CHAPTER 9: Decision Trees

ETHEM ALPAYDIN
© The MIT Press, 2010
Edited and expanded for CS 4641 by Chris Simpkins

alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml2e
Overview

- Univariate decision trees
- Building classification trees
- Dealing with overfitting
- Extracting rules from decision trees
Tree Uses Nodes, and Leaves
Divide and Conquer

• Internal decision nodes
  • Univariate: Uses a single attribute, $x_i$
    • Numeric $x_i$: Binary split: $x_i > w_m$
    • Discrete $x_i$: $n$-way split for $n$ possible values
  • Multivariate: Uses all attributes, $x$

• Leaves
  • Classification: Class labels, or proportions
  • Regression: Numeric; $r$ average, or local fit

• Learning is greedy; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)
Classification Trees (ID3, CART, C4.5)

- For node $m$, $N_m$ instances reach $m$, $N^i_m$ belong to $C_i$

$$\hat{P}(C_i | x, m) \equiv p^i_m = \frac{N^i_m}{N_m}$$

- Node $m$ is pure if $p^i_m$ is 0 or 1
- Measure of impurity is entropy

$$I_m = - \sum_{i=1}^{K} p^i_m \log_2 p^i_m$$
Best Split

- If node $m$ is pure, generate a leaf and stop, otherwise split and continue recursively.
- Impurity after split: $N_{mj}$ of $N_m$ take branch $j$. $N^i_{mj}$ belong to $C_i$

$$\hat{P}(C_i \mid x, m, j) = p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$

$$J'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{k} p^i_{mj} \log_2 p^i_{mj}$$

- Find the variable and split that min impurity (among all variables -- and split positions for numeric variables)
GenerateTree($\mathcal{X}$)

If NodeEntropy($\mathcal{X}$) < $\theta_I$ /* eq. 9.3
Create leaf labelled by majority class in $\mathcal{X}$
Return

$i \leftarrow$ SplitAttribute($\mathcal{X}$)

For each branch of $x_i$

Find $\mathcal{X}_i$ falling in branch

GenerateTree($\mathcal{X}_i$)

SplitAttribute($\mathcal{X}$)

MinEnt $\leftarrow$ MAX

For all attributes $i = 1, \ldots, d$

If $x_i$ is discrete with $n$ values

Split $\mathcal{X}$ into $\mathcal{X}_1, \ldots, \mathcal{X}_n$ by $x_i$

$e \leftarrow$ SplitEntropy($\mathcal{X}_1, \ldots, \mathcal{X}_n$) /* eq. 9.8 */

If $e < $MinEnt MinEnt $\leftarrow e$; bestf $\leftarrow i$

Else /* $x_i$ is numeric */

For all possible splits

Split $\mathcal{X}$ into $\mathcal{X}_1, \mathcal{X}_2$ on $x_i$

$e \leftarrow $SplitEntropy($\mathcal{X}_1, \mathcal{X}_2$)

If $e < $MinEnt MinEnt $\leftarrow e$; bestf $\leftarrow i$

Return bestf
Summary of Main Loop

- Pick $A$, the “best” decision attribute for examples at current node using impurity measure
- For each value of $A$, create new descendant leaf nodes of current node
- Sort training examples to leaf nodes according to their values for attribute $A$
- If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes (turning some of them into internal nodes)
Example: Play tennis today?¹

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperatur</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</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>High</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>Normal</td>
<td>Strong</td>
<td>Yes</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>Cool</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>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</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>

¹Example from Mitchell 1997
Choosing the first split attribute

- **Outlook:**

\[
I'_{\text{root}} = - \sum_{j \in \{\text{Sunny, Overcast, Rain}\}} \frac{N_j}{N} \sum_{i \in \{\text{Yes, No}\}} p_j^i \log_2 p_j^i
\]

\[
= -\left[\frac{5}{14} \left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5}\right) + \frac{4}{14} \left(\frac{4}{4} \log_2 \frac{4}{4} + \frac{0}{4} \log_2 \frac{0}{4}\right) + \frac{5}{14} \left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5}\right)\right]
\]

\[= 0.693\]

