Decision Tree
One of the most popular off-the-shelf classifiers
Decision Tree for Playing Tennis

- Each internal node test on an attribute $x_i$
- Each branch from a node takes a particular value of $x_i$
- Each leaf node predicts a class label
(outlook=sunny, wind=strong, humidity=normal, ?)
DT for prediction C-section risks

Learned from medical records of 1000 women
Negative examples are C-sections

[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
|   Previous_Csection = 0: [767+,81-] .90+ .10-
|   |   Primiparous = 0: [399+,13-] .97+ .03-
|   |   Primiparous = 1: [368+,68-] .84+ .16-
|   |   |   Fetal_Distress = 0: [334+,47-] .88+ .12-
|   |   |   Birth_Weight < 3349: [201+,10.6-] .95+ .00-
|   |   |   Birth_Weight >= 3349: [133+,36.4-] .78+ .22-
|   |   |   Fetal_Distress = 1: [34+,21-] .62+ .38-
|   |   Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
Characteristics of Decision Trees

• Decision trees have many appealing properties
  – Similar to human decision process, easy to understand
  – Deal with both discrete and continuous features
  – Highly flexible hypothesis space, as the # of nodes (or depth) of the tree increase, decision tree can represent increasingly complex decision boundaries

Definition: Hypothesis space $H$

The space of solutions that a learning algorithm can possibly output. For example,

• For Perceptron: the hypothesis space is the space of all straight lines
• For nearest neighbor: the hypothesis space is infinitely complex
• For decision tree: it is a flexible space, as we increase the depth of the tree, the hypothesis space grows larger and larger
DT can represent arbitrarily complex decision boundaries

If needed, the tree can keep on growing until all examples are correctly classified! Although it may not be the best idea
How to learn decision trees?

• Possible goal: find a decision tree $h$ that achieves minimum error on training data
  – Trivially achievable – if use a large enough tree

• Another possibility: find the smallest decision tree that achieves the minimum training error
  – NP-hard
Greedy Learning For DT

We will study a top-down, greedy search approach. Instead of trying to optimize the whole tree together, we try to find one test at a time.

Basic idea: (assuming discrete features, relax later)

1. Choose the best attribute to test on at the root of the tree.
2. Create a descendant node for each possible outcome of the test.
3. Training examples in training set S are sent to the appropriate descendant node.
4. Recursively apply the algorithm at each descendant node to select the best attribute to test using its associated training examples.
   • If all examples in a node belong to the same class, turn it into a leaf node, label with the majority class.
Building DT: an example

Training data contains

13 15
One possible question: is $x < 0.5$?
This could keep on going, until all examples are correctly classified.
Choosing the best test

Which one is better?
Choosing the best test

Assuming that we will stop growing the tree after selecting this test. How many mistakes will we make?
Greedy Learning For DT

We will study a top-down, greedy search approach. Instead of trying to optimize the whole tree together, we try to find one test at a time.

Basic idea: (assuming discrete features, relax later)

1. Choose the best attribute to test on at the root of the tree.
2. Create a descendant node for each possible outcome of the test.
3. Training examples in training set S are sent to the appropriate descendant node.
4. Recursively apply the algorithm at each descendant node to select the best attribute to test using its associated training examples.
   • If all examples in a node belong to the same class, turn it into a leaf node, label with the majority class.
Choosing the Best test: A General View

$S$: current set of training examples

$m$ branches, one for each possible outcome of the test

$S_1, S_2, \ldots, S_m$: $m$ subsets of training examples

$p_i$: The portion of examples in $S$ that takes branch $i$

Benefit of split = $U(S) - \sum_{i}^{m} p_i U(S_i)$

Uncertainty of the class label in $S$

Total Expected Remaining Uncertainty after the test
Choosing the best test

This can be viewed as using the error rate as a measure of the uncertainty.
Using error rate as a measure of uncertainty does not always work well

Consider this scenario:

\[
\begin{array}{c}
\text{T} \\
21 + 3 - \\
\text{F} \\
5 + 4 - \\
\end{array}
\]

\[
\frac{24}{33} \times \frac{3}{24} + \frac{9}{33} \times \frac{4}{9} = \frac{7}{33}
\]

In this example, testing on \( t \) did not reduce the error. However, it actually is actually making some “progress” toward a good tree.
A Better Measure: Entropy

- Given a set of training examples $S$
  - Let $y$ denote the label of an example randomly drawn from $S$
  - If all examples belong to one class, $y$ has zero uncertainty
  - If $y$ takes the positive and negative values with a 50%-50% chance, we have the highest amount of uncertainty in $y$

- In information theory, **entropy** is the measure of uncertainty of a random variable

**Definition**

Let $y$ be a categorical random variable that can take $k$ different values: $v_1, v_2, ..., v_k$; and $p_i = P(y = v_i)$ for $i = 1, ..., k$.

