CLASSIFICATION TREES

Concha Bielza, Pedro Larrañaga

Computational Intelligence Group
Departmento de Inteligencia Artificial
Universidad Politécnica de Madrid

Machine Learning

Outline

Outline

Introduction

Representation as a tree

- Each node (not leaf) specifies a test of some attribute of the instance
- Each branch descending corresponds to one of the possible values for this attribute
- Each leaf node provides the classification of the instance
- Unseen instances are classified by sorting them down the tree from the root to some leaf node testing the attribute specified at each node. At the leaf we have its classification

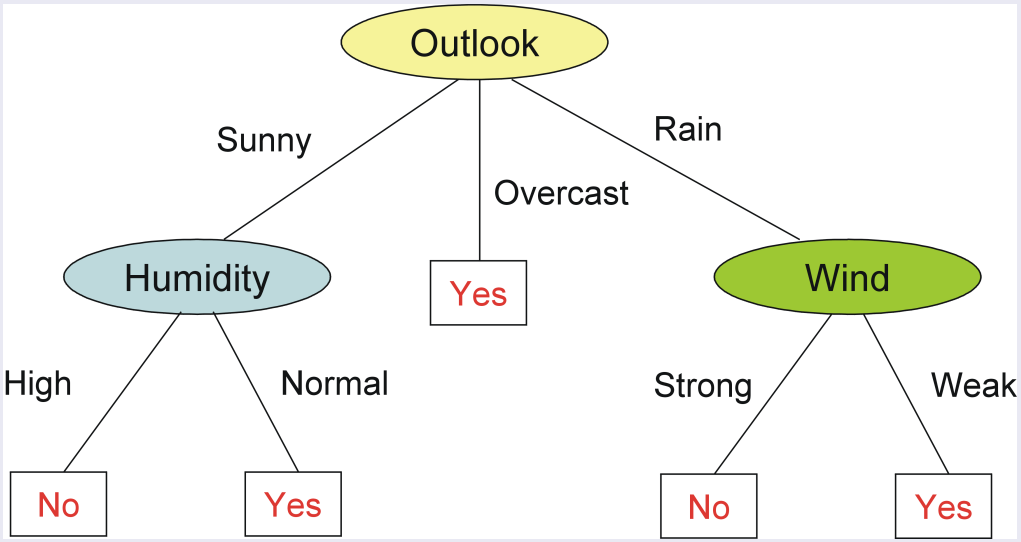
Introduction

Example: PlayTennis

Classify Saturday mornings according to whether they're suitable for playing tennis

Decision trees allow selecting features

Humidity is only relevant when the outlook is sunny -> DTs give such level of detail



Instance:

