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.

Learning element

Design of learning element is dictated by:
- what type of performance element is used
- which functional component is to be learned
- how that functional component is represented
- what kind of feedback is available

Example scenarios:

<table>
<thead>
<tr>
<th>Performance element</th>
<th>Component</th>
<th>Representation</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha-beta search</td>
<td>Eval. fn.</td>
<td>Weighted linear function</td>
<td>Win/loss</td>
</tr>
<tr>
<td>Logical agent</td>
<td>Transition model</td>
<td>Successor-state axioms</td>
<td>Outcome</td>
</tr>
<tr>
<td>Utility-based agent</td>
<td>Transition model</td>
<td>Dynamic Bayes net</td>
<td>Outcome</td>
</tr>
<tr>
<td>Simple reflex agent</td>
<td>Percept-action fn</td>
<td>Neural net</td>
<td>Correct action</td>
</tr>
</tbody>
</table>

Supervised learning: correct answers for each instance
Reinforcement learning: occasional rewards

Inductive learning (a.k.a. Science)

Simplest form: learn a function from examples (tabula rasa)

\[ f \text{ is the target function} \]

An example is a pair \( x, f(x), \) e.g., \( \frac{O}{O} \frac{X}{X} \), +1

Problem: find a(n) 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 examples are given
- Assumes that the agent wants to learn \( f \)—why?)
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:

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</th>
<th>WillWait</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>Alt Bar Fri Hun</td>
<td>F F T</td>
<td>F</td>
</tr>
<tr>
<td>X₂</td>
<td>T F F T Full</td>
<td>$ $</td>
<td>F</td>
</tr>
<tr>
<td>X₃</td>
<td>F T F T Some</td>
<td>$ F $</td>
<td>F</td>
</tr>
<tr>
<td>X₄</td>
<td>T F T T Full</td>
<td>$ $</td>
<td>F</td>
</tr>
<tr>
<td>X₅</td>
<td>T F F T Full</td>
<td>$ $</td>
<td>F</td>
</tr>
<tr>
<td>X₆</td>
<td>F T F T Some</td>
<td>$ $</td>
<td>T</td>
</tr>
<tr>
<td>X₇</td>
<td>F F F F None</td>
<td>$ $</td>
<td>T</td>
</tr>
<tr>
<td>X₈</td>
<td>F F F F Some</td>
<td>$ $</td>
<td>T</td>
</tr>
<tr>
<td>X₉</td>
<td>T T T T Full</td>
<td>$ $</td>
<td>$</td>
</tr>
<tr>
<td>X₁₀</td>
<td>T T T T Full</td>
<td>$ $</td>
<td>$</td>
</tr>
<tr>
<td>X₁₁</td>
<td>F F F F None</td>
<td>$ $</td>
<td>F</td>
</tr>
<tr>
<td>X₁₂</td>
<td>T T T T Full</td>
<td>$ $</td>
<td>F</td>
</tr>
</tbody>
</table>

Classification of examples is positive (T) or negative (F)

Decision trees

One possible representation for hypotheses
E.g., here is the “true” tree for deciding whether to wait:

Expressiveness

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

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

= number of distinct truth tables with 2ⁿ rows
Hypothesis spaces

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

\[ \text{number of Boolean functions} \]

\[ \text{number of distinct truth tables with } 2^n \text{ rows} = 2^{2^n} \]

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

How many purely conjunctive hypotheses (e.g., \( \text{Hungry} \land \neg \text{Rain} \))??

Each attribute can be in (positive), in (negative), or out \( \Rightarrow 3^n \) distinct conjunctive hypotheses

More expressive hypothesis space

– increases chance that target function can be expressed

– increases number of hypotheses consistent w/ training set

\( \Rightarrow \) may get worse predictions

Decision tree learning

Aim: find a small tree consistent with the training examples

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

\[
\text{function } \text{DTL}(\text{examples}, \text{attributes}, \text{default}) \text{ returns a decision tree}
\]

\[
\text{if } \text{examples} \text{ is empty then return default}
\]

\[
\text{else if all } \text{examples} \text{ have the same classification then return the classification}
\]

\[
\text{else if attributes is empty then return } \text{Mode}(\text{examples})
\]

\[
\text{else best } \leftarrow \text{Choose-Attribute}(\text{attributes, examples})
\]

\[
\text{tree } \leftarrow \text{a new decision tree with root test best}
\]

\[
\text{for each value } v_i \text{ of best do}
\]

\[
\text{examples}_i \leftarrow \{ \text{elements of } \text{examples} \text{ with } \text{best} = v_i \}
\]

\[
\text{subtree } \leftarrow \text{DTL}(\text{examples}_i, \text{attributes} - \text{best}, \text{Mode}(\text{examples}))
\]

\[
\text{add a branch to } \text{tree} \text{ with label } v_i \text{ and subtree } \text{subtree}
\]

return \text{tree}

Choosing an attribute

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

\[ \text{Patrons? is a better choice—gives information about the classification} \]
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 Boolean question with prior (0.5, 0.5)

Information in an answer when prior is \((P_1, \ldots, P_n)\) is

\[ H((P_1, \ldots, P_n)) = \sum_{i=1}^{n} -P_i \log_2 P_i \]

(also called entropy of the prior)

Example contd.

Decision tree learned from the 12 examples:

```
<table>
<thead>
<tr>
<th>Patrons?</th>
<th>None</th>
<th>Some</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hungry?</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type?</td>
<td>French</td>
<td>Italian</td>
<td>Thai</td>
</tr>
<tr>
<td>Fri/Sat?</td>
<td>None</td>
<td>Some</td>
<td>Full</td>
</tr>
</tbody>
</table>
```

Substantially simpler than “true” tree—a more complex hypothesis isn’t justified by small amount of data

Performance measurement

How do we know that \(h \approx f\)? (Hume’s Problem of Induction)

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 (e.g., thresholded linear function)
  - redundant expressiveness (e.g., loads of irrelevant attributes)

Summary

Learning needed for unknown environments, 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 using information gain

Learning performance = prediction accuracy measured on test set