Classification and regression trees

Pierre Geurts
p.geurts@ulg.ac.be

Last update: 22/09/2013
Outline

• Supervised learning
• Decision tree representation
• Decision tree learning
• Extensions
• Regression trees
• By-products
**Database**

- A collection of objects (rows) described by attributes (columns)

<table>
<thead>
<tr>
<th>checkingaccount</th>
<th>duration</th>
<th>purpose</th>
<th>amount</th>
<th>savings</th>
<th>yearsemployed</th>
<th>age</th>
<th>good or bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>0&lt;=…&lt;200 DM</td>
<td>48</td>
<td>radiotv</td>
<td>5951</td>
<td>...&lt;100 DM</td>
<td>1&lt;...&lt;4</td>
<td>22</td>
<td>bad</td>
</tr>
<tr>
<td>...&lt;0 DM</td>
<td>6</td>
<td>radiotv</td>
<td>1169</td>
<td>unknown</td>
<td>...&gt;7</td>
<td>67</td>
<td>good</td>
</tr>
<tr>
<td>no</td>
<td>12</td>
<td>education</td>
<td>2096</td>
<td>...&lt;100 DM</td>
<td>4&lt;...&lt;7</td>
<td>49</td>
<td>good</td>
</tr>
<tr>
<td>...&lt;0 DM</td>
<td>42</td>
<td>furniture</td>
<td>7882</td>
<td>...&lt;100 DM</td>
<td>4&lt;...&lt;7</td>
<td>45</td>
<td>good</td>
</tr>
<tr>
<td>...&lt;0 DM</td>
<td>24</td>
<td>newcar</td>
<td>4870</td>
<td>...&lt;100 DM</td>
<td>1&lt;...&lt;4</td>
<td>53</td>
<td>bad</td>
</tr>
<tr>
<td>no</td>
<td>36</td>
<td>education</td>
<td>9055</td>
<td>unknown</td>
<td>1&lt;...&lt;4</td>
<td>35</td>
<td>good</td>
</tr>
<tr>
<td>no</td>
<td>24</td>
<td>furniture</td>
<td>2835</td>
<td>500&lt;...&lt;1000 DM</td>
<td>...&gt;7</td>
<td>53</td>
<td>good</td>
</tr>
<tr>
<td>0&lt;=…&lt;200 DM</td>
<td>36</td>
<td>usedcar</td>
<td>6948</td>
<td>...&lt;100 DM</td>
<td>1&lt;...&lt;4</td>
<td>35</td>
<td>good</td>
</tr>
<tr>
<td>no</td>
<td>12</td>
<td>radiotv</td>
<td>3059</td>
<td>...&gt;1000 DM</td>
<td>4&lt;...&lt;7</td>
<td>61</td>
<td>good</td>
</tr>
<tr>
<td>0&lt;=…&lt;200 DM</td>
<td>30</td>
<td>newcar</td>
<td>5234</td>
<td>...&lt;100 DM</td>
<td>unemployed</td>
<td>28</td>
<td>bad</td>
</tr>
<tr>
<td>0&lt;=…&lt;200 DM</td>
<td>12</td>
<td>newcar</td>
<td>1295</td>
<td>...&lt;100 DM</td>
<td>...&lt;1</td>
<td>25</td>
<td>bad</td>
</tr>
<tr>
<td>...&lt;0 DM</td>
<td>48</td>
<td>business</td>
<td>4308</td>
<td>...&lt;100 DM</td>
<td>...&lt;1</td>
<td>24</td>
<td>bad</td>
</tr>
<tr>
<td>0&lt;=…&lt;200 DM</td>
<td>12</td>
<td>radiotv</td>
<td>1567</td>
<td>...&lt;100 DM</td>
<td>1&lt;...&lt;4</td>
<td>22</td>
<td>good</td>
</tr>
</tbody>
</table>
Supervised learning

- **Goal:** from the database, find a function $f$ of the inputs that approximate at best the output

- **Discrete output $→$ classification problem**
- **Continuous output $→$ regression problem**
Examples of application (1)

• Predict whether a bank client will be a good debtor or not

• Image classification:
  – Handwritten characters recognition:

