These slides are adopted from Polo, Andrew w. Moore, and Vivek Srikumar
Visual Introduction to Decision Tree

Building a tree to distinguish homes in New York from homes in San Francisco
### Decision Tree: Example (2)

#### Will I play tennis today?

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
</tbody>
</table>

**Outlook:** Sunny, Overcast, Rainy

**Temperature:** Hot, Medium, Cool

**Humidity:** High, Normal, Low

**Wind:** Strong, Weak
Decision trees (DT)

The classifier:

\( f_T(x) \): majority class in the leaf in the tree \( T \) containing \( x \)

Model parameters: The tree structure and size
Decision trees

Pieces:

1. Find the best attribute to split on
2. Find the best split on the chosen attribute
3. Decide on when to stop splitting
Categorical or Discrete attributes

- Three variables:
  - Hair = \{blond, dark\}
  - Height = \{tall, short\}
- Label:
  - Country = \{Gromland, Polvia\}

Training data:
- (B,T,P)
- (B,T,P)
- (B,S,G)
- (D,S,G)
- (D,T,G)
- (B,S,G)
At each level of the tree, we split the data according to the value of one of the attributes.

After enough splits, only one class is represented in the node. This is a terminal leaf of the tree. We call that class the output class for that node.

'G' is the output for this node.
General Decision Tree (Discrete Attributes)

\[ X_1 = \text{first possible value for } X_1? \]

\[ X_1 = \text{nth possible value for } X_1? \]

Output class \( Y = y_1 \)

\[ X_j = \text{ith possible value for } X_j? \]

Output class \( Y = y_c \)
Continuous attributes

Decision Tree Example

$X_2 = 0.5$

$X_1 = 0.5$

$X_2 < 0.5$??

$X_1 < 0.5$??

$3 : 4$

$7 : 4$

$4 : 0$

$3 : 0$

$0 : 4$
The class of a new input can be classified by following the tree all the way down to a leaf and by reporting the output of the leaf. For example: 
(0.2,0.8) is classified as \(\times\) 
(0.8,0.2) is classified as \(\circ\)
General Decision Tree (Continuous Attributes)

\[ X_1 < t_1? \]

Output class \( Y = y_1 \)

\[ X_j < t_j? \]

Output class \( Y = y_c \)
Basic Questions

• How to choose the attribute/value to split on at each level of the tree?
• When to stop splitting? When should a node be declared a leaf?
• If a leaf node is impure, how should the class label be assigned?
• If the tree is too large, how can it be pruned?
How to choose the attribute/value to split on at each level of the tree?

- Two classes (red circles/green crosses)
- Two attributes: $X_1$ and $X_2$
- 11 points in training data
- Idea → Construct a decision tree such that the leaf nodes predict correctly the class for all the training examples
How to choose the attribute/value to split on at each level of the tree?
We want to find the most compact, smallest size tree (Occam’s razor), that classifies the training data correctly → We want to find the split choices that will get us the fastest to pure nodes.

This node is “pure” because there is only one class left → No ambiguity in the class label.

This node is almost “pure” → Little ambiguity in the class label.

These nodes contain a mixture of classes → Do not disambiguate between the classes.
Information Content

Coin flip

Which coin will give us the purest information?

Lower uncertainty, higher information gain

Entropy ~ Uncertainty

\[ H(X) = - \sum_{i=1}^{N} P(x = i) \log_2 P(x = i) \]

Entropy = \(-0 \log 0 - 1 \log 1 = -0 - 0 = 0\)

Entropy = \(-\frac{1}{6} \log_2 \frac{1}{6} - \frac{5}{6} \log_2 \frac{1}{6} = 0.65\)

Entropy = \(-\frac{2}{6} \log_2 \frac{2}{6} - \frac{4}{6} \log_2 \frac{4}{6} = 0.92\)
Entropy

• In general, the average number of bits necessary to encode $n$ values is the entropy:

