

Advanced School in Artificial Intelligence

Supervised Learning Tree Models

Elena Bellodi
elena.bellodi@unife.it

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021



**Università
degli Studi
di Ferrara**

Outline

- Tree Models
 - Decision Trees
 - classification trees
 - regression trees
 - Bagging
 - Random Forests

Outline

- Tree Models
 - Decision Trees
 - classification trees
 - regression trees
 - Bagging
 - Random Forests

Tree models

- Among the most popular models in machine learning
- Supervised learning
 - Used for classification and regression
- Non-linear models
- Easy to use
- Easy to interpret

Attribute-value Language

- Each instance is described by a set of attributes (fixed for all examples)
- **Literals** are logical expressions representing equalities of the form *Attribute = Value* and, for numerical features, also inequalities of the form *Attribute > Value*
- **Example**
 - Domain: set of athletes
 - Instances described by attributes: height, weight, favourite_music
 - Example of an instance

`height = 1.85m, weight = 110kg, favorite_music = rock`



Decision Trees

- Cope naturally with numerical and discrete data
 - height is numerical
 - color is discrete (values in {red, blue, green})
- **Each node corresponds to a test on an attribute** (equality, inequality) and **each branch** starting from the node **is labeled with a literal** (the test result)
 - The set of literals at a node is called a **split**
- **Each leaf** of the tree represents a **logical expression**, which is the conjunction of literals encountered on the path from the root of the tree to the leaf

Decision Trees

- **Each leaf node has an output label**

- classification : Yes/No
- regression : numeric value
- new instances falling in a certain leaf will have the same label (classification) or similar numeric output (regression)



- **Tree structure is not fixed a priori but it grows during learning depending on the complexity of the problem**

Classification Trees

- The tennis problem: *should I play tennis, given some weather conditions?*
- 9 Examples say 'yes', 5 examples say 'no' in the training set

	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



Tree Induction

Recursive procedure to build the tree based on ‘divide-and-conquer’ technique

Algorithm 1 *GrowTree(D, F)* – grow a feature tree from training data.

Require: data D ; set of features F

Ensure: feature tree T with labelled leaves

```
1: if Homogeneous( $D$ ) then
2:   return Label( $D$ );
3: end if
4:  $S \leftarrow \text{BestSplit}(D, F)$ ;
5: split  $D$  into subsets  $D_i$  according to the literals in  $S$ ;
6: for each  $i$  do
7:   if  $D_i \neq \emptyset$  then
8:      $T_i \leftarrow \text{GrowTree}(D_i, F)$ 
9:   else
10:     $T_i$  is a leaf labelled with Label( $D$ );
11:   end if
12: end for
13: return a tree whose root is labelled with  $S$  and whose children are  $T_i$ 
```



Tree Induction


1. INPUT

- **D: set of examples**
- **{C1, C2,..., Ck}: set of classes**

2. Consider the set D:

- D contains one or more examples, all belonging to the same class => single leaf labeled the class (line 2)
- D does not contain any example (empty set) => single leaf with label the most frequent class in the parent set
- D contains cases that belong to multiple classes => partitioning of D into multiple subsets D1, D2, ..., Dn according to a test on an attribute (line 5)
 - that node is associated with the test (literal S), with a subtree for each possible result of the test itself
 - this splits up the example set into subsets, one for every value of the literal
 - *call the algorithm on each branch / subset*

Tree Induction

- Systems that learn decision trees: CLS, IDR, C4, ASSISTANT, ID5, CART, ID3 etc.
- c4.5: Evolution of ID3, another system by the same author, J.R. Quinlan
- c4.5 does not stop only when it finds uniform or empty sets, **it also stops if no test is such that at least two subsets contain a minimum number of cases**
 - default is 2 examples 

