Decision Trees

- *Decision tree learning* is a form of supervised learning for concept classification. A decision tree represents a procedure for classifying objects based on their attributes.

- Decision trees are perhaps the most popular ML/KDD algorithm.

- Internal node: an attribute to test. Leaf node: a classification.

- To classify an object: start at the root node, traverse the branches corresponding to the attribute values of the object.
A Concept Learning Task

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp</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>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>
Decision Tree Representation

- Internal nodes test the value of a particular feature, and branch according to the outcome of the test.

- Leaf nodes specify the class $h(x)$.

Example: $<\text{Outlook} = \text{sunny}, \text{Temperature} = \text{Hot}, \text{Humidity} = \text{High}, \text{Wind} = \text{Strong} > \text{Class}$.
Decision Trees Represent Disjunction of Conjunctions

- Each path from the tree root to a leaf corresponds to a conjunction of attribute tests.
- The tree itself corresponds to a disjunction of these conjunctions.
- Example – decision tree of last slide corresponds to the expression:
  \[(Outlook = Sunny \land Humidity = Normal) \lor (Outlook = Overcast) \lor (Outlook = Rain \land Wind = Weak)\]
- A decision tree can be converted to an equivalent rule set.
Appropriate Problems for Decision Tree Learning

- Instances represented by attribute-value pairs
- Target function has a discrete number of output values
- Disjunctive descriptions may be required
- Training data may contain errors
- Training data may contain missing attribute values
Designing a Decision Tree Learning Algorithm

**Goal:** Construct a decision tree that agrees (is consistent) with the training set.

**Trivial solution:** construct a decision tree that has one path to a leaf for every example.

Problem with trivial solution?

**Non-trivial solution:** find a concise decision tree that agrees with the training data.

Problem?

Solution?
ID3

- ID3 is a well-known decision tree algorithm that uses a top-down greedy search through the hypothesis space.

- ID3 was designed to handle large training sets with many attributes.

- ID3 tries to generate fairly simple trees, but is not guaranteed to produce the best one.

- Two of the most widely used decision tree algorithms, C4.5 and C5.0, are descendents of ID3.
Basic Decision Tree Algorithm

Approach: construct tree from the top down, starting with the question: which attribute should be tested at the root of the tree.

Algorithm:

If all instances from same class
    then tree is leaf with that class name
else

    *pick test for decision node*

    partition instances by test outcome

    construct one branch for each possible outcome

    build subtrees recursively
Example

CS Major Database

<table>
<thead>
<tr>
<th>Height</th>
<th>Eyes</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>short</td>
<td>brown</td>
<td>hacker</td>
</tr>
<tr>
<td>tall</td>
<td>blue</td>
<td>theorist</td>
</tr>
<tr>
<td>tall</td>
<td>brown</td>
<td>hacker</td>
</tr>
<tr>
<td>short</td>
<td>blue</td>
<td>theorist</td>
</tr>
</tbody>
</table>
Which attribute is best?

A1=?

A2=?
Characteristics of Tests

Imagine that we have examples from two classes $P$ and $N$. How do we decide which attribute to split on?

Let's first take a look at some characteristics of tests:

Let $S$ contain 20 occurrences of $P$ and 20 of $N$.

Imagine a Boolean test that splits the data into two subsets $S_1$ and $S_2$,

**best case:** $S_1 = 20P$'s and $S_2 = 20N$'s

**worst case:** $S_1 = 10P, 10N$ and $S_2 = 10P, 10N$

**intermediate case:** $S_1 = 17P, 1N$ and $S_2 = 3P, 19N$

Q: Why is the third case better than the second?
• $S$ is a sample of training examples
• $p$ is the proportion of positive examples in $S$
• $n$ is the proportion of negative examples in $S$
• Entropy measures the impurity of $S$

$$Entropy(S) \equiv -p \log_2 p - n \log_2 n$$
Some Examples

- All points of class P
  - \( \frac{p}{p+n} = 1.0 \)
  - \( \frac{n}{p+n} = 0.0 \)
  - \(-1 \log_2(1.0) - 0 \log_2(0) = 0.0 \)
  - No disorder

- Half and Half
  - \( \frac{p}{p+n} = 0.5 \)
  - \( \frac{n}{p+n} = 0.5 \)
  - \(-0.5 \log_2(0.5) - 0.5 \log_2(0.5) = 1.0 \)
  - Maximum disorder
Multiclass

