LEARNING FROM OBSERVATIONS

Chapter 18, Sections 1–3
Outline

♦ Inductive learning
♦ Decision tree learning
♦ Measuring learning performance
Learning

Learning is essential for unknown environments, i.e., when designer lacks omniscience.

Learning is useful as a system construction method, i.e., expose the agent to reality rather than trying to write it down.

Learning modifies the agent’s decision mechanisms to improve performance.

Different kinds of learning:
- **Supervised learning**: we get correct answers for each training instance.
- **Reinforcement learning**: we get occasional rewards.
- **Unsupervised learning**: we don’t know anything...
Inductive learning

Simplest form: learn a function from examples

\( f \) is the target function

An example is a pair \( x, f(x) \), e.g.,

\[
\begin{array}{c|c|c}
O & O & X \\
X & & \\
X & & \\
\end{array}
\]

, \( +1 \)

Problem: find a hypothesis \( h \) such that \( h \approx f \) given a training set of examples

(This is a highly simplified model of real learning:

– Ignores prior knowledge
– Assumes a deterministic, observable “environment”
– Assumes that the examples are given)
Inductive learning method

Construct/adjust \( h \) to agree with \( f \) on training set
\( (h \) is consistent if it agrees with \( f \) on all examples)\)

E.g., curve fitting:

\[
f(x)
\]

\( x \)
Inductive learning method

Construct/adjust $h$ to agree with $f$ on training set
($h$ is consistent if it agrees with $f$ on all examples)

E.g., curve fitting:

\[ f(x) \]

\[ x \]
Inductive learning method

Construct/adjust $h$ to agree with $f$ on training set
($h$ is consistent if it agrees with $f$ on all examples)

E.g., curve fitting:
Inductive learning method

Construct/adjust $h$ to agree with $f$ on training set
($h$ is consistent if it agrees with $f$ on all examples)

E.g., curve fitting:
Inductive learning method

Construct/adjust $h$ to agree with $f$ on training set
($h$ is consistent if it agrees with $f$ on all examples)

E.g., curve fitting:
Inductive learning method

Construct/adjust \( h \) to agree with \( f \) on training set
(\( h \) is consistent if it agrees with \( f \) on all examples)

E.g., curve fitting:

Ockham’s razor: maximize a combination of consistency and simplicity
Attribute-based representations

Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won’t wait for a table:

<table>
<thead>
<tr>
<th>Example</th>
<th>Attributes</th>
<th>Target WillWait</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alt Bar Fri Hun Pat Price Rain Res Type Est</td>
<td></td>
</tr>
<tr>
<td>$X_1$</td>
<td>T F F T Some $$$ F T</td>
<td>French 0–10 T</td>
</tr>
<tr>
<td>$X_2$</td>
<td>T F F F Full $ F F</td>
<td>Thai 30–60 F</td>
</tr>
<tr>
<td>$X_3$</td>
<td>F T F F Some $ F F</td>
<td>Burger 0–10 T</td>
</tr>
<tr>
<td>$X_4$</td>
<td>T F T T Full $ F F</td>
<td>Thai 10–30 T</td>
</tr>
<tr>
<td>$X_5$</td>
<td>T F T F Full $$$ F T</td>
<td>French &gt;60 F</td>
</tr>
<tr>
<td>$X_6$</td>
<td>F T F T Some $$$ T T</td>
<td>Italian 0–10 T</td>
</tr>
<tr>
<td>$X_7$</td>
<td>F T F F None $ T F</td>
<td>Burger 0–10 F</td>
</tr>
<tr>
<td>$X_8$</td>
<td>F F F T Some $$$ T T</td>
<td>Thai 0–10 T</td>
</tr>
<tr>
<td>$X_9$</td>
<td>F T T T Full $ T F</td>
<td>Burger &gt;60 F</td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>T T T T Full $$$ F T</td>
<td>Italian 10–30 F</td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>F F F F None $ F F</td>
<td>Thai 0–10 F</td>
</tr>
<tr>
<td>$X_{12}$</td>
<td>T T T T Full $ F F</td>
<td>Burger 30–60 T</td>
</tr>
</tbody>
</table>

*Alt(ernate), Fri(day), Hun(gry), Pat(rons), Res(ervation), Est(imated waiting time)
Decision trees are one possible representation for hypotheses, e.g.:
Expressiveness

