Classification

Decision Trees
Learning Methods

• Learning methods differ in terms of:
  – The form of hypothesis (or function)
  – The way the computer finds a hypothesis from the data (the algorithm)

• One of the most popular learning algorithm makes hypotheses in the form of **decision trees**.

• In a decision tree, each node represents a question, and the arcs represent possible answers.
Example of a Decision Tree

Training Data

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Model: Decision Tree

Splitting Attributes

Refund

MarSt: Single, Divorced

TaxInc:

< 80K

Yes

NO

> 80K

YES

NO
Another Example of Decision Tree

There could be more than one tree that fits the same data!
Apply Model to Test Data

Start from the root of tree.

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund

Start from the root of tree.
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund

- Yes
  - NO
- No
  - MarSt
    - Single, Divorced
    - TaxInc
      - < 80K
        - NO
      - > 80K
        - YES
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund → NO

Marital Status

Married

Taxable Income

< 80K

> 80K

Cheat

NO

YES
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Decision Tree:
- Refund: No
- Marital Status: Married
- Taxable Income: 80K
- Cheat: ?
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund

Refund

MarSt

Married

TaxInc

< 80K

< 80K

> 80K

> 80K

NO

NO

YES

NO

NO

YES
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Assign Cheat to “No”
Digression: Entropy
Information - Bits

- We are watching a set of independent random samples of $X$.
- We see that $X$ has four possible values.

<table>
<thead>
<tr>
<th>$P(X=A)$</th>
<th>$P(X=B)$</th>
<th>$P(X=C)$</th>
<th>$P(X=D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
</tbody>
</table>

- So we might see: BAACBADCDDDDA...
- We transmit data over a binary serial link. We can encode each reading with two bits (e.g. $A=00$, $B=01$, $C=10$, $D = 11$)

01000010010011101100111111100...
Fewer Bits

• Someone tells us that the probabilities are not equal

| P(X=A) = 1/2 | P(X=B) = 1/4 | P(X=C) = 1/8 | P(X=D) = 1/8 |

• It’s possible...

...to invent a coding for your transmission that only uses 1.75 bits on average per symbol. Here is one.

<table>
<thead>
<tr>
<th>A</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
</tr>
<tr>
<td>D</td>
<td>111</td>
</tr>
</tbody>
</table>
General Case

• Suppose $X$ can have one of $m$ values...

| $P(X=V_1) = p_1$ | $P(X=V_2) = p_2$ | .... | $P(X=V_m) = p_m$ |

• What’s the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from $X$’s distribution? It’s

$$\text{entropy}(p_1, \ldots, p_m) = - p_1 \log_2 p_1 - \ldots - p_m \log_2 p_m$$

• Shannon (1948) got to this formula by setting down several desirable properties for uncertainty, and then finding it.
Entropy

\[ H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \ldots - p_m \log_2 p_m \]

\[ = - \sum_{j=1}^{m} p_j \log_2 p_j \]

\[ H(X) = \text{The entropy of } X \]

- **“High Entropy”** means \( X \) is from a uniform (boring) distribution.
- **“Low Entropy”** means \( X \) is from varied (peaks and valleys) distribution

A histogram of the frequency distribution of the values of \( X \) would be flat.

... and so the values sampled from it would be all over the place.

A histogram of the frequency distribution of the values of \( X \) would have many lows and a couple of highs.

... and so the values sampled from it would be more predictable.
Back to Decision Trees
Constructing decision trees (ID3)

- Normal procedure: top-down in a recursive divide-and-conquer fashion
  - First: an attribute is selected for root node and a branch is created for each possible attribute value
  - Then: the instances are split into subsets (one for each branch extending from the node)
  - Finally: the same procedure is repeated recursively for each branch, using only instances that reach the branch
- Process stops if all instances in a branch have the same class, or there is no attribute to split on
## Weather data

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>
Which attribute to select?

(a) outlook
- sunny: yes, yes, no, no
- overcast: yes, yes, yes, yes
- rainy: yes, yes, no, no

(b) temperature
- hot: yes, yes, yes, yes
- mild: yes, yes, no, no
- cool: yes, yes, no, no

(c) humidity
- high: yes, yes, yes, yes
- normal: no, no, no, no

(d) windy
- false: yes, yes, yes, no
- true: yes, yes, no, no
A criterion for attribute selection

- Which is the best attribute?

- The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the “purest” nodes

- Popular impurity criterion: entropy of nodes
  - Lower the entropy purer the node.

