Decision Trees

- Decision Tree: a hierarchical model for supervised learning whereby the local region is identified in a sequence of recursive splits.
- Internal Decision nodes: Each node $m$ implements a test function $f_m(x)$ with discrete outcomes labeling the branches. Given an input, the test is applied and one of the branches is taken depending on the outcome.
- Terminal leaves: output: class code (for classification) or numeric value (for regression).
- Each $f_m(x)$ defines a discriminant in the $d$-dimensional input space dividing it into smaller regions which are further subdivided as we take a path from the root down.
- Each terminal leaf defines a localized region in the input space where instances falling in this region have the same label.

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp</th>
<th>Hum.</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>Nml</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Nml</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Nml</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>Nml</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Nml</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Nml</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>Nml</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>
- Decision Tree represents a disjunction of conjunctions of the constraints on the attributes.
- Each path from the root to a leaf correspond to a conjunction of attribute tests.
- The tree itself is a disjunction of these conjunctions.

\[(\text{Outlook}=\text{Sunny} \land \text{Humidity}=\text{Normal}) \lor (\text{Outlook}=\text{Overcast}) \lor (\text{Outlook}=\text{Rain} \land \text{Wind}=\text{Weak})\]

Different kinds of decision Trees:

- 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)
Univariate Tree:
- One attribute at a time;
- Each decision node defines axis-parallel hyperplane.
- Each leaf defines a hyper rectangular decision surface.

Multivariate decision tree:
- All attributes each time;
- Each decision node defines a hyperplane
- Each leaf defines a polyhedral decision surface

Whence Decision Trees?

Consider: Discrete Univariate Classification Trees
- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data or missing attribute values

Examples:
- Equipment or medical diagnosis
- Credit risk analysis
- Many successful applications that outperform human experts.
Evolution of Decision Trees:
- CLS (Concept Learning System) Earl Hunt 1960's
- ID3 (Interactive Dichotemizer 3) Quinlin 70’s and 80’s
- C4.5 Quinlin 90’s

Top-Down Induction

Main loop:
1. $A \leftarrow$ the “best” decision attribute for next node
2. Assign $A$ as decision attribute for node
3. For each value of $A$, create new descendant of node
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP,
   Else iterate over new leaf nodes
Which Attribute is Best?

What is a good quantitative measure of the worth of an attribute?

Measuring Entropy

- $S$ is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in $S$
- $p_\otimes$ is the proportion of negative examples in $S$

Entropy measures the impurity of $S$

$$Entropy(S) = -p_\oplus \log p_\oplus - p_\otimes \log p_\otimes$$
Entropy

Entropy$(S) = \text{expected number of bits needed to encode class } (\oplus \text{ or } \otimes) \text{ of a randomly drawn member of } S \text{ (under the optimal, shortest-length code)}$

Why?

Information theory: optimal length code assigns $- \log_2 p$ bits to message having probability $p$.

So, expected number of bits to encode $\oplus$ or $\otimes$ of a random member of $S$:

$$p_\oplus (- \log p_\oplus) + p_\otimes (- \log p_\otimes)$$
Information Gain

Gain(S, A) = expected reduction in entropy due to sorting S on A

Gain(S, A) = Entropy(S) - \sum_v \frac{|S_v||S|}{|S|} \text{Entropy}(S_v)

Here, S_v is the set of training instances remaining from S after restricting to those for which attribute A has value v.

Which Attribute is Best?

Which attribute should be used first?
## Training Examples

<table>
<thead>
<tr>
<th>Day Outlook</th>
<th>Temp</th>
<th>Hum.</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1 Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2 Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3 Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4 Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5 Rain</td>
<td>Cool</td>
<td>Nml</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6 Rain</td>
<td>Cool</td>
<td>Nml</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7 Overcast</td>
<td>Cool</td>
<td>Nml</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8 Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9 Sunny</td>
<td>Cool</td>
<td>Nml</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10 Rain</td>
<td>Mild</td>
<td>Nml</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11 Sunny</td>
<td>Mild</td>
<td>Nml</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12 Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13 Overcast</td>
<td>Hot</td>
<td>Nml</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14 Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

## Selecting the Next Attribute

**S: [9+,5-]**

\[
E = 0.940
\]

**Humidity**

- **High**
  - \([3+,4+]\)
  - \(E = 0.985\)
- **Normal**
  - \([6+,1-]\)
  - \(E = 0.592\)

**S: [9+,5-]**

\[
E = 0.940
\]

**Wind**

- **Weak**
  - \([6+,2-]\)
  - \(E = 0.811\)
- **Strong**
  - \([3+,3+]\)
  - \(E = 1.00\)

Which attribute is the best classifier?

\[
Gain(S, \text{Humidity}) = .940 \cdot (7/14) \cdot 0.985 \cdot (7/14) \cdot 0.592 = .151
\]

\[
Gain(S, \text{Wind}) = .940 \cdot (8/14) \cdot 0.811 \cdot (6/14) \cdot 1.0 = .048
\]
Comparing Attributes

\( S_{\text{sunny}} = \{D1, D2, D8, D9, D11\} \)

