Designing a Learning Agent

- What type of performance element?
- Which functional component to be learned?
- How that functional component is represented
- What type of feedback is available?

<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 fn.</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 network</td>
<td>Correct action</td>
</tr>
</tbody>
</table>
Types of Learning

- **Supervised learning**
  - Give correct answer for each instance
  - Learn a function from examples of inputs/outputs

- **Unsupervised learning**
  - No correct answers known
  - Can learn patterns in the input
  - Can’t learn what to do w/o feedback (don’t know whether states are desirable/undesirable)
  - But you can learn a probability distribution

- **Reinforcement learning**
  - Sometimes you get a reward, sometimes you get punished
  - Example: a waiter will learn to prefer certain behaviors because he gets bigger tips
  - Typically, trying to learn how the environment works
Induction

• Example: curve fitting

\[ f(x) \]
Induction

• Example: curve fitting

\[ f(x) \]

\[ x \]

\[ h1 \]
Induction

- Example: curve fitting

![Graph showing curve fitting]

\[ f(x) \]

\[ x \]

\[ h2 \]
Induction

- Example: curve fitting

H is consistent if it agrees with all examples
Induction

- Example: curve fitting

Given multiple consistent hypotheses, pick the simplest one

(OCHAM’S RAZOR)
Learning Decision Trees

• A simple technique whereby the computer learns to make decisions that emulate human decision-making

• Can also be used to learn to classify
  – A decision can be thought of as a classification problem

• An object or situation is described as a set of attributes
  – Attributes can have discrete or continuous values

• Predict an outcome (decision or classification)
  – Can be discrete or continuous
  – We assume positive (true) or negative (false)
Eat at a restaurant?

• Attributes:
  – Alternate: suitable alternate restaurant nearby (y/n)
  – Bar: A bar to wait in (y/n)
  – Fri/Sat: it’s a Friday or Saturday (y/n)
  – Hungry: y/n
  – Price: price range ($, $$, $$$)
  – Raining: y/n
  – Reservation: we made a reservation (y/n)
  – Type: french, italian, thai, burger
  – WaitEstimate: 0-10, 10-30, 30-60, >60
  – Patrons: none, some, full
<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Will/Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X_2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
</tr>
<tr>
<td>X_3</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X_4</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
</tr>
<tr>
<td>X_5</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X_6</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X_7</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X_8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X_9</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X_10</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
</tr>
<tr>
<td>X_11</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X_12</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
</tr>
</tbody>
</table>
Supervised Learning

• Training set
• Test set
Pos: 1 3 4 6 8 12
Neg: 2 5 7 9 10 11

Patrons

None

Pos: nil
Neg: 7 11

NO

1

3

4

6

8

12

Full

Pos: 4 12
Neg: 2 5 9 10

Hungry

Yes

4

12

No

Pos: nil
Neg: 5 9

NO

5

9

2

10
• Learned from the 12 examples
• Why doesn’t it look like the previous tree?
  – Not enough examples
  – No reason to use rain or reservations
  – Hasn’t seen all cases
• Learning is only as good as your training data
Which attribute to choose?

• The one that gives you the most information (aka the most diagnostic)
• Information theory
  – Answers the question: how much information does something contain?
  – Ask a question
  – Answer is information
  – Amount of information depends on how much you already knew
• Example: flipping a coin
  – If you don’t know that coin flipping is random: 1 bit of information is gained
  – If you do know: 0 bits of information is gained
If there are $n$ possible answers, $v_1 \ldots v_n$ and $v_i$ has probability $P(v_i)$ of being the right answer, then the amount of information is:

$$I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i)$$

- Example: coin toss
• For a training set:
  \( p = \# \) of positive examples
  \( n = \# \) of negative examples

  \[
  I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}
  \]

  Probability of a positive example  Probability of a negative example

• For our restaurant behavior
  – \( p = n = 6 \)
  – \( I() = 1 \)
  – Would not be 1 if training set weren’t 50/50 yes/no, but the point is to arrange attributes to increase information gain
Measuring attributes

• Information gain is a function of how much more information you need after applying an attribute
  – If I use attribute A next, how much more information will I need?
  – Use this to compare attributes

