Why Learning?

- **Learning is essential for unknown environments**
  e.g., when designer lacks omniscience

- **Learning is necessary in dynamic environments**
  Agent can adapt to changes in environment not foreseen at design time

- **Learning is useful as a system construction method**
  Expose the agent to reality rather than trying to approximate it through equations etc.

- **Learning modifies the agent's decision mechanisms to improve performance**
Types of Learning

• **Supervised learning**: correct answers for each input is provided
  E.g., decision trees, backprop neural networks

• **Unsupervised learning**: correct answers not given, must discover patterns in input data
  E.g., clustering, principal component analysis

• **Reinforcement learning**: occasional rewards (or punishments) given
  E.g., Q learning, MDPs

Inductive learning

A form of **Supervised Learning**:
Learn a function from examples

\( f \) is the target function. Examples are pairs \((x, f(x))\)

Problem: learn a function ("hypothesis") \( h \)
  such that \( h \approx f \) (\( h \) approximates \( f \) as best as possible)
given a training set of examples

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

- Construct $h$ to agree with $f$ on training set
  - $h$ is consistent if it agrees with $f$ on all training examples
- E.g., curve fitting (regression):

```
\begin{tikzpicture}
  \draw[->] (0,0) -- (5,0) node[below] {$x$};
  \draw[->] (0,0) -- (0,5) node[left] {$f(x)$};
  \filldraw (0,1) circle (1pt);
  \filldraw (1,2) circle (1pt);
  \filldraw (2,3) circle (1pt);
  \filldraw (3,4) circle (1pt);
  \filldraw (4,5) circle (1pt);
  \node at (4.5,0.5) {$x = \text{Input data point (a training example)}$};
\end{tikzpicture}
```

```
\begin{tikzpicture}
  \draw[->] (0,0) -- (5,0) node[below] {$x$};
  \draw[->] (0,0) -- (0,5) node[left] {$f(x)$};
  \draw[thick, red, domain=0:4] plot function {x+1};
  \filldraw (0,1) circle (1pt);
  \filldraw (1,2) circle (1pt);
  \filldraw (2,3) circle (1pt);
  \filldraw (3,4) circle (1pt);
  \filldraw (4,5) circle (1pt);
\end{tikzpicture}
```

$h = \text{Straight line?}$
Inductive learning example

What about a quadratic function?

Finally, a function that satisfies all!
Inductive learning example

But so does this one...

\[ f(x) \]

Ockham’s razor principle

- Ockham’s razor: prefer the simplest hypothesis consistent with data
  - Related to KISS principle ("keep it simple stupid")
  - Smooth blue function preferable over wiggly yellow one
  - If noise known to exist in this data, even linear might be better (the lowest x might be due to noise)
Example data for learning the concept “Good day for tennis”

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Humid</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>s</td>
<td>h</td>
<td>w</td>
<td>n</td>
</tr>
<tr>
<td>d2</td>
<td>s</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d3</td>
<td>o</td>
<td>h</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d4</td>
<td>r</td>
<td>h</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d5</td>
<td>r</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d6</td>
<td>r</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d7</td>
<td>o</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d8</td>
<td>s</td>
<td>h</td>
<td>w</td>
<td>n</td>
</tr>
<tr>
<td>d9</td>
<td>s</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d10</td>
<td>r</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d11</td>
<td>s</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d12</td>
<td>o</td>
<td>h</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d13</td>
<td>o</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d14</td>
<td>r</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
</tbody>
</table>

- **Outlook** = sunny, overcast, rain
- **Humidity** = high, normal
- **Wind** = weak, strong

A Decision Tree for the Same Data

Leaves = classification
Arcs = choice of value for parent attribute

Decision tree is equivalent to logic in disjunctive normal form
PlayTennis ⇔ (Sunny ∧ Normal) ∨ Overcast ∨ (Rain ∧ Weak)
Decision Trees

