LEARNING FROM OBSERVATIONS

CHAPTER 18, SECTIONS 1–3
Outline

- Learning agents
- 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
Inductive learning (a.k.a. Science)

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

$f$ is the target function

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

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>O</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+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 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:

![Diagram of curve fitting](image)
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:

\[ f(x) \]
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:

\[ f(x) \]
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:

\[ f(x) \]
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:

$Ockham'sazor$: 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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alt</td>
<td>Bar</td>
</tr>
<tr>
<td>X₁</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X₂</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X₃</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>X₄</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X₅</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>X₆</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>X₇</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>X₈</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>X₉</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>X₁₀</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>X₁₁</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>X₁₂</td>
<td>T</td>
<td>T</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:

```
Patrons?
None F
Some T
Full

WaitEstimate?
>60 F
30-60 T
10-30

Alternate?
F
No
T
Yes

Hungry?
0-10 T

Reservation?
Fri/Sat?

Bar?
T
F
No
Yes

Alternate?
T
Yes

Raining?
F
T
No
Yes
```
Expressiveness

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A xor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

Trivially, there is a consistent decision tree for any training set w/ one path to leaf for each example (unless $f$ nondeterministic in $x$) but it probably won’t generalize to new examples

Prefer to find more **compact** decision trees
Hypothesis spaces

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

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

= number of Boolean functions
Hypothesis spaces

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

= number of Boolean functions
= number of distinct truth tables with \( 2^n \) rows
Hypothesis spaces

How many distinct decision trees with $n$ Boolean attributes??

= number of Boolean functions
= number of distinct truth tables with $2^n$ rows = $2^{2n}$
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
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

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

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

\[
\begin{align*}
\text{if } \text{examples} & \text{ is empty then return default} \\
\text{else if all } \text{examples} & \text{ have the same classification then return the classification} \\
\text{else if attributes} & \text{ is empty then return } \text{MODE(\text{examples})} \\
\text{else}
\end{align*}
\]

\[
\begin{align*}
\text{best} & \leftarrow \text{CHOOSE-ATTRIBUTE(\text{attributes, examples})} \\
\text{tree} & \leftarrow \text{a new decision tree with root test best} \\
\text{for each value } v_i & \text{ of best do} \\
\text{examples}_i & \leftarrow \{\text{elements of examples with best } = v_i\} \\
\text{subtree} & \leftarrow \text{DTL(\text{examples}_i, attributes – best, MODE(\text{examples})}} \\
& \text{add a branch to tree with label } v_i \text{ and subtree subtree} \\
\text{return } \text{tree}
\end{align*}
\]
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—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 $\langle 0.5, 0.5 \rangle$

Information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is

$$H(\langle P_1, \ldots, P_n \rangle) = \sum_{i=1}^{n} - P_i \log_2 P_i$$

(also called entropy of the prior)
Suppose we have $p$ positive and $n$ negative examples at the root
$\Rightarrow H(\langle p/(p+n), n/(p+n) \rangle)$ bits needed to classify a new example
E.g., for 12 restaurant examples, $p = n = 6$ 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
$\Rightarrow H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i) \rangle)$ bits needed to classify a new example
$\Rightarrow$ 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
$\Rightarrow$ choose the attribute that minimizes the remaining information needed
Example contd.

Decision tree learned from the 12 examples:

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