**Inductive learning**

- Simplest form: learn a function from examples
  
  $f$ is the target function

An example is a pair $(x, f(x))$

Problem: find a hypothesis $h$
such that $h = f$
given a training set of examples

(This is a highly simplified model of real learning:
- Ignores prior knowledge
- Assumes examples are given)

---

**Decision Trees**

CS 540 section 2
Louis Oliphant
oliphant@cs.wisc.edu

---

**Learning decision trees**

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:
1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

---

**Attribute-based representations**

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

<table>
<thead>
<tr>
<th>Example</th>
<th>Attributes</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>$X_2$</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>$X_3$</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>$X_4$</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>$X_5$</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>$X_6$</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>$X_7$</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>$X_8$</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>$X_9$</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

- Classic
**Decision trees**

- One possible representation for hypotheses
- E.g., here is the “true” tree for deciding whether to wait:

![Decision Tree Diagram]

**Expressiveness**

- 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 with one path to leaf for each example (unless nondeterministic in $\gamma$) but it probably won’t generalize to new examples.
- Prefer to find more compact decision trees.

**Decision tree learning**

- 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 = elements of examples with best = $v_i$;
        subtree = DTL(examples, attributes – best, Mode(examples));
        add a branch to tree with label $v_i$ and subtree as subtrees;
    return tree;
```

**Choosing an attribute**

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

```
Patrons?

- Parse
- Stars
- Full
- French
- Thai
- Japanese
- Patrons?
```

- *Patrons?* is a better choice
**Using information theory**

- To implement Choose-Attribute in the DTL algorithm
- Information Content (Entropy):
  \[ I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i) \]
- For a training set containing \( p \) positive examples and \( n \) negative examples:
  \[
  I(p, n) = - \frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}
  \]

**Information function with two categories**

Entropy reflects the lack of “purity” of some particular set \( S \)

As the proportion of positives \( p \) approaches 0.5 (very impure), the Entropy of \( S \) converges to 1.0

**Information gain**

- A chosen attribute \( A \) divides the training set \( E \) into subsets \( E_i, \ldots, E_v \), according to their values for \( A \), where \( A \) has \( v \) distinct values.
  \[
  \text{remainder}(A) = \sum_{i=1}^{v} p_i + n_i I(p_i, n_i)
  \]
- Information Gain (IG) or reduction in entropy from the attribute test:
  \[
  IG(A) = I(p, n) - \text{remainder}(A)
  \]
- Choose the attribute with the largest IG

**Information gain**

For the training set, \( p = n = 6 \), \( I(6/12, 6/12) = 1 \) bit

Consider the attributes *Patrons* and *Type* (and others too):

\[
\begin{align*}
IG(\text{Patrons}) &= 1 - \frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{4}{6} I(2,4) + \frac{4}{6} I(4,2) = 0.541 \text{ bits} \\
IG(\text{Type}) &= 1 - \frac{2}{12} I(1,1) + \frac{4}{12} I(1,2) + \frac{4}{4} I(2,2) + \frac{4}{4} I(4,4) = 0.285 \text{ bits}
\end{align*}
\]

*Patrons* has the highest IG of all attributes and so is chosen by the DTL algorithm as the root.
Example contd.

• Decision tree learned from the 12 examples:

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

\[
I(p, n) = - \frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}
\]

\[
\text{remainder}(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I\left( \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i} \right)
\]

\[
\text{IG}(A) = I\left( \frac{p}{p+n}, \frac{n}{p+n} \right) - \text{remainder}(A)
\]

<table>
<thead>
<tr>
<th>Size</th>
<th>Shape</th>
<th>Color</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>square</td>
<td>red</td>
<td>+</td>
</tr>
<tr>
<td>large</td>
<td>circle</td>
<td>blue</td>
<td>+</td>
</tr>
<tr>
<td>small</td>
<td>triangle</td>
<td>blue</td>
<td>-</td>
</tr>
<tr>
<td>large</td>
<td>circle</td>
<td>red</td>
<td>-</td>
</tr>
<tr>
<td>small</td>
<td>circle</td>
<td>blue</td>
<td>+</td>
</tr>
<tr>
<td>small</td>
<td>circle</td>
<td>red</td>
<td>-</td>
</tr>
</tbody>
</table>

Make The Decision Tree

Pick Features that maximize IG

Handling Continuous Features

• One way of dealing with a continuous feature \( F \) is to treat them like Boolean features, partitioned on a dynamically chosen threshold \( t \):
  - Sort the examples in \( S \) according to \( F \)
  - Identify adjacent examples with differing class labels
  - Compute InfoGain with \( t \) equal to the average of the values of at these boundaries
  - Can also be generalized to multiple thresholds

**Handling Continuous Features**

- There are two candidates for threshold $t$ in this example:

<table>
<thead>
<tr>
<th>Temperature</th>
<th>40</th>
<th>48</th>
<th>60</th>
<th>72</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt;1.000?$</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

$t = (48+60)/2 = 54$  
$t = (80+90)/2 = 85$

- The dynamically-created Boolean features $Temp_{s1}$ and $Temp_{s3}$ can now compete with the other Boolean and discrete features in the dataset.

**Noisy Data**

- Noisy data could be in the examples:
  - examples have the same attribute values, but different classifications (rare case: if is empty of attributes)
  - classification is wrong
  - attributes values are incorrect because of errors getting or preprocessing the data
  - irrelevant attributes used in the decision-making process

- Use Pruning to improve performance when working with noisy data.

**Missing Data**

- Missing data:
  - while learning: replace with most likely value
  - while learning: use NotKnown as a value
  - while classifying: follow arc for all values and weight each by the frequency of exs. crossing that arc

**Tree Induction as Search**

- We can think of inducing the “best tree” as an optimization search problem:
  - **States**: possible (sub-)trees
  - **Actions**: add a feature as a node of the tree
  - **Objective Function**: increase the overall information gain of the tree

- Essentially, Decision Tree Learner is a hill-climbing search through the hypothesis space, where the heuristic picks features that are likely to lead to small trees.
Pruning

- Overfitting
  meaningless regularity is found in the data
  - irrelevant attributes confound the
    true, important, distinguishing features
  - fix by pruning lower nodes in the decision tree
  - if gain of best attribute is below a threshold,
    make this node a leaf rather than
    generating child nodes

