CS 349: Machine Learning
Bryan Pardo & Bongjun Kim

Topic: Decision Trees

(Includes content provided by: Tom Mitchell, Russel & Norvig, D. Downie, P. Domingos)
General Learning Task

There is a set of possible examples $X = \{\vec{x}_1, \ldots, \vec{x}_n\}$

Each example is an n-tuple of attribute values

$\vec{x}_1 = \langle a_1, \ldots, a_k \rangle$

There is a target function that maps $X$ onto some finite set $Y$

$f : X \rightarrow Y$

The DATA is a set of duples <example, target function values>

$D = \{\langle \vec{x}_1, f(\vec{x}_1) \rangle, \ldots, \langle \vec{x}_m, f(\vec{x}_m) \rangle\}$

Find a hypothesis $h$ such that...

$\forall \vec{x}, h(\vec{x}) \approx f(\vec{x})$
Attribute-based representations

Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won’t wait for a table:

<table>
<thead>
<tr>
<th>Example</th>
<th>Attr</th>
<th>Attributes</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alt</td>
<td>Bar</td>
<td>Fri</td>
</tr>
<tr>
<td>X₁</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>X₂</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>X₃</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X₄</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>X₅</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>X₆</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X₇</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X₈</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>X₉</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>X₁₀</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>X₁₁</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>X₁₂</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Classification of examples is positive (T) or negative (F)
One possible representation for hypotheses
E.g., here is the “true” tree for deciding whether to wait:

```
Decision Tree

Patrons?
  None    Some    Full
     F      T      WaitEstimate?
        >60  30-60  10-30
            F    T    Hungry?
                No  Yes  0-10 T

Alternate?
  F      T
     No  Yes

Reservation?  Fri/Sat?
  T      T
     No  No  Yes

Bar?
  T      F      T
     No  Yes

Raining?
  F      T
     No  Yes
```
Expressiveness of D-Trees

Decision trees can express any function of the input attributes. E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A xor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

Trivially, there is a consistent decision tree for any training set w/ one path to leaf for each example (unless $f$ nondeterministic in $x$) but it probably won’t generalize to new examples

Prefer to find more compact decision trees
Decision Trees represent *disjunctions of conjunctions*

\[
f(x) = yes \text{ iff...} \\
(\text{Outlook} = \text{Sunny} \land \text{Humidity} = \text{Normal}) \lor \\
(\text{Outlook} = \text{overcast}) \lor \\
(\text{Outlook} = \text{rain} \land \text{Wind} = \text{weak})
\]
Decision Tree Boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the $K$ classes.
A learned decision tree

Decision tree learned from the 12 examples:

Substantially simpler than “true” tree—a more complex hypothesis isn’t justified by small amount of data
Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”

Patrons? is a better choice—gives information about the classification

The more skewed the examples in a bin, the better.

We’re going to use ENTROPY to as a measure of how skewed each bin is.
Counts as probabilities

$P_1 = \text{probability I will wait for a table}$

$P_2 = \text{probability I will NOT wait for a table}$

$P_1 = 0.5$

$P_2 = 0.5$

$P_1 = 0.333$

$P_2 = 0.667$
Information

Information answers questions

The more clueless I am about the answer initially, the more information contained in the answer

Scale: 1 bit = answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$

Information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is

$$H(\langle P_1, \ldots, P_n \rangle) = \sum_{i=1}^{n} - P_i \log_2 P_i$$

(also called entropy of the prior)
About ID3

• A recursive, greedy algorithm to build a decision tree

• At each step it picks the best variable to split the data on, and then moves on

• It is “greedy” because it makes the optimal choice at the current step, without considering anything beyond the current step.

