Today

- Decision Trees
  - entropy
  - mutual information
Another Classification Idea

- We could view the decision boundary as being the composition of several simple boundaries.
Decision Tree: Example

- **width > 6.5cm?**
  - Yes
    - **height > 9.5cm?**
      - Yes (Lemon)
      - No (Orange)
  - No
    - **height > 6.0cm?**
      - Yes (Orange)
      - No (Lemon)
Decision Trees

- Internal nodes test attributes
Decision Trees

Internal nodes test attributes

Branching is determined by attribute value
Decision Trees

- Internal nodes test attributes
- Branching is determined by attribute value
- Leaf nodes are outputs (class assignments)
- Internal nodes test attributes
- Branching is determined by attribute value
- Leaf nodes are outputs (class assignments)
- In general, a decision tree can represent any binary function
Choose an attribute on which to descend at each level.
Choose an attribute on which to descend at each level.

Condition on earlier (higher) choices.
Decision Tree: Algorithm

- Choose an attribute on which to descend at each level.
- Condition on earlier (higher) choices.
- Generally, restrict only one dimension at a time.
Choose an attribute on which to descend at each level.
Condition on earlier (higher) choices.
Generally, restrict only one dimension at a time.
Declare an output value when you get to the bottom.
Choose an attribute on which to descend at each level.
Condition on earlier (higher) choices.
Generally, restrict only one dimension at a time.
Declare an output value when you get to the bottom.
In the orange/lemon example, we only split each dimension once, but that is not required.
Choose an attribute on which to descend at each level.

Condition on earlier (higher) choices.

Generally, restrict only one dimension at a time.

Declare an output value when you get to the bottom.

In the orange/lemon example, we only split each dimension once, but that is not required.

How do you construct a useful decision tree?
Choose an attribute on which to descend at each level.
Condition on earlier (higher) choices.
Generally, restrict only one dimension at a time.
Declare an output value when you get to the bottom.
In the orange/lemon example, we only split each dimension once, but that is not required.

How do you construct a useful decision tree?
We use information theory to guide us.
Two Binary Sequences

Sequence 1:
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

Sequence 2:
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?
Two Binary Sequences

Sequence 1:
0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

Sequence 2:
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?

versus

16

versus

8 10

16

2

0 1
Quantifying Uncertainty

\[ H(X) = - \sum_{x \in X} p(x) \log_2 p(x) \]

\[
\begin{align*}
- \frac{8}{9} \log_2 \frac{8}{9} - \frac{1}{9} \log_2 \frac{1}{9} & \approx \frac{1}{2} \\
- \frac{4}{9} \log_2 \frac{4}{9} - \frac{5}{9} \log_2 \frac{5}{9} & \approx 0.99
\end{align*}
\]
Quantifying Uncertainty

\[ H(X) = - \sum_{x \in X} p(x) \log_2 p(x) \]

\[ -\frac{8}{9} \log_2 \frac{8}{9} - \frac{1}{9} \log_2 \frac{1}{9} \approx \frac{1}{2} \]

\[ -\frac{4}{9} \log_2 \frac{4}{9} - \frac{5}{9} \log_2 \frac{5}{9} \approx 0.99 \]

- How surprised are we by a new value in the sequence?
Quantifying Uncertainty

\[ H(X) = - \sum_{x \in X} p(x) \log_2 p(x) \]

\[ -\frac{8}{9} \log_2 \frac{8}{9} - \frac{1}{9} \log_2 \frac{1}{9} \approx \frac{1}{2} \]

\[ -\frac{4}{9} \log_2 \frac{4}{9} - \frac{5}{9} \log_2 \frac{5}{9} \approx 0.99 \]

- How surprised are we by a new value in the sequence?
- How much information does it convey?
Quantifying Uncertainty: Shannon Entropy

\[ H(X) = - \sum_{x \in X} p(x) \log_2 p(x) \]

- Shannon Entropy is an extremely powerful concept.
- It tells you how much you can compress your data!
• For a discrete random variable $X$, where $P(X=x_i) = p(x_i)$, the entropy of a random variable is:

$$H(p) = - \sum_i p(x_i) \log p(x_i).$$

• Distributions that are sharply picked around a few values will have a relatively low entropy, whereas those that are spread more evenly across many values will have higher entropy.