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

$$
\begin{array}{ccc}
A & B & A \ xor \ B \\
F & F & F \\
F & T & T \\
T & F & T \\
T & T & F \\
\end{array}
$$

Trivially, there is a consistent decision tree for any training set with one path to a leaf for each example – but it does probably not generalize to new examples

We prefer to find more **compact** decision trees
Hypothesis spaces

How many distinct decision trees are there with \( n \) Boolean attributes?

\[ \begin{align*}
\text{number of Boolean functions} &= \text{number of distinct truth tables with } 2^n \text{ rows} \\
&= 2^{2^n} \text{ distinct decision trees}
\end{align*} \]

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees
Decision tree learning

Aim: find a small tree consistent with the training examples

Idea: (recursively) choose “most significant” attribute as root of (sub)tree

function \text{DTL}(\text{examples}, \text{attributes}, \text{parent-exs}) returns a decision tree

if \text{examples} is empty then return \text{Plurality-Value}(\text{parent-exs})
else if all \text{examples} have the same classification then return the classification
else if \text{attributes} is empty then return \text{Plurality-Value}(\text{examples})
else
    \begin{align*}
    & A \leftarrow \text{arg max}_{a \in \text{attributes}} \text{Importance}(a, \text{examples}) \\
    & \text{tree} \leftarrow \text{a new decision tree with root test } A \\
    & \text{for each value } v_i \text{ of } A \text{ do} \\
    & \quad \text{exs} \leftarrow \{e \in \text{examples} \text{ such that } e[A] = v_i\} \\
    & \quad \text{subtree} \leftarrow \text{DTL(exs, attributes−A, examples)} \\
    & \quad \text{add a branch to } \text{tree} \text{ with label } (A = v_i) \text{ and subtree } \text{subtree} \\
    & \text{return } \text{tree}
    \end{align*}
Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”

Patrons? is a better choice—it gives information about the classification
Information

Information answers questions

The more clueless I am about the answer initially, the more information is contained in the answer

Scale: 1 bit = answer to a Boolean question with prior \( \langle 0.5, 0.5 \rangle \)

The information in an answer when prior is \( V = \langle P_1, \ldots, P_n \rangle \) is

\[
H(V) = \sum_{k=1}^{n} P_k \log_2 \frac{1}{P_k} \\
= -\sum_{i=1}^{n} P_k \log_2 P_k
\]

(this is called the entropy of \( V \))
Suppose we have $p$ positive and $n$ negative examples at the root

$\implies$ we need $H(\langle p/(p+n), \ n/(p+n) \rangle)$ bits to classify a new example

E.g., for our example with 12 restaurants, $p = n = 6$ so we need 1 bit

An attribute splits the examples $E$ into subsets $E_i$, each of which (we hope) needs less information to complete the classification

Let $E_i$ have $p_i$ positive and $n_i$ negative examples

$\implies$ we need $H(\langle p_i/(p_i+n_i), \ n_i/(p_i+n_i) \rangle)$ bits to classify a new example

The expected number of bits per example over all branches is

$$\sum_i \frac{p_i + n_i}{p + n} H(\langle p_i/(p_i+n_i), \ n_i/(p_i+n_i) \rangle)$$

For Patrons?, this is 0.459 bits, for Type this is (still) 1 bit

$\implies$ choose the attribute that minimizes the remaining information needed
Example contd.

Decision tree learned from the 12 examples:

Substantially simpler than the “true” tree
– a more complex hypothesis isn’t justified by that small amount of data
Performance measurement

How do we know that $h \approx f$?

1) Use theorems of computational/statistical learning theory

2) Try $h$ on a new test set of examples
   (use same distribution over example space as training set)

Learning curve = % correct on test set as a function of training set size
Performance measurement contd.

Learning curve depends on

- **realizable** (can express target function) vs. non-realizable
  
  non-realizability can be due to missing attributes or restricted hypothesis class

- redundant expressiveness (e.g., loads of irrelevant attributes)
Learning is needed for unknown environments, or for lazy designers

Learning agent = performance element + learning element

Learning method depends on type of performance element, available feedback, type of component to be improved, and its representation

For supervised learning, the aim is to find a simple hypothesis that is approximately consistent with training examples

Decision tree learning is using information gain, or entropy

Learning performance = prediction accuracy measured on test set
  – the test set should contain new examples, but with the same distribution