    3 → 3  5 → 5

  – Face recognition
Examples of application (2)

- Classification of cancer types from gene expression profiles (Golub et al (1999))

<table>
<thead>
<tr>
<th>Nº patient</th>
<th>Gene 1</th>
<th>Gene 2</th>
<th>...</th>
<th>Gene 7129</th>
<th>Leucimia</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-134</td>
<td>28</td>
<td>...</td>
<td>123</td>
<td>AML</td>
</tr>
<tr>
<td>2</td>
<td>-123</td>
<td>0</td>
<td>...</td>
<td>17</td>
<td>AML</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
<td>-123</td>
<td>...</td>
<td>-23</td>
<td>ALL</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>72</td>
<td>89</td>
<td>-123</td>
<td>...</td>
<td>12</td>
<td>ALL</td>
</tr>
</tbody>
</table>
Learning algorithm

• It receives a learning sample and returns a function $h$
• A learning algorithm is defined by:
  – A hypothesis space $H$ (=a family of candidate models)
  – A quality measure for a model
  – An optimisation strategy

A model ($h \in H$) obtained by automatic learning
Decision (classification) trees

• A learning algorithm that can handle:
  – Classification problems (binary or multi-valued)
  – Attributes may be discrete (binary or multi-valued) or continuous.

• Classification trees were invented twice:
  – By statisticians: CART (Breiman et al.)
  – By the AI community: ID3, C4.5 (Quinlan et al.)
A decision tree is a tree where:
- Each *interior node* tests an attribute
- Each *branch* corresponds to an attribute value
- Each *leaf* node is labelled with a class
# A simple database: playtennis

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
A decision tree for playtennis

- **Outlook**
  - Sunny
  - Overcast
  - Rain
- **Humidity**
  - High
  - Normal
- **Wind**
  - Strong
  - Weak
Tree learning

- Tree learning = choose the tree structure and determine the predictions at leaf nodes
- Predictions: to minimize the misclassification error, associate the majority class among the learning sample cases reaching this node

```
<table>
<thead>
<tr>
<th>Outlook</th>
<th>Humidity</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>High</td>
<td>Strong</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Weak</td>
</tr>
<tr>
<td>Overcast</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Rain</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
```

- 25 yes, 40 no
- 15 yes, 10 no
- 14 yes, 2 no
How to generate trees? (1)

- What properties do we want the decision tree to have?

1. It should be consistent with the learning sample (for the moment)
   - Trivial algorithm: construct a decision tree that has one path to a leaf for each example
   - Problem: it does not capture useful information from the database
How to generate trees? (2)

• What properties do we want the decision tree to have?

2. It should be at the same time as simple as possible
   – Trivial algorithm: generate all trees and pick the simplest one that is consistent with the learning sample.
   – Problem: intractable, there are too many trees
Top-down induction of DTs (1)

- Choose « best » attribute
- Split the learning sample
- Proceed recursively until each object is correctly classified

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Top-down induction of DTs (2)

Procedure learn_dt(learning sample, LS)

• If all objects from LS have the same class
  – Create a leaf with that class
• Else
  – Find the « best » splitting attribute A
  – Create a test node for this attribute
  – For each value $a$ of $A$
    • Build $LS_a = \{o \in LS \mid A(o) \text{ is } a\}$
    • Use Learn_dt($LS_a$) to grow a subtree from $LS_a$. 
Properties of TDIDT

• Hill-climbing algorithm in the space of possible decision trees.
  – It adds a sub-tree to the current tree and continues its search
  – It does not backtrack

• Sub-optimal but very fast

• Highly dependent upon the criterion for selecting attributes to test
Which attribute is best?

