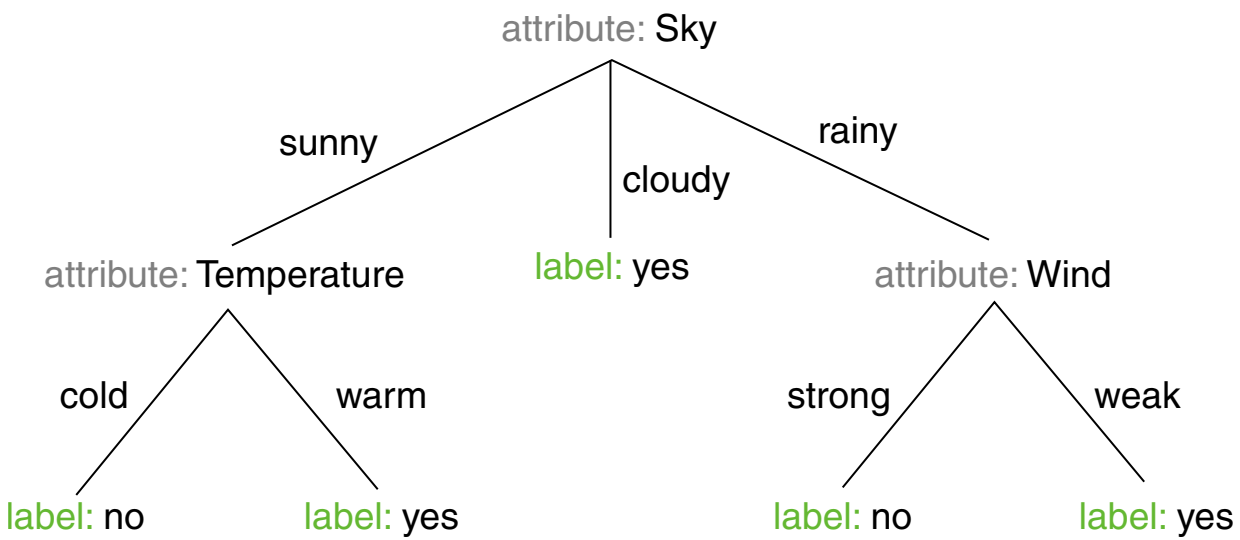
Characterization of the model (model world):

- 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:

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

3. Binary splitting induced by an ordinal feature A:

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

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.