• This can lead to trouble, if one needs to consider things beyond a single variable (e.g. multiple variables) when making a choice. (Try it on XOR)
Decision Tree Learning (ID3)

Aim: find a small tree consistent with the training examples

Idea: (recursively) choose “most significant” attribute as root of (sub)tree

```plaintext
function DTL(examples, attributes, default) returns a decision tree

    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value \( v_i \) of best do
            examples_i ← \{ elements of examples with best = v_i \}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label \( v_i \) and subtree subtree
        return tree
```
Choosing an attribute in ID3

- For each attribute, find the entropy $H$ of the example set AFTER splitting on that example.
  *note, this means taking the entropy of each subset created by splitting on the attribute, and then combining these entropies...weighted by the size of each subset.

- Pick the attribute that creates the lowest overall entropy.
Entropy prior to splitting

$P_1 =$ probability I will wait for a table

$P_2 =$ probability I will NOT wait for a table

$$H_0 \langle P_1, P_2 \rangle = \sum_j -P_j \log_2 P_j$$

$$= -P_1 \log_2 P_1 - P_2 \log_2 P_2$$

$$= 1$$
If we split on Patrons

\[
H_{\text{Patrons}} = W_{\text{none}} H_{\text{none}} + W_{\text{some}} H_{\text{some}} + W_{\text{full}} H_{\text{full}}
\]

\[
= \frac{2}{12} \cdot 0 + \frac{4}{12} \cdot 0 + \frac{6}{12} \left( -\frac{2}{6} \log_2 \frac{2}{6} - \frac{4}{6} \log_2 \frac{4}{6} \right) = 0.459
\]
If we split on Type

\[ H_{\text{Type}} = W_{\text{french}} H_{\text{french}} + W_{\text{italian}} H_{\text{italian}} + W_{\text{thai}} H_{\text{thai}} + W_{\text{burger}} H_{\text{burger}} \]

\[ = \frac{2}{12} \cdot 1 + \frac{2}{12} \cdot 1 + \frac{4}{12} \cdot 1 + \frac{4}{12} \cdot 1 = 1 \]
Decision Tree Learning (ID3)

Aim: find a small tree consistent with the training examples

Idea: (recursively) choose “most significant” attribute as root of (sub)tree

function DTL(examples, attributes, default) returns a decision tree

    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value \( v_i \) of best do
            examples_{i} ← \{elements of examples with best = v_i\}
            subtree ← DTL(examples_{i}, attributes – best, MODE(examples))
            add a branch to tree with label \( v_i \) and subtree subtree
        return tree
A fully worked-out example

- Let’s build a tree with ID3 to decide whether, on a particular day, I will play tennis.
Someone followed me around for two weeks to see if I played tennis each day and collected the following data to train a model.

<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>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>
Information Gain

- The expected reduction in Entropy
- Entropy (before split) – Entropy (after split)

\[ G_{\text{gain}}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \]

- \( S \) = the set of examples
- \(|S|\) = the number of examples in the set \( S \)
- \( A \) = the attribute you’re splitting the data on
- \( v \) = one of the values attribute \( A \) can take
- \( S_v \) = the subset of \( S \) where the examples take value \( v \) on attribute \( A \)
How to choose the best attribute

- Pick the one with the most information gain
  - The expected reduction in Entropy
  - Entropy (before split) – Entropy (after split)

\[
Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v)
\]
Entropy before & after

\( P_1 = \text{probability I will play tennis} \)
\( P_2 = \text{probability I will NOT play tennis} \)

\[
\text{Entropy}(S) = \sum_j -P_j \log_2 P_j
\]

\[
= -P_1 \log_2 P_1 - P_2 \log_2 P_2
\]

S: [9+, 5-], Entropy = \(-(9/14)\log_2(9/14) -(5/14)\log_2(5/14)\)
= 0.94

Humidity

- high
- Normal

[3+, 4-]  [6+, 1-]
Entropy=0.985  Entropy=0.592
Information gain for Humidity

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

- \( S: [9+, 5-] \), \( E=0.94 \)
- \( \text{Humidity} \)
  - high: \([3+, 4-]\), \( H=0.985 \)
  - Normal: \([6+, 1-]\), \( H=0.592 \)

\[ \text{Gain}(S, \text{Humidity}) = 0.94 - ((7/14)*0.985+(7/14)*0.592 = 0.151 \]

(remember, we’re using \( H \) to mean ‘entropy’)

\( G \)
### Gain(S, outlook)

- Gain(S, outlook) = 0.246

### Gain(S, Temp)

- Gain(S, Temp) = 0.029

### Gain(S, Humidity)

- Gain(S, Humidity) = 0.151

### Gain(S, wind)

- Gain(S, wind) = 0.048

---

<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>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>
Since ‘outlook’ has the most information gain, this becomes the root of the decision tree.