- We want a small tree
  - We should maximize the class separation at each step, i.e. make successors as pure as possible
  - \( \Rightarrow \) it will favour short paths in the trees
Impurity

• Let $LS$ be a sample of objects, $p_j$ the proportions of objects of class $j$ ($j=1,\ldots,J$) in $LS$,

• Define an **impurity** measure $I(LS)$ that satisfies:
  
  – $I(LS)$ is minimum only when $p_i=1$ and $p_j=0$ for $j\neq i$  
    (all objects are of the same class)  
  – $I(LS)$ is maximum only when $p_j=1/J$  
    (there is exactly the same number of objects of all classes)  
  – $I(LS)$ is symmetric with respect to $p_1,\ldots,p_J$
Reduction of impurity

• The “best” split is the split that maximizes the expected reduction of impurity

\[ \Delta I(LS, A) = I(LS) - \sum_a \frac{|LS_a|}{|LS|} I(LS_a) \]

where \( LS_a \) is the subset of objects from \( LS \) such that \( A=a \).

• \( \Delta I \) is called a score measure or a splitting criterion

• There are many other ways to define a splitting criterion that do not rely on an impurity measure
Example of impurity measure (1)

- Shannon’s entropy:
  - \( H(\mathcal{LS}) = -\sum_j p_j \log p_j \)
  - If two classes, \( p_1 = 1 - p_2 \)

- Entropy measures impurity, uncertainty, surprise...
- The reduction of entropy is called the information gain
Example of impurity measure (2)

• Which attribute is best?

\[ \Delta I(LS, A1) = 0.99 - \left( \frac{26}{64} \right) 0.71 - \left( \frac{38}{64} \right) 0.75 \]
\[ = 0.25 \]

\[ \Delta I(LS, A2) = 0.99 - \left( \frac{51}{64} \right) 0.94 - \left( \frac{13}{64} \right) 0.62 \]
\[ = 0.12 \]
Other impurity measures

- Gini index:
  \[ I(\text{LS}) = \sum_j p_j (1-p_j) \]

- Misclassification error rate:
  \[ I(\text{LS}) = 1 - \max_j p_j \]

- two-class case:
Playtennis problem

- Which attribute should be tested here?
  - $\Delta I(LS,\text{Temp.}) = 0.970 - (3/5) 0.918 - (1/5) 0.0 - (1/5) 0.0 = 0.419$
  - $\Delta I(LS,\text{Hum.}) = 0.970 - (3/5) 0.0 - (2/5) 0.0 = 0.970$
  - $\Delta I(LS,\text{Wind}) = 0.970 - (2/5) 1.0 - (3/5) 0.918 = 0.019$

- $\Rightarrow$ the best attribute is Humidity
Overfitting (1)

• Our trees are perfectly consistent with the learning sample
• But, often, we would like them to be good at predicting classes of unseen data from the same distribution (generalization).
• A tree $T$ overfits the learning sample iff $\exists T'$ such that:
  – $Error_{LS}(T) < Error_{LS}(T')$
  – $Error_{unseen}(T) > Error_{unseen}(T')$
• In practice, $\text{Error}_{\text{unseen}}(T)$ is estimated from a separate test sample
Reasons for overfitting (1)

- Data is noisy or attributes don’t completely predict the outcome

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D15</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

Add a test here
Reasons for overfitting (2)

• Data is incomplete (not all cases covered)

• We do not have enough data in some part of the learning sample to make a good decision
How can we avoid overfitting?

- **Pre-pruning**: stop growing the tree earlier, before it reaches the point where it perfectly classifies the learning sample.

- **Post-pruning**: allow the tree to overfit and then post-prune the tree.

- **Ensemble methods** (this afternoon)
Pre-pruning

• Stop splitting a node if
  – The number of objects is too small
  – The impurity is low enough
  – The best test is not statistically significant
    (according to some statistical test)

• Problem:
  – the optimum value of the parameter \( n, I_{th}, \)
    significance level) is problem dependent.
  – We may miss the optimum
Post-pruning (1)

• Split the learning sample $LS$ into two sets:
  – a growing sample $GS$ to build the tree
  – A validation sample $VS$ to evaluate its generalization error
• Build a complete tree from $GS$
• Compute a sequence of trees $\{T_1, T_2, \ldots\}$ where
  – $T_1$ is the complete tree
  – $T_i$ is obtained by removing some test nodes from $T_{i-1}$
• Select the tree $T_i^*$ from the sequence that minimizes the error on $VS$
Post-pruning (2)

