Classification and Regression

- We want to learn functions of the form:
  - \( y = f(x) \)

- \( Y \) is discrete valued:
  - Classification

- \( Y \) is continuous
  - Regression

- \( X \) can be one or more continuous or discrete values.
Classification

- Estimate a discrete-valued quantity in terms of a number of features
  - Example: Car or Motorcycle?
    - Features:
      - Size in pixels
      - Aspect ratio
      - Average color
      - ...

Regression

- Estimate a continuous-valued quantity in terms of a number of features
- Example: APPL stock price
  - Features:
    - Number of news articles about upcoming products
    - Last quarter’s revenue
    - Cash on hand
    - Whether Steve Jobs is CEO
- Example: Movie rating predictions
  - Features:
    - How much did the user like other movies?
    - How much did other users like this movie?
Basics

- Training dataset
  - Data used to learn our model

- Test dataset
  - Data used to see how well we’ve learned f(x)
  - Why is this separate from training data?

Classification: roadmap

- kNN
- Decision Trees
- Boosting
- SVM
- Neural networks
kNN

Given feature vector $\mathbf{x}$, estimate $y$ based on previously seen examples close to $\mathbf{x}$

K-Nearest Neighbors
- Find k closest examples
  - Majority vote
- Special data structures make nearest-neighbor lookups relatively fast. (How would you do it?)

Very simple, effective, little parameter tuning
- A good “first try” method
Nearest Neighbor

- Example: Predict MPG given:
  - # of cylinders
  - car mass
  - Distance = \((c_i - c_j)^2 + (m_i - m_j)^2\)

- What happens?
  - # of cylinders doesn’t matter much at all!
  - Scaling matters!
    - Normalization

Decision Trees
Decision Trees

- Classify attribute vectors into two or more classes
- Boolean case: learn goal predicate

Which boolean functions can we learn?

Mushroom Decision Tree

(Large ∧ ¬Yellow) ∨ (¬Large ∧ Spotted ∧ OnPizza)

from Ginsberg, Essentials of AI
Building Decision Trees

- Given set of examples, derive consistent decision tree
  - Idea: just include path for each positive example
    - What’s wrong with this?
    - How can we do better?

Ockham’s Razor

“Pluralitas non est ponenda sine necessitate” —William of Ockham, 14th century

- Plurality should not be posited without necessity
- Prefer the simplest consistent hypothesis
- Allows for generalization
Building Decision Tree

- **Bad news**
  - Finding smallest possible tree intractable

- **Greedy approach**
  - Starting from root (containing all examples)
  - Until stuck:
    - Pick a node in which not all examples are the same
      - (And at least one attribute is left)
    - Pick attribute most effective in distinguishing among examples
    - Split node using attribute.

### Mushroom Instances

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Size</th>
<th>Color</th>
<th>OnPizza</th>
<th>Edible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>Yellow</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>Yellow</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>NY</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>NY</td>
<td>No</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Small</td>
<td>Yellow</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Small</td>
<td>NY</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Large</td>
<td>Yellow</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Large</td>
<td>Yellow</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Large</td>
<td>NY</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>No Spots</td>
<td>Large</td>
<td>NY</td>
<td>No</td>
<td>YES</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>Yellow</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>Yellow</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>NY</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>NY</td>
<td>No</td>
<td>NO</td>
</tr>
</tbody>
</table>
Choose-Attribute in DTL

Measuring Information Value

- Consider binary event with probability \( p \).
- Finding out event resolves uncertainty.
  - \( p = 1 \) or 0. Already knew it, **no new information**.
  - \( p = 1/2 \). Maximal information from event: **1 bit**.
- General formula:
  \[
  I(q) = -q \log_2 q - (1-q) \log_2 (1-q)
  \]
Information Gain

- Set of \( p \) positive examples, \( n \) negative.
- Information value, \( I(p/[p+n]) \).
- After observing binary attribute, average information is:

\[
\frac{p_i + n_i}{p+n} I(p_i/[p_i+n_i]) + \frac{p_f + n_f}{p+n} I(p_f/[p_f+n_f])
\]

Information gain is difference between information before and after observing attribute.

Choose-Attribute in DTL

Diagram showing decision tree with attributes Size, Pattern, Color, and Pizza.
Calculating Initial Information

Initially:

\[
I(5/12) = -\frac{5}{12} \log_2 \left(\frac{5}{12}\right) - \left(\frac{7}{12}\right) \log_2 \left(\frac{7}{12}\right)
\]

\[
= -\frac{5}{12}(-1.263) - \frac{7}{12}(-0.778)
\]

\[= .980\]

Fair amount of uncertainty!

Attribute Information Calculations

After observing “Large” (remainder):

\[
(6/12) I(4/6) + (6/12) I(1/6) = .784
\]

So Gain(Large) = .980 – .784 = .196

After observing “Spotted” (remainder):

\[
(6/12) I(3/6) + (6/12) I(2/6) = .959
\]

So Gain(Spotted) = .980 – .959 = .021

After observing “Yellow” (remainder):

\[
(4/12) I(0) + (8/12) I(5/8) = .636
\]

So Gain(Yellow) = .980 - .636 = .354

After observing “OnPizza” (remainder):

Same as Spotted.

So, split on Yellow: positive = NO, negative is 8 cases.
Remaining Mushroom Instances

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Size</th>
<th>Color</th>
<th>OnPizza</th>
<th>Edible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>Yellow</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>Yellow</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>NY</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Large</td>
<td>NY</td>
<td>No</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Small</td>
<td>Yellow</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Small</td>
<td>Yellow</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>Spotted</td>
<td>Small</td>
<td>NY</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Spotted</td>
<td>Small</td>
<td>NY</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Large</td>
<td>Yellow</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Large</td>
<td>Yellow</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Large</td>
<td>NY</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>No Spots</td>
<td>Large</td>
<td>NY</td>
<td>No</td>
<td>YES</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>Yellow</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>Yellow</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>NY</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>No Spots</td>
<td>Small</td>
<td>NY</td>
<td>No</td>
<td>NO</td>
</tr>
</tbody>
</table>

Induced Tree

\[ (\neg \text{Yellow} \land \text{Large}) \lor (\neg \text{Yellow} \land \neg \text{Large} \land \text{Spotted} \land \text{OnPizza}) \]