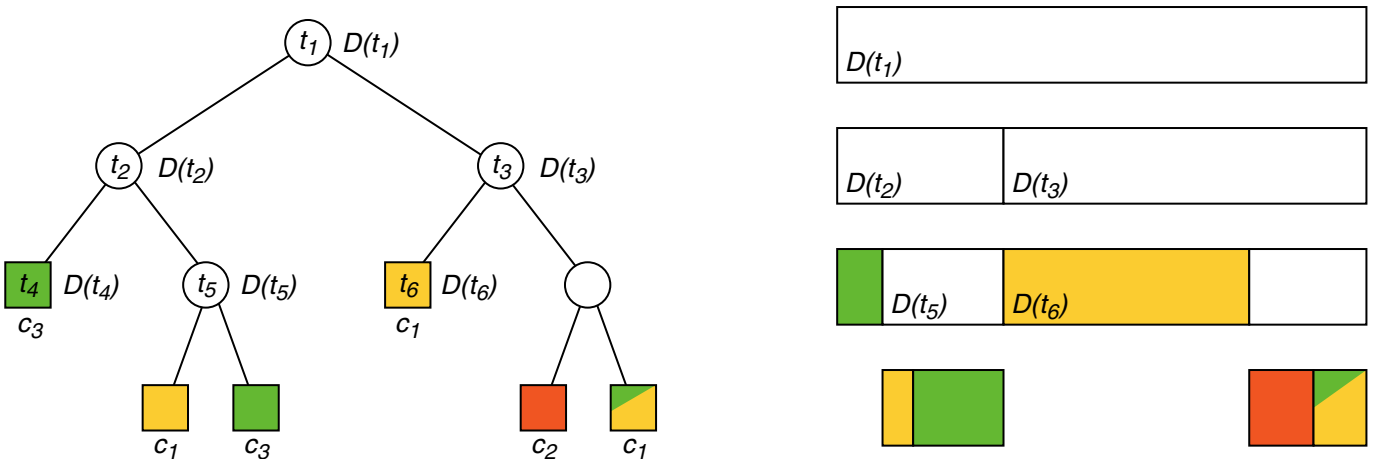
Decision Trees Basics

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−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*$(D, \textit{criterion})$
    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.

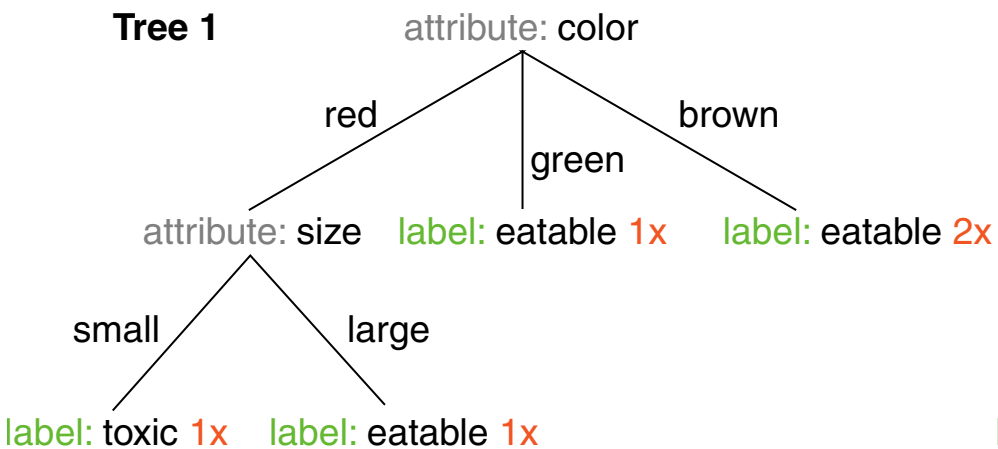
- Weighted external path length

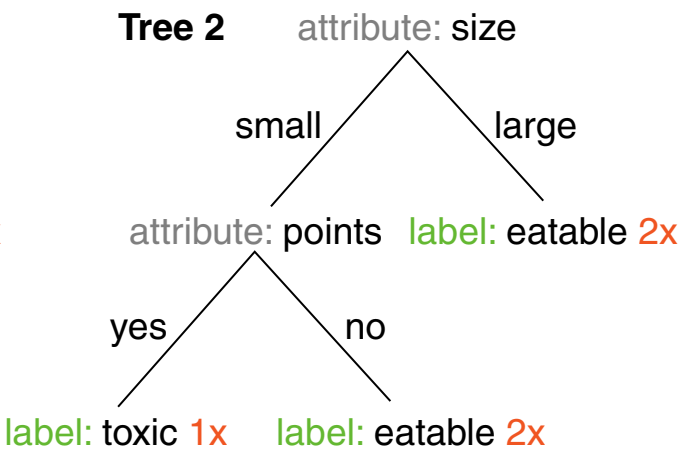
  The weighted external path length is defined as the external path length where each length value is weighted by the number of examples in D that are classified by this path.

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:

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

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:

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

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:

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

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 $\operatorname{argmax}(f)$. 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:
  - $\max_{c \in C}(f(c)) = \max\{f(c) \mid c \in C\}$
  - $\operatorname{argmax}_{c \in C}(f(c)) = c^* \quad \Rightarrow \; f(c^*) = \max_{c \in C}(f(c))$

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

- The true misclassification rate $\mathit{Err}^*(T)$ 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:

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

- If D has been used as training set, a reliable interpretation of the (training) error $\mathit{Err}(T, D)$ in terms of $\mathit{Err}^*(T)$ 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:

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

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:

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

Misclassification costs of 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:

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

Misclassification costs of 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 $\operatorname{argmin}(f)$. 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.