Thus far we have assumed that the target class is Boolean. More generally, the class can take on \( c \) values, then the entropy of \( S \) relative to this \( c \)-wise classification is defined as:

\[
Entropy(S) = \sum_{i=1}^{c} -\frac{|S_i|}{|S|} \log_2\left(\frac{|S_i|}{|S|}\right)
\]

Where \( S_i \) is the proportion of \( S \) belonging to class \( i \).
Information Gain

Measures the expected reduction in entropy caused by partitioning the examples according to the attribute.

\[
Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)
\]
Calculation for Attribute Humidity

<table>
<thead>
<tr>
<th>branch</th>
<th>value</th>
<th>( n_{bp} )</th>
<th>( n_{bn} )</th>
<th>disorder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>high</td>
<td>3</td>
<td>4</td>
<td>.99</td>
</tr>
<tr>
<td>2</td>
<td>normal</td>
<td>6</td>
<td>1</td>
<td>.58</td>
</tr>
</tbody>
</table>

Entropy (high) = \(-\frac{3}{7} \log_2(\frac{3}{7}) - \frac{4}{7} \log_2(\frac{4}{7}) = .99\)

Entropy (normal) = \(-\frac{6}{7} \log_2(\frac{6}{7}) - \frac{1}{7} \log_2(\frac{1}{7}) = .58\)

Gain(S, Humidity) =

\[ \text{Entropy}(S) - \frac{7}{14} \text{Entropy}(\text{high}) - \frac{7}{14} \text{Entropy}(\text{normal}) = \]

\[ 0.940 - \frac{7}{14}(.99) - \frac{7}{14}(.58) = 0.151 \]
Which attribute is the best classifier?

**Humidity**

- High: [3+,4-]
  - $E = 0.985$
- Normal: [6+,1-]
  - $E = 0.592$

Gain ($S$, Humidity)

\[
\begin{align*}
&= 0.940 - \frac{7}{14} \times 0.985 - \frac{7}{14} \times 0.592 \\
&= 0.151
\end{align*}
\]

**Wind**

- Weak: [6+,2-]
  - $E = 0.811$
- Strong: [3+,3-]
  - $E = 1.00$

Gain ($S$, Wind)

\[
\begin{align*}
&= 0.940 - \frac{8}{14} \times 0.811 - \frac{6}{14} \times 1.0 \\
&= 0.048
\end{align*}
\]
Which attribute should be tested here?

\[S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}\]

\[Gain (S_{\text{sunny}}, \text{Humidity}) = .970 - (2/5) 0.0 - (2/5) 0.0 = .970\]

\[Gain (S_{\text{sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570\]

\[Gain (S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019\]
ID3’s Hypothesis Space

Slide CS478–19
ID3’s Hypothesis Space

- ID3’s hypothesis space is the set of all possible decision trees. This is a complete space of finite discrete-valued functions (relative to the given attributes).

  BOTTOM LINE: The hypothesis space is guaranteed to contain the target concept.

- Outputs a single hypothesis.

- Performs a simple-to-complex hill-climbing search: information gain metric guides the search.

- No backtracking: Can get stuck in local maxima!

- Statistically-based search heuristic: robust to noise.
ID3’s Inductive Bias

- prefers shallow trees over deeper ones.

  ID3 can be viewed a heuristic approximation of breadth-first search for decision tree generation!

- prefers to branch on attributes with highest information gain first.
Preference vs. Restriction Biases

Preference Bias: preference for certain hypotheses over others, but no restriction on the hypothesis space.

*ID3 searches a complete hypothesis space, but searches it incompletely.*

Restriction Bias: a priori restriction on the hypothesis space, but no preference for one hypothesis over another.

*The candidate elimination algorithm searches an incomplete hypothesis space, but searches it completely.*

Some learning algorithms use both preference and restriction biases, such as evaluation function learning methods.
Occam’s Razor Bias

**Occam’s Razor heuristics:** Prefer the simplest hypothesis that fits the data.

Why prefer simpler hypotheses over complex hypotheses?

*Common rationale:* There are fewer simpler hypotheses, so it is less likely to be coincidence if a simple hypothesis fits the data.

**Counter arguments:**

- But there are lots of meaningless small hypothesis sets!
- The complexity of a hypothesis depends on the representation used by the learner.