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
## 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. In.</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)

\( f \) is the target function

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

\[
\begin{array}{c|c|c|c}
O & O & X \\
X & X & +1 \\
\end{array}
\]

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
\( (h \text{ is consistent if it agrees with } f \text{ on all examples}) \)

E.g., curve fitting:

\[
f(x)
\]
Inductive learning method

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

E.g., curve fitting:

\[ f(x) \]

\[ x \]

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>T F F T</td>
<td>Some</td>
<td>$$$</td>
</tr>
<tr>
<td>X2</td>
<td>T F F T</td>
<td>Full</td>
<td>$</td>
</tr>
<tr>
<td>X3</td>
<td>F T F F</td>
<td>Some</td>
<td>$</td>
</tr>
<tr>
<td>X4</td>
<td>T F T T</td>
<td>Full</td>
<td>$$$</td>
</tr>
<tr>
<td>X5</td>
<td>T F T F</td>
<td>Full</td>
<td>$$$</td>
</tr>
<tr>
<td>X6</td>
<td>T F T F</td>
<td>Some</td>
<td>$</td>
</tr>
<tr>
<td>X7</td>
<td>F T F F</td>
<td>None</td>
<td>$</td>
</tr>
<tr>
<td>X8</td>
<td>F F T F</td>
<td>Some</td>
<td>$$$</td>
</tr>
<tr>
<td>X9</td>
<td>F T T F</td>
<td>Full</td>
<td>$</td>
</tr>
<tr>
<td>X10</td>
<td>T T T T</td>
<td>Full</td>
<td>$$$</td>
</tr>
<tr>
<td>X11</td>
<td>F F F T</td>
<td>None</td>
<td>$</td>
</tr>
<tr>
<td>X12</td>
<td>T T T T</td>
<td>Full</td>
<td>$</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 boolean function of the input attributes.
E.g., for Boolean attributes, truth table row → 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 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

= 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
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
⇒ $3^n$ distinct conjunctive hypotheses

More expressive hypothesis space
– increases chance that target function can be expressed
– increases number of hypotheses consistent w/ training set
⇒ 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(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$_i$))
        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”

<table>
<thead>
<tr>
<th>Patrons?</th>
<th>Type?</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>French</td>
</tr>
<tr>
<td>Some</td>
<td>Italian</td>
</tr>
<tr>
<td>Full</td>
<td>Thai</td>
</tr>
<tr>
<td></td>
<td>Burger</td>
</tr>
</tbody>
</table>

Information Theory

- Consider communicating two messages (T and F) between two parties
- Bits are used to measure message size
- If \( P(T) = 1 \) and \( P(F) = 0 \), how many bits are needed?
- If \( P(T) = .5 \) and \( P(F) = .5 \), how many bits are needed?
- Information: \( I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i) \)
- \( I(1, 0) = 0 \) bit
- \( I(0.5, 0.5) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1 \) bit

Information Theory

- Consider communicating two messages (T and F) between two parties
- Bits are used to measure message size
- If \( P(T) = 1 \) and \( P(F) = 0 \), how many bits are needed?
- If \( P(T) = .5 \) and \( P(F) = .5 \), how many bits are needed?
- Information: \( I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i) \)
- \( I(1, 0) = 0 \) bit
- \( I(0.5, 0.5) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1 \) bit
- \( I \) measures the information content for communication (or uncertainty in what is already known)
- The more one knows, the less to be communicated, the smaller is \( I \)
- The less one knows, the more to be communicated, the larger is \( I \)
Using Information Theory

- \((P(\text{pos}), P(\text{neg}))\): probabilities of positive ("message T") and negative ("message F")
- Attribute color: black (1.0), white (0.1)
- Attribute size: large (0.5), small (0.5)

Before adding an attribute

- How much uncertainty/confusion before adding an attribute (e.g., color)?
  - \(p = \) number of positive examples, \(n = \) number of negative examples
  - Estimating probabilities: \(P(\text{pos}) = \frac{p}{p+n}, P(\text{neg}) = \frac{n}{p+n}\)
  - \(\text{Before}() = I(P(\text{pos}), P(\text{neg}))\)