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i$$

• $P_i =$ probability of occurrence of value $i$
  - High entropy $\Rightarrow$ All the classes are (nearly) equally likely
  - Low entropy $\Rightarrow$ A few classes are likely; most of the classes are rarely observed
The entropy captures the degree of “purity” of the distribution.
Example Entropy Calculation

\[ N_A = 1 \]
\[ N_B = 6 \]

\[ p_A = \frac{N_A}{(N_A+N_B)} = \frac{1}{7} \]
\[ p_B = \frac{N_B}{(N_A+N_B)} = \frac{6}{7} \]

\[ H_1 = -p_A \log_2 p_A - p_B \log_2 p_B = 0.59 \]

\[ N_A = 3 \]
\[ N_B = 2 \]

\[ p_A = \frac{N_A}{(N_A+N_B)} = \frac{3}{5} \]
\[ p_B = \frac{N_B}{(N_A+N_B)} = \frac{2}{5} \]

\[ H_2 = -p_A \log_2 p_A - p_B \log_2 p_B = 0.97 \]

\[ H_1 < H_2 \Rightarrow \text{(2) less pure than (1)} \]
Conditional Entropy

Entropy before splitting: $H$

After splitting, a fraction $P_L$ of the data goes to the left node, which has entropy $H_L$

After splitting, a fraction $P_R$ of the data goes to the right node, which has entropy $H_R$

The average entropy after splitting is:

$$H_L \times P_L + H_R \times P_R$$
Information Gain

We want nodes as pure as possible
→ We want to reduce the entropy as much as possible
→ We want to maximize the difference between the entropy of the parent node and the expected entropy of the children

Information Gain (IG) = Amount by which the ambiguity is decreased by splitting the node

Maximize:

\[ IG = H - (H_L \times P_L + H_R \times P_R) \]
Notations

• Entropy: $H(Y) =$ Entropy of the distribution of classes at a node

• Conditional Entropy:
  – *Discrete*: $H(Y|X_j) =$ Entropy after splitting with respect to variable $j$
  – *Continuous*: $H(Y|X_j,t) =$ Entropy after splitting with respect to variable $j$ with threshold $t$

• Information gain:
  – *Discrete*: $IG(Y|X_j) = H(Y) - H(Y|X_j) =$ Entropy after splitting with respect to variable $j$
  – *Continuous*: $IG(Y|X_j,t) = H(Y) - H(Y|X_j,t) =$ Entropy after splitting with respect to variable $j$ with threshold $t$
\[ H = 0.99 \]

\[ IG = H - (H_L \times 4/11 + H_R \times 7/11) \]

\[ H_L = 0 \quad H_R = 0.58 \]

\[ H = 0.99 \]

\[ IG = H - (H_L \times 5/11 + H_R \times 6/11) \]

\[ H_L = 0.97 \quad H_R = 0.92 \]
Choose this split because the information gain is greater than with the other split.
More Complete Example

- ● = 20 training examples from class A
- ✗ = 20 training examples from class B
- Attributes = $X_1$ and $X_2$ coordinates
Best split value (max Information Gain) for $X_1$ attribute: 0.24 with $IG = 0.138$
Best split value (max Information Gain) for $X_2$ attribute: 0.234 with $\text{IG} = 0.202$
Best $X_1$ split: 0.24, IG = 0.138
Best $X_2$ split: 0.234, IG = 0.202

Split on $X_2$ with 0.234

Total data points = 7
7 A
0 B

Total data points = 33
13 A
20 B
There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output ‘A’

This node is not pure so we need to split further

Total data points = 7
7 A
0 B

Total data points = 33
13 A
20 B
Best split value (max Information Gain) for $X_1$ attribute: 0.22 with $IG \sim 0.182$
Best split value (max Information Gain) for $X_2$ attribute: 0.75 with IG $\sim$ 0.353
Best $X_1$ split: 0.22, IG = 0.182
Best $X_2$ split: 0.75, IG = 0.353

Split on $X_2$ with 0.75
There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output ‘A’.
Each of the leaf nodes is pure \( \rightarrow \) contains data from only one class.
Final decision tree
Given an input \((X,Y)\) →
Follow the tree down to a leaf.
Return corresponding output class for this leaf

Example \((X,Y) = (0.5,0.5)\)
Basic Questions

• How to choose the attribute/value to split on at each level of the tree?
• When to stop splitting? When should a node be declared a leaf?
• If a leaf node is impure, how should the class label be assigned?
• If the tree is too large, how can it be pruned?
Pure and Impure Leaves and When to Stop Splitting

All the data in the node comes from a single class → We declare the node to be a leaf and stop splitting. This leaf will output the class of the data it contains.