Input: Description of an object or a situation through a set of attributes

Output: a decision that is the predicted output value for the input

Both input and output can be discrete or continuous

Discrete-valued functions lead to classification problems

Learning a continuous function is called regression

Example: Classification of Continuous Valued Inputs

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the $K$ classes.
### Expressiveness

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:

  \[
  \begin{array}{ccc}
  A & B & A \oplus B \\
  F & F & F \\
  F & T & T \\
  T & F & T \\
  T & T & F \\
  \end{array}
  \]

  - Trivially, there is a consistent decision tree for any training set with one path to leaf for each example
  - But most likely won’t generalize to new examples

- Prefer to find more compact decision trees

### Learning Decision Trees

**Example:** When should I wait for a table at a restaurant?

**Attributes (features) relevant to Wait? decision:**

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)
Example Decision tree

A decision tree for \textit{Wait?} based on personal “rules of thumb”:

\begin{itemize}
  \item \textbf{Patrons?}
    \begin{itemize}
      \item None
      \item Some
      \item Full
    \end{itemize}

  \item Wait Estimate?
    \begin{itemize}
      \item $>60$
      \item $30-60$
      \item $10-30$
      \item $0-10$
    \end{itemize}

  \item Alternate?
    \begin{itemize}
      \item Yes
      \item No
    \end{itemize}

  \item Hungry?
    \begin{itemize}
      \item Yes
      \item No
    \end{itemize}

  \item Reservation?
    \begin{itemize}
      \item Yes
      \item No
    \end{itemize}

  \item FriSat?
    \begin{itemize}
      \item Yes
      \item No
    \end{itemize}

  \item Rainy?
    \begin{itemize}
      \item Yes
      \item No
    \end{itemize}

  \item Bar?
    \begin{itemize}
      \item Yes
      \item No
    \end{itemize}

  \item Classification of examples is positive (T) or negative (F)
\end{itemize}

Input Data for Learning

- Past examples where I did/did not wait for a table:

\begin{table}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
Example & Alt & Bar & Fri & Hun & Pat & Price & Rain & Res & Type & Est & Target & Wait \\
\hline
$X_1$ & T & F & F & T & Some & $$ & F & T & French & 0-10 & T \\
$X_2$ & T & F & F & T & Full & $ & F & F & Thai & 30-60 & F \\
$X_3$ & F & T & F & F & Some & $ & F & F & Burger & 0-10 & T \\
$X_4$ & T & F & T & T & Full & $ & F & F & Thai & 10-30 & T \\
$X_5$ & T & F & T & F & Some & $$ & F & T & French & >60 & F \\
$X_6$ & F & T & F & T & Some & $$ & T & T & Italian & 0-10 & T \\
$X_7$ & F & T & F & F & None & $ & T & F & Burger & 0-10 & F \\
$X_8$ & F & F & F & T & Some & $$ & T & T & Thai & 0-10 & T \\
$X_9$ & F & T & T & F & Full & $ & T & F & Burger & >60 & F \\
$X_{10}$ & T & T & T & T & Full & $$ & T & T & Italian & 10-30 & F \\
$X_{11}$ & F & F & F & F & None & $ & F & F & Thai & 0-10 & F \\
$X_{12}$ & T & T & T & T & Full & $ & F & F & Burger & 30-60 & T \\
\hline
\end{tabular}
\end{table}
Decision Tree Learning

- **Aim:** find a small tree consistent with training examples
- **Idea:** (recursively) choose "most significant" attribute as root of (sub)tree

```plaintext
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 of best do
            examples ← {elements of examples with best = v}
            subtree ← DTL(examples, attributes - best, MODE(examples))
            add a branch to tree with label v, and subtree subtree
        return tree
```

Choosing an attribute to split on

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

- *Patrons?* is a better choice
Next Time

• How to choose attributes to split on?
  Using information theory and entropy
• The more, the merrier (and better) - combining classifiers
  Ensemble learning via boosting