Supervised Learning: Summary So Far

- **Decision Trees**
- **Naive Bayes**
- **Boosting**

Learning as function approximation:

- **k-NN**: distance function (any attributes), any labels
- **Neural network**: numeric attributes, numeric or binary labels
- **Regression**: incremental training with LMS
- **3-Layer ANN**: train with BackProp

What about discrete attributes and labels?
Decision Trees

- Example
- Construction
- Break

Naive Bayes

Boosting
### Example: WillWait

#### Decision Trees
- **Example**
- **Construction**
- **Break**

#### Naive Bayes

#### Boosting

#### Table: Examples for the restaurant domain.

<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.
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 \( P(x_i) \)
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 \( A = a \)
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 \( A = a \)

prune branches when gain is small (\( \chi^2 \) test, see p.705) or cross-validate
Naive Bayes
learning as function approximation

**$k$-NN:** distance function (any attributes), any labels

**Neural network:** numeric attributes, numeric or binary labels

**Regression:** incremental training with LMS

**3-Layer ANN:** train with BackProp

**Decision Trees:** easier with discrete attributes and labels

learning as density estimation
Bayes’ Theorem

\[ P(H|D) = \frac{P(H)P(D|H)}{P(D)} \]
Bayes’ Theorem

\[ P(H|D) = \frac{P(H)P(D|H)}{P(D)} \]

\[ P(H) = 0.0001 \]
\[ P(D|H) = 0.99 \]
\[ P(D) = 0.01 \]

\[ P(H|D) = \]
Bayes’ Theorem

\[ P(H|D) = \frac{P(H)P(D|H)}{P(D)} \]

If you don’t have \( P(D) \),
Bayes’ Theorem

\[
P(H|D) = \frac{P(H)P(D|H)}{P(D)}
\]

\[
\begin{align*}
P(H) &= 0.0001 \\
P(D|H) &= 0.99 \\
P(D) &= 0.01
\end{align*}
\]

\[
P(H|D) = \quad \text{(Calculate based on given probabilities)}
\]

If you don’t have \(P(D)\), sometimes it helps to note that

\[
P(D) = P(D|H)P(H) + P(D|\neg H)P(\neg H)
\]
Bayes’ Theorem:

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$
Bayes’ Theorem:

\[ P(H|D) = \frac{P(H)P(D|H)}{P(D)} \]

naive model:

\[ P(D|H) = P(x_1, \ldots, x_n|H) = \prod_i P(x_i|H) \]
Bayes’ Theorem:

\[ P(H|D) = \frac{P(H)P(D|H)}{P(D)} \]

naive model:

\[ P(D|H) = P(x_1, \ldots, x_n|H) = \prod_i P(x_i|H) \]

attributes independent, given class
Bayes’ Theorem:

\[ P(H|D) = \frac{P(H)P(D|H)}{P(D)} \]

naive model:

\[ P(D|H) = P(x_1, \ldots, x_n|H) = \prod_i P(x_i|H) \]

attributes independent, given class

\[ P(H|x_1, \ldots, x_n) = \alpha P(H) \prod_i P(x_i|H) \]
The ‘Naive Bayes’ Classifier

\[ P(H|x_1, \ldots, x_n) = \alpha P(H) \prod_{i} P(x_i|H) \]

attributes independent, given class

maximum \textit{a posteriori} = pick highest
maximum likelihood = ignore prior

watch for sparse data when learning!

learning as density estimation
<table>
<thead>
<tr>
<th>Decision Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
</tr>
</tbody>
</table>

**Boosting**
- Ensembles
- AdaBoost
- Behavior
- Summary
- EOLQs
committees, ensembles
weak vs strong learners
reduce variance, expand hypothesis space (eg, half-spaces)
$N$ examples, $T$ rounds, $L$ a weak learner on weighted examples

$p \leftarrow$ uniform distribution over the $N$ examples

for $t = 1$ to $T$ do

\begin{itemize}
  \item $h_t \leftarrow$ call $L$ with weights $p$
  \item $\epsilon_t \leftarrow h_t$'s weighted misclassification probability
  \item if $\epsilon_t = 0$, return $h_t$
  \item $\alpha_t \leftarrow \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$
  \item for each example $i$
    \begin{itemize}
      \item if $h_t(i)$ is correct, $p_i \leftarrow p_i e^{-\alpha_t}$
      \item else, $p_i \leftarrow p_i e^{\alpha_t}$
    \end{itemize}
  \end{itemize}

normalize $p$ to sum to 1

return the $h$ weighted by the $\alpha$

to classify, choose label with highest sum of weighted votes
doesn’t overfit (maximizes margin even when no error) outliers get high weight, can be inspected
problems:
■ not enough data
■ hypothesis class too small
■ boosting: learner too weak, too strong
Supervised Learning: Summary

**Decision Trees**
- \(k\)-NN: distance function (any attributes), any labels
- **Neural network:** numeric attributes, numeric or binary labels

**Boosting**
- Regression: incremental training with LMS
- **3-Layer ANN:** BackProp learning

**Decision Trees:** easier with discrete attributes and labels
**Naive Bayes:** easier with discrete attributes and labels
**Boosting:** general wrapper to improve performance

Didn’t cover: RBFs, SVMs, deep learning, structured output learning
■ 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!