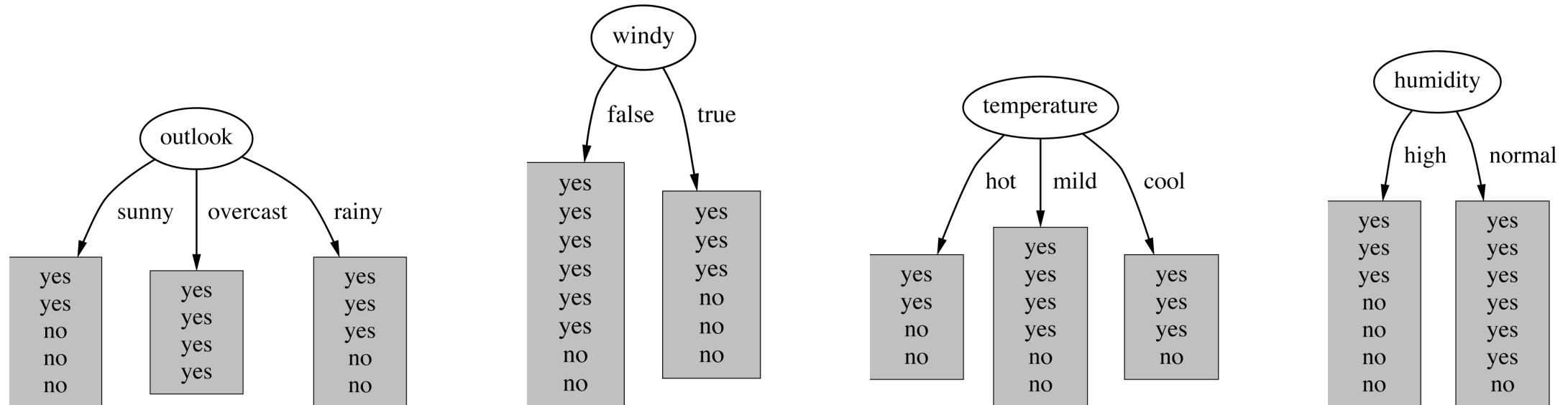
Tree Induction

- For a given training set, there exist many trees that code it with no error, and, for simplicity, **we are interested in finding the smallest among them**, where tree size is measured as the number of nodes in the tree and the complexity of the decision nodes
- Finding the smallest tree is NP-complete, and we are forced to use local search procedures based on heuristics that give reasonable trees in reasonable time
 - Heuristics with no backtracking

How do we choose the test?

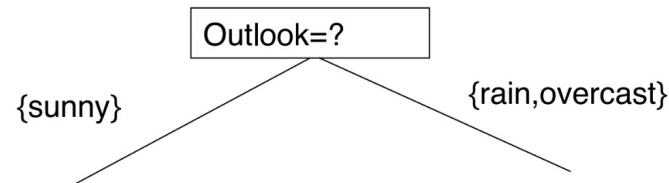
- Which attribute should be used as the test?

Intuitively, you would prefer the one that separates the training examples as much as possible.



How do we choose the test?

- **Discrete attribute X with n possible values x_1, \dots, x_n**
 - Equality with a constant: $X = \text{const}$, 2 possible outcomes: yes, no
 - Equality test: $X = ?$, n possible outcomes
 - Membership in a set: $X \in S$, 2 possible outcomes: yes, no
 - Membership in a set of a partition of $\{x_1, \dots, x_n\}$: one outcome per set
- Example of membership in a set of a partition:
 - Attribute *Outlook*, partition of the set of values $\{\{\text{sunny}\}, \{\text{rain}, \text{overcast}\}\}$



How do we choose the test?

- **Continuous attribute X**
 - Comparison with a threshold $X \leq \text{const}$, 2 possible outcomes: yes, no 


How do we choose the test?