Outlook=Sunny, Temperature=Hot, Humidity=High, Wind=Strong

sorted down the leftmost branch, classified PlayTennis=No

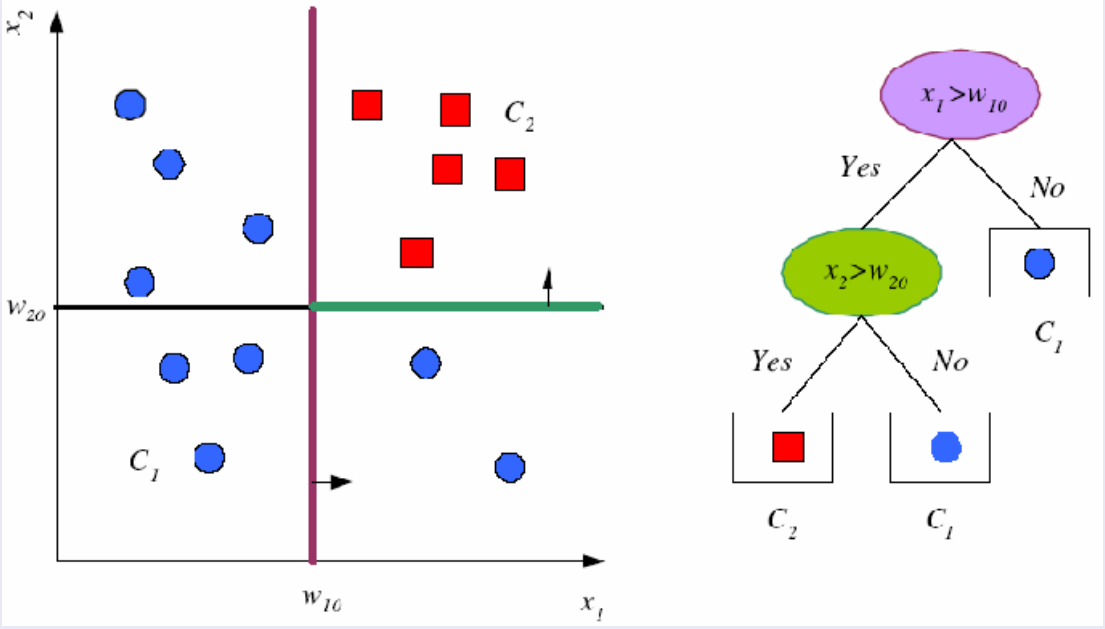
Introduction

Rule generation

R1: If *(Outlook=Sunny)* AND *(Humidity=High)* then PlayTennis=No
R2: If *(Outlook=Sunny)* AND *(Humidity=Normal)* then PlayTennis=Yes
R3: If *(Outlook=Overcast)* then PlayTennis=Yes
R4: If *(Outlook=Rain)* AND *(Wind=Strong)* then PlayTennis=No
R5: If *(Outlook=Rain)* AND *(Wind=Weak)* then PlayTennis=Yes

Introduction

Example



- Space is divided into regions labelled with one class; they're hyperectangles

Outline

Basic algorithm: ID3

It stops when all the instances in a branch have the same label

ID3=Iterative dicotomiser [Quinlan, 1986]

- Greedy search strategy through the space of all possible classification trees
- Construct the tree top-down, asking: which attribute should be tested at the root?
- Answer by evaluating each attribute to determine how well it alone classifies training examples
- Best attribute is selected, a descendant of the root is created for each value of this attribute and the training examples are sorted to the appropriate descendant
- Repeat this process using the examples associated with each descendant node to select the best attribute at that point in the tree (always forward, searching among the not yet used attributes in this path)
- Stop when the tree correctly classifies the examples or when all attributes have been used
- Label each leaf node with the class of the examples

Basic algorithm: ID3

ID3

- Key step: how to select which attribute to test at each node in the tree?
- We would like the most useful for classifying examples; that separates well the examples
- ID3 chooses mutual information as a measure of the worth of an attribute (maximize)

$$I(C, X_i) = H(C) - H(C|X_i) \qquad \text{(information gain)}$$

$$H(C) = -\sum_c p(c) \log_2 p(c), \quad H(C|X) = -\sum_c \sum_x p(x, c) \log_2 p(c|x)$$

- Expected reduction in entropy (uncertainty), caused by partitioning examples according to this attribute

Basic algorithm: ID3

Example ID3: PlayTennis

Data:

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

Basic algorithm: ID3

Example ID3: PlayTennis

Wind?

- $H(C) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$
- $H(C|\text{Wind}) =$
  $-p(Strong, Yes)\log_2 p(Yes|Strong) - p(Strong, No)\log_2 p(No|Strong) -$
  $p(Weak, Yes)\log_2 p(Yes|Weak) - p(Weak, No)\log_2 p(No|Weak) =$
  $-\frac{3}{14}\log_2\frac{3}{6} - \frac{3}{14}\log_2\frac{3}{6} - \frac{6}{14}\log_2\frac{6}{8} - \frac{2}{14}\log_2\frac{2}{8} = 0.892$