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. Ockham's Razor.

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 $D(t)$.
   $D(t)$ will not be partitioned further if the number of examples, $|D(t)|$, is below a certain threshold.

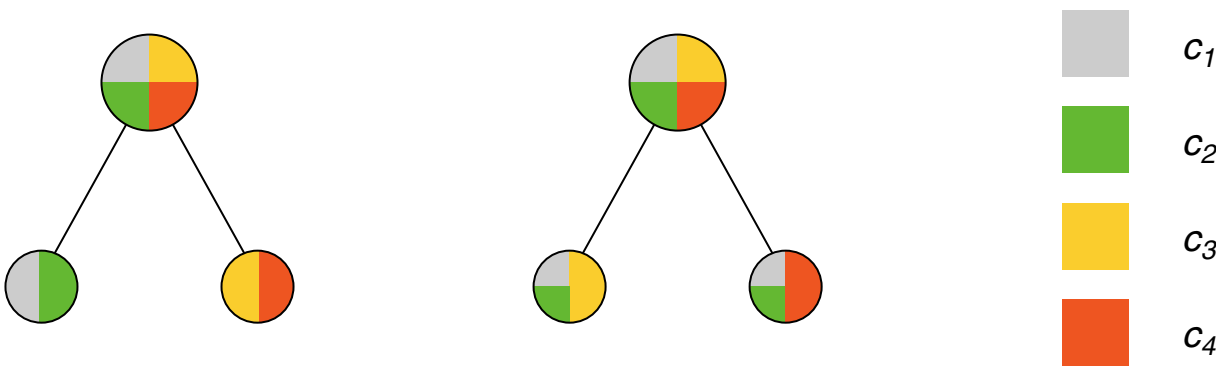
2. Purity of $D(t)$.
   $D(t)$ will not be partitioned further if all examples in D are members of the same class.

3. Ockham's Razor.
   $D(t)$ will not be partitioned further if the resulting decision tree is not improved significantly by the splitting.

Impurity Functions

Splitting (continued)

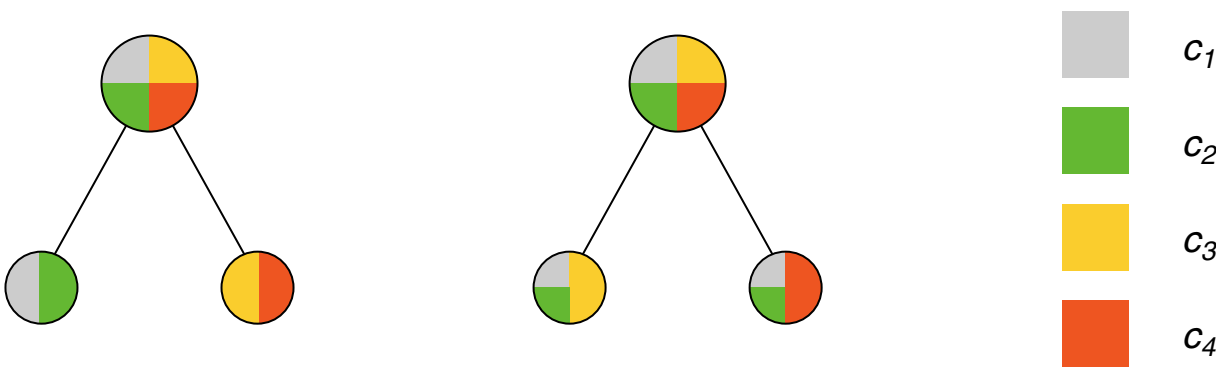
Let D be a set of examples over a feature space X and a set of classes $C = \{c_1, c_2, c_3, c_4\}$. Distribution of D for two possible splittings of X :



Impurity Functions

Splitting (continued)

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



- The left splitting should be preferred, since it minimizes the *impurity* of the subsets of D in the leaf nodes. The argumentation presumes that the misclassification costs are independent of the classes in C.
- The impurity is a function defined on $\mathcal{P}(D)$, the set of all subsets of an example set D.