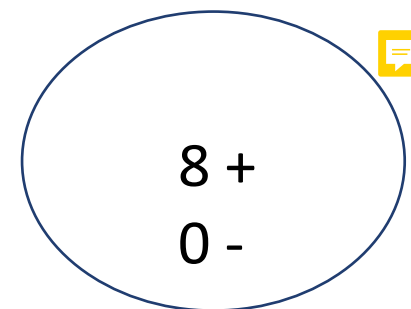
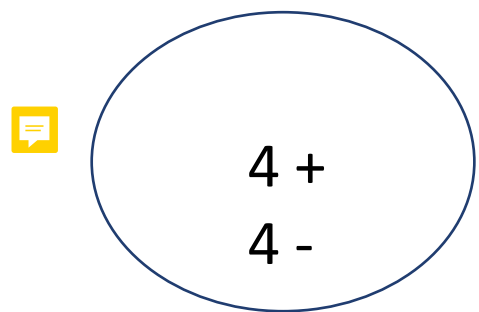
- **For Discrete attributes**
 - Choice of the type of test: usually only the equality test $X=?$ is used
 - the attribute must be chosen
 - possibly choice of the constant or partition
- **For Continuous attributes**
 - Choice of the threshold
- **Constraints that the test must satisfy:** at least two among D_1, D_2, \dots, D_n must contain a minimum number of examples

How do we choose the test?

- We measure the “purity” of each node
- A split is pure if after the split, for all branches, all the instances going down a branch belong to the same class C_i —> a leaf is reached
- Different **purity measures**
 - classification: entropy, information gain, gini index, misclassification error: there is not a significant difference between them
 - regression: if at a node the mean square error is acceptable, then a leaf node is created and it stores the regression value (a real value)
- For all attributes, discrete and numeric, and for a numeric attribute for all split positions, we calculate the impurity and choose the attribute that has the minimum one on the training set
- Stop when all nodes are pure

Entropy and purity

- Entropy measures the purity of a set:
 - It is minimal when the set is most pure, i.e., when it contains examples from only one class 
 - It is maximal when the set is most impure, i.e., when it contains an equal number of examples of the two classes



The distribution is less uniform
Entropy is lower
The node is purer

Entropy

- The *expected amount of information* when observing the output of a random variable C , the class of an instance randomly selected from a set D

$$H(C) = - \sum_{j=1}^k P(c_j) \log_2 P(c_j) = E[-\log_2 P(C)]$$

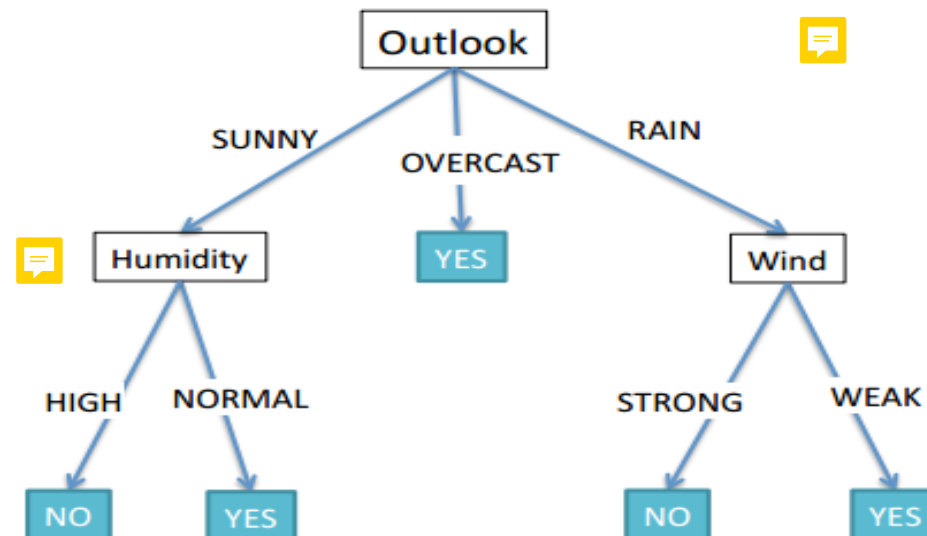
- $P(c_j) = |D_j| / |D|$

where D_j is the set of examples from D that belong to class c_j

Gini Index

$$\text{gini}(D) = 1 - \sum_{j=1}^k P(C_j)^2$$

Classification trees



```
if ( Outlook==sunny AND Humidity==high )      then NO
if ( Outlook==sunny AND Humidity==normal )    then YES
if ( Outlook==overcast )                      then YES
if ( Outlook==rain AND Wind==strong )         then NO
if ( Outlook==rain AND Wind==weak )           then YES
```

Classification Trees: *prediction*

Given the model (decision tree) learnt before...