Error vs. Complexity

Underfitting

Overfitting

Tree growing

Tree pruning

Error on VS

Error on GS

Optimal tree
Post-pruning (3)

• How to build the sequence of trees?
  – Reduced error pruning:
    • At each step, remove the node that most decreases the error on $VS$
  – Cost-complexity pruning:
    • Define a cost-complexity criterion:
      – $\text{Error}_{GS}(T) + \alpha \cdot \text{Complexity}(T)$
    • Build the sequence of trees that minimize this criterion for increasing $\alpha$
Post-pruning (4)

$T_1$

Error $_{GS}=0\%$, Error $_{VS}=10\%$

$T_2$

Error $_{GS}=6\%$, Error $_{VS}=8\%$

$T_3$

Error $_{GS}=13\%$, Error $_{VS}=15\%$

$T_4$

Error $_{GS}=27\%$, Error $_{VS}=25\%$

$T_5$

Error $_{GS}=33\%$, Error $_{VS}=35\%$
Post-pruning (5)

• Problem: require to dedicate one part of the learning sample as a validation set ⇒ may be a problem in the case of a small database

• Solution: $N$-fold cross-validation
  – Split the training set into $N$ parts (often 10)
  – Generate $N$ trees, each leaving one part among $N$
  – Make a prediction for each learning object with the (only) tree built without this case.
  – Estimate the error of this prediction

• May be combined with pruning
How to use decision trees?

• Large datasets (ideal case):
  – Split the dataset into three parts: $GS$, $VS$, $TS$
  – Grow a tree from $GS$
  – Post-prune it from $VS$
  – Test it on $TS$

• Small datasets (often)
  – Grow a tree from the whole database
  – Pre-prune with default parameters (risky), post-prune it by 10-fold cross-validation (costly)
  – Estimate its accuracy by 10-fold cross-validation
Outline

• Supervised learning
• Tree representation
• Tree learning
• Extensions
  – Continuous attributes
  – Attributes with many values
  – Missing values
• Regression trees
• By-products
Continuous attributes (1)

• Example: temperature as a number instead of a discrete value

• Two solutions:
  – Pre-discretize: Cold if Temperature<70, Mild between 70 and 75, Hot if Temperature>75
  – Discretize during tree growing:

```
<table>
<thead>
<tr>
<th>Temperature</th>
<th>≤65.4</th>
<th>&gt;65.4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
```

• How to find the cut-point?
### Continuous attributes (2)

<table>
<thead>
<tr>
<th>Temp.</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>No</td>
</tr>
<tr>
<td>85</td>
<td>No</td>
</tr>
<tr>
<td>83</td>
<td>Yes</td>
</tr>
<tr>
<td>75</td>
<td>Yes</td>
</tr>
<tr>
<td>68</td>
<td>Yes</td>
</tr>
<tr>
<td>65</td>
<td>No</td>
</tr>
<tr>
<td>64</td>
<td>Yes</td>
</tr>
<tr>
<td>72</td>
<td>No</td>
</tr>
<tr>
<td>71</td>
<td>No</td>
</tr>
<tr>
<td>75</td>
<td>Yes</td>
</tr>
<tr>
<td>70</td>
<td>Yes</td>
</tr>
<tr>
<td>69</td>
<td>Yes</td>
</tr>
<tr>
<td>81</td>
<td>Yes</td>
</tr>
<tr>
<td>72</td>
<td>Yes</td>
</tr>
<tr>
<td>80</td>
<td>No</td>
</tr>
<tr>
<td>81</td>
<td>Yes</td>
</tr>
<tr>
<td>83</td>
<td>Yes</td>
</tr>
<tr>
<td>85</td>
<td>No</td>
</tr>
</tbody>
</table>

- **Temp.**< 64.5  \(\Delta I=0.048\)
- **Temp.**< 66.5  \(\Delta I=0.010\)
- **Temp.**< 68.5  \(\Delta I=0.000\)
- **Temp.**< 69.5  \(\Delta I=0.015\)
- **Temp.**< 70.5  \(\Delta I=0.045\)
- **Temp.**< 71.5  \(\Delta I=0.001\)
- **Temp.**< 73.5  \(\Delta I=0.001\)
- **Temp.**< 77.5  \(\Delta I=0.025\)
- **Temp.**< 80.5  \(\Delta I=0.000\)
- **Temp.**< 82   \(\Delta I=0.010\)
- **Temp.**< 84   \(\Delta I=0.113\)
## Continuous attribute (3)

The table below shows the values for A1 and A2 for different samples, along with their corresponding colors.

