Intelligent Agents

Chapter 2
Outline

- Agents and environments
- Rationality
- Task environment:
  - **PEAS**:
    - Performance measure
    - Environment
    - Actuators
    - Sensors
- Environment types
- Agent types
Agents and Environments

• An *agent* is anything that can be viewed as perceiving its *environment* through *sensors* and acting in that environment through *actuators*.
Agents and Environments

- An *agent* is anything that can be viewed as perceiving its *environment* through *sensors* and acting in that environment through *actuators*.

  ![Diagram of agent-environment interaction](image)

- *Agents* include humans, robots, softbots, thermostats, etc.
- The *agent function* maps from percept histories to actions:
  \[ f : \mathcal{P}^* \rightarrow \mathcal{A} \]
- The *agent program* runs on a physical *architecture* to give \( f \)
Vacuum-cleaner world

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

Agent function:

<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>
</tbody>
</table>

Note: This says *how* the agent should function.

- It says nothing about how this should be implemented.
A vacuum-cleaner agent

Agent program:

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

Ask:

- What is the *right* function for implementing a specification?
- Can it be implemented in a small agent program?
Rationality

Informally a *rational* agent is one that does the “right thing”.

• How well an agent does is given by a performance measure.
• A fixed performance measure evaluates a sequence of environment states.
  Examples:
  • 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?
• A rational agent selects an action which maximizes the expected value of the performance measure given the percept sequence to date and its own knowledge.
• The action selection may range from being hardwired (e.g. in an insect or reflexive agent) to involving substantial reasoning.
Rationality

Informally a *rational* agent is one that does the “right thing”.

- *How well* an agent does is given by a *performance measure*.
Rationality

Informally a *rational* agent is one that does the “right thing”.

- *How well* an agent does is given by a *performance measure*.
- A fixed *performance measure* evaluates a sequence of environment states.
- Examples:
  - 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？
Rationality

Informally a *rational* agent is one that does the “right thing”.

- *How well* an agent does is given by a *performance measure*.
- A fixed *performance measure* evaluates a sequence of environment states.
- **Examples:**
  - 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?
- A *rational agent* selects an action which maximizes the expected value of the performance measure *given the percept sequence to date* and its own knowledge.
- The action selection may range from being hardwired (e.g. in an insect or reflexive agent) to involving substantial reasoning.
Rationality

Notes:

- Rational $\neq$ omniscient
  - percepts may not supply all the relevant information
- Rational $\neq$ clairvoyant
- action outcomes may not be as expected
- Hence, rational $\neq$ successful
- Full, general rationality requires exploration, learning, autonomy
Rationality

Notes:

• Rational $\neq$ omniscient
  • percepts may not supply all the relevant information
• Rational $\neq$ clairvoyant
  • action outcomes may not be as expected
Rationality

Notes:

- Rational $\neq$ omniscient
  - percepts may not supply all the relevant information
- Rational $\neq$ clairvoyant
  - action outcomes may not be as expected
- Hence, rational $\neq$ successful
- Full, general rationality requires exploration, learning, autonomy
The Task Environment

• To design a rational agent, we must specify the *task environment*

• The task environment has the following components:
  • Performance measure
  • Environment
  • Actuators
  • Sensors

• Acronym: PEAS
Consider, e.g., the task of designing an automated taxi:

**Performance measure:** safety, destination, profits, legality, comfort, ...

**Environment:** streets/freeways, traffic, pedestrians, weather, ...

**Actuators:** steering, accelerator, brake, horn, speaker, ...

**Sensors:** video, accelerometers, gauges, engine sensors, keyboard, GPS, ...
Internet shopping agent

Performance measure: ??

Environment: ??
  Actuators: ??
  Sensors: ??
Internet shopping agent

Performance measure: price, quality, appropriateness, efficiency

Environment: ??
  Actuators: ??
  Sensors: ??
Internet shopping agent

**Performance measure:** price, quality, appropriateness, efficiency

**Environment:** current and future WWW sites, vendors, shippers

**Actuators:** ??

**Sensors:** ??
Internet shopping agent

Performance measure: price, quality, appropriateness, efficiency

Environment: current and future WWW sites, vendors, shippers

Actuators: display to user, follow URL, fill in form

Sensors: ??
Internet shopping agent

**Performance measure:** price, quality, appropriateness, efficiency

**Environment:** current and future WWW sites, vendors, shippers

**Actuators:** display to user, follow URL, fill in form

**Sensors:** HTML pages (text, graphics, scripts)
Environment Types

- Fully observable vs. partially observable
  - If the agent has access to full state of the environment or not
Environment Types

• *Fully observable vs. partially observable*
  
  • If the agent has access to full state of the environment or not

• *Deterministic vs. stochastic*
  
  • Deterministic: Next state is completely determined by the agent’s actions. (Or the set of agents in a multiagent env.)
Environment Types

- Fully observable vs. partially observable
  - If the agent has access to full state of the environment or not

- Deterministic vs. stochastic
  - Deterministic: Next state is completely determined by the agent’s actions. (Or the set of agents in a multiagent env.)