Impurity Functions

Impurity Functions Based on the Misclassification Rate

Definition for two classes:

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

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

Impurity Functions

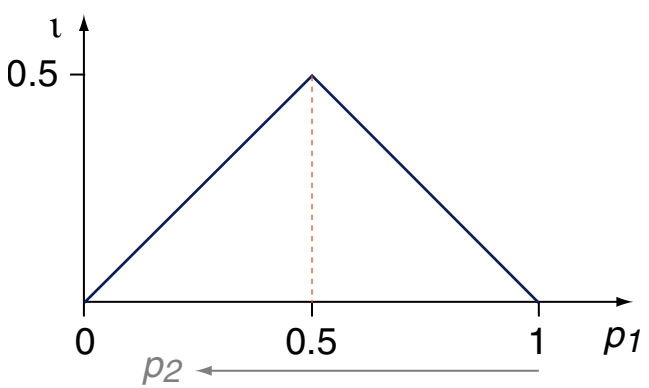
Impurity Functions Based on the Misclassification Rate

Definition for two classes:

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

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

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



Impurity Functions

Impurity Functions Based on the Misclassification Rate (continued)

Definition for k classes:

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

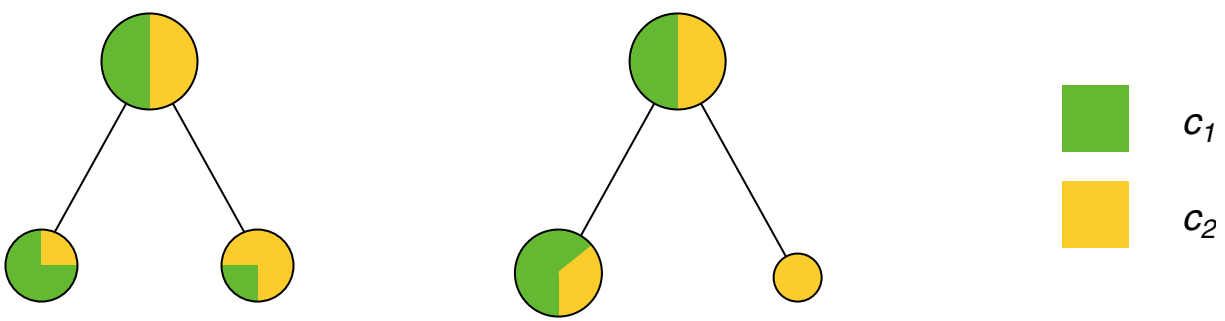
$$\iota_{misclass}(D) = 1 - \max_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c\}|}{|D|}$$

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 (see splitting on the right-hand side):

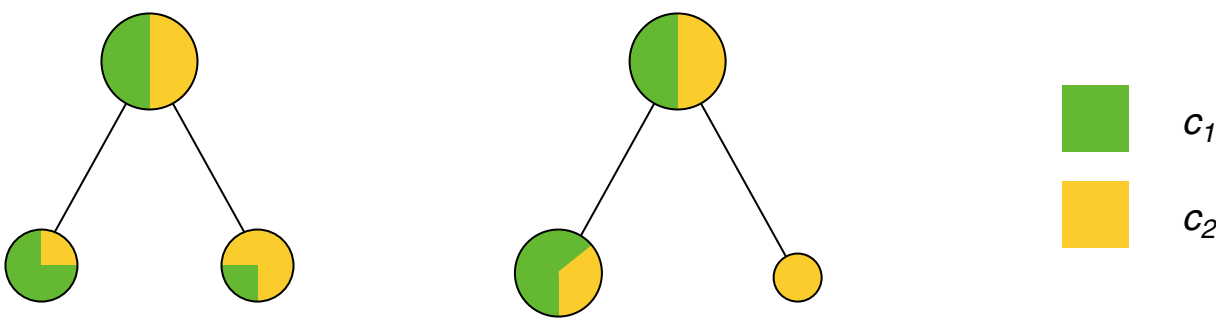


Impurity Functions

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 (see splitting on the right-hand side):