<table>
<thead>
<tr>
<th>Number</th>
<th>A1</th>
<th>A2</th>
<th>Colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.58</td>
<td>0.75</td>
<td>Red</td>
</tr>
<tr>
<td>2</td>
<td>0.78</td>
<td>0.65</td>
<td>Red</td>
</tr>
<tr>
<td>3</td>
<td>0.89</td>
<td>0.23</td>
<td>Green</td>
</tr>
<tr>
<td>4</td>
<td>0.12</td>
<td>0.98</td>
<td>Red</td>
</tr>
<tr>
<td>5</td>
<td>0.17</td>
<td>0.26</td>
<td>Green</td>
</tr>
<tr>
<td>6</td>
<td>0.50</td>
<td>0.48</td>
<td>Red</td>
</tr>
<tr>
<td>7</td>
<td>0.45</td>
<td>0.16</td>
<td>Green</td>
</tr>
<tr>
<td>8</td>
<td>0.80</td>
<td>0.75</td>
<td>Green</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>100</td>
<td>0.75</td>
<td>0.13</td>
<td>Green</td>
</tr>
</tbody>
</table>

The scatter plot on the right side of the page provides a visual representation of the data, with A1 and A2 plotted against each other. The data points are color-coded according to the color attribute, and a decision tree is shown, indicating the classification based on the values of A2.

### Decision Tree
- If A2 < 0.33, then the sample is classified as good.
- If A2 ≥ 0.33, then:
  - If A2 < 0.65, then the sample is classified as bad.
  - If A2 ≥ 0.65, then the sample is classified as good.
Attributes with many values (1)

- Problem:
  - Not good splits: they fragment the data too quickly, leaving insufficient data at the next level
  - The reduction of impurity of such test is often high (example: split on the object id).

- Two solutions:
  - Change the splitting criterion to penalize attributes with many values
  - Consider only binary splits (preferable)
Attributes with many values (2)

• Modified splitting criterion:
  – Gainratio($LS,A$) = $\Delta H(LS,A)/\text{Splitinformation}(LS,A)$
  – Splitinformation($LS,A$) = $-\sum_a |LS_a|/|LS| \log(|LS_a|/|LS|)$
  – The split information is high when there are many values

• Example: outlook in the playtennis
  – $\Delta H(LS,\text{outlook}) = 0.246$
  – Splitinformation($LS,\text{outlook}$) = 1.577
  – Gainratio($LS,\text{outlook}$) = $0.246/1.577=0.156 < 0.246$

• Problem: the gain ratio favours unbalanced tests
Attributes with many values (3)

- Allow binary tests only:

- There are $2^N - 1$ possible subsets for $N$ values
- If $N$ is small, determination of the best subsets by enumeration
- If $N$ is large, heuristics exist (e.g. greedy approach)
Missing attribute values

• Not all attribute values known for every objects when learning or when testing

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D15</td>
<td>Sunny</td>
<td>Hot</td>
<td>?</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

• Three strategies:
  – Assign most common value in the learning sample
  – Assign most common value in tree
  – Assign probability to each possible value
Regression trees (1)

- Tree for regression: exactly the same model but with a number in each leaf instead of a class

```
<table>
<thead>
<tr>
<th>Outlook</th>
<th>Humidity</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>22.3</td>
<td></td>
</tr>
<tr>
<td>Overcast</td>
<td></td>
<td>45.6</td>
</tr>
<tr>
<td>Rain</td>
<td>64.4</td>
<td>7.4</td>
</tr>
</tbody>
</table>

Temperature:
- <71: 1.2
- >71: 3.4
```
Regression trees (2)

