<table>
<thead>
<tr>
<th>BackProp</th>
<th>1 handouts: slides</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Trees</td>
<td>asst 4 is due</td>
</tr>
<tr>
<td></td>
<td>730W blog entries were due</td>
</tr>
</tbody>
</table>
BackProp

- Three layers
- Nonlinear
- BackProp
- Break

Decision Trees

BackProp
Supervised Learning: Summary So Far

- **BackProp**
  - Three layers
  - Nonlinear
  - BackProp
  - Break

- **Decision Trees**

- **k-NN**: distance function (any attributes), any labels
- **Neural network**: numeric attributes, numeric or binary labels
  - **Regression**: incremental training with LMS
  - **3-Layer ANN**: non-linear wrt features
- **Inductive Logic Programming**: logical concepts
The Three-layer Architecture

‘hidden layer’
non-linear!
training: backwards error propagation
recurrence
Figure 18.23  (a) The result of combining two opposite-facing soft threshold functions to produce a ridge. (b) The result of combining two ridges to produce a bump.
$k$ inputs, $j$ hidden units, $i$ outputs
$g'(in_i)$ is derivative of activation function wrt input $i$

$$
\Delta_i = g'(in_i)(\hat{y} - y)
$$
$$
W_{j,i} = W_{j,i} - \alpha a_j \Delta_i
$$
Backwards Error Propagation

$k$ inputs, $j$ hidden units, $i$ outputs

$g'(in_i)$ is derivative of activation function wrt input $i$

\[
\Delta_i = g'(in_i)(\hat{y} - y)
\]

\[
W_{j,i} = W_{j,i} - \alpha a_j \Delta_i
\]

\[
\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i
\]

\[
W_{k,j} = W_{k,j} - \alpha a_k \Delta_j
\]

only locally optimal, dependence on structure
asst 4
asst 5: data, tool, reference
projects!
Decision Trees
### Example: WillWait

#### Attributes

<table>
<thead>
<tr>
<th>Example</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>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>0–10</td>
<td>Yes</td>
</tr>
<tr>
<td>$X_2$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
<td>30–60</td>
<td>No</td>
</tr>
<tr>
<td>$X_3$</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Some</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>0–10</td>
<td>Yes</td>
</tr>
<tr>
<td>$X_4$</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Thai</td>
<td>10–30</td>
<td>Yes</td>
</tr>
<tr>
<td>$X_5$</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Full</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>&gt;60</td>
<td>No</td>
</tr>
<tr>
<td>$X_6$</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
<td>Yes</td>
<td>Italian</td>
<td>0–10</td>
<td>Yes</td>
</tr>
<tr>
<td>$X_7$</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Burger</td>
<td>0–10</td>
<td>No</td>
</tr>
<tr>
<td>$X_8$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
<td>Yes</td>
<td>Thai</td>
<td>0–10</td>
<td>Yes</td>
</tr>
<tr>
<td>$X_9$</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Burger</td>
<td>&gt;60</td>
<td>No</td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>Italian</td>
<td>10–30</td>
<td>No</td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
<td>0–10</td>
<td>No</td>
</tr>
<tr>
<td>$X_{12}$</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>30–60</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Figure 18.3** Examples for the restaurant domain.

---

Wheeler Ruml (UNH)
**Building a Decision Tree**

**DTLearn** *(examples, attributes, default)*

- if no examples, return default
- if all same label, return it
- \( m \leftarrow \) majority label
- if no attributes, return \( m \)
- else
  - \( a \leftarrow \) choose attribute
  - make node that branches on \( a \)
  - remove \( a \) from attributes
  - for each value \( v \) of \( a \)
    - subtree \( \leftarrow \) DTLearn *(examples with \( a = v \), attributes, \( m \))*
    - add branch to subtree for \( v \) at node
- return node
want attribute that reduces uncertainty
want attribute that reduces uncertainty = entropy =

\[ H(X) = - \sum_i P(x_i) \log_2 P(x_i) \]

where \( X \) is random var that takes value \( x_i \) with prob \( P(x_i) \)
Branching

want attribute that reduces uncertainty = entropy =

$$H(X) = - \sum_i P(x_i) \log_2 P(x_i)$$

where $X$ is random var that takes value $x_i$ with prob $P(x_i)$

information gain of attribute $A$:

$$H(X) - \sum_{a \in A} P(a) H(X_a)$$

where $X_a$ contains only examples with $A = a$
Branching

want attribute that reduces uncertainty = entropy =

\[ H(X) = - \sum_i P(x_i) \log_2 P(x_i) \]

where \(X\) is random var that takes value \(x_i\) with prob \(P(x_i)\)

information gain of attribute \(A\):

\[ H(X) - \sum_{a \in A} P(a) H(X_a) \]

where \(X_a\) contains only examples with \(A = a\)

stop when gain is small (\(\chi^2\) test, see p.705) or cross-validate
What question didn’t you get to ask today?

What’s still confusing?

What would you like to hear more about?

Please write down your most pressing question about AI and put it in the box on your way out.

Thanks!