After adding an attribute

- How much uncertainty/confusion after adding an attribute (e.g., color)?
  - \(p_i = \) number of positive examples for value \(i\) (e.g., black), \(n_i = \) number of negative ones
  - Estimating probabilities for value \(i\): \(P_i(\text{pos}) = \frac{n_i}{p_i+n_i}, P_i(\text{neg}) = \frac{n}{p_i+n_i}\)
  - Uncertainty from value \(i\): \(I(P_i(\text{pos}), P_i(\text{neg}))\)
  - But we have \(v\) values for attribute \(A\) (e.g., 2 for color)
  - How do we combine the uncertainty from the different attribute values?

\[
\text{Remainder}(A) = \text{After}(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I(P_i(\text{pos}), P_i(\text{neg})) \quad [\text{expected uncertainty}]
\]
Choosing an Attribute

◊ “Information Gain” (reduction in uncertainty)

• $\text{Gain}(A) = \text{Before()} - \text{After}(A)$
• Why $\text{Before()} - \text{After}(A)$, not $\text{After}(A) - \text{Before}()$?
• $\text{Before}()$ should have more uncertainty
• Choose attribute $A$ with the largest $\text{Gain}(A)$

Example contd.

Decision tree learned from the 12 examples:

![Decision Tree Diagram]

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

Performance measurement

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

How do we know that \( h \approx f \)?

How about measuring the accuracy of \( h \) on the examples that were used to learn \( h \)?)

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
   - divide into two disjoint subsets: training and test sets
   - prediction accuracy: accuracy on the (unseen) test set

Learning curve = \% correct on test set as a function of training set size

- realizable (can express target function) vs. non-realizable

  non-realizability can be due to:
  - missing attributes
  - and/or restricted hypothesis class (e.g., thresholded linear function)

  redundant expressiveness (e.g., loads of irrelevant attributes)

\[ \begin{array}{c}
\text{% correct} \\
\text{# of examples}
\end{array} \]

\[ \begin{array}{c}
\text{realizable} \\
\text{redundant} \\
\text{nonrealizable}
\end{array} \]
Irrelevant Attributes

- Consider adding the attribute: Date (month and day)
- How can it affect the learned tree?

Significance Test

- “Null hypothesis” (in statistics): attribute is irrelevant (gain is not significant)
- “Alternative hypothesis”: attribute is relevant
- Calculating the deviation
  - expected \( \bar{p}_i = p \times \frac{E_i + n_i}{p + n} \)
  - expected \( \bar{n}_i = n \times \frac{E_i + n_i}{p + n} \)
  - Deviation (from expected):
    \[
    D = \sum_{i=1}^{v} \frac{(p_i - \bar{p}_i)^2}{\bar{p}_i} + \frac{(n_i - \bar{n}_i)^2}{\bar{n}_i}
    \]
  - \( D \) is \( \chi^2 \) (chi-squared) distributed with \( v - 1 \) degrees of freedom
  - \( \chi^2 \) Test in statistics
- With a confidence level (e.g. 95%), if \( D > \chi^2 \) value, attribute is relevant (Null hypothesis is rejected)

Overfitting

- More attributes \( \Rightarrow \) larger hypothesis space
- Larger hypothesis space can lead to more hypotheses that represent meaningless "regularity/patterns"
- Overfitting: high accuracy on training set, but low accuracy on test set—low prediction accuracy
- Select the attribute with the largest information gain
  - however, is the gain significant?
  - (statistical) significance test
- Pruning
  - do not include the attribute if information gain is not statistically significant
  - potentially, less than 100% accurate on the training set, why?
  - however, improved prediction accuracy on the test set

Additional Issues

- Missing attribute values.
- Gain() biases to attributes with more values.
- Continuous-valued (numeric) attributes have infinite number of values.
Learning as search

- What is the state space in learning decision trees?
- State-space formulation

- State: a decision tree
- Initial state: an empty decision tree
- Action: add an attribute to the tree
- Goal test: all examples in each leaf have the same classification

What kind of search is DTL?

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