Learning from Observations

Chapter 18
Section 1 – 3

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

Learning agents

![Diagram of learning agents]
Learning element

• Design of a learning element is affected by
  – Which components of the performance element are to be learned
  – What feedback is available to learn these components
  – What representation is used for the components

• Type of feedback:
  – Supervised learning: correct answers for each example
  – Unsupervised learning: correct answers not given
  – Reinforcement learning: occasional rewards

Inductive learning

• Simplest form: learn a function from examples
  \( f \) is the target function
  An example is a pair \((x, f(x))\)
  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:

\[ f(x) \]
\[ x \]
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:

\[
\begin{align*}
\text{\( f(x) \)} & \quad \text{x} \\
\end{align*}
\]
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:

\[ \text{E.g., curve fitting:} \]

\[ \text{Inductive learning method} \]

• 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:

\[ \text{E.g., curve fitting:} \]

• Ockham’s razor: prefer the simplest hypothesis consistent with data

Learning decision trees

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)
Attribute-based representations

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

<table>
<thead>
<tr>
<th>Example</th>
<th>Full?</th>
<th>Wait?</th>
<th>Type?</th>
<th>Type</th>
<th>Time?</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_3</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_4</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_5</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_6</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_7</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_9</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_10</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>0-10</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:

```
Decision trees

- Another possible representation for hypotheses
```

<table>
<thead>
<tr>
<th>Example</th>
<th>Full?</th>
<th>Wait?</th>
<th>Type?</th>
<th>Type</th>
<th>Time?</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_3</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_4</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_5</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_6</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_7</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_9</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
<tr>
<td>x_10</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>0-10</td>
</tr>
</tbody>
</table>

```Decision trees

- Another possible representation for hypotheses
```
Expressiveness

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

```
A  B  A xor B
F  F  F
F  T  T
T  F  T
T  T  F
```

- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless non-deterministic 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?
= number of Boolean functions
= number of distinct truth tables with $2^n$ 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., Hungry $\land \neg$ 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 with 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

Choosing an attribute

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

Using information theory

- To implement Choose-Attribute in the DTL algorithm
- Information Content (Entropy):
  \[ I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i) \]
- For a training set containing \( p \) positive examples and \( n \) negative examples:
  \[ I(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} -\frac{n}{p+n} \log_2 \frac{n}{p+n} \]
Information gain

- A chosen attribute \( A \) divides the training set \( E \) into subsets \( E_1, \ldots, E_v \) according to their values for \( A \), where \( A \) has \( v \) distinct values.

\[
\text{remainder}(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} \cdot \log \left( \frac{p_i + n_i}{p+n} \right)
\]

- Information Gain (IG) or reduction in entropy from the attribute test:

\[
\text{IG}(A) = I \left( \frac{p}{p+n}, \frac{n}{p+n} \right) - \text{remainder}(A)
\]

- Choose the attribute with the largest IG

Computing Entropy and IG

\[
I \left( \frac{p}{p+n}, \frac{n}{p+n} \right) = - \frac{p}{p+n} \log \left( \frac{p}{p+n} \right) - \frac{n}{p+n} \log \left( \frac{n}{p+n} \right)
\]

\[
\text{remainder} \left( A \right) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} \cdot \log \left( \frac{p_i + n_i}{p+n} \right)
\]

\[
\text{IG} \left( A \right) = I \left( \frac{p}{p+n}, \frac{n}{p+n} \right) - \text{remainder} \left( A \right)
\]

Information gain

For the training set, \( p = n = 6 \), \( I(6/12, 6/12) = 1 \) bit

Consider the attributes \( \text{Patrons} \) and \( \text{Type} \) (and others too):

\[
\text{IG(\text{Patrons})} = 1 - \frac{2}{12} \cdot \frac{1}{2} + \frac{4}{12} \cdot \frac{1}{2} + \frac{4}{12} \cdot \frac{1}{2} + \frac{4}{12} \cdot \frac{1}{2} + \frac{4}{12} \cdot \frac{1}{2} = 0.541 \text{ bits}
\]

\[
\text{IG(\text{Type})} = 1 - \frac{2}{12} \cdot \frac{1}{2} + \frac{2}{12} \cdot \frac{1}{2} + \frac{2}{12} \cdot \frac{1}{2} + \frac{4}{12} \cdot \frac{1}{2} + \frac{4}{12} \cdot \frac{1}{2} + \frac{4}{12} \cdot \frac{1}{2} = 0 \text{ bits}
\]

\( \text{Patrons} \) has the highest IG of all attributes and so is chosen by the DTL algorithm as the root.
Example contd.

• Decision tree learned from the 12 examples:

• Substantially simpler than the first tree—a more complex hypothesis isn’t justified by small amount of data

Performance measurement

• How do we know that $h = f$?
  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

Why Learning Works

• There is a theoretic foundation: Computational Learning Theory.
• The underlying principle: Any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong: it must be Probably Approximately Correct (PAC).
• The Stationarity Assumption: The training and test sets are drawn randomly from the same population of examples using the same probability distribution.
Summary

• Learning needed for unknown environments, lazy designers
• Learning agent = performance element + learning element
• For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples
• Decision tree learning using information gain
• Learning performance = prediction accuracy measured on test set