Intelligent Agents

Chapter 2
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:
  \[ f: \mathcal{P}^* \rightarrow \mathcal{A} \]

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

Percepts: location and contents, e.g., [A, Dirty]

Actions: Left, Right, Suck, NoOp
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
else

<table>
<thead>
<tr>
<th>Percept sequence</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<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>[A, Clean] [B, Clean]</td>
<td>left</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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: A fixed objective criterion for success of an agent's behavior.

- E.g., performance measure of a vacuum-cleaner agent could be:
  - one point per square cleaned up in time T
  - one point per clean square per time step, minus one per move
  - penalize for > k dirty squares
  - Penalize for > m units of electricity consumed per time step
  - Penalize for amount of noise generated
Rational agents

- **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) – action outcomes may not be as expected
- Rational is not equal to clairvoyant – percepts may not be complete
- Thus rational is not equal to successful!

- Rationality ➔ exploration, learning, autonomy
  - Agents can perform actions to explore environment to obtain useful information (learning by 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
PEAS

- 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)
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
Agent: Program playing the game of checkers
Performance measure: Maximize the number of games won
Environment: A human opponent player
Actuators: Screen display (the move chosen by the program)
Sensors: Keyboard (the move chosen by the human player)
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 w/o a clock</th>
<th>Backgammon</th>
<th>Taxi driving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully observable</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Strategic</td>
<td>Strategic</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Episodic</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static</td>
<td>Semi</td>
<td>Yes</td>
<td>Semi</td>
<td>No</td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</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>
<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
  append *percept* to the end of *percepts*
  *action* ← Lookup (*percepts*, *table*)
  Return *action*

- 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 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
Simple reflex agents

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
Model-based reflex agents

- State
- How the world evolves
- What my actions do
- Condition-action rules
- What the world is like now
- What action I should do now
- Actuators
- Sensors

Agent

Environment
Model-based reflex agents

Function Reflex-Agent-w-St (percept)

Static: state, rules, action

\[
\text{state} \leftarrow \text{Update-State} \ (\text{state}, \text{action}, \text{percept})
\]

\[
\text{rule} \leftarrow \text{Rule-Match} \ (\text{state}, \text{rules})
\]

\[
\text{action} \leftarrow \text{Rule-Action} \ [\text{rule}]
\]

Return action
Goal-based agents
Utility-based agents
Learning agents