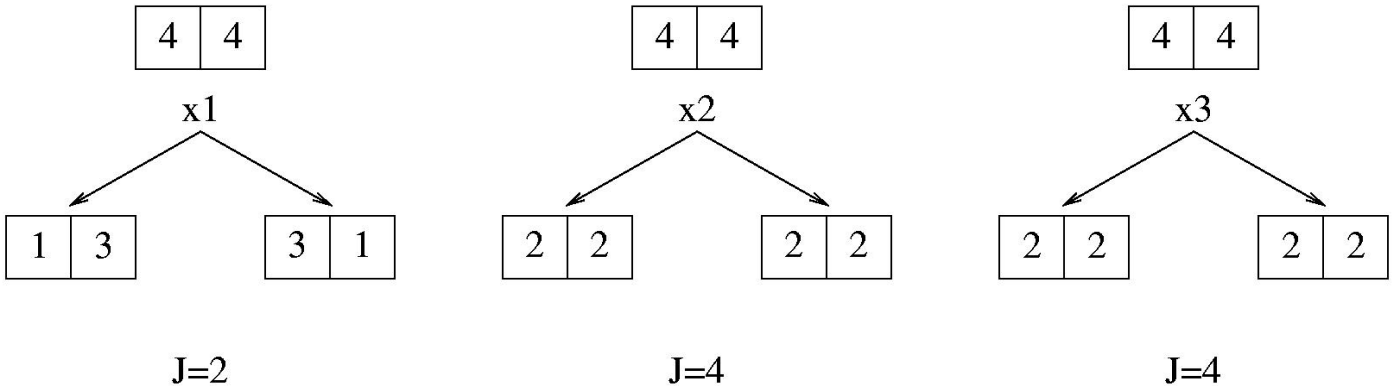
$$\Delta\iota_{misclass} = \iota_{misclass}(D) - \left(\frac{|D_1|}{|D|} \cdot \iota_{misclass}(D_1) + \frac{|D_2|}{|D|} \cdot \iota_{misclass}(D_2)\right)$$

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

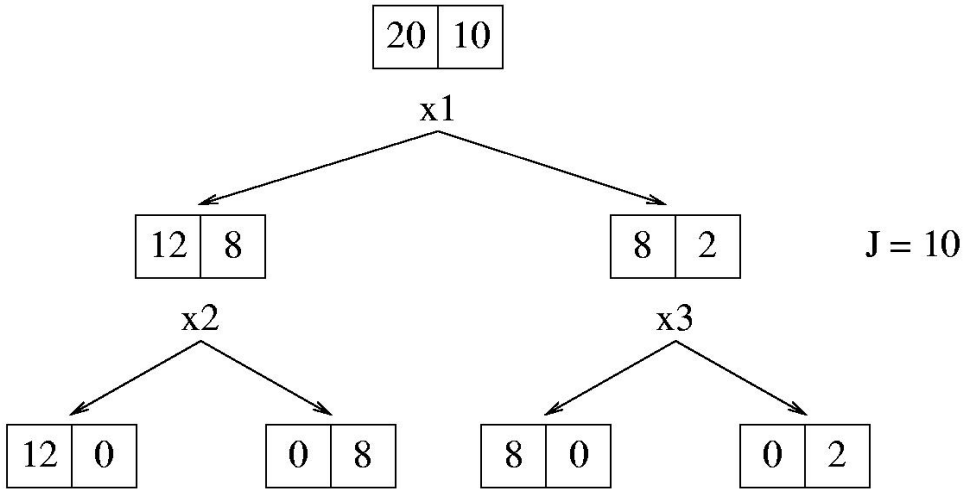
Choosing the Best Attribute—An Example

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Choosing the Best Attribute (3)

Unfortunately, this measure does not always work well, because it does not detect cases where we are making "progress" toward a good tree.



