



Decision Trees

Department of Computer Languages and Systems

Decision trees: introduction

Objectives of a **decision tree**

- inference, learning, tree generation:
 - to achieve the best possible split, at leaf nodes, of the (training) samples of different classes
 - to build small trees
- use, exploitation:
 - to predict the class (classification)
 - to predict the value of a target variable (regression)

Decision trees: introduction

Strategy to build a decision tree from a training set X :

- progressive splitting of the training set into smaller and smaller subsets
- recursive splitting of X based on the value of attributes
- creating a labelled leaf, when all instances of a subset belong to the same class
- pruning: a branch with a mixed subset of distinct classes is labelled with the majority class (the resulting subtree is pruned)

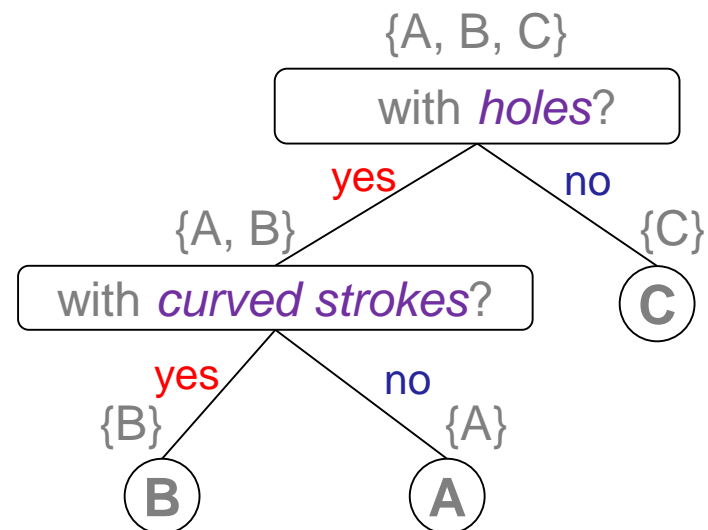
Decision trees: an example

You want to define a **system to recognise 3 capital letters** (e.g., Arial) based on the presence or absence of 2 features: *holes* and *curved strokes*.

| <i>class</i> | <i>holes</i> | <i>curved strokes</i> |
|--------------|--------------|-----------------------|
| A | yes | no |
| B | yes | yes |
| C | no | yes |

comments:

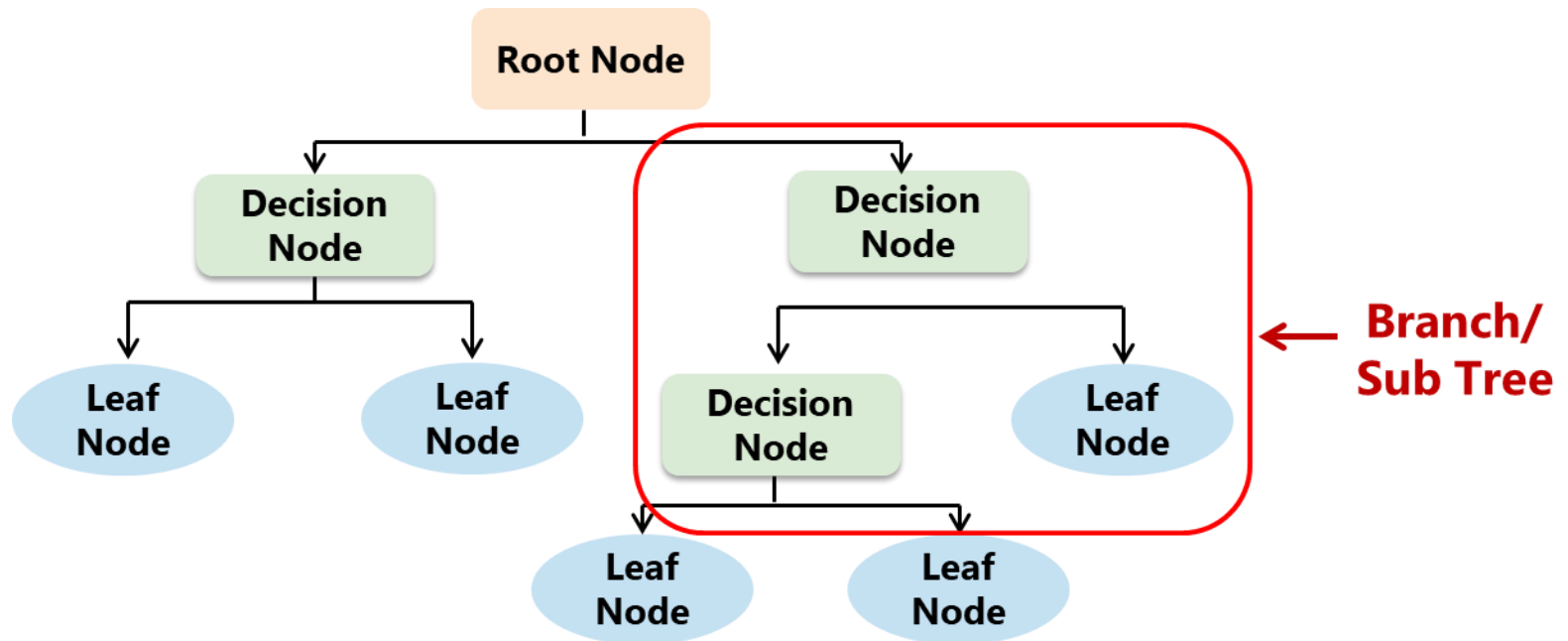
- 3 classes: A, B, C
- 1 instance per class
- 2 discrete features



