High-Level Agent Design

1/22/16
Reading Quiz

Q1: Which of the following was **NOT** mentioned as a dimension of complexity?

- a) finite horizon / indefinite horizon
- b) static environment / dynamic environment **(Correct Answer)**
- c) perfect rationality / bounded rationality
- d) single-agent / multi-agent
Reading Quiz

Q2: A **causal transduction** maps a _______ to a _______.

a) percept $\rightarrow$ command

b) history $\rightarrow$ command

c) belief state & percept $\rightarrow$ belief state

d) belief state & percept $\rightarrow$ command trace
Design Dimensions

- modularity
- representation scheme
- planning horizon
- uncertainty
- number of agents
- learning
- computational limitations

Beyond the book’s list: discreteness, dynamics
Modularity

- **Flat**
  - Many agents in this class will appear flat.
  - examples: SearchAgent, a thermostat
- **Modular**
  - Most interesting agents are modular.
  - examples: a smart home, a self-driving car
- **Hierarchical**
  - This is a specific way of organizing modularity.
  - We can think of each level as an agent.
Representation Scheme

- **states**
  - SearchAgent: a state is a (row, col) pair.
  - Chess: a state is the board configuration.

- **features**
  - SearchAgent: a feature could be the row or column.
  - Chess: a feature could be the black queen’s position.

- **relational descriptions**
  - SearchAgent: is_wall(state) could be a relation.
  - Chess: threatens(black knight, white queen) could be a relation.
Discreteness

Does the agent model the environment as:

- **Discrete**
  - Some software agents may truly live in a discrete world.

- **Continuous**
  - The real world is continuous, but a discrete model can often improve agent reasoning.
  - Discrete and continuous modules can co-exist, e.g. discrete route planning and continuous motor control.

Temperature is continuous, but a discrete state model simplifies the thermostat.
Planning Horizon

- **non-planning**
  - thermostat

- **fixed finite horizon**
  - tic-tac-toe player

- **indefinite finite horizon**
  - chess player

- **infinite horizon**
  - smart home

This difference is an adaptation to computational constraints.

Different components of the same system may operate on different horizons.
Uncertainty

● Input uncertainty: the world may be fully or partially observable.
  ○ noisy LIDAR sensors
  ○ opponent’s poker hand

● Output uncertainty: actions may have deterministic or stochastic effects.
  ○ uncertain effects: wheel slippage
  ○ dynamic environment
Dynamic environment

- If the world is modeled as static, we assume that the environment only changes as a result of the agent’s actions.

- In a dynamic environment, the world can change on its own.
Number of Agents

Additional agents can be modeled as:

- **part of the environment**
  - This will always make the environment dynamic.

- **competitors**
  - The agent will need to reason about their intentions with game theory.

- **collaborators**
  - The agent may be able to offload some of its physical or computational work on others.
  - The agent may need to assist other agents.
Learning

An agent may need to learn:

- The state of the environment.
- Its action function.
- Its goals or preferences.

Learning introduces a whole new set of design dimensions.
Computational Limitations

● Does the agent look for optimal actions or adequate ones?
  ○ These can have very different computational complexity.
  ○ There may be a trade-off between acting optimally and acting quickly.

● What capability do other agents have?
  ○ This relates to whether they are part of the environment or competitors.
<table>
<thead>
<tr>
<th>modular</th>
<th>finite horizon</th>
<th>discrete</th>
<th>uncertainty</th>
<th>dynamic environment</th>
<th>single agent</th>
<th>learning</th>
<th>bounded rationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>🟢⬜⬜⬜⬜⬜⬜⬜</td>
<td><img src="image" alt="Lab 0" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mars rover</td>
<td><img src="image" alt="Mars rover" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stock trading</td>
<td><img src="image" alt="stock trading" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Agent Function

We can think of the entire agent, or the controller, or any given module as implementing some function.

The presence of internal state means that the input to the agent function is the entire history of percepts.

The output at any given time-step is an action.

\[
\begin{align*}
  f (\text{history}) &= \text{action} \\
  f (\text{percept, state}) &= \text{command}
\end{align*}
\]