The **entropy** of $y$, denoted $H(y)$, is defined as

$$H(y) = \sum_{i=1}^{k} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{k} p_i \log_2 p_i$$
Entropy of a Binary $y$

- Entropy is a concave function downward

Minimum uncertainty occurs when $p_0=0$ or $1$
The **Information Gain** approach:
Measuring uncertainty using entropy:

\[
H(S) = -\frac{26}{33} \log_2 \frac{26}{33} - \frac{7}{33} \log_2 \frac{7}{33} = 0.7455
\]

\[
H(S_1) = -\frac{21}{24} \log_2 \frac{21}{24} - \frac{3}{24} \log_2 \frac{3}{24} = 0.5436
\]

\[
H(S_2) = -\frac{5}{9} \log_2 \frac{5}{9} - \frac{4}{9} \log_2 \frac{4}{9} = 0.9911
\]

\[
H(S) - (p_1 H(S_1) + p_2 H(S_2)) = 0.7455 - \frac{24}{33} \cdot 0.5436 - \frac{9}{33} \cdot 0.9911 = 0.0799
\]

- This measures the **Mutual information** between \( t \) and \( y \) \( I(t, y) \)
- Hence the name information gain
Choosing the Best Feature: Summary

Measures of Uncertainty

<table>
<thead>
<tr>
<th>Error</th>
<th>( \min(p_+, p_-) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy</td>
<td>(-p_+ \log_2 p_+ - p_- \log_2 p_-)</td>
</tr>
<tr>
<td>Gini Index</td>
<td>( p_+ p_- )</td>
</tr>
<tr>
<td>Day</td>
<td>Outlook</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>D1</td>
<td>Sunny</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
</tr>
</tbody>
</table>
Selecting the root test using information gain

\[
S \quad \begin{array}{c|c}
H(S) & 0.940 \\
\hline
9 & 5 \\
\hline
+ & - \\
\hline
Humidity
\end{array}
\]

\[
H(S_1) = 0.985 \quad H(S_2) = 0.592
\]

\[
Gain(humidity) = 0.940 - \frac{1}{2}0.985 - \frac{1}{2}0.592 = 0.151
\]

\[
S \quad \begin{array}{c|c}
H(S) & 0.940 \\
\hline
9 & 5 \\
\hline
+ & - \\
\hline
Outlook
\end{array}
\]

\[
\begin{array}{c|c|c|c}
 & sunny & Overcast & Rain \\
\hline
2 & 3 & 4 & 0 \\
\hline
+ & - & + & - \\
\hline
H(S_1) = 0.971 & H(S_2) = 0 & H(S_3) = 0.971
\end{array}
\]

\[
Gain(Outlook) = 0.940 - \frac{5}{14}0.971 - \frac{5}{14}0.971 = 0.2464
\]
Continue building the tree

\[
\begin{align*}
\{D_1, D_2, \ldots, D_{14}\} & \\
S & \\
9 & 5 \\
+ & - \\
\text{Outlook} & \\
\text{sunny} & \\
\text{Overcast} & \\
\{D_1, D_2, D_3, D_9, D_{11}\} & \\
2 & 3 \\
+ & - \\
? & \\
\{D_3, D_7, D_{12}, D_{13}\} & \\
3 & 2 \\
+ & - \\
\{D_4, D_5, D_6, D_{10}, D_{14}\} & \\
\text{Humidity} & \\
\text{High} & \\
0 & 3 \\
+ & - \\
\text{Normal} & \\
2 & 0 \\
+ & - \\
\end{align*}
\]

Which test should be placed here?
Issues with Multi-nomial Features

• Multi-nomial features: more than 2 possible values
• Consider two features, one is binary, the other has 100 possible values, which one you expect to have higher information gain?
• Conditional entropy of Y given the 100-valued feature will be low – why?
• This bias will prefer multinomial features to binary features

Method 1: To avoid this, we can rescale the information gain:

\[
\arg \max_j \left( \frac{H(y) - H(y | x_j)}{H(x_j)} \right)
\]

Information gain of \(x_j\)

Method 2: Test for one value versus all of the others
Method 3: Group the values into two disjoint sets and test one set against the other
Dealing with Continuous Features

• Test against a threshold

• How to compute the best threshold $\theta_j$ for $x_j$?
  – Sort the examples according to $x_j$.
  – Move the threshold $\theta$ from the smallest to the largest value
  – Select $\theta$ that gives the best information gain
  – Trick: only need to compute information gain when class label changes

• Note that continuous features can be tested for multiple times on the same path in a DT
Considering both discrete and continuous features

• If a data set contains both types of features, do we need special handling?
• No, we simply consider all possibly splits in every step of the decision tree building process, and choose the one that gives the highest information gain
  – This include all possible (meaningful) thresholds
Issue of Over-fitting

- Decision tree has a very flexible hypothesis space
- As the nodes increase, we can represent arbitrarily complex decision boundaries
- This can lead to over-fitting

Possibly just noise, but the tree is grown larger to capture these examples
Over-fitting
Avoid Overfitting

• Early stop
  – Stop growing the tree when data split does not offer large benefit (e.g., compare information gain to a threshold, or perform statistical testing to decide if the gain is significant)

• Post pruning
  – Separate training data into training set and validating set
  – Evaluate impact on validation set when pruning each possible node
  – Greedily prune the node that most improves the validation set performance
Effect of Pruning
Summary

• Decision tree is a very flexible classifier
  – Can model arbitrarily complex decision boundaries
  – By changing the depth of the tree (or # of nodes in the tree), we can increase or decrease the model complexity
  – Handling both continuous and discrete features

• Learning of the decision tree
  – Greedy top-down induction
  – Not guaranteed to find an optimal decision tree

• DT can overfitting to noise and outliers
  – Can be controlled by early stopping or post pruning