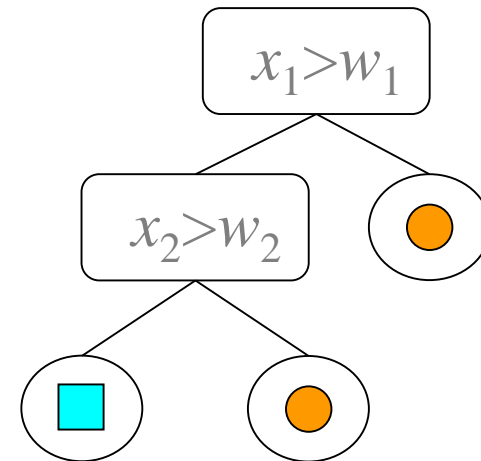
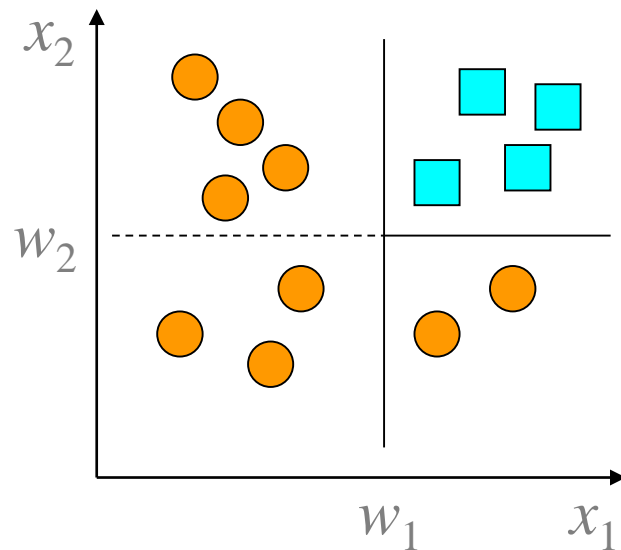
Decision trees: introduction

- **tree structure** with the following types of nodes:
 - **root node** is the top-most decision node and represents the entire set X that further gets divided according to the value of attributes
 - **internal decision node** $f_m(x)$ based on an attribute m ; it produces as many branches as allowed by the splitting criterion adopted
 - **leaf node** (or **terminal node**), with output value (class, prediction); it groups instances, for example, of a single class
- **decision**: for each new input instance x , $f_m(x)$ **guides** the analysis of x **from the root to a leaf** that provides the result (**class**)