Predict/Classify the target class for the new examples:

No	Outlook	Temp	Humid	Windy	Class
D15	overcast	Mild	Normal	Strong	?
D16	sunny	Cool	Normal	Strong	?
D17	rainy	Hot	High	Weak	?
D18	sunny	Hot	High	Weak	?
D19	rainy	Mild	High	Strong	?

Regression Trees

- The target variable is continuous
- Each leaf is labelled with a numeric value instead of a class
- A leaf defines a localized region in the input space where **instances falling in this region have very similar numeric outputs**

Regression Trees

- The tree induction procedure is adapted to continuous values
- $\text{Label}(Y)$ is similarly adapted to return the mean value in Y
- function $\text{Homogeneous}(Y)$ returns true if the variance of the target values in Y is zero (or smaller than a low threshold)
- Impurity measure is replaced by the **variance** of a set Y of target values as the average squared distance from the mean:

$$\text{Var}(Y) = \frac{1}{|Y|} \sum_{y \in Y} (y - \bar{y})^2$$

- where \bar{y} is the mean of the target values in Y
- The goal is minimizing variance over all possible splits of a given parent \rightarrow choose the split with the minimum variance

Regression Trees Example

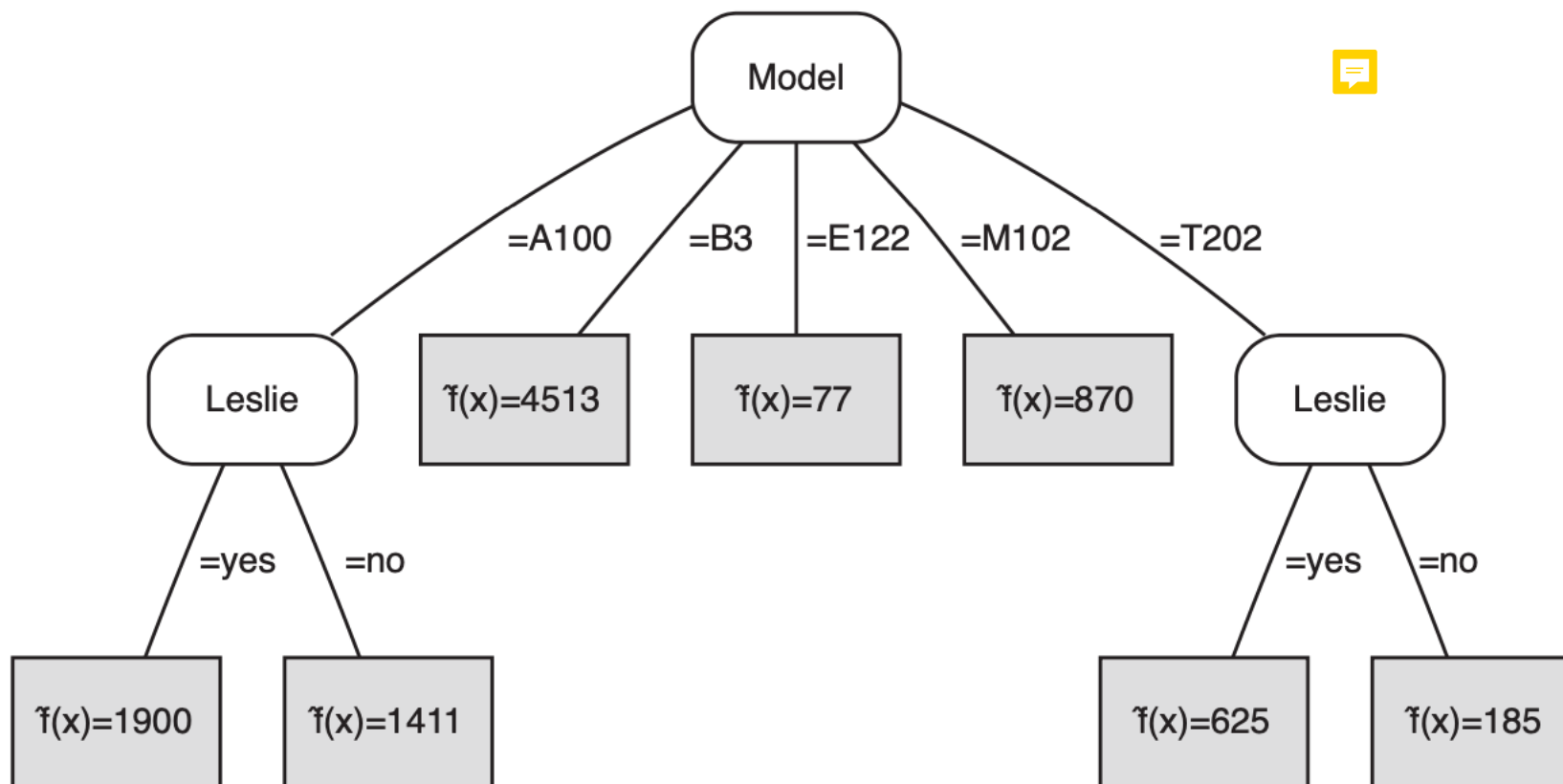
You have been monitoring an online auction site, from which you collected some data about interesting transactions.