• Histograms of two probability distributions over 30 bins.

• The largest entropy will arise from a uniform distribution $H = -\ln(1/30) = 3.40$.

• For a density defined over continuous random variable, the differential entropy is given by:

$$H(p) = - \int p(x) \log p(x) dx.$$
Decision Tree: Algorithm

- Choose an attribute on which to descend at each level.
- Condition on earlier (higher) choices
- Generally, restrict only one dimension at a time.
- How do you construct a useful decision tree?
- We use information theory to guide us
### Entropy of a Joint Distribution

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

\[
H(X, Y) = - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(x, y)
\]

\[
= - \frac{24}{100} \log_2 \frac{24}{100} - \frac{1}{100} \log_2 \frac{1}{100} - \frac{25}{100} \log_2 \frac{25}{100} - \frac{50}{100} \log_2 \frac{50}{100}
\]

\[
\approx 1.56 \text{bits}
\]
What is the entropy of cloudiness, given that it is raining?

\[ H(X|Y = y) = \sum_{x \in X} p(x|y) \log_2 p(x|y) \]

\[ = - \frac{24}{25} \log_2 \frac{24}{25} - \frac{1}{25} \log_2 \frac{1}{25} \]

\[ \approx 0.24 \text{ bits} \]
## (Non-Specific) Conditional Entropy

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Raining</strong></td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td><strong>Not Raining</strong></td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

The expected conditional entropy:

\[
H(X|Y) = \sum_{y \in Y} p(y)H(X|Y = y) \\
= -\sum_{y \in Y} \sum_{x \in X} p(x, y) \log_2 p(x|y)
\]
(Non-Specific) Conditional Entropy

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

What is the entropy of cloudiness, given the knowledge of whether or not it is raining?

\[ H(X|Y) = \sum_{y \in Y} p(y)H(X|Y = y) \]

\[ = \frac{1}{4}H(\text{clouds|is raining}) + \frac{3}{4}H(\text{clouds|not raining}) \]

\[ \approx 0.75 \text{ bits} \]
How much information about cloudiness do we get by discovering whether it is raining?

$$IG(X \mid Y) = H(X) - H(X \mid Y)$$

$$\approx 0.25 \text{ bits}$$
Mutual Information

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

- How much information about cloudiness do we get by discovering whether it is raining?

\[ IG(X|Y) = H(X) - H(X|Y) \]
\[ \approx 0.25 \text{ bits} \]

- Also called information gain in $X$ due to $Y$
How much information about cloudiness do we get by discovering whether it is raining?

\[ IG(X|Y) = H(X) - H(X|Y) \approx 0.25 \text{ bits} \]

- Also called information gain in \( X \) due to \( Y \)
- For decision trees, \( X \) is the class/label and \( Y \) is an attribute
I made the fruit data partitioning just by eyeballing it.
I made the fruit data partitioning just by eyeballing it.

We can use the **mutual information** to automate the process.
I made the fruit data partitioning just by eyeballing it.

We can use the **mutual information** to automate the process.

At each level, one must choose:
I made the fruit data partitioning just by eyeballing it.

We can use the mutual information to automate the process.

At each level, one must choose:

1. Which variable to split.
I made the fruit data partitioning just by eyeballing it.

We can use the **mutual information** to automate the process.

At each level, one must choose:

1. Which variable to split.
2. Possibly where to split it.
I made the fruit data partitioning just by eyeballing it.

We can use the **mutual information** to automate the process.

At each level, one must choose:

1. Which variable to split.
2. Possibly where to split it.

Choose them based on how much information we would gain from the decision!
Simple, greedy, recursive approach, builds up tree node-by-node

1. pick an attribute to split at a non-terminal node
2. split examples into groups based on attribute value
3. for each group:
   ▶ if no examples – return majority from parent
   ▶ else if all examples in same class – return class
   ▶ else loop to step 1
### Decision Tree Example: Data

<table>
<thead>
<tr>
<th>Ex.</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>WillWait</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_2$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
</tr>
<tr>
<td>$X_3$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_4$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
</tr>
<tr>
<td>$X_5$</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>$X_6$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_7$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>$X_8$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_9$</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>T</td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>$X_{12}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
</tr>
</tbody>
</table>