We split the data into subsets, based on the outlook.

…and we remove outlook as an attribute to consider.
Humidity

<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>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</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>
</tbody>
</table>

Gain = 0.971 - (2/5)*1.0 = 0.570

Wind

<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>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</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>
</tbody>
</table>

Gain < 0.971

Temp.

<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>
</tbody>
</table>

Gain = 0.971 - (2/5)*1.0 = 0.570

Humidity

<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>
</tbody>
</table>

Gain = 0.971

Sunny

Gain = 0.971

Overcast

Gain = 0.971

Rain

Gain < 0.971
### Humidity

<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>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>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Outlook

- **Sunny**
  - D1: Sunny, Hot, High, Weak, No
  - D2: Sunny, Hot, High, Strong, No
- **Overcast**
  - D3: Overcast, Hot, High, Weak, Yes
  - D7: Overcast, Cool, Normal, Strong, Yes
- **Rain**
  - D4: Rain, Mild, High, Weak, Yes
  - D5: Rain, Cool, Normal, Weak, Yes
  - D6: Rain, Cool, Normal, Strong, No
  - D10: Rain, Mild, Normal, Weak, Yes
  - D14: Rain, Mild, High, Strong, No

### Normal

<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>D9</td>
<td>Sunny</td>
<td>Cool</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>
</tbody>
</table>
If the data subset at a branch is all one category, you can stop and return that category as the answer.
The final tree

- **outlook**
  - sunny
  - overcast
  - rain

- **Humidity**
  - high
  - Normal
    - No
    - Yes

- **wind**
  - strong
  - weak
    - No
    - Yes
Measuring Performance

How do we know that $h \approx f$? (Hume’s **Problem of Induction**)

1) Use theorems of computational/statistical learning theory

2) Try $h$ on a new **test set** of examples
   (use **same distribution over example space** as training set)

**Learning curve** = % correct on test set as a function of training set size
What the learning curve tells us

Learning curve depends on
- realizable (can express target function) vs. non-realizable
  non-realizability can be due to missing attributes
  or restricted hypothesis class (e.g., thresholded linear function)
- redundant expressiveness (e.g., loads of irrelevant attributes)
Rule #2 of Machine Learning

The *best* (i.e. the one that generalizes well) hypothesis almost never achieves 100% accuracy on the training data.

(Rule #1 was: you can’t learn anything without inductive bias)
Avoiding Overfitting

• Approaches
  – Stop splitting when information gain is low or when split is not statistically significant.
  – Grow full tree and then prune it when done

• How to pick the “best” tree?
  – Performance on training data?
  – Performance on validation data?
  – Complexity penalty?
Reduced Error Pruning

- Split data into a training and a validation set

- Repeat until pruning hurts performance measure
  1. Try removing each leaf node (one by one) and measure the resulting performance on the validation set
  2. Remove the leaf that most improves performance
C4.5 Algorithm

• Builds a decision tree from labeled training data

• Also by Ross Quinlan

• Generalizes ID3 by
  – Allowing continuous value attributes
  – Allows missing attributes in examples
  – Prunes tree after building to improve generality
Rule post pruning

- Used in C4.5
- Steps
  1. Build the decision tree
  2. Convert it to a set of logical rules
  3. Prune each rule independently
  4. Sort rules into desired sequence for use
Take away about decision trees

- Used as classifiers
- Supervised learning algorithms (ID3, C4.5)
- (mostly) Batch processing
- Good for situations where
  - The classification categories are finite
  - The data can be represented as vectors of attributes
  - You want to be able to UNDERSTAND how the classifier makes its choices