- **Temperature:**

\[
I'_{\text{root}} = - \sum_{j \in \{\text{Hot, Mild, Cool}\}} \frac{N_j}{N} \sum_{i \in \{\text{Yes, No}\}} p_j^i \log_2 p_j^i
\]

\[
= -\left[\frac{4}{14} \left(\frac{2}{4} \log_2 \frac{2}{4} + \frac{2}{4} \log_2 \frac{2}{4}\right) + \frac{6}{14} \left(\frac{4}{6} \log_2 \frac{4}{6} + \frac{2}{6} \log_2 \frac{2}{6}\right) + \frac{4}{14} \left(\frac{3}{4} \log_2 \frac{3}{4} + \frac{1}{4} \log_2 \frac{1}{4}\right)\right]
\]

\[= 0.915\]

- **Humidity:**

\[
I'_{\text{root}} = - \sum_{j \in \{\text{High, Normal}\}} \frac{N_j}{N} \sum_{i \in \{\text{Yes, No}\}} p_j^i \log_2 p_j^i
\]

\[
= -\left[\frac{7}{14} \left(\frac{3}{7} \log_2 \frac{3}{7} + \frac{4}{7} \log_2 \frac{4}{7}\right) + \frac{7}{14} \left(\frac{6}{7} \log_2 \frac{6}{7} + \frac{1}{7} \log_2 \frac{1}{7}\right)\right]
\]

\[= 0.789\]

- **Wind:**

\[
I'_{\text{root}} = - \sum_{j \in \{\text{Strong, Weak}\}} \frac{N_j}{N} \sum_{i \in \{\text{Yes, No}\}} p_j^i \log_2 p_j^i
\]

\[
= -\left[\frac{6}{14} \left(\frac{3}{6} \log_2 \frac{3}{6} + \frac{3}{6} \log_2 \frac{3}{6}\right) + \frac{8}{14} \left(\frac{6}{8} \log_2 \frac{6}{8} + \frac{2}{8} \log_2 \frac{2}{8}\right)\right]
\]

\[= 0.892\]
Which attribute should be tested here?
Choosing an attribute for Sunny node

- $X_{\text{Sunny}} = \{D_1, D_2, D_8, D_9, D_{11}\}$

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperatur</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</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>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D_{11}</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- **Humidity:** 
  
  $I'_{\text{sunny,humidity}} = - \sum_{j \in \{\text{High, Normal}\}}^{N_j} \frac{N_j}{N} \sum_{i \in \{\text{Yes, No}\}} p_i^j \log_2 p_i^j$
  
  $= -\left[\frac{3}{5}(\frac{0}{3}\log_2 \frac{0}{3} + \frac{3}{3}\log_2 \frac{3}{3}) + \frac{2}{5}(\frac{2}{2}\log_2 \frac{2}{2} + \frac{0}{2}\log_2 \frac{0}{2})\right]$
  
  $= 0$

- Do same calculation for Temperature and Wind
- Humidity has lowest entropy, so Humidity is split attribute at Sunny node
Observations

- Temperature was not needed
  - Decision tree can be used for feature extraction (a.k.a. feature selection)
- Tree is short
  - ID3 family of algorithms have preference bias for short trees
  - Occam’s Razor: shorter trees are better
Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set

- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)
Rule Extraction from Trees

C4.5Rules
(Quinlan, 1993)

R1: IF (age>38.5) AND (years-in-job>2.5) THEN y = 0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y = 0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y = 0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y = 0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y = 0.2
Conclusion

- Decision trees good when:
  - Instances described by attribute-value pairs
  - Target function is discrete
  - Interpretability of learned hypothesis (e.g., as a rule set) is desired
  - Training data may be noisy
- Decision trees are a good first algorithm to try