Russell & Norvig example
Attribute Selection

\[
IG(Y) = H(X) - H(X|Y)
\]

\[
IG(\text{type}) = 1 - \left[ \frac{2}{12} H\left( \frac{1}{2}, \frac{1}{2} \right) + \frac{2}{12} H\left( \frac{1}{2}, \frac{1}{2} \right) + \frac{4}{12} H\left( \frac{2}{4}, \frac{2}{4} \right) + \frac{4}{12} H\left( \frac{2}{4}, \frac{2}{4} \right) \right] = 0
\]

\[
IG(\text{Patrons}) = 1 - \left[ \frac{2}{12} H(0, 1) + \frac{4}{12} H(1, 0) + \frac{6}{12} H\left( \frac{2}{6}, \frac{4}{6} \right) \right] \approx 0.541
\]
Which Tree is Better?

Urtasun & Zemel (UofT)

CSC 411: 06-Decision Trees

Sep 30, 2015 20 / 24
What Makes a Good Tree?

- Not too small: need to handle important but possibly subtle distinctions in data
- Not too big:
  - Computational efficiency (avoid redundant, spurious attributes)
  - Avoid over-fitting training examples
- Occam’s Razor: find the simplest hypothesis (smallest tree) that fits the observations
- Inductive bias: small trees with informative nodes near the root
What Makes a Good Tree?

- Not too small: need to handle important but possibly subtle distinctions in data
- Not too big:
  - Computational efficiency (avoid redundant, spurious attributes)
  - Avoid over-fitting training examples
What Makes a Good Tree?

- Not too small: need to handle important but possibly subtle distinctions in data
- Not too big:
  - Computational efficiency (avoid redundant, spurious attributes)
  - Avoid over-fitting training examples
- Occam’s Razor: find the simplest hypothesis (smallest tree) that fits the observations
What Makes a Good Tree?

- Not too small: need to handle important but possibly subtle distinctions in data
- Not too big:
  - Computational efficiency (avoid redundant, spurious attributes)
  - Avoid over-fitting training examples
- **Occam’s Razor**: find the simplest hypothesis (smallest tree) that fits the observations
- **Inductive bias**: small trees with informative nodes near the root
Problems:

- You have exponentially less data at lower levels.
- Too big of a tree can overfit the data.
- Greedy algorithms don't necessarily yield the global optimum.

In practice, one often regularizes the construction process to try to get small but highly-informative trees.

Decision trees can also be used for regression on real-valued outputs, but it requires a different formalism.
Problems:

- You have exponentially less data at lower levels.
Problems:

- You have exponentially less data at lower levels.
- Too big of a tree can overfit the data.
Problems:

- You have exponentially less data at lower levels.
- Too big of a tree can overfit the data.
- Greedy algorithms don’t necessarily yield the global optimum.
Problems:

- You have exponentially less data at lower levels.
- Too big of a tree can overfit the data.
- Greedy algorithms don’t necessarily yield the global optimum.

In practice, one often regularizes the construction process to try to get small but highly-informative trees.
Decision Tree Miscellany

Problems:

- You have exponentially less data at lower levels.
- Too big of a tree can overfit the data.
- Greedy algorithms don’t necessarily yield the global optimum.

In practice, one often regularizes the construction process to try to get small but highly-informative trees.

Decision trees can also be used for regression on real-valued outputs, but it requires a different formalism.
Comparison to k-NN

K-Nearest Neighbors
- Decision boundaries: piece-wise
- Test complexity: non-parametric, few parameters besides (all?) training examples

Decision Trees
- Decision boundaries: axis-aligned, tree structured
- Test complexity: attributes and splits
Applications of Decision Trees

- Can express any Boolean function, but most useful when function depends critically on few attributes
Applications of Decision Trees

- Can express any Boolean function, but most useful when function depends critically on few attributes
- Bad on: parity, majority functions; also not well-suited to continuous attributes
Applications of Decision Trees

- Can express any Boolean function, but most useful when function depends critically on few attributes
- Bad on: parity, majority functions; also not well-suited to continuous attributes
- Practical Applications:
  - Flight simulator: 20 state variables; 90K examples based on expert pilot’s actions; auto-pilot tree
  - Yahoo Ranking Challenge
  - Random Forests