Decision trees: types of nodes



Decision trees: learning (induction) of a decision tree



Data set

2 classes: \circ , \square

2 dimensions: x_1, x_2

Decision tree

rectangular nodes \rightarrow decision nodes

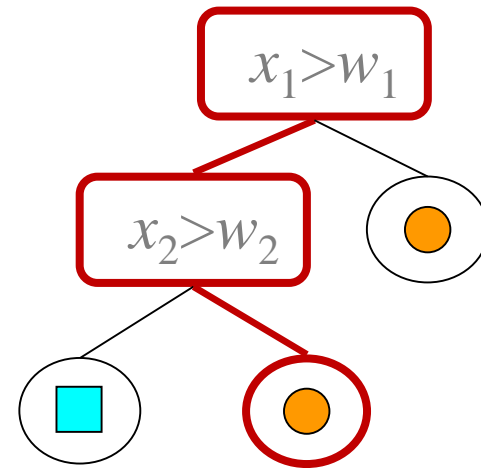
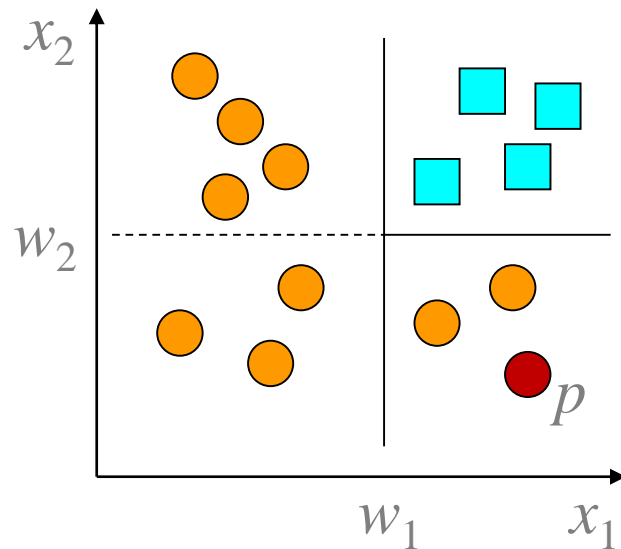
circular nodes \rightarrow leaf nodes

1st split: $x_1 > w_1$

2nd split: $x_2 > w_2$ (orthogonal to 1st)

Decision trees: classification

Given a point $p = (w_1 + \delta, w_2 - \delta)$, $\delta > 0$, the induced tree classifies p as belonging to class \bigcirc (see **coloured path**): **it goes through the tree structure from the root node until reaching a leaf**



Decision trees: interpretability

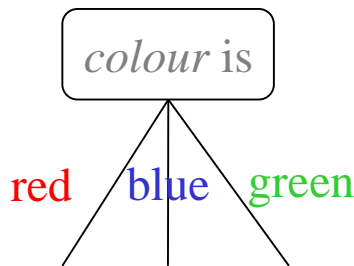
- decision tree \equiv set of rules **IF-THEN-ELSE**
- for example, the previous tree is equivalent to:

```
IF  $x_1 > w_1$  THEN  
    IF  $x_2 > w_2$  THEN  
        class =  $\square$   
    ELSE  
        class =  $\bigcirc$   
ELSE  
    class =  $\bigcirc$ 
```

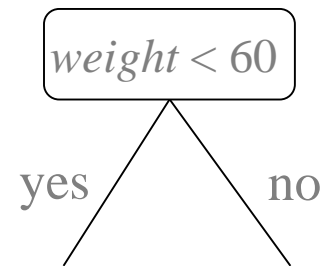
Univariate trees: introduction

the $f_m(x)$ of each internal node is defined in terms of a single dimension (attribute) x_i (see *previous example*)

if x_i is **discrete**, for example: x_i is an attribute *colour* ...



if x_i is **numeric** (ordered), for example: x_i is an attribute *weight* ...



