Lecture 22: Learning 1

Victor R. Lesser
CMPSCI 683
Fall 2010

Today’s Lecture

• The structure of a learning agent
• Basic problems: bias, Ockham’s razor, expressiveness
• Decision-tree algorithms for Classification

Commonsense Definition

Learning is change within a system that improves its performance

This admits a lot of different behaviors, but identifies the basic preconditions of learning:
- Learning systems must be capable of change
- Learning systems must do something differently as a result of the change

Why Should Systems Learn?

• Learning can simplify the complexity of problem solving.
  - Replace procedural/declarative knowledge, inferencing, and search with learned functions and policies
• Learning increases efficiency, robustness, survivability, and autonomy of a system.
  - Key to operating in “open” environments
  - Re-evaluate key assumptions in light of what is happening
• A learning program can become better than its teacher.
  
A viable alternative to problem solving.
Types of Learned Knowledge*

- A direct mapping from conditions on the current state to actions.
- Weighting of parameters of multi-attribute decision process.
- A means to infer relevant properties of the world from the percept sequence.
- Information about the way the world evolves.
  - Allow prediction of future events

How Does this Relate to Systems We Have Studied?

Types of Learned Knowledge cont.

- Information about the results of possible actions the agent can take.
- Utility information indicating the desirability of world states.
- Action-value information indicating the desirability of particular actions in particular states.
- Goals that describe classes of states whose achievement maximizes the agent’s utility.

How Does this Relate to Systems We Have Studied?

Examples from My Lab

- Meta-level Control
  - Learning policy for balancing thinking/coordinating and acting in a sophisticated agent
- Agent Plans – SRTA
  - Learned other used agent plans; avoided planning overhead
- Agent Behavior Statistics – SRTA
  - Learned statistical distribution of performance of agent actions than used in planning and scheduling of agent activities
- Information Gathering -- BRG
  - Learned text extraction strategy
- Agent Coordination
  - Learned new coordination rules
  - Learned situation specific context for applying coordination rules
  - Learning routing policies in a peer-to-peer IR
  - Learning distributed task allocation policy
- Blackboard control
  - Learned tactical control for when to invoke specific Kbs
- Model Acquisition for Sound Understanding – IPUS
  - Learned models for characterizing never before heard sounds

Characterizing Learning Systems

- What changes as a result of learning?
- How does the system find out change is needed?
- How does the system localize the problem to find out what changes are necessary?
- What is the mechanism of change?
Available Feedback

- **Supervised learning**
  - Is told by a “teacher” what action is best in a specific situation
  - Learning to brake

- **Reinforcement Learning**
  - Gets feedback about the consequences of a specific sequence of actions in a certain situation
  - Can also be thought of as supervised learning with a less informative feedback signal.
  - Training a dog

- **Unsupervised Learning**
  - No feedback about actions
  - Learns to predict future precepts given its previous precepts
  - Can’t learn what to do unless it already has a utility function that defines appropriateness of a given situation (built-in feedback signal).
  - Learning traffic patterns

A Model of Learning Agents

- **Critic** — tells learning element how well agent is doing
  - Fixed standard of performance
- **Learning element** — modifies performance element (usually its knowledge) in response to feedback
- **Problem generator** — suggests actions that will lead to new and informative experiences also called exploration
  - Related to decision to acquire information

Design of Learning Element

- **Goals**:
  - Learn better actions that lead to higher long-term utility
  - Speed up performance element
  - Which components of the performance element are to be improved.
  - What representation is used for those components.
  - What feedback is available
  - What prior information is available.
Dimensions of Learning

- The type of training instances
  - The beginning data for the learning task.
- The language used to represent knowledge.
  - Specific training instances must be translated into this representation language.
  - In some programs the training instances are in the same language as the internal knowledge base and this step is unnecessary.
- A set of operations on representations.
  - Typical operations generalize or specialize existing knowledge, combine units of knowledge, or otherwise modify the program’s existing knowledge or the representation of the training instances.

Dimensions of Learning cont.

- The concept space.
  - The operations that define a space of possible knowledge structures that is searched to find the appropriate characterization of the training instances and similar problems.
  - Learning as Search?
- The learning algorithms and heuristics employed to search the concept space.
  - The order of the search and the use of heuristics to guide the search.

