Learning Decision Trees

To play or not to play?
A “personal” decision tree for deciding whether to wait at a restaurant

- A decision tree for *Wait?* based on personal “rules of thumb”:
### Input Data for Learning

- **Past examples when I did/did not wait for a table:**

<table>
<thead>
<tr>
<th>Example (X_i)</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>(Wait)</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>T</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>T</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>T</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</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>
Decision Tree Learning

- **Aim:** Find a small tree *consistent* with 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
        \[ best \leftarrow \text{CHOOSE-ATTRIBUTE}(attributes, examples) \]
        tree \leftarrow \text{a new decision tree with root test } best
        for each value \( v_i \) of \( best \) do
            \[ \text{examples}_i \leftarrow \{\text{elements of examples with } best = v_i\} \]
            \[ \text{subtree} \leftarrow \text{DTL}(\text{examples}_i, \text{attributes} - best, \text{MODE}(\text{examples})) \]
            add a branch to \( \text{tree} \) with label \( v_i \) and subtree \( \text{subtree} \)
        \]
    return \( \text{tree} \)
```
Choosing an attribute to split on

- Idea: a good attribute should reduce uncertainty
  - E.g., splits the examples into subsets that are (ideally) "all positive" (T) or "all negative" (F)
- *Patrons?* is a better choice

For *Type?*, to wait or not to wait is still at 50%
Reduce uncertainty?
How do you quantify uncertainty?
Use information theory!

- **Entropy** measures the amount of uncertainty in a probability distribution.

- **Entropy** (or information content in bits) of an answer to a question with $n$ possible answers $v_1, \ldots, v_n$:

$$I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i)$$
Using information theory

- Suppose we have $p$ examples with $\text{Wait} = \text{True}$ (positive) and $n$ examples with $\text{Wait} = \text{False}$ (negative).

- Our best estimate of the probabilities of $\text{Wait} = \text{true}$ or $\text{false}$ is given by:
  
  \[
P(\text{true}) \approx \frac{p}{p + n} \\
  p(\text{false}) \approx \frac{n}{p + n}
  \]

- Hence the entropy (in bits) is given by:
  
  \[
  I\left(\frac{p}{p + n}, \frac{n}{p + n}\right) = -\frac{p}{p + n}\log_2 \frac{p}{p + n} - \frac{n}{p + n}\log_2 \frac{n}{p + n}
  \]
Entropy is highest when uncertainty is greatest.

- $P(Wait = T)$
- $Wait = F$
- $Wait = T$

$P(Wait = T)$
Choosing an attribute to split on

- Idea: a good attribute should reduce *uncertainty* and result in “gain in information”
- How much information do we gain if we disclose the value of some attribute?

- Answer:
  uncertainty before – uncertainty after
Back at the Restaurant

Before choosing an attribute: 6 True and 6 False

Entropy = - \frac{6}{12} \log\left(\frac{6}{12}\right) - \frac{6}{12} \log\left(\frac{6}{12}\right)
= - \log\left(\frac{1}{2}\right) = \log(2) = 1 \text{ bit}

There is “1 bit of information to be discovered”
Choosing an Attribute

If we choose **Type**: Along “French”: entropy = 1 bit.
Information gain = 1-1 = 0. (same for other branches)

If we choose **Patrons**:
In branches “None” and “Some”, entropy = 0
For “Full”, entropy = \(-\frac{2}{6} \log(\frac{2}{6}) - \frac{4}{6} \log(\frac{4}{6}) = 0.92\)

So info gain = (1-0) or (1-0.92) bits > 0 in all cases
**Choosing Patrons gains more information!**
Combining entropy across branches

- Computing average entropy
- Weight entropies according to probability of branches
  - 2/12 times we entered “None” so weight for “None” = 1/6
  - “Some” has weight: 4/12 = 1/3
  - “Full” has weight: 6/12 = ½

AvgEntropy = \sum_{i=1}^N \frac{p_i + n_i}{p + n} \text{Entropy} \left( \frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i} \right)

Sum over all N branches weight for each branch entropy for each branch
Information gain

- Information Gain (IG) (= reduction in entropy) when choosing attribute A:

\[ IG(A) = \text{Entropy before choosing} - \text{AvgEntropy after choosing A} \]

- When constructing each level of decision tree, choose attribute with largest IG
Information gain in our example

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

\[ IG(\text{Patrons}) = 1 - \left[ \frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .541 \text{ bits} \]

*Patrons* has highest IG of all attributes

⇒ DTL algorithm chooses *Patrons* as the root
Decision tree learned from the 12 examples:

- Substantially simpler than “rules-of-thumb” tree
  - more complex hypothesis not justified by small amount of data
Performance Evaluation

- How do we know that the learned tree $h \approx true f$?
- Answer: Try $h$ on a new test set of examples
- Learning curve = % correct on test set as a function of training set size
Generalization

- How do we know the classifier function we have learned is good?
  - Look at generalization error on test data
    - Method 1: Split data into separate training and test sets (the “hold out” method)
      - What if the split you chose was bad?
    - Method 2: Cross-Validation
Cross-validation

- **K-fold cross-validation:**
  - Divide data into K subsets of equal size
  - Train learning algorithm K times, each time leaving out one of the subsets, and compute error on left-out subset
  - Report average error over all subsets

- **Leave-1-out cross-validation:**
  - Train on all but 1 data point, test on that data point; repeat for each point
  - Report average error over all points
Next Time

- Other classification methods
  - Linear Classification
  - Support Vector Machines

- To Do:
  - Project 4
  - Read Chapter 18