Decision Trees

Razvan C. Bunescu

School of Electrical Engineering and Computer Science

bunescu@ohio.edu
Decision Tree Learning

- Target output is discrete (i.e. binary, or multiple classes).
  - PlayTennis $\in \{Yes, No\}$.

- Features have finite cardinality (i.e. nominal features).
  - Outlook $\in \{Sunny, Overcast, Rain\}$.
  - Temperature $\in \{Cool, Mild, Hot\}$.
  - Humidity $\in \{Normal, High\}$.
  - Wind $\in \{Weak, Strong\}$.

- Target model requires disjunctive description in terms of features $\Rightarrow$ use **Decision Trees**.
Decision Trees

Decision Tree \iff Disjunction of conjunctions of constraints on the attribute values of instances.

(Outlook = Sunny \land Humidity = Normal) 
\lor (Outlook = Overcast) 
\lor (Outlook = Rain \land Wind = Weak)
## Decision Tree Learning

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</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>Cold</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cold</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cold</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>Cold</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 Learning

- There may be many decision trees consistent with a set of training examples:
  - Q: Which decision tree should be selected?
  - A: Prefer shorter trees over larger trees.
  - ⇒ the **ID3 Algorithm** for learning decision trees.

**Occam’s razor:**

Prefer the simplest hypothesis that fits the data.
The ID3 Algorithm

• At each node:
  – Select the feature that results in the largest *expected reduction in entropy* for the target label.
  ⇔ select the feature with largest *information gain*.

• $D =$ the training data
• $T =$ the random variable corresponding to PlayTennis.

\[
p(T = \text{yes}) = \frac{9}{14}, \quad p(T = \text{no}) = \frac{5}{14}
\]

\[
\Rightarrow H(T; D) = - \sum_i p(x_i) \log p(x_i)
\]

\[
= - \left\{ \frac{9}{14} \log \frac{9}{14} + \frac{5}{14} \log \frac{5}{14} \right\} \approx 0.940
\]
The ID3 Algorithm

• Suppose we split on feature $X$ that has $k$ values \{x_1, \ldots, x_k\}
• Let $D_i$ be the set of instances where $X = x_i$.
• The expected reduction in entropy is:

$$IG(X; D) = H(T; D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} H(T; D_i)$$

• Choose the feature that maximizes the information gain:

$$\hat{X} = \arg \max_X IG(X; D)$$
The ID3 Algorithm

\[ D = \{9+,5-\} \]
\[ H(T;D) = 0.940 \]

\[ D_1 = \{6+,2-\} \]
\[ H(T;D_1) = 0.811 \]

\[ D_2 = \{3+,3-\} \]
\[ H(T;D_2) = 1.00 \]

\[ IG(\text{Wind};D) = 0.940 - (8/14)*0.811 - (6/14)*1.00 = 0.048 \]
The ID3 Algorithm

\[ IG(Wind; D) = 0.048 \]
\[ IG(Humidity; D) = 0.151 \]
\[ IG(Temperature; D) = 0.029 \]
\[ IG(Outlook; D) = 0.246 \]

\[ \Rightarrow \text{select } X = \text{Outlook to split at the root.} \]
The ID3 Algorithm

- Repeatedly apply the Information Gain criterion to select the best attribute for each nonterminal node.
  - set $D$ to the training examples for that node.
  - use only remaining attributes.

$D = \{9+,5-\}$

Outlook

- Sunny
  - $\{2+,3-\}$
  - ?
- Rain
  - $\{3+,2-\}$
  - ?
- Overcast
  - $\{4+,0-\}$
  - Yes
The ID3 Algorithm

Algorithm ID3(Training data $D$, Features $F$):

if all examples in $D$ have the same label:
    return a leaf node with that label

let $X \in F$ be the feature with the largest information gain
let $T$ be a tree root labeled with feature $X$.
let $D_1, D_2, \ldots, D_k$ be the partition produced by splitting $D$ on feature $X$
for each $D_i \in \{D_1, D_2, \ldots, D_k\}$:
    let $T_i = \text{ID3}(D_i, F-\{X\})$
    add $T_i$ as a new branch of $T$

return $T$
Overfitting

- ID3 learns a tree that classifies the training data perfectly.
- The learned tree may not lead to the tree with the best generalization performance on unseen (test) data.

learning which patients have a form of diabetes.

Overfitting

1. Maximum number of leaf nodes is $N = \# \text{ training examples}$  
   $\Rightarrow$ no generalization outside of the training data.

2. When small number of examples are associated with leaf nodes:
   - some attribute that is unrelated to the actual target function happens to partition the examples very well.

3. When training examples contain random noise:
   - consider adding (false) negative example:

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D15</td>
<td>Sunny</td>
<td>Hot</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Overfitting

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D15</td>
<td>Sunny</td>
<td>Hot</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

Outlook

- Sunny
- Overcast
- Rain

Humidity

- High
- Normal

Wind

- Strong
- Weak

\{+D_9,+D_{11},-D_{15}\}
Methods for Reducing Overfitting

- Two types of methods:
  1) Stop growing the tree before it perfectly classifies the training data.
  2) Allow the tree to overfit the data, then prune the tree.
  3) Use ensembles of trees: bagged trees & random forests.

1. Criteria for determining the right size of the tree:
   - Use a validation set to evaluate utility of pruning nodes.
   - Use a statistical test ($\chi^2$) to determine whether expanding a particular node is likely to produce improvement over entire instance distribution.
   - Minimal Description Length (MDL): determine if additional nodes leads to less complex hypothesis than just remembering any exceptions that result from pruning.
2. Reduced Error Pruning

1. grow a complete tree from the training data
2. while accuracy on validation set is non-decreasing:
3. for each internal node in the tree:
4. temporarily prune the subtree and replace it with a leaf labeled with the majority class
5. measure the accuracy of the pruned tree on validation set
6. permanently prune the node that results in greatest accuracy.

⇒ leaf nodes created due to chance regularities in training data are likely to be pruned.

- Drawback: “wastes” validation dataset, instead of using it for training.
2. Reduced Error Pruning

[Quinlan, 1987]
3. Bagged Decision Trees

- Training set has $N$ examples, each example has $K$ features.

1. for $t = 1$ to $T$:
2. draw $n < N$ samples with replacement.
3. train a decision tree $D_t$ on the $n$ samples.
4. construct final classifier by majority voting over trees $D_t$.
   - for regression, average predictions of trees $D_t$. 
3. Random Forests

- Training set has $N$ examples, each example has $K$ features.

1. for $t = 1$ to $T$:
2. draw $n < N$ samples with replacement.
3. sample $k < K$ features at random.
4. train a decision tree $D_t$ on the $n$ samples and $k$ features.
5. construct final classifier by majority voting over trees $D_t$.
   - for regression, average predictions of trees $D_t$. 