- A regression tree is a piecewise constant function of the input attributes

\[
X_1 \leq t_1 \quad X_2 \leq t_2 \quad X_1 \leq t_3 \quad X_2 \leq t_4
\]

\[
r_1 \quad r_2 \quad r_3 \quad r_4 \quad r_5
\]
Regression tree growing

- To minimize the square error on the learning sample, the prediction at a leaf is the average output of the learning cases reaching that leaf.
- Impurity of a sample is defined by the variance of the output in that sample:
  \[
  I(LS) = \text{var}_{y|LS}\{y\} = E_{y|LS}\{(y - E_{y|LS}\{y\})^2\}
  \]
- The best split is the one that reduces the most variance:
  \[
  \Delta I(LS, A) = \text{var}_{y|LS}\{y\} - \sum_{a} \left| \frac{LS_a}{LS} \right| \text{var}_{y|LS_a}\{y\}
  \]
Regression tree pruning

- Exactly the same algorithms apply: pre-pruning and post-pruning.
- In post-pruning, the tree that minimizes the squared error on $V_S$ is selected.

- In practice, pruning is more important in regression because full trees are much more complex (often all objects have a different output values and hence the full tree has as many leaves as there are objects in the learning sample)
Outline

• Supervised learning
• Tree representation
• Tree learning
• Extensions
• Regression trees
• By-products
  – Interpretability
  – Variable selection
  – Variable importance
Interpretability (1)

• Obvious

• Compare with a neural networks:
Interpretability (2)

- A tree may be converted into a set of rules
  - If (outlook=sunny) and (humidity=high) then PlayTennis=No
  - If (outlook=sunny) and (humidity=normal) then PlayTennis=Yes
  - If (outlook=overcast) then PlayTennis=Yes
  - If (outlook=rain) and (wind=strong) then PlayTennis=No
  - If (outlook=rain) and (wind=weak) then PlayTennis=Yes
Attribute selection

• If some attributes are not useful for classification, they will not be selected in the (pruned) tree

• Of practical importance, if measuring the value of an attribute is costly (e.g. medical diagnosis)

• Decision trees are often used as a pre-processing for other learning algorithms that suffer more when there are irrelevant variables
Variable importance

• In many applications, all variables do not contribute equally in predicting the output.
• We can evaluate variable importance by computing the total reduction of impurity brought by each variable:
  \[ \text{Imp}(A) = \sum_{\text{nodes where } A \text{ is tested}} |L_{S_{\text{node}}}| \Delta I(L_{S_{\text{node}}}, A) \]
When are decision trees useful?

• Advantages
  – Very fast: can handle very large datasets with many attributes (Complexity $O(n.N \log N)$)
  – Flexible: several attribute types, classification and regression problems, missing values...
  – Interpretability: provide rules and attribute importance

• Disadvantages
  – Instability of the trees (high variance)
  – Not always competitive with other algorithms in terms of accuracy
Further extensions and research

- Cost and un-balanced learning sample
- Oblique trees (test like $\sum \alpha_i A_i < a_{th}$)
- Using predictive models in leaves (e.g. linear regression)
- Induction graphs
- Fuzzy decision trees (from a crisp partition to a fuzzy partition of the learning sample)
Demo

• Illustration on two datasets:
  – titanic
    • http://www.cs.toronto.edu/~delve/data/titanic/desc.html
  – splice junction
    • http://www.cs.toronto.edu/~delve/data/splice/desc.html
References

• About tree algorithms:
  – *Classification and regression trees*, L.Breiman et al., Wadsworth, 1984
  – *C4.5: programs for machine learning*, J.R.Quinlan, Morgan Kaufmann, 1993

• More general textbooks:
  – *Pattern classification*, R.O.Duda et al., John Wiley and sons, 2000
Softwares

• scikit-learn:
  – http://scikit-learn.org

• Weka
  – J48
  – http://www.cs.waikato.ac.nz/ml/weka

• In R:
  – Packages tree and rpart

• C4.5:
  – http://www.cse.unwe.edu.au/~quinlan

• Java applet:
  – http://www.montefiore.ulg.ac.be/~geurts/