binary partition of space

$L = \{\text{people} \mid \text{weight} < 60\}$

$R = \{\text{people} \mid \text{weight} \geq 60\}$

Univariate trees: tree learning algorithm

- tree induction \equiv tree learning
- algorithms of local search (greedy):
 - begin: root node with the whole set of instances
 - heuristic (recursive): to divide the set that arrives at a node m using the most discriminating attribute, that is, the one that generates more class homogeneous (i.e., pure) disjoint subsets
 - if the attribute is numeric, binary division; if the attribute is discrete, as many “children” (new nodes) as possible values
 - end: until get (labelled) pure nodes

Univariate trees: impurity of a node

objective: evaluate the **degree of homogeneity** of the set of instances X that reaches a node m

example: let $C = \{\omega_1=A, \omega_2=B\}$ be the set of classes

- case 1: $X^1 = \{(x_1, A), (x_2, B), (x_3, B), (x_4, A), (x_5, A), (x_6, B)\}$
- case 2: $X^2 = \{(x_1, A), (x_2, B), (x_3, A), (x_4, A), (x_5, A), (x_6, A)\}$

observations:

- X^1 has 3 instances of each class (**heterogeneous**)
- X^2 has 5 instances of A and 1 of B (**almost homogeneous**)
- intuition: X^1 is **more impure** than X^2 or X^2 is **more pure** than X^1

Univariate trees: an impurity measure

impurity measure of a node m ; given:

- N_m , the number of instances that reach node m
- $N_{i,m}$, the number of instances in node m that belong to class ω_i
- $P_{i,m} = N_{i,m}/N_m$, the probability of class ω_i in node m
- **I_m , entropy**, a measure of impurity (c classes):

$$I_m = - \sum_{i=1}^c P_{i,m} \log_2 P_{i,m}$$

(Note: $0 \log 0 = 0$)

Univariate trees: another impurity measure

impurity measure of a node m ; given:

- N_m , the number of instances that reach node m
- $N_{i,m}$, the number of instances in node m that belong to class ω_i
- $P_{i,m} = N_{i,m}/N_m$, the probability of class ω_i in node m
- G_m , **Gini index**, a measure of impurity (c classes):

$$G_m = 1 - \sum_{i=1}^c P_{i,m}^2$$

Univariate trees: how to apply the measure (entropy)

... it continues from previous example: suppose we can choose whether node m is reached by X^1 or X^2

- in both cases $N_m = 6$
- in X^1 : $N_{1,m} = 3, N_{2,m} = 3 \Rightarrow p_{1,m} = 0.50, p_{2,m} = 0.50$
- in X^2 : $N_{1,m} = 5, N_{2,m} = 1 \Rightarrow p_{1,m} = 0.83, p_{2,m} = 0.17$

evaluation of **entropy** (impurity measure):

- in X^1 , $I_m = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$ (+ *impure*)
- in X^2 , $I_m = -(0.83 \log_2 0.83 + 0.17 \log_2 0.17) = 0.66$ (+ *pure*)

it would be better X^2 to reach m because its entropy is lower (i.e. higher purity)

Univariate trees: how to apply the measure (Gini index)

... it continues from previous example: suppose we can choose whether node m is reached by X^1 or X^2