From this data, you want to construct a regression tree that will help you determine a reasonable price for your next purchase.

#	Model	Condition	Leslie	Price
1.	B3	excellent	no	4513
2.	T202	fair	yes	625
3.	A100	good	no	1051
4.	T202	good	no	270
5.	M102	good	yes	870
6.	A100	excellent	no	1770
7.	T202	fair	no	99
8.	A100	good	yes	1900
9.	E112	fair	no	77

Target variable

Regression Trees Example



Advantages

- **Very fast to both train and test** (highly parallelizable):
 - if the decisions are binary, each decision eliminates half of the cases. If there are b regions, then in the best case, the correct region can be found in $\log_2 b$
- **Interpretability:** the tree can be converted to a set of IF-THEN rules that are easily understandable. For this reason, decision trees are very popular.

Conversion into rules

- It's possible to convert a decision tree into IF-THEN rules
- **Standard form of rules**
 1. IF *Conditions* THEN *Class*
 2. *Class* IF *Conditions*
 3. *Conditions* \rightarrow *Class*

Conditions = c_1 and c_2 and ... and c_m

c_i is of the form $a \text{ rel } v$

a = attribute; v = its value; $\text{rel} \in \{=, >, <, \geq, \leq\}$

- Example: **IF** *outlook = rainy and windy = weak* **THEN** *Play = yes*

Overfitting

- You can perfectly fit the tree to any training data
- **Zero bias, high variance**
- Two approaches against excessive tree growth
 1. Stop the growth before reaching a tree that perfectly classifies the training examples (**prepruning**) - faster
 2. Grow a full tree, then prune the tree, by eliminating nodes (**postpruning**) – more accurate

Other practical problems

- **Attributes with many values** → many branches → methods to penalize such attributes
- **Noise (misabeled instances)** → possible large and overfitted trees → set a threshold for purity

Applications

- Top-down induction of decision trees is one of the most extensively researched method of machine learning

1. *Classification and Regression Trees (CART) algorithm*

- (Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984). Classification And Regression Trees (1st ed.). Routledge.)
- allows to generate multivariate trees: trees with tests involving more than one attribute in a node (previously we saw univariate trees)
- available in scikit-learn, an open source suite of ML tools in Python

Applications

2. **ID3 algorithm** (Quinlan 1986), and its extension **C4.5** (Quinlan 1993)
 - *C5.0, the most recent version of C4.5, is available commercially*
 - **J48** *is an open source Java implementation of the C4.5 algorithm in the Weka data mining tool*

(<http://www.cs.waikato.ac.nz/ml/weka/>)

Next...

- Bagging
- Random Forests