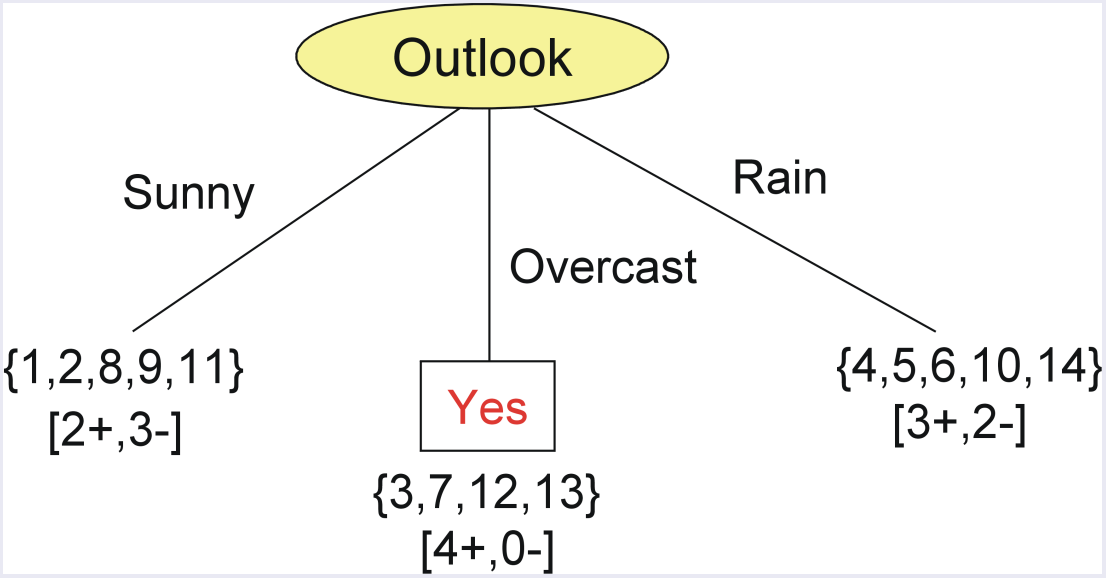
$\Rightarrow I(C, \text{Wind}) = 0.940 - 0.892 = 0.048$

Analogously,

- $I(C, \text{Humidity}) = 0.151$
- $I(C, \text{Outlook}) = 0.246 \Leftarrow$ Choose Outlook as root node
- $I(C, \text{Temperature}) = 0.029$

Basic algorithm: ID3

Example ID3: PlayTennis



- All the examples with *Outlook=Overcast* are positive $\Rightarrow$ It becomes a leaf node
- The rest have nonzero entropy and the tree still grows

Basic algorithm: ID3

Example ID3: PlayTennis

E.g., through the branch *Sunny* (with 5 instances), we search for the next attribute:

- $H(C_{sunny}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$
- $I(C_{sunny}, \text{Temperature}) = 0.97 - 0.4 = 0.57$
- $I(C_{sunny}, \text{Humidity}) = 1.94$
- $I(C_{sunny}, \text{Wind}) = 0.019$

$\Rightarrow$ ...Final tree is the one above

- Any attribute only appears at most once along any path

Outline

ID3 improvements

Practical issues

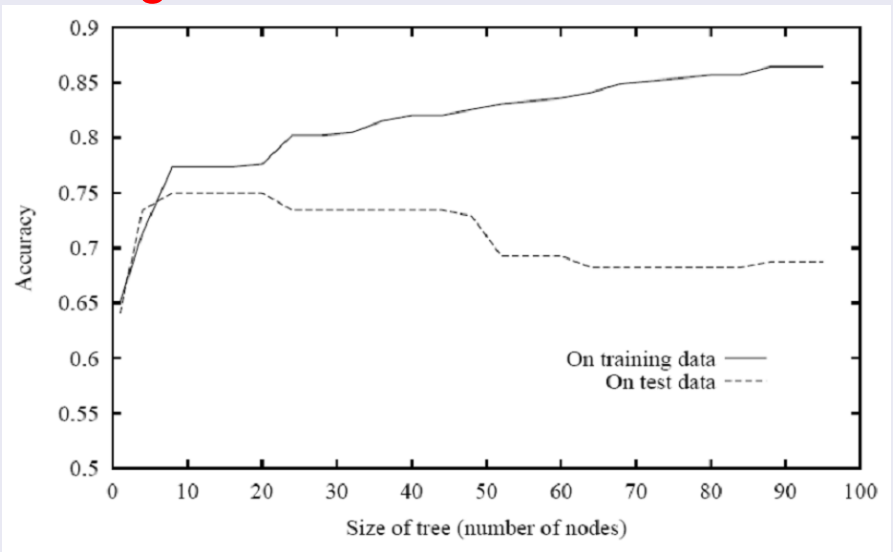
- Determine how deeply to grow the tree
- Handling continuous attributes