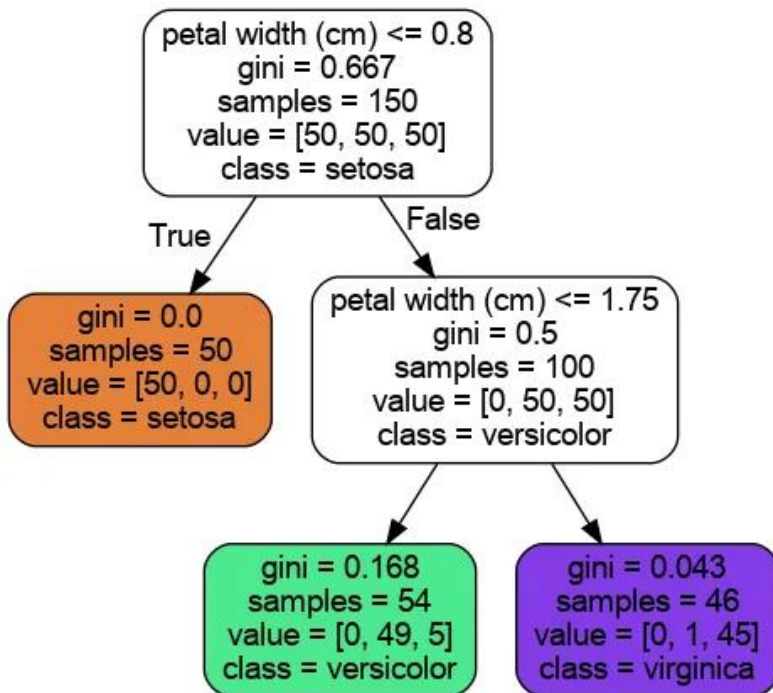
- in both cases $N_m = 6$
- in X^1 : $N_{1,m} = 3, N_{2,m} = 3 \Rightarrow p_{1,m} = 0.50, p_{2,m} = 0.50$
- in X^2 : $N_{1,m} = 5, N_{2,m} = 1 \Rightarrow p_{1,m} = 0.83, p_{2,m} = 0.17$

evaluation of **Gini index** (impurity measure):

- in X^1 , $G_m = 1 - (0.5^2 + 0.5^2) = 0.50$ (+ *impure*)
- in X^2 , $G_m = 1 - (0.83^2 + 0.17^2) = 0.28$ (+ *pure*)

it would be better X^2 to reach m because its Gini index is lower (i.e. higher purity)

Univariate trees: another example



Gini index:

$$1 - \left[\left(\frac{0}{54} \right)^2 + \left(\frac{49}{54} \right)^2 + \left(\frac{5}{54} \right)^2 \right] = 0.168$$

Entropy:

$$-\left(\frac{0}{54} \right) \log \left(\frac{0}{54} \right) - \left(\frac{49}{54} \right) \log \left(\frac{49}{54} \right) - \left(\frac{5}{54} \right) \log \left(\frac{5}{54} \right) = 0.134$$

Univariate trees: generalization to 2 classes

context: impurity value at node m for 2 classes

- $N_m = N_{1,m} + N_{2,m} \Rightarrow$
 $p_{1,m} + p_{2,m} = 1 \Rightarrow$
 $p_{2,m} = 1 - p_{1,m}$
- I_m , entropy for $c = 2$ classes
(let it be $p \equiv p_{1,m}$):

$$I_m = -p \log_2(p) - (1 - p) \log_2(1 - p)$$

Univariate tree: pure node (impurity 0)

a **node** m is **pure** if all instances reaching m are of the same class, that is, if $N_{1,m} = 0$ or $N_{2,m} = 0$; suppose $N_{1,m} = 0$, then

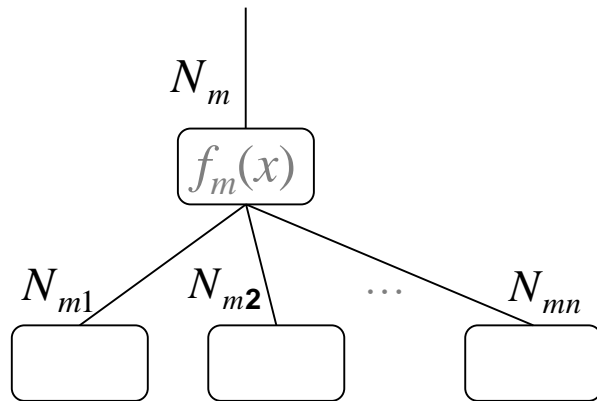
$$N_{1,m} = 0 \Rightarrow p_{1,m} = 0, p_{2,m} = 1 \Rightarrow I_m = 0, G_m = 0$$

a node m is pure $\Leftrightarrow I_m = 0$ or $G_m = 0$ (impurity 0)

a **pure node** becomes a **leaf** that is assigned the class label of its instances