- \( \text{Gain}(S_{\text{sunny}} \cdot \text{Humidity}) \)
  \[ = .970 \cdot \left( \frac{3}{5} \right) 0.0 - \left( \frac{2}{5} \right) 0.0 = .970 \]

- \( \text{Gain}(S_{\text{sunny}} \cdot \text{Temp}) \)
  \[ = .970 \cdot \left( \frac{2}{5} \right) 0.0 - \left( \frac{2}{5} \right) 1.0 - \left( \frac{1}{5} \right) 0.0 = .570 \]

- \( \text{Gain}(S_{\text{sunny}} \cdot \text{Wind}) \)
  \[ = .970 \cdot \left( \frac{2}{5} \right) 1.0 - \left( \frac{3}{5} \right) .918 = .019 \]
What is ID3 Optimizing?

The hypothesis space searched by ID3 is the set of possible decision trees. Simple-to-Complex hill-climbing (greedy) search guided by information gain measure.

How would you find a tree that minimizes:
- misclassified examples?
- expected entropy?
- expected number of tests?
- depth of tree given a fixed accuracy?
- etc.?

How decide if one tree beats another?

Hypothesis Space Search by ID3

ID3:
- representation: trees
- scoring : entropy
- search : greedy
Hypothesis Space Search by ID3

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can't generate all consistent hypotheses... Single Concept.
- No back tracking
  - Local minima... Not globally optimal.
- Statistically-based search choices
  - Robust to noisy data...
- Inductive bias "prefer shortest tree"

Inductive Bias in ID3

Note \( H \) is the power set of instances \( X \)

- Unbiased?
  Not really...
  - Preference for short trees, and for those with high information gain attributes near the root
  - Bias is a \textit{preference} for some hypotheses, rather than a \textit{restriction} of hypothesis space \( H \)
  - Occam’s razor: prefer the shortest hypothesis that fits the data
Occam's Razor

Why prefer short hypotheses?
Argument in favor:
• Fewer short hyps than long hyps.
  – A short hyp that fits data unlikely to be coincidence
  – A long hyp that fits data might be coincidence
Argument opposed:
• There are many ways to define small sets of hyps
• E.g., all trees with a prime number of nodes that use attributes beginning with “Z”
• What’s so special about small sets based on size of hypothesis??

Overfitting

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

```
Outlook
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Overcast</td>
</tr>
<tr>
<td>Humidity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Rain</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Overfitting

Consider error of hypothesis $h$ over
- training data: $\text{error}_{\text{train}}(h)$
- entire distribution $D$ of data: $\text{error}_D(h)$

Hypothesis $h$ in $H$ **overfits** training data if there is an alternative hypothesis $h'$ in $H$ such that
- $\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')$, and
- $\text{error}_D(h) > \text{error}_D(h')$

**Overfitting in Learning**

![Graph showing accuracy vs. size of tree (number of nodes)]
Overfitting in Learning

Avoiding Overfitting

How can we avoid overfitting?
• stop growing when data split not statistically significant
• grow full tree, then post-prune (DP alg!)

How to select “best” tree:
• Measure performance over training data
• Measure performance over separate validation data set
• MDL: minimize
  \[ \text{size(tree)} + \text{size(misclassifications(tree))} \]
Reduced-Error Pruning

Split data into training and validation set
Do until further pruning is harmful:
1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves validation set accuracy
   • produces smallest version of most accurate subtree
   • What if data is limited?

Effect of Pruning
Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use
   Perhaps most frequently used method (e.g., C4.5)
The Rules

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

Attributes with Many Values - C4.5

Problem:
• If one attribute has many values compared to the others, Gain will select it
• Imagine using Date = Jun_3_1996 as attribute
One approach: use GainRatio instead

\[
GainRatio(S,A) = \frac{Gain(S,A)}{SplitInfo(S,A)}
\]

\[
SplitInfo(S,A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}
\]

where \( S_i \) is subset of \( S \) for which \( A \) has value \( v_i \)
Attributes with Costs

Consider

- medical diagnosis, BloodTest has cost $150
- robotics, Width_from_1ft has cost 23 sec.

How to learn a consistent tree with low expected cost? Find min cost tree.

Another approach: replace gain by

- Tan and Schlimmer (1990)
  \[ \text{Gain}^2(S,A)/\text{Cost}(A) \]
- Nunez (1988) [w in [0,1]: importance]
  \[ (2\text{Gain}(S,A)-1)/(\text{Cost}(A)+1)^w \]

Unknown Attribute Values

Some examples missing values of \( A \)?

Use training example anyway, sort it

- If node \( n \) tests \( A \), assign most common value of \( A \) among other examples sorted to node \( n \)
- assign most common value of \( A \) among other examples with same target value
- assign probability \( p_i \) to each possible value \( v_i \) of \( A \) (perhaps as above)
  - assign fraction \( p_i \) of example to each descendant in tree
- Classify new examples in same fashion
Sources

• ML: Chapter 3
• i2ML: Chapter 9
• Slides by Ethem Alpaydin
• Slides by Tom Mitchell as provided by Michael Littman