Introduction

Decision trees form a simple, easily-interpretable hypothesis

- Interpretability useful in independent validation and explanation
Quick to train
Quick to evaluate new instances
Effective “off-the-shelf” learning method
Can be combined with boosting, including using “stumps”

Decision Tree for PlayTennis (Mitchell)

With Numeric Attributes

Decision Tree Representation

High-Level Learning Algorithm (IDS, C4.5, CART)

Entropy

\[ I_m = -p_m \log_2 p_m - (1 - p_m) \log_2 (1 - p_m) \]

or for \( K \) classes,

\[ I_m = - \sum_{i=1}^{K} p_{m_i} \log_2 p_{m_i} \]
Total Impurity

- Now can look for an attribute $A$, when used to partition $X_m$ by value, produces the most pure (lowest-entropy) subsets
- Weight each subset by relative size
- E.g., size-3 subsets should carry less influence than size-300 ones
- Let $N_m = |X_m| = $ number of instances reaching node $m$
- Let $N_m(i) = $ number of these instances with value $i \in \{1, \ldots, K\}$ for attribute $A$
- Let $N_m(i) = $ number of these instances with label
- Let $p_{ij} = N_m(i)/N_m$
- Then the total impurity is

$$T_m(A) = -\sum_{i=1}^{n} \sum_{j=1}^{K} p_{ij} \log_2 p_{ij}$$

Learning Algorithm

```
GenerateTree(X)
if NodeEntropy(X) < 0 /* equation 9.3 */
Create leaf labelled by majority class in X
Return
i = SplitAttribute(X)
for each branch of i
Find Xi falling in branch
GenerateTree(Xi)
```

Example Run

### Training Examples

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</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>Cool</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>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>

### Example Run

### Selecting the First Attribute

Comparing Humidity to Wind:

- $I_m^{Humidity} = (7/14)0.985 + (7/14)0.592 = 0.789$
- $I_m^{Wind} = (8/14)0.811 + (6/14)1.000 = 0.892$
- $I_m^{Outlook} = (5/14)0.971 + (4/14)0.0 + (5/14)0.971 = 0.694$
- $I_m^{Temp} = (4/14)1.000 + (6/14)0.918 + (4/14)0.811 = 0.911$

### Example Run

### Selecting the Next Attribute

Regression Trees

- A regression tree is similar to a decision tree, but with real-valued labels at the leaves
- To measure impurity at a node $m$, replace entropy with variance of labels:

$$E_m = \frac{1}{N_m} \sum_{(x',m) \in X_m} \left( x' - g_m \right)^2,$$

where $g_m$ is the mean (or median) label in $X_m$
### Attributes with Many Values

Problem:
- If attribute $A$ has many values, it might artificially minimize $T_m^j(A)$
- E.g., if $Date$ is attribute, $T_m^j(A)$ will be low because several very small subsets will be created

One approach: penalize $A$ with a measure of split information, which measures how broadly and uniformly attribute $A$ splits data:

$$S(A) = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \log_2 \left( \frac{N_{mj}}{N_m} \right) \in [0, \log_2 n]$$

### Continuous-Valued Attributes

Use threshold to map continuous to boolean, e.g. $(Temperature > 72.3) \in \{t,f\}$

<table>
<thead>
<tr>
<th>Temperature</th>
<th>40</th>
<th>48</th>
<th>60</th>
<th>72</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayTennis</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

- Can show that threshold minimizing impurity must lie between two adjacent attribute values in $\mathcal{X}$ such that label changed, so try all such values, e.g., $(48 + 60)/2 = 54$ and $(80 + 90)/2 = 85$
- Now (dynamically) replace continuous attribute with boolean attributes $Temperature_{\leq 54}$ and $Temperature_{> 85}$ and run algorithm normally
- Other options: Split into multiple intervals rather than two; use thresholded linear combinations of continuous attributes (Sec 9.6)

### Inductive Bias of Learning Algorithm

- Hypothesis space $\mathcal{H}$ is complete, in that any function can be represented
- Thus inductive bias does not come from restricting $\mathcal{H}$, but from preferring some trees over others
  - Tends to prefer shorter trees
  - Computationally intractable to find a guaranteed shortest tree, so heuristically apply greedy approach to locally minimize impurity

### Overfitting

- Consider adding noisy training example #15: Sunny, Hot, Normal, Strong, PlayTennis = No
- What effect on earlier tree?

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Humidity</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Normal</td>
<td>No</td>
</tr>
<tr>
<td>Rain</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Weak</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Expect old tree to generalize better since new one fits noisy example
Consider error of hypothesis $h$ over:
- training data (empirical error): $\text{error}_{\text{train}}(h)$
- entire distribution $D$ of data (generalization error): $\text{error}_D(h)$

Hypothesis $h \in \mathcal{H}$ overfits training data if there is an alternative hypothesis $h' \in \mathcal{H}$ such that:

$$\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')$$

and

$$\text{error}_D(h) > \text{error}_D(h')$$

To prevent trees from growing too much and overfitting the data, we can prune them:
- In spirit of Occam’s Razor, minimum description length
- In prepruning, we allow skipping a recursive call on set $\mathcal{X}_m$ and instead insert a leaf, even if $\mathcal{X}_m$ is not pure
  - Can do this when entropy (or variance) is below a threshold ($\theta$ in pseudocode)
  - Can do this when $|\mathcal{X}_m|$ is below a threshold, e.g., 5
- In postpruning, we grow the tree until it has zero error on training set and then prune it back afterwards
  - First, set aside a pruning set not used in initial training
  - Then repeat until pruning is harmful:
    - Evaluate impact on validation set of pruning each possible node (plus those below it)
    - Greedily remove the one that most improves validation set accuracy

Convert tree to equivalent set of rules
- Prune each rule independently of others by removing selected preconditions (the ones that improve accuracy the most)
- Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g. C4.5)

Converting A Tree to Rules

IF (Outlook = Sunny) \land (Humidity = High) THEN PlayTennis = No

IF (Outlook = No) THEN PlayTennis = Yes

...