A Better Heuristic From Information Theory

Let V be a random variable with the following probability distribution:

$P(V = 0)$	$P(V = 1)$
0.2	0.8

The *surprise*, $S(V = v)$ of each value of V is defined to be

$$S(V = v) = -\lg P(V = v).$$

An event with probability 1 gives us zero surprise.

An event with probability 0 gives us infinite surprise!

It turns out that the surprise is equal to the number of bits of information that need to be transmitted to a recipient who knows the probabilities of the results.

This is also called the *description length* of $V = v$.

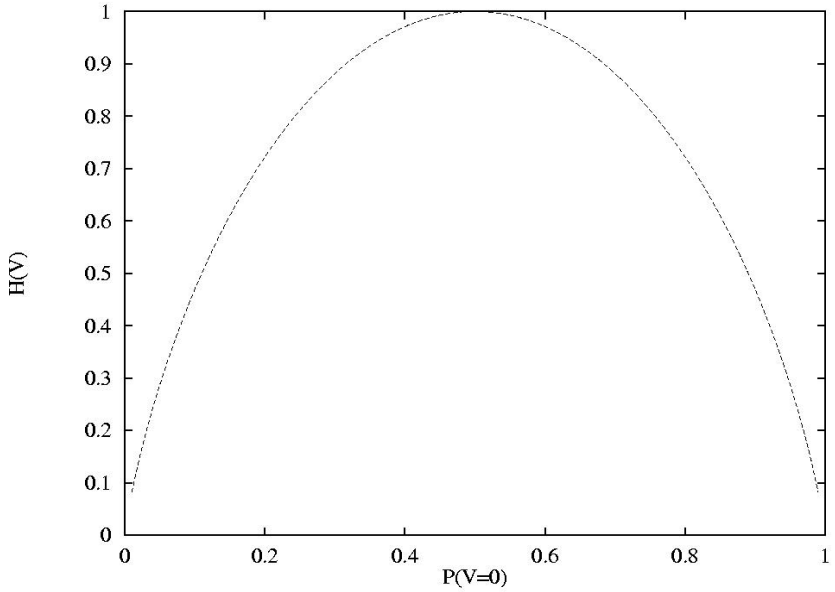
Fractional bits only make sense if they are part of a longer message (e.g., describe a whole sequence of coin tosses).

Entropy

The *entropy* of V, denoted $H(V)$ is defined as follows:

$$H(V) = \sum_{v=0}^{1} -P(H = v) \lg P(H = v).$$

This is the average surprise of describing the result of one "trial" of V (one coin toss).



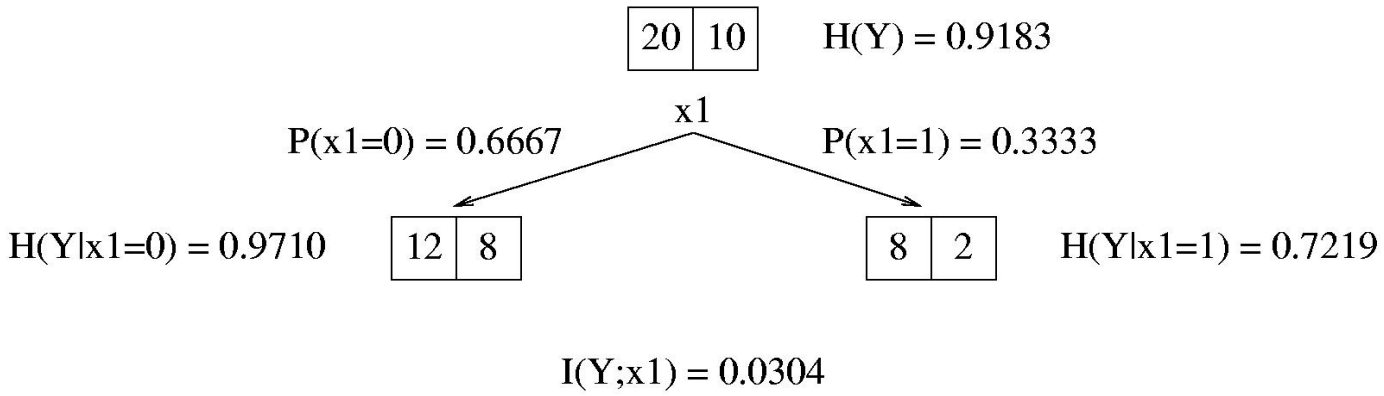
Entropy can be viewed as a measure of uncertainty.

