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

- **Outlook**
  - Sunny
  - Overcast
    - Yes
  - Rain
- **Humidity**
  - High
    - No
  - Normal
    - Yes
- **Wind**
  - Strong
    - No
  - Weak
    - Yes
Definition

• **Internal nodes**
  – Each tests an attribute
  – Branch according to attribute values
  – Discrete attributes – branching is naturally defined
  – Continuous attributes – branching by comparing to a threshold

• **Leaf nodes**
  – Each assign a class label
(outlook=sunny, wind=strong, humidity=normal, ? )
Decision Tree Decision Boundaries

- For continuous attributes, a decision tree divides the input space into *axis-parallel rectangles* and label each rectangle with one of the K classes.
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.
So far we have looked at what is a decision tree, and what kind of decision boundaries decision trees produce, and its appealing properties. We now need to address:

**How to learn decision trees**

- **Goal:** Find a decision tree $h$ that achieves _minimum misclassification errors on the training data_
- As our previous slides suggest, we can always achieve this by using large trees
- In fact, we can achieve this trivially: just create a decision tree with one path from root to leaf for each training example
  - Problem: Such a tree would just memorize the training data. It would not _generalize to new data points_ — i.e., _capture regularities that are applicable to unseen data_
- Alternatively: find the _smallest_ tree $h$ that minimizes training error
  - Problem: This is NP-Hard
There are different ways to construct trees from data. We will focus on the 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 $a^*$ to place at the root of the tree.
2. Separate training set $S$ into subsets $\{S_1, S_2, \ldots, S_k\}$ where each subset $S_i$ contains examples having the same value for $a^*$
3. Recursively apply the algorithm on each new subset until all examples have the same class or there are few of them.
4. Label the leaf nodes with the dominating class among all examples in that node
Building DT: an example

Training data contains
13 15

If we had to make a decision now, we’d pick 15. But there’s too much uncertainty.

Based on training data, with probability 13/28 I would be wrong.

Now if you are allowed to ask one question about your example to help the decision, which question will you ask?
One possible question: is $x < 0.5$?
Continue
Building a decision tree

1. Choosing the best attribute $a^*$ to place at the root of the tree.
   - What do we mean by “best” – reduce the most uncertainty about our prediction of the class labels
2. Separate the training set $S$ into subsets $\{S_1, S_2, .., S_k\}$ where each subset $S_i$ contains examples having the same value for $a^*$
3. Recursively apply the algorithm on each new subset until all examples have the same class label
4. Label each leaf node by its majority class
Choosing split: example

If we were to stop here, what labels to give to the leaf nodes?
Left branch: $y=1$, right branch: $y=0$
How many mistakes we will be making? 2

We can use the number of training errors as a way of measuring the uncertainty.
Continue

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Training examples
Choosing the Best Feature: A General View

U(S): uncertainty of y before knowing the outcome of test

Benefit of split = U(S) – [P_L * U(S_L) + P_R * U(S_R)]

Expected Remaining Uncertainty after knowing the outcome
Using training error as a measure of uncertainty does not always work well.

In this example, testing on \( x_1 \) 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 \( y \) be a random variable representing the label of a randomly drawn example from \( S \)
  – If all examples belong to one class, \( y \) has zero uncertainty
  – If the examples are distributed 50%-50%, we have the highest amount of uncertainty

• In information theory, \textit{entropy} is the measure of uncertainty of a random variable

\textbf{Definition}

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

The \textit{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

- Entropy is a concave function downward

Minimum uncertainty occurs when \( p_0 = 0 \) or \( 1 \)
Measuring uncertainty using entropy: the **Information Gain** approach

\[
H(y) - p(x_1 = 0)H(y | x_1 = 0) + p(x_1 = 1)H(y | x_1 = 1)
\]

\[
= H(y) - H(y | x_1) = I(x_1, y)
\]

- \(H(y | x_1)\): **conditional entropy** of \(y\) given \(x_1\) --- the remaining uncertainty of \(y\) after knowing the value of \(x_1\)
- \(I(x_1, y)\): **Mutual information** between \(x_1\) and \(y\) - the amount of info. \(x_1\) has about \(y\)

* Note that because \(H(y)\) is constant across all possible tests, so we can simply compute \(H(y|x_i)\) for each \(x_i\) and choose \(a^*\) that minimizes \(H(y|x_i)\)
\[
P(X_1=0) = 0.6677 \quad \text{P}(X_1=1) = 0.3333
\]

\[
H(Y|X_1=0) = -0.6 \log_2 0.6 - 0.4 \log_2 0.4 = 0.9710
\]

\[
H(Y|X_1=1) = -0.8 \log_2 0.8 - 0.2 \log_2 0.2 = 0.7219
\]

\[
H(Y|X_1) = 0.6667 \times 0.9710 + 0.3333 \times 0.7219 = 0.8873
\]

\[
I(X_1, Y) = 0.0304
\]
Choosing the Best Feature: Summary

Benefit of split = $U(S) - [P_L \cdot U(S_L) + P_R \cdot U(S_R)]$

Expected Remaining Uncertainty

Measures of Uncertainty

<table>
<thead>
<tr>
<th></th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>$\min{p, 1 - p}$</td>
</tr>
<tr>
<td>Entropy</td>
<td>$-p \log p - (1 - p) \log 1 - p$</td>
</tr>
<tr>
<td>Gini Index</td>
<td>$2p(1 - p)$</td>
</tr>
</tbody>
</table>
Issues with Multi-nominal Features

- Multi-nominal features: more than 2 possible values
- Comparing two features, one is binary, the other has 100 possible values, which one you expect to have higher information gain?
  - The conditional entropy of Y given this feature will be low
  - But is this meaningful?
  - This bias will inherently prefer such multinomial features to binary features
  - Method 1: To avoid this, we can rescale the information gain:
    \[
    \arg\max_j \frac{H(y) - H(y \mid x_j)}{H(x_j)}
    \]
  - 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
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
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
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)

• 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