Intelligent Agents

Chapter 2

Some slide credits to Hwee Tou Ng (Singapore)
Outline

• Agents and environments
• Rationality
• PEAS (Performance measure, Environment, Actuators, Sensors)
• Environment types
• Agent types
Agents

• An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators
• Human agent: eyes, ears, and other organs for sensors; hands,
• legs, mouth, and other body parts for actuators
• Robotic agent: cameras and infrared range finders for sensors;
• various motors for actuators
Agents and environments

- The **agent function** maps from percept histories to actions:
  \[ f: P^* \rightarrow A \]

- The **agent program** runs on the physical architecture to produce \( f \)

- agent = architecture + program
agent = architecture + program?
Vacuum-cleaner world

- Percepts: location and contents, e.g., [A, Dirty]
- Actions: Left, Right, Suck, NoOp
A vacuum-cleaner agent

<table>
<thead>
<tr>
<th>Percept sequence</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[B, Clean]</td>
<td>Left</td>
</tr>
<tr>
<td>[B, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[A, Clean], [A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Clean], [A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>[A, Clean], [A, Clean], [A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Clean], [A, Clean], [A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Figure 2.3** Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.
Rational agents

• An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform. The right action is the one that will cause the agent to be most successful.

• Performance measure: An objective criterion for success of an agent's behavior.

• E.g., performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.

• Which measure is the best?
Rational agents: the right thing?

- **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.
Rational agents

• Rationality is distinct from omniscience (all-knowing with infinite knowledge)
• Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, exploration)
• An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt)
PEAS

- PEAS: Performance measure, Environment, Actuators, Sensors
- Must first specify the setting for intelligent agent design
- Consider, e.g., the task of designing an automated taxi driver:
  - Performance measure
  - Environment
  - Actuators
  - Sensors
Must first specify the setting for intelligent agent design
Consider, e.g., the task of designing an automated taxi driver:
- Performance measure: Safe, fast, legal, comfortable trip, maximize profits
- Environment: Roads, other traffic, pedestrians, customers
- Actuators: Steering wheel, accelerator, brake, signal, horn
- Sensors: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard
PEAS

• Agent: Medical diagnosis system
• Performance measure: Healthy patient, minimize costs, lawsuits
• Environment: Patient, hospital, staff
• Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
• Sensors: Keyboard (entry of symptoms, findings, patient's answers)
PEAS

- **Agent**: Part-picking robot
- **Performance measure**: Percentage of parts in correct bins
- **Environment**: Conveyor belt with parts, bins
- **Actuators**: Jointed arm and hand
- **Sensors**: Camera, joint angle sensors
PEAS

• Agent: Interactive English tutor
• Performance measure: Maximize student's score on test
• Environment: Set of students
• Actuators: Screen display (exercises, suggestions, corrections)
• Sensors: Keyboard
Environment types

- **Fully observable** (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.

- **Deterministic** (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)

- **Episodic** (vs. sequential): The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.
Environment types

• Static (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is *semidynamic* if the environment itself does not change with the passage of time but the agent's performance score does)

• Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.

• Single agent (vs. multiagent): An agent operating by itself in an environment.
### Environment types

<table>
<thead>
<tr>
<th></th>
<th>Chess with a clock</th>
<th>Chess without a clock</th>
<th>Taxi driving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully observable</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Strategic</td>
<td>Strategic</td>
<td>No</td>
</tr>
<tr>
<td>Episodic</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static</td>
<td>Semi</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Single agent</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent
Agent functions and programs

• An agent is completely specified by the agent function mapping percept sequences to actions
• One agent function (or a small equivalence class) is rational
• Aim: find a way to implement the rational agent function concisely
Table-lookup agent

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

Figure 2.7 The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

- **Drawbacks:**
  - Huge table
  - Take a long time to build the table
  - No autonomy
  - Even with learning, need a long time to learn the table entries
Agent program for a vacuum-cleaner agent

function REFLEX-VACUUM-AGENT([location, status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left

Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

Drawbacks:
- Too simple?
Agent types

Four basic types in order of increasing generality:

• Simple reflex agents
• Model-based reflex agents
• Goal-based agents
• Utility-based agents
Simple reflex agents
function SIMPLE-REFLEX-AGENT(\textit{percept}) \textbf{returns} an action
\textbf{persistent}: \textit{rules}, a set of condition–action rules

\begin{itemize}
\item \textit{state} $\leftarrow$ \textsc{interpret-input}(\textit{percept})
\item \textit{rule} $\leftarrow$ \textsc{rule-match}(\textit{state}, \textit{rules})
\item \textit{action} $\leftarrow$ \textit{rule}.ACTION
\end{itemize}
\textbf{return} \textit{action}

\begin{figure}[h]
\caption{A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.}
\end{figure}
Model-based reflex agents
Model-based reflex agents

```plaintext
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
    persistent: state, the agent’s current conception of the world state
        model, a description of how the next state depends on current state and action
        rules, a set of condition–action rules
        action, the most recent action, initially none

    state ← UPDATE-STATE(state, action, percept, model)
    rule ← RULE-MATCH(state, rules)
    action ← rule.ACTION
    return action
```

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.
Goal-based agents

- Sensor
- What the world is like now
- What it will be like if I do action A
- What my actions do
- How the world evolves
- State
- What action I should do now
- Goals
- Actuators
- Environment
Utility-based agents

- State
- How the world evolves
- What my actions do
- Utility

Sensors

What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now

Agent

Environment

Actuators