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

Instances of the attribute
Positive examples for this answer
Negative examples for this answer

Total answers
Different answers
Examples classified by A
\[
\text{Remainder(type)} = \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) = 1 \text{ bit}
\]
Patrons

None
- Pos: nil
- Neg: 7 11

Some
- Pos: 1 3 6 8
- Neg: nil

Full
- Pos: 4 12
- Neg: 2 5 9 10

Remainder(patrons) = \frac{2}{12} I\left(\frac{0}{2}, \frac{2}{2}\right) + \frac{4}{12} I\left(\frac{4}{4}, \frac{0}{4}\right) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) ≈ 0.459 \text{ bit}
• Not done yet
• Need to measure information \textbf{gained} by an attribute

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

• Pick the biggest
• Example:
  – \text{Gain}(\text{type}) = I(\frac{1}{2}, \frac{1}{2}) - \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}
  
  – \text{Gain}(\text{patrons}) = I(\frac{1}{2}, \frac{1}{2}) - \left(\frac{2}{12} I\left(\frac{0}{2}, \frac{2}{2}\right) + \frac{4}{12} I\left(\frac{4}{4}, \frac{2}{4}\right) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right)\right)
    \approx 0.541 \text{ bits}
Patrons

Patrons=full, hungry=yes

Patrons=full, hungry=no

gain(hungry) = \( I\left(\frac{2}{6}, \frac{4}{6}\right) - \left[ \frac{2}{6} I\left(\frac{0}{2}, \frac{2}{2}\right) + \frac{4}{6} I\left(\frac{2}{4}, \frac{2}{4}\right) \right] \)

= 0.9182958 – [ 0 + (4/6)(1)]

≈ 0.251 bits
Decision-tree-learning (examples, attributes, default)

IF examples is empty THEN RETURN default
ELSE IF all examples have same classification THEN RETURN classification
ELSE IF attributes is empty RETURN majority-value(examples)
ELSE
  best = choose(attributes, example)
  tree = new decision tree with best as root
  m = majority-value(examples)
  FOREACH answer $v_i$ of best DO
    examples$_i$ = {elements of examples with best=$v_i$}
    subtree$_i$ = decision-tree-learning(examples$_i$, attributes-{best}, m)
    add a branch to tree based on $v_i$ and subtree$_i$
  RETURN tree
How many hypotheses?

• How many distinct trees?
  – N attributes
    = # of boolean functions
    = # of distinct truth tables with $2^n$ rows
    = $2^2^n$
  – With 6 attributes: > 18 quintillion possible trees
How do we assess?

• How do we know $h \approx f$?
• A learning algorithm is good if it produces hypotheses that do a good job of predicting decisions/classifications from unseen examples

1. Collect a large set of examples (with answers)
2. Divide into training set and test set
3. Use training set to produce hypothesis $h$
4. Apply $h$ to test set (w/o answers)
   – Measure % examples that are correctly classified
5. Repeat 2-4 for different sizes of training sets, randomly selecting examples for training and test
   – Vary size of training set $m$
   – Vary which $m$ examples are training
• Plot a learning curve
  – % correct on test set, as a function of training set size

![Graph showing learning curve]

• As training set grows, prediction quality should increase
  – Called a “happy graph”
  – There is a pattern in the data AND the algorithm is picking it up!
Noise

• Suppose 2 or more examples with same description (Same assignment of attributes) have different answers
• Examples: on two identical* situations, I do two different things
• You can’t have a consistent hypothesis (it must contradict at least one example)
• Report majority classification or report probability
Overfitting

• Learn a hypothesis that is consistent using irrelevant attributes
  – Coincidental circumstances result in spurious distinctions among examples
  – Why does this happen?
    • You gave a bunch of attributes because you didn’t know what would be important
    • If you knew which attributes were important, you might not have had to do learning in the first place

• Example: Day, month, or color of die in predicting a die roll
  – As long as no two examples are identical, we can find an exact hypothesis
  – Should be random 1-6, but if I roll once every day and each day results in a different number, the learning algorithm will conclude that day determines the roll

• Applies to all learning algorithms