Learning from Observations

Chapter 18
Section 1 – 3
Outline

• Learning agents
• Inductive learning
• Decision tree learning
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
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 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:
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:
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:

\[ f(x) \]

- 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 minutes)
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>Attrs</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₅</td>
<td>Alt: T, Bar: F, Fri: T, Hun: F</td>
<td>Full, $$$, F, T</td>
</tr>
<tr>
<td>X₁₀</td>
<td>Alt: T, Bar: T, Fri: T, Hun: T</td>
<td>Full, $$$, F, T</td>
</tr>
<tr>
<td>X₁₁</td>
<td>Alt: F, Bar: F, Fri: F, Hun: T</td>
<td>None, $, F, 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:

<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 with 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?

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

• 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?
= 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., \( \text{Hungry} \land \neg \text{Rain} \))?
• 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 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 tree or subtree

```
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 \( v_i \) of best do
            examples_i ← \{ elements of examples with best = \( v_i \) \}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label \( v_i \) and subtree subtree
        return tree
```
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
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.

$$\text{remainder}(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

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

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

- Choose the attribute with the largest $IG$
Information gain

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

Consider the attributes *Patrons* and *Type* (and others too):

\[
IG(Patrons) = 1 - \left[ \frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .0541 \text{ bits}
\]

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

Compute \( IG(Hungry), IG(Fri/Sat), IG(Raining), \) ….etc.

*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 “true” tree---a more complex hypothesis isn’t justified by a 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
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