Mutual Information

Now consider two random variables A and B that are not necessarily independent. The *mutual information* between A and B is the amount of information we learn about B by knowning the value of A (and vice versa—it is symmetric). It is computed as follows:

$$I(A;B) = H(B) - \sum_b P(B=b) \cdot H(A|B=b)$$

In particular, consider the class Y of each training example and the value of feature x_1 to be random variables. Then the mutual information quantifies how much x_1 tells us about the value of the class Y.



Impurity Functions

Impurity Functions Based on Entropy

Definition 8 (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:

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

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 $0 \cdot \log_2(0) = 0$.

Impurity Functions

Impurity Functions Based on Entropy (continued)

Definition 9 (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:

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

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

Impurity Functions

Impurity Functions Based on Entropy (continued)

Definition 9 (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:

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

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

Impurity Functions

Impurity Functions Based on Entropy (continued)

Definition 9 (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:

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

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

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 conditional entropy $H(\mathcal{A} \mid \mathcal{B})$ of class $c(\mathbf{x})$ will become smaller if one learns 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; it is zero if the involved events are statistically independent.
- 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:

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

Impurity Functions

Impurity Functions Based on Entropy (continued)

Definition for two classes:

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

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

Impurity Functions

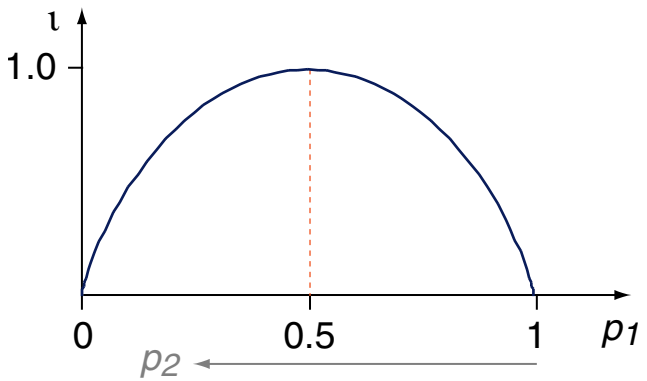
Impurity Functions Based on Entropy (continued)

Definition for two classes:

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

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

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



Impurity Functions

Impurity Functions Based on Entropy (continued)

Definition for k classes:

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

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

Impurity Functions

Impurity Functions Based on Entropy (continued)

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

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

Impurity Functions

Impurity Functions Based on Entropy (continued)

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

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

Legend:

- $\iota_{entropy}(D) = \iota_{entropy}(P(A_1), \ldots, P(A_k))$
- $\iota_{entropy}(D_j) = \iota_{entropy}(P(A_1 \mid B_j), \ldots, P(A_k \mid B_j)),\ j = 1, \ldots, s$
- $\iota_{entropy}(p_1, \ldots, p_k) = -\sum_{i=1}^{k} p_i \cdot \log_2(p_i)$
- $\frac{|D_j|}{|D|} = P(B_j),\ j = 1, \ldots, s$
- $A_i,\ i = 1, \ldots, k$, 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: $X(t) = X(t_1) \cup \ldots \cup X(t_s) = \{\mathbf{x} \in X(t) : \mathbf{x}|_B = b_1\} \cup \ldots \cup \{\mathbf{x} \in X(t) : \mathbf{x}|_B = b_s\}$
- $P(A_i), P(B_j), P(A_i \mid B_j)$ are estimated as relative frequencies based on D.