```java
//find better tree
progressMade = false;
currentTree = bestTree;
for (each interiorNode N in currentTree) { //start at root
    prunedTree = pruned copy of currentTree;
    newAccuracy = accuracy of prunedTree on TUNE set;
    if (newAccuracy >= bestAccuracy) {
        bestAccuracy = newAccuracy;
        bestTree = prunedTree;
        progressMade = true;
    }
}
```

Pruning

randomly partition training examples into:
TRAIN set (80% of training exs.)
TUNE set (10% of training exs.)
TEST set (10% of training exs.)

build decision tree as usual using TRAIN set:
bestTree = decision tree produced on the TRAIN set;
bestAccuracy = accuracy of bestTree on the TUNE set;
progressMade = true;
while (progressMade) { //while accuracy on TUNE improves
    find better tree;
    //starting at root, consider various pruned versions
    //of the current tree and see if any are better than
    //the best tree found so far
}
return bestTree;
use TEST to determine performance accuracy;

```
//pruned copy of currentTree
replace interiorNode N in currentTree by a leaf node
label leaf node with the majorityClass among TRAIN set
examples that reached node N
break ties in favor of '-'
```
**Case Studies**

- Decision trees have been shown to be at least as accurate as human experts.
- Diagnosing breast cancer
  - humans correct 65% of the time
  - decision tree classified 72% correct
- BP designed a decision tree for gas-oil separation for offshore oil platforms
- Cessna designed a flight controller using 90,000 exs. and 20 attributes per ex.

**Conclusions**

- Information (Entropy) of a set of data
- Information Gain using some feature on a set of data
- Handling
  - continuous features
  - noisy data
  - missing values
- Pruning
- Constructing decision trees as Search