Univariate trees: impurity of a split

- K classes or categories
- x is the attribute chosen to split an impure node



$$\sum_{j=1}^n N_{m,j} = N_m$$

$N_{m,j}^i$ instances of the $N_{m,j}$ belong to ω_i such that $\sum_{i=1}^K N_{m,j}^i = N_{m,j}$
 $p_{m,j}^i = \frac{N_{m,j}^i}{N_{m,j}}$ is the probability of ω_i in the branch j

Impurity of a split:

$$I_m^* = - \sum_{j=1}^n \frac{N_{m,j}}{N_m} \left(\sum_{i=1}^K p_{m,j}^i \log_2 p_{m,j}^i \right)$$

Univariate trees: choice of the attribute; local optimization

Local optimization:

in each node (**not pure**) in which it is intended to continue splitting the training set, the **most discriminating attribute** will be chosen, that is, the one that produces a split that is:

- the least impure \equiv
- the most homogeneous \equiv
- with the greatest class separation

Univariate trees: general tree induction algorithm

```
generate_tree ( $T_{tra}, \alpha$ ):  $m$   
  if entropy  $I_m(T_{tra}) \leq \alpha$ :  
     $m \leftarrow$  class label of the most represented class in  $T_{tra}$   
  else  
     $a =$  the_most_discriminating_attribute( $T_{tra}$ )  
     $m \leftarrow a$   
    for each branch  $a_i$  of  $a$ :  
       $tra_i = \{x \in T_{tra} \mid x \text{ satisfies the condition of } a_i\}$   
       $h =$  generate_tree( $tra_i$ )  
      add_child( $m, h$ )  
  return  $m$ 
```

Univariate trees: function “the most discriminating attribute”

```
the_most_discriminating_attribute(tra): a  
  mín_entr  $\leftarrow$  MAX  
  for each attribute  $x_i$ :  
    if  $x_i$  is discrete:  
      divide tra into  $tra_1, \dots, tra_N$  based on the values of  $x_i$   
       $e \leftarrow I_m^*(tra_1, tra_2, \dots, tra_N)$   
      if  $e < mín\_entr$ :  $mín\_entr \leftarrow e$ ;  $a \leftarrow x_i$   
    else //  $x_i$  is numeric  
      for each possible binary division ( $tra_1, tra_2$ ):  
         $e \leftarrow I_m^*(tra_1, tra_2)$   
        if  $e < mín\_entr$ :  $mín\_entr \leftarrow e$ ;  $a \leftarrow x_i$   
  return a
```

Pruning: introduction

Context

- it is common to find high overlapping or *outliers*
 - to generate a “pure” tree trying to learn from:
 - *outliers*
 - small sample size (small number of instances)
 - overlapping
- reduces the generalization of the model (overfitting)

Alternative

- pre-pruning: do not split nodes with few instances while growing the tree
- post-pruning: after building the tree to its depth, remove unnecessary subtrees

Pruning: pre-pruning

Idea

- let m be a node and N_m the number of samples (instances)
- stop splitting the training set
if m is pure (“*natural end*”) or if $N_m < \theta$ (pruning)
- label m with the majority class among the N_m instances

Pruning: post-pruning

Idea: to generate 3 independent (and disjoint) sets for training, pruning, and test

- **training:** generate the complete tree, with all its pure leaves, using the *training set*
- **pruning** (it is part of the learning stage):
 - replace each subtree by a leaf node with the label of the majority instances covered by that subtree
 - prune the subtree if the “surrogate” leaf node does not worsen its performance with the *pruning set*
- **evaluation:** obtain a performance measure of the final (pruned) tree by classifying the *test set*

Pruning: summary

Experiments show that:

- **post-pruning** produces **more accurate trees** (both classification and regression) than pre-pruning
- **pre-pruning** is **much faster** than post-pruning

Decision trees: advantages

- **Comprehensive:** it is good for interpreting data in a highly visual way
- **Simplicity:** it is one of the simplest algorithms since it has no complex formulas or data structures

Decision trees: disadvantages

- It is **computationally expensive**. At each node, the candidate split must be sorted before determining the best
- It is sometimes **unstable as small variations in data** might lead to the formation of a new tree