How deeply to grow the tree

Overfitting problem

- Problems if we grow the tree just deeply enough to perfectly classify the training examples:
  - If there is noise in the data, we can learn the noise!
  - If we have a few examples associated with leaves, hard to produce a representative sample of the true target function

  $\Rightarrow$ ID3 produces trees that overfit the training data (and it doesn't work well with new unseen examples). The model is unable to generalize



Overfitting

Avoiding overfitting

$\Rightarrow$ Pruning: allow the tree to overfit the data, and then post-prune it replacing subtrees by a leaf

Post-pruning

- In down-top manner, pruning means removing the subtree rooted at a node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node
- Prune is done only if the resulting tree performs no worse than the original, this being computed over a pruning set
- Prune iteratively, choosing the node whose removal most increases the accuracy
- ...until further pruning is harmful (accuracy decreases)

Continuous attributes

Discretize them

- Partition into a discrete set of intervals, e.g. 2 intervals $A < c$ and $A \geq c$
- How to select c? we would like one that produces the greatest info gain
- It can be shown that this value always lies where the class value changes, once the examples are sorted according to the continuous attribute A

$\Rightarrow$ Sort the examples according to A, take 2 adjacent A-values with different C-labels and average them

$\Rightarrow$ Evaluate each candidate to be c by computing the info gain associated with each

$\Rightarrow$ With this chosen c, this new attribute competes with the other discrete candidate attributes

$\Rightarrow$ If A is not chosen, c may be different later

$\Rightarrow$ I.e., the attribute is created dynamically

Continuous attributes

Discretize them

- Example: candidates are $c_1 = (48 + 60)/2 = 54$ and $c_2 = (80 + 90)/2 = 85$:

Temperature	40	48	60	72	80	90
PlayTennis	No	No	Yes	Yes	Yes	No

Order the instances according to the continuous variable

The continuous variables can appear several times on the nodes. This can't happen with the discrete variables

Outline

C4.5 algorithm [Quinlan, 1993]

C4.5

- Choose attributes by using gain ratio (maximize it). Penalizes attributes with many values and many uniformly distributed values

It avoids problems with the variables that have high entropy

$$I(C, X_i)/H(X_i) \quad \text{(gain ratio)}$$

- Incorporate post-pruning rules: generate rules (1 per path) and eliminate preconditions (antecedents) whenever it doesn't worse the error
  - Convert the tree into a set of rules R
  - *Error* = classification error with R
  - For each rule r from R: If we remove all the antecedents from a rule, we will remove the whole rule
    - *New-error* = error after eliminating antecedent j from r
    - If *New-error* $\leq$ *Error*, then *New-error* = *Error* and eliminate from r this antecedent
  - If there are no more antecedents in r, delete r from R
- We can prune contexts (paths), rather than subtrees Prune the rules
- Last version: C4.8, which is C4.5 revision 8, last public version of this family of algorithms before the commercial implementation C5.0
  $\Rightarrow$ J48 is the name in Weka

Outline

Software

Trees with Weka

`Classifier` ⇒ `Trees`

Id3 J48

Conclusions

Classification trees

- ID3 greedily selects the next attribute to be added in the tree
- Avoiding overfitting is important to have a tree that generalizes well $\Rightarrow$ Pruning the tree
- Extensions to ID3 to handle continuous attributes
- Trees are robust to errors in (training) data, both in the class variable and attributes
- Data may contain *missing* attribute values
- ...extensions to real-valued outputs: regression trees, with linear regressions at the leaves –algorithms M5 and CART–

Bibliography

Texts

- Alpaydin, E (2004) *Introduction to Machine Learning*, MIT Press [Chap. 9]
- Duda, R., Hart, P.E., Stork, D.G. (2001) *Pattern Classification*, Wiley [Chap. 8]
- Mitchell, T. (1997) *Machine Learning*, McGraw-Hill [Chap. 3]
- Webb, A. (2002) *Statistical Pattern Recognition* Wiley [Chap. 7]
- Witten, I., Frank, E. (2005) *Data Mining*, Morgan Kaufmann, 2nd ed. [Sections 4.3, 6.1, 6.5, 10.4]

Papers

- Quinlan, J.R. (1986) Induction of trees, *Machine Learning*, 1, 81-106. [ID3]
- Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann. [C4.5]