CS 331: Artificial Intelligence
Intelligent Agents

General Properties of AI Systems

This part is called an agent.
Agent: anything that perceives its environment through sensors and acts on that environment through actuators

Example: Vacuum Cleaner Agent

Percept Sequence | Action
--- | ---
[A, Clean] | Right
[A, Dirty] | Stuck
[B, Clean] | Left
[B, Dirty] | Stuck
[A, Clean][A, Clean] | Right
[A, Clean][A, Dirty] | Stuck
[A, Clean][A, Clean][A, Clean] | Right
[A, Clean][A, Clean][A, Dirty] | Stuck

Agent-Related Terms

- **Percept sequence**: A complete history of everything the agent has ever perceived. Think of this as the state of the world from the agent’s perspective.
- **Agent function (or Policy)**: Maps percept sequence to action (determines agent behavior)
- **Agent program**: Implements the agent function

Question

What’s the difference between the agent function and the agent program?

Rationality

- **Rationality**: do the action that causes the agent to be most successful
- **How do you define success?** Need a performance measure
- **E.g.** reward agent with one point for each clean square at each time step (could penalize for costs and noise)

Important point: Design performance measures according to what one wants in the environment, not according to how one thinks the agent should behave
Rationality

Rationality depends on 4 things:
1. Performance measure of success
2. Agent’s prior knowledge of environment
3. Actions agent can perform
4. Agent’s percept sequence to date

Rational agent: for each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Learning

Successful agents split task of computing policy in 3 periods:
1. Initially, designers compute some prior knowledge to include in policy
2. When deciding its next action, agent does some computation
3. Agent learns from experience to modify its behavior

Autonomous agents: Learn from experience to compensate for partial or incorrect prior knowledge

Properties of Environments

| Fully observable: can access complete state of environment at each point in time | vs Partially observable: could be due to noisy, inaccurate or incomplete sensor data |
| Deterministic: if next state of the environment completely determined by current state and agent’s action | vs Stochastic: a partially observable environment can appear to be stochastic. (Strategic: environment is deterministic except for actions of other agents) |
| Episodic: agent’s experience divided into independent, atomic episodes in which agent perceives and performs a single action in each episode | vs Sequential: current decision affects all future decisions |
| State: agent doesn’t need to keep sensing while decides what action to take, doesn’t need to worry about time | vs Dynamic: environment changes while agent is thinking. (Semi Dynamic: environment doesn’t change with time but agent’s performance does) |
| Discrete (note: discrete/continuous distinction applies to states, time, percepts, or actions) | vs Continuous |
| Single agent | vs Multiagent: agents affect each others performance measure – cooperative or competitive |

PEAS Descriptions of Task Environments

Performance, Environment, Actuators, Sensors

Example: Automated taxi driver

<table>
<thead>
<tr>
<th>Performance</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe, fast, legal, comfortable trip, maximize profits</td>
<td>Roads, other pedestrians, customers</td>
<td>Steering, accelerator, brake, signal, horn, display</td>
<td>Camera, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard</td>
</tr>
</tbody>
</table>

Examples of Task Environments

<table>
<thead>
<tr>
<th>Task Environment</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
<th>Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossword puzzle</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Single</td>
</tr>
<tr>
<td>Chess with a clock</td>
<td>Fully</td>
<td>Strategic</td>
<td>Sequential</td>
<td>Semi</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Poker</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Backgammon</td>
<td>Fully</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Tennis driving</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Multi</td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Multi</td>
</tr>
<tr>
<td>Image analysis</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Episodic</td>
<td>Semi</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Refinery controller</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Multi</td>
</tr>
<tr>
<td>Interactive English tutor</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
</tbody>
</table>

In-class Exercise

Develop a PEAS description of the task environment for a movie recommendation agent

<table>
<thead>
<tr>
<th>Performance Measure</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie database</td>
<td>Movie</td>
<td>Rating</td>
<td>Profile</td>
</tr>
<tr>
<td>User preferences</td>
<td>User</td>
<td>History</td>
<td>Feedback</td>
</tr>
<tr>
<td>Recommendation engine</td>
<td>Engine</td>
<td>Algorithm</td>
<td>Data</td>
</tr>
</tbody>
</table>
In-class Exercise
Describe the task environment for the movie recommendation agent

<table>
<thead>
<tr>
<th>Fully Observable</th>
<th>Partially Observable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Stochastic</td>
</tr>
<tr>
<td>Episodic</td>
<td>Sequential</td>
</tr>
<tr>
<td>Static</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Discrete</td>
<td>Continuous</td>
</tr>
<tr>
<td>Single agent</td>
<td>Multi-agent</td>
</tr>
</tbody>
</table>

Agent Programs
- Agent program: implements the policy
- Simplest agent program is a table-driven agent

```
function TABLE-DRIVEN-AGENT(percept) returns an action
static: percepts, a sequence, initially empty
table, a table of actions, indexed by percept sequences, initially fully specific
append percept to the end of percepts
action ← LOOKUP(percepts, table)
return action
```

This is a BIG table... clearly not feasible!

4 Kinds of Agent Programs
- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

Simple Reflex Agent
- Selects actions using only the current percept
- Works on condition-action rules:
  if condition then action

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
static: rules, a set of condition-action rules
state ← INTERPRET-INPUT(percept)
rule ← RULE-MATCH(state, rules)
action ← RULE-ACTION(rule)
return action
```

Simple Reflex Agents
- Advantages:
  - Easy to implement
  - Uses much less memory than the table-driven agent
- Disadvantages:
  - Will only work correctly if the environment is fully observable
  - Infinite loops
Model-based Reflex Agents

- Maintain some internal state that keeps track of the part of the world it can’t see now
- Needs model (encodes knowledge about how the world works)

```python
function REFLEX-AGENT(WITH-STATE)(percept) returns an action
  static: state, a description of the current world state
  rules, a set of condition-action rules
  action, the most recent action, initially none
  state ← UPDATE-STATE(state, action, percept)
  rule ← RULE-MATCH(state, rules)
  action ← RULE-ACTION(rule)
  return action
```

Goal-based Agents

- Goal information guides agent’s actions (looks to the future)
- Sometimes achieving goal is simple e.g. from a single action
- Other times, goal requires reasoning about long sequences of actions
- Flexible: simply reprogram the agent by changing goals

Utility-based Agents

- What if there are many paths to the goal?
- Utility measures which states are preferable to other states
- Maps state to real number (utility or “happiness”)
Learning Agents

Think of this as outside the agent since you don’t want it to be changed by the agent.

Maps percepts to actions

In-class Exercise

• Select a suitable agent design for the movie recommendation agent

What you should know

• What it means to be rational
• Be able to do a PEAS description of a task environment
• Be able to determine the properties of a task environment
• Know which agent program is appropriate for your task