- **Strategy**: choose attribute that results in lowest entropy of the children nodes.
Attribute “Outlook”

outlook=sunny

\[
\text{info}(\[2,3\]) = \text{entropy}(2/5,3/5) = -2/5 \log(2/5) -3/5 \log(3/5) = .971
\]

outlook=overcast

\[
\text{info}(\[4,0\]) = \text{entropy}(4/4,0/4) = -1 \log(1) -0 \log(0) = 0
\]

outlook=rainy

\[
\text{info}(\[3,2\]) = \text{entropy}(3/5,2/5) = -3/5 \log(3/5)-2/5 \log(2/5) = .971
\]

Expected info:

\[
.971*(5/14) + 0*(4/14) + .971*(5/14) = .693
\]
Attribute “Temperature”

temperature=hot

\[\text{info([2,2])} = \text{entropy}(2/4,2/4) = -2/4\log(2/4) - 2/4\log(2/4) = 1\]

temperature=mild

\[\text{info([4,2])} = \text{entropy}(4/6,2/6) = -4/6\log(1) - 2/6\log(2/6) = .528\]

temperature=cool

\[\text{info([3,1])} = \text{entropy}(3/4,1/4) = -3/4\log(3/4) - 1/4\log(1/4) = .811\]

Expected info:

\[1 \times (4/14) + .528 \times (6/14) + .811 \times (4/14) = .744\]
Attribute “Humidity”

humidity=high

\[ \text{info}([3,4]) = \text{entropy}(\frac{3}{7}, \frac{4}{7}) = -\frac{3}{7}\log\left(\frac{3}{7}\right) -\frac{4}{7}\log\left(\frac{4}{7}\right) = .985 \]

humidity=normal

\[ \text{info}([6,1]) = \text{entropy}(\frac{6}{7}, \frac{1}{7}) = -\frac{6}{7}\log\left(\frac{6}{7}\right) -\frac{1}{7}\log\left(\frac{1}{7}\right) = .592 \]

Expected info:

\[ .985 \times \frac{7}{14} + .592 \times \frac{7}{14} = .788 \]
Attribute “Windy”

windy=false

\[ \text{info}([6,2]) = \text{entropy}(6/8,2/8) = -6/8 \log(6/8) -2/8 \log(2/8) = .811 \]

windy=true

\[ \text{info}([3,3]) = \text{entropy}(3/6,3/6) = -3/6 \log(3/6) -3/6 \log(3/6) = 1 \]

Expected info:
\[ .811 \times \frac{8}{14} + 1 \times \frac{6}{14} = .892 \]
And the winner is...

"Outlook"  .693
Temperature  .744
Humidity  .788
Windy  .899

...So, the root will be "Outlook"
Continuing to split (for Outlook="Sunny")

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Which one to choose?
Continuing to split (for Outlook="Sunny")

- **temperature=hot**: $\text{info}([2,0]) = \text{entropy}(2/2,0/2) = 0$
- **temperature=mild**: $\text{info}([1,1]) = \text{entropy}(1/2,1/2) = 1$
- **temperature=cool**: $\text{info}([1,0]) = \text{entropy}(1/1,0/1) = 0$

  **Expected info**: $(2/5)*0 + (2/5)*1 + (1/5) *0 = 0.4$

- **humidity=high**: $\text{info}([3,0]) = 0$
- **humidity=normal**: $\text{info}([2,0]) = 0$

  **Expected info**: $(3/5)*0 + (2/5) *0 = 0$

- **windy=false**: $\text{info}([1,2]) = \text{entropy}(1/3,2/3) =$
  
  
  $-1/3\log(1/3) -2/3\log(2/3) = 0.918$

- **windy=true**: $\text{info}([1,1]) = \text{entropy}(1/2,1/2) = 1$

  **Expected info**: $(3/5)* 0.918 + (2/5)*1 = 0.951$

**Winner is "humidity"**
Tree so far

outlook

- sunny
- overcast
- rainy

humidity

- high
- normal

- no
- yes
Continuing to split (for Outlook="Overcast")

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Nothing to split here, "play" is always "yes".

Tree so far
Continuing to split (for Outlook="Rainy")

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>

- We can easily see that "Windy" is the one to choose. (Why?)
The final decision tree

- Note: not all leaves need to be pure; sometimes identical instances have different classes

⇒ Splitting stops when data can't be split any further
Information gain

- Sometimes people don’t use directly the entropy of a node. Rather the information gain is being used.
- The result though is exactly the same.

- Info-gain =
  
  _information before split_ – _information after split_.

\[
\begin{align*}
\text{gain(Outlook)} & = \text{info([9,5])}-\text{info([2,3],[4,0],[3,2])} = .940-.693 = .247 \text{ bits} \\
\text{gain(Temp)} & = \text{info([9,5])}-\text{info([2,2],[4,2],[3,1])} = .940-.744 = .196 \text{ bits} \\
\text{gain(Humidity)} & = \text{info([9,5])}-\text{info([3,4],[6,1])} = .940-.788 = .152 \text{ bits} \\
\text{gain(Windy)} & = \text{info([9,5])}-\text{info([6,2],[3,3])} = .940-.892 = .048 \text{ bits}
\end{align*}
\]

- Clearly, greater the information gain better the purity of a node. So, we choose “Outlook” for the root.
Highly-branching attributes