Several data points have exactly the same attributes even though they are from different class → We cannot split any further → We still declare the node to be a leaf, but it will output the class that is the majority of the classes in the node (in this example, ‘B’).
Decision Tree Algorithm (Continuous Attributes)

- LearnTree\((X, Y)\)
  - Input:
    - Set \(X\) of \(R\) training vectors, each containing the values \((x_1, \ldots, x_M)\) of \(M\) attributes \((X_1, \ldots, X_M)\)
    - A vector \(Y\) of \(R\) elements, where \(y_j\) = class of the \(j\)th datapoint
  - If all the datapoints in \(X\) have the same class value \(y\)
    - Return a leaf node that predicts \(y\) as output
  - If all the datapoints in \(X\) have the same attribute value \((x_1, \ldots, x_M)\)
    - Return a leaf node that predicts the majority of the class values in \(Y\) as output
  - Try all the possible attributes \(X_j\) and threshold \(t\) and choose the one, \(j^*\), for which \(IG(Y|X_j,t)\) is maximum
  - \(X_L, Y_L\) = set of datapoints for which \(x_{j^*} < t\) and corresponding classes
  - \(X_H, Y_H\) = set of datapoints for which \(x_{j^*} \geq t\) and corresponding classes
  - Left Child \(\leftarrow\) LearnTree\((X_L, Y_L)\)
  - Right Child \(\leftarrow\) LearnTree\((X_H, Y_H)\)
Decision Tree Algorithm (Discrete Attributes)

- **LearnTree**(X, Y)
  - Input:
    - Set X of R training vectors, each containing the values \((x_1, \ldots, x_M)\) of M attributes \((X_1, \ldots, X_M)\)
    - A vector Y of R elements, where \(y_j = \) class of the \(j^{th}\) datapoint
  - If all the datapoints in X have the same class value \(y\)
    - Return a leaf node that predicts \(y\) as output
  - If all the datapoints in X have the same attribute value \((x_1, \ldots, x_M)\)
    - Return a leaf node that predicts the majority of the class values in Y as output
  - Try all the possible attributes \(X_j\) and choose the one, \(j^*\), for which \(IG(Y|X_j)\) is maximum
  - For every possible value \(v\) of \(X_j^*\):
    - \(X_v, Y_v = \) set of datapoints for which \(x_j = v\) and corresponding classes
    - \(Child_v \leftarrow \) LearnTree\((X_v, Y_v)\)
Decision Trees So Far

- Given \( R \) observations from training data, each with \( M \) attributes \( X \) and a class attribute \( Y \), construct a sequence of tests (decision tree) to predict the class attribute \( Y \) from the attributes \( X \).
- Basic strategy for defining the tests ("when to split") \( \rightarrow \) maximize the information gain on the training data set at each node of the tree.
- Problems (next):
  - Computational issues \( \rightarrow \) How expensive is it to compute the IG
  - The tree will end up being much too big \( \rightarrow \) pruning
  - Evaluating the tree on training data is dangerous \( \rightarrow \) overfitting
Basic Questions

• How to choose the attribute/value to split on at each level of the tree?
• When to stop splitting? When should a node be declared a leaf?
• If a leaf node is impure, how should the class label be assigned?
• If the tree is too large, how can it be pruned?
What will happen if a tree is too large?

- Overfitting
- High variance
- Instability in predicting test data
How to avoid overfitting?

• Acquire more training data

• Remove irrelevant attributes (manual process – not always possible)

• Grow full tree, then post-prune

• Ensemble learning
Reduced-Error Pruning

Split data into training and validation sets

Grow tree based on training 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 node that most improves validation set accuracy
How to decide to remove it a node using pruning

- Pruning of the decision tree is done by replacing a whole subtree by a leaf node.
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf.

If we had simply predicted the majority class (negative), we make 2 errors instead of 4.