- Uncertain: not fully observable or not deterministic
Environment Types

- Fully observable vs. partially observable
  - If the agent has access to full state of the environment or not

- Deterministic vs. stochastic
  - Deterministic: Next state is completely determined by the agent’s actions. (Or the set of agents in a multiagent env.)
  - Uncertain: not fully observable or not deterministic

- Episodic vs. sequential
  - Episodic: Agent’s experience is divided into independent episodes (e.g. classification)
Environment Types

- **Fully observable vs. partially observable**
  - If the agent has access to full state of the environment or not

- **Deterministic vs. stochastic**
  - Deterministic: Next state is completely determined by the agent’s actions. (Or the set of agents in a multiagent env.)

- **Uncertain**: not fully observable or not deterministic

- **Episodic vs. sequential**
  - Episodic: Agent’s experience is divided into independent episodes (e.g. classification)

- **Static vs. dynamic**
  - Dynamic: Environment may change while agent is deliberating.
Environment Types

- Fully observable vs. partially observable
  - If the agent has access to full state of the environment or not
- Deterministic vs. stochastic
  - Deterministic: Next state is completely determined by the agent’s actions. (Or the set of agents in a multiagent env.)
- Uncertain: not fully observable or not deterministic
- Episodic vs. sequential
  - Episodic: Agent’s experience is divided into independent episodes (e.g. classification)
- Static vs. dynamic
  - Dynamic: Environment may change while agent is deliberating.
- Discrete vs. continuous
Environment Types

- **Fully observable vs. partially observable**
  - If the agent has access to full state of the environment or not

- **Deterministic vs. stochastic**
  - Deterministic: Next state is completely determined by the agent’s actions. (Or the set of agents in a multiagent env.)

- **Uncertain**: not fully observable or not deterministic

- **Episodic vs. sequential**
  - Episodic: Agent’s experience is divided into independent episodes (e.g. classification)

- **Static vs. dynamic**
  - Dynamic: Environment may change while agent is deliberating.

- **Discrete vs. continuous**

- **Single-agent vs. multiagent**
## Environment types

<table>
<thead>
<tr>
<th>Observable</th>
<th>Crossword</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Episodic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-agent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Environment types

<table>
<thead>
<tr>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
<th>Single-agent</th>
<th>Crossword</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
## Environment types

<table>
<thead>
<tr>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
<th>Single-agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Partly</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
## Environment types

<table>
<thead>
<tr>
<th></th>
<th>Crossword</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes</td>
<td>No</td>
<td>Partly</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>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Discrete</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Single-agent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Environment types

<table>
<thead>
<tr>
<th></th>
<th>Crossword</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes</td>
<td>No</td>
<td>Partly</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>Yes</td>
<td>Yes</td>
<td>Semi</td>
<td>No</td>
</tr>
<tr>
<td>Discrete</td>
<td>Single-agent</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Environment types

<table>
<thead>
<tr>
<th>Observable</th>
<th>Crossword</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes</td>
<td>No</td>
<td>Partly</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>Yes</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>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
## Environment types

<table>
<thead>
<tr>
<th></th>
<th>Crossword</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes</td>
<td>No</td>
<td>Partly</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>Yes</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>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
The environment type largely determines the agent design

- The real world is:

<table>
<thead>
<tr>
<th>Environment Type</th>
<th>Crossword</th>
<th>Backgammon</th>
<th>Internet Shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes</td>
<td>No</td>
<td>Partly</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>Yes</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>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
### Environment types

<table>
<thead>
<tr>
<th></th>
<th>Crossword</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes</td>
<td>No</td>
<td>Partly</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>Yes</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>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

*The environment type largely determines the agent design*

- The real world is:
  - partially observable,
  - stochastic,
  - sequential,
  - dynamic,
  - continuous, and
  - multi-agent
Agent types

There are four basic types in order of increasing generality:

- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents

All these can have a learning component added
Simple reflex agents

- Action is selected according to the current percept
- No knowledge of percept history.
A simple reflex agent algorithm

Function \texttt{Simple-Reflex-Agent}(\texttt{percept}) \textbf{returns} an action

\textbf{persistent:} \texttt{rules} a set of condition-action rules

\texttt{state} ← \texttt{Interpret-Input(\texttt{percept})}
\texttt{rule} ← \texttt{Rule-Match(\texttt{state},\texttt{rules})}
\texttt{action} ← \texttt{rule.Action}
\textbf{return} \texttt{action}
Example

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
Reflex agents with state

- Also called a “model-based reflex agent”
- Agent keeps track of what it knows about the world.
- Useful for partial observability
A simple reflex agent algorithm

Function Reflex-Agent-With-State(percept) returns an action

persistent: state: the agent’s conception of the world state
model: The transition model — how the next state depends on the present 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
• Agent’s actions are determined in part by its goals.
• Example: Classical planning.
Utility-based agents

In addition to goals, use a notion of how “good” an action sequence is.
  - E.g.: Taxi to airport should be safe, efficient, etc.
Learning agents

Performance standard

Environment

Agent

Critic

Sensors

feedback

changes

knowledge

learning goals

Problem generator

Learning element

Performance element

Actuators
Summary

- **Agents** interact with **environments** through **actuators** and **sensors**
- The **agent function** describes what the agent does in all circumstances
- The **performance measure** evaluates the environment sequence
- A **rational** agent maximizes expected performance
- **Agent programs** implement agent functions
- **PEAS** descriptions define task environments
- Environments are categorized along several dimensions: observable? deterministic? episodic? static? discrete? single-agent?
- Several basic agent architectures exist: reflex, reflex with state, goal-based, utility-based