- The weather data with ID code

<table>
<thead>
<tr>
<th>ID code</th>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>B</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>C</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>D</td>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>E</td>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>G</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>H</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>I</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>J</td>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>K</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>L</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>M</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>N</td>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>
• Entropy of split:
  \[ \text{info}([0,1],[0,1],[1,0],...,1,0],[0,1]) = 0 \]
• Info-gain is maximal, namely
  \[ \text{info\_gain} = \text{info} ([9,5]) - \text{info} ([0,1],[0,1],[1,0],...,1,0],[0,1]) \]
  \[ = .940 \]
Highly-branching attributes

So,

• Subsets are more likely to be pure if there is a large number of values:
  – Information gain is biased towards choosing attributes with a large number of values
  – May result in overfitting (selection of an attribute that is non-optimal for prediction)
Gain ratio and Intrinsic Information

- **Gain ratio**: a modification of the information gain that reduces its bias
  - It corrects the information gain by taking the intrinsic information of a split into account
  - Gain ratio takes number and size of branches into account when choosing an attribute

- **Intrinsic information**: entropy of split (with respect to the attribute on focus).
  - E.g. for attribute “ID Code”:
    \[
    \text{intrinsic(ID code)} = \text{info}[1,1,\ldots,1] \\
    = 14\cdot\left(-\frac{1}{14}\log\left(\frac{1}{14}\right)\right) = 3.807
    \]
Gain Ratio

gain_ratio(Attribute) =

\[
\frac{\text{gain(Attribute)}}{\text{intrinsic_info(Attribute)}}
\]

Example:

\[
gain\_ratio(\text{ID Code}) = \frac{\text{gain(ID code)}}{\text{intrinsic(ID code)}}
= 0.940 / 3.807 = 0.246
\]

\[
gain\_ratio(\text{Outlook}) = \frac{\text{gain(Outlook)}}{\text{intrinsic(Outlook)}}
= 0.247 / \text{info}[5,4,5] = 0.247 / 1.429 = 0.173
\]

“ID code” still wins, but its advantage is greatly reduced.
## Gain ratios for weather data

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Info:</td>
<td>0.693</td>
<td>Info:</td>
<td>0.911</td>
</tr>
<tr>
<td>Gain: 0.940-0.693</td>
<td>0.247</td>
<td>Gain:</td>
<td>0.940-0.911</td>
</tr>
<tr>
<td>Split info: info([5,4,5])</td>
<td>1.577</td>
<td>Split info: info([4,6,4])</td>
<td>1.362</td>
</tr>
<tr>
<td>Gain ratio: 0.247/1.577</td>
<td>0.156</td>
<td>Gain ratio: 0.029/1.362</td>
<td>0.021</td>
</tr>
<tr>
<td>Humidity</td>
<td>Windy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Info:</td>
<td>0.788</td>
<td>Info:</td>
<td>0.892</td>
</tr>
<tr>
<td>Gain: 0.940-0.788</td>
<td>0.152</td>
<td>Gain:</td>
<td>0.940-0.892</td>
</tr>
<tr>
<td>Split info: info([7,7])</td>
<td>1.000</td>
<td>Split info: info([8,6])</td>
<td>0.985</td>
</tr>
<tr>
<td>Gain ratio: 0.152/1</td>
<td>0.152</td>
<td>Gain ratio: 0.048/0.985</td>
<td>0.049</td>
</tr>
</tbody>
</table>
More on the gain ratio

• “Outlook” still comes out top but “Humidity” is now a much closer contender because it splits the data into two subsets instead of three.

• However: “ID code” has still greater gain ratio. But its advantage is greatly reduced.

• Problem with gain ratio: it may overcompensate
  – May choose an attribute just because its intrinsic information is very low
  – **Standard fix**: choose an attribute that maximizes the gain ratio, provided the information gain for that attribute is at least as great as the average information gain for all the attributes examined.
Discussion

• Algorithm for top-down induction of decision trees ("ID3") was developed by Ross Quinlan
  – University of Sydney Australia

• Gain-ratio is just one modification of this basic algorithm
  Led to development of C4.5, which can deal with
  – numeric attributes
  – missing values
  – noisy data
Random Forest Construction

Each tree is constructed using the following algorithm:

**Input**
- $N$ training cases with $M$ attributes each.
- Number $m <<< M$ of attributes to be used to determine the decision at each tree.

**Algorithm**
- Sample $N$ instances at random, with replacement from the original $N$ training cases. This sample will be the training data to grow the tree.
- For each tree, $m$ attributes are selected at random out of the $M$ and for each node in the tree the best on these $m$ attributes is used to split the data.
- ($m$ is constant during forest construction)
- Fully grow the tree.
Random Forest Prediction

- The new sample is pushed down a tree.
- It is assigned the label of the terminal node it ends up in.
- This procedure is iterated over all trees in the ensemble (forest), and the majority vote of all trees is reported as random forest prediction.