Decision making in episodic environments

- We have just looked at decision making in sequential environments
- Now let’s consider the “easier” problem of episodic environments
  - The agent gets a series of unrelated problem instances and has to make some decision or inference about each of them
  - This is what most of “machine learning” is about
Example: Image classification

<table>
<thead>
<tr>
<th>input</th>
<th>desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td></td>
</tr>
<tr>
<td>pear</td>
<td></td>
</tr>
<tr>
<td>tomato</td>
<td></td>
</tr>
<tr>
<td>cow</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td></td>
</tr>
<tr>
<td>horse</td>
<td></td>
</tr>
</tbody>
</table>
Example: Spam Filter

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential and top secret. ...

Ok, I know this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use. I know it was working fine being stuck in the corner, but when I plugged it in, hit the power nothing happened.

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99
Example: Seismic data

![Seismic data diagram](image)

- Earthquakes
- Nuclear explosions

Surface wave magnitude vs. Body wave magnitude plot.
The basic classification framework

\[ y = f(x) \]

- **Learning:** given a *training set* of labeled examples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \), estimate the parameters of the prediction function \( f \)
- **Inference:** apply \( f \) to a never before seen *test example* \( x \) and output the predicted value \( y = f(x) \)
Example: Training and testing

- Key challenge: *generalization* to unseen examples
Naïve Bayes classifier

\[ f(x) = \arg \max_y P(y | x) \]

\[ \propto \arg \max_y P(y) P(x | y) \]

\[ = \arg \max_y P(y) \prod_d P(x_d | y) \]

A single dimension or attribute of \( x \)
Decision tree classifier

Example 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)
## Decision tree classifier

### Example Data

<table>
<thead>
<tr>
<th>Example</th>
<th>Attrs</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alt Bar Fri Hun Pat Price Rain Res Type Est Wait</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>T F F T</td>
<td>Some $$$$ F T French 0–10 T</td>
</tr>
<tr>
<td>X2</td>
<td>T F F T</td>
<td>Full $ F F Thai 30–60 F</td>
</tr>
<tr>
<td>X3</td>
<td>F T F F</td>
<td>Some $ F F Burger 0–10 T</td>
</tr>
<tr>
<td>X4</td>
<td>T F T T</td>
<td>Full $ F F Thai 10–30 T</td>
</tr>
<tr>
<td>X5</td>
<td>T F T F</td>
<td>Full $$$$ F T French &gt;60 F</td>
</tr>
<tr>
<td>X6</td>
<td>F T F T</td>
<td>Some $$ T T Italian 0–10 T</td>
</tr>
<tr>
<td>X7</td>
<td>F T F F</td>
<td>None $ T F Burger 0–10 F</td>
</tr>
<tr>
<td>X8</td>
<td>F F F T</td>
<td>Some $$ T T Thai 0–10 T</td>
</tr>
<tr>
<td>X9</td>
<td>F T T F</td>
<td>Full $ T F Burger &gt;60 F</td>
</tr>
<tr>
<td>X10</td>
<td>T T T T</td>
<td>Full $$$$ F T Italian 10–30 F</td>
</tr>
<tr>
<td>X11</td>
<td>F F F F</td>
<td>None $ F F Thai 0–10 F</td>
</tr>
<tr>
<td>X12</td>
<td>T T T T</td>
<td>Full $ F F Burger 30–60 T</td>
</tr>
</tbody>
</table>
Decision tree classifier

- Patrons?
  - None
  - Some
  - Full
- WaitEstimate?
  - >60
  - 30-60
  - 10-30
  - 0-10
- Alternate?
- Hungry?
  - No
  - Yes
- Reservation?
- Fri/Sat?
  - No
  - Yes
  - No
  - Yes
- Bar?
  - No
  - Yes
  - No
  - Yes
- Alternate?
  - No
  - Yes
- Raining?
Nearest neighbor classifier

\[ f(x) = \text{label of the training example nearest to } x \]

- All we need is a distance function for our inputs
- No training required!
• Find a *linear function* to separate the classes

\[ f(x) = \text{sgn}(w_1x_1 + w_2x_2 + \ldots + w_Dx_D) = \text{sgn}(w \cdot x) \]
Perceptron

Input

Weights

\[ x_1 \quad w_1 \]
\[ x_2 \quad w_2 \]
\[ x_3 \quad w_3 \]
\[ \vdots \]
\[ x_D \quad w_D \]

Output: \( \text{sgn}(w \cdot x + b) \)
Linear separability

$x_1 \text{ and } x_2$

$x_1 \text{ or } x_2$

$x_1 \text{ xor } x_2$
Multi-Layer Neural Network

• Can learn nonlinear functions
• **Training**: find network weights to minimize the error between true and estimated labels of training examples:

\[ E(f) = \sum_{i=1}^{N} (y_i - f(x_i))^2 \]

• Minimization can be done by gradient descent provided \( f \) is differentiable
  – This training method is called **back-propagation**
Differentiable perceptron

Input

Weights

\[ w_1 \]

\[ w_2 \]

\[ w_3 \]

\[ \cdots \]

\[ w_d \]

Output:

\[ \sigma(w \cdot x + b) \]

Sigmoid function:

\[ \sigma(t) = \frac{1}{1 + e^{-t}} \]
Review: Types of classifiers

- Naïve Bayes
- Decision tree
- Nearest neighbor
- Linear classifier
- Nonlinear classifier