Types of Knowledge Representations for Learning

- numerical parameters
- decision trees
- formal grammars
- production rules
- logical theories
- graphs and networks
- frames and schemas
- computer programs (procedural encoding)

Learning Functions

All learning can be seen as learning the representation of a function/mapping

- Choice of representation of a function
  - Trade-off between expressiveness and efficiency
  - Is what you want representable?
  - Is what you want learnable (# of examples, cost of search)?
- Choice of training data
  - Correctly reflects past experiences
  - Correctly predicts future experiences
- How to judge the goodness of the learned function
Some Additional Thoughts

- Importance of Prior Knowledge
  - Prior knowledge (e.g., first principles) can significantly speed up learning process
  - EBL: Explanation-Based Learning
- Learning as a search process
  - Finding the “best” function
- Incremental Process (on-line) vs. off-line

Inductive (Supervised) Learning

Let an example be \((x, f(x))\)

- Give a collection of examples of \(f\), return a function \(h\) that approximates \(f\).
- This function \(h\) is called a hypothesis:
  - Feedback is relation between \(f(x)\) and \(h(x)\)
  - \((x, f(x))\) could only be approximately correct
    - Noise – observation of \(f(x)\) not always accurate
    - Missing components of \(x\) – ambiguity of whether missing component is key to decision (output of \(f(x)\))

Problems

- Many hypotheses \(h\)'s are approximately consistent with the training set
- Curve-fitting ...

- A preference for one hypothesis over another beyond consistency is called Bias

Ockham’s Razor

- “Simple” hypotheses that are consistent with data are preferred

- We want to maximize some metric of consistency and simplicity in the choice of the most appropriate function
Learning Classification Decision Trees

- Restricted representation of logical sentences
  - Boolean functions
- Takes as input a situation described by a set of properties and outputs a "yes/no" decision
  - Tree of property value tests
  - Terminals are decisions
- Not all attributes of situation need to be used
- Decision tree as a performance element

Learn, based on conditions of the situation, whether to wait at a restaurant for a table.

Decision trees

- A (classification) decision tree takes as input a situation described by a set of attributes and returns a "decision."
  - Reaches its decision by performing a sequence of incremental tests
  - Each internal node corresponds to a test of one of the attributes of the situation
- Can express any boolean function of the input attributes.
- How to choose between equally consistent trees

Expressions of Decision Tree

- Any Boolean function can be written as a decision tree
  - $\forall r \text{ Patrons}(r, \text{Full}) \land \text{WaitEstimate}(r, 10-30) \land \text{Hungry}(r, N) \implies \text{WillWait}(r)$
  - Row of truth table path in decision tree
  - $2^n$ rows given $n$ literals, $2^m$ functions

Limits on Expressability

- Cannot use decision tree to represent tests that refer to two or more different objects
  - $\exists r_1, \text{Nearby}(r_1, r) \land \text{Price}(r, p) \land \text{Price}(r_2, p_2) \land \text{Cheaper}(p, p_2)$
- New Boolean attribute: CheaperRestaurantNearby but intractable to add all such attributes
- Some truth tables cannot be compactly represented in decision tree -- analogous to Bayesian Joint Distribution
  - Parity function
    - returns 1 if and only if an even number of inputs are 1
    - exponentially large decision tree will be needed.
  - Majority function
    - which returns 1 if more than half of its inputs are 1
Example: Waiting for a Table

- Alternate restaurant exists
- Bar that you can wait
- Fri/Sat
- Hungry
- Patrons (None, Some, Full)
- Price ($, $$, $$$)
- Raining
- Reservation
- Type (French, Italian, Thai, Burger)
- WaitEstimate (0-10, 10-30, 30-60, >60)

Data available for decision whether to wait for a table

Inducing Decision Trees from Examples

<table>
<thead>
<tr>
<th>Example</th>
<th>Ask</th>
<th>Bar</th>
<th>Fri</th>
<th>Rain</th>
<th>Pat</th>
<th>Price</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Some</td>
<td>$$</td>
<td>No</td>
<td>No</td>
<td>Fri</td>
<td>10-30</td>
</tr>
<tr>
<td>x2</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$$</td>
<td>No</td>
<td>No</td>
<td>Sat</td>
<td>10-30</td>
</tr>
<tr>
<td>x3</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Some</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Fri</td>
<td>30-60</td>
</tr>
<tr>
<td>x4</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Fri</td>
<td>&lt;10</td>
</tr>
<tr>
<td>x5</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
<td>No</td>
<td>Fri</td>
<td>&lt;10</td>
</tr>
<tr>
<td>x6</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
<td>No</td>
<td>Sat</td>
<td>&lt;10</td>
</tr>
<tr>
<td>x7</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
<td>No</td>
<td>Sat</td>
<td>&lt;10</td>
</tr>
<tr>
<td>x8</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$$</td>
<td>No</td>
<td>No</td>
<td>Fri</td>
<td>&lt;10</td>
</tr>
<tr>
<td>x9</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Fri</td>
<td>&lt;10</td>
</tr>
</tbody>
</table>

Next Lecture

- Continuation of Decision Trees
- Neural Networks