Chapter 2: Intelligent Agents

Outline

What is an Agent?

Definition

An **agent** is anything that can be viewed as:

- **Perceiving** its environment through **sensors**
- **Acting** upon that environment through **actuators**

Examples:

- **Human agent:** Eyes, ears (sensors) $\rightarrow$ Hands, legs, vocal tract (actuators)
- **Robotic agent:** Cameras, infrared sensors $\rightarrow$ Motors, grippers
- **Software agent:** File contents, network packets $\rightarrow$ Display output, file writes

Agent Interaction with Environment

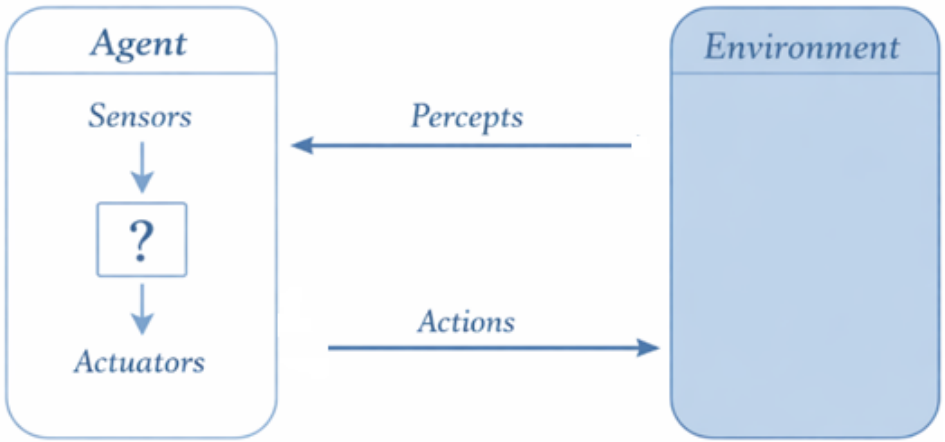
Agent $\leftrightarrow$ Environment Loop

- **Percept:** Content an agent's sensors are perceiving at any given instant
- **Percept sequence:** Complete history of everything the agent has ever perceived
- **Agent function:** Maps any given percept sequence to an action
- **Agent program:** Concrete implementation of the agent function

Key Equation

agent = architecture + program

Agent, Environment and the Universe



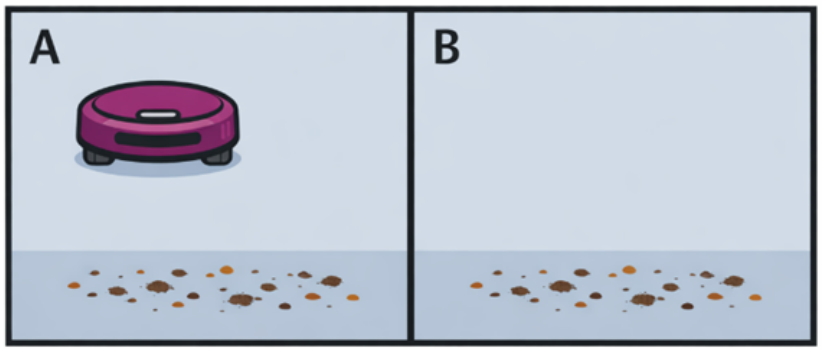
The Vacuum World Example

Simple environment for illustrating agent concepts:

- Two locations: Square A and Square B
- Each square can be *Clean* or *Dirty*
- Agent perceives: its location and dirt status
- Available actions: `Left`, `Right`, `Suck`, `NoOp`

Simple Agent Function:

- If current square is dirty $\rightarrow$ `Suck`
- Otherwise $\rightarrow$ move to the other square



This simple reflex behavior can be implemented with just a few if-then rules!

Good Behavior: Performance Measure

Performance Measure

An **objective criterion** for success of an agent's behavior

For the vacuum world:

- Award one point for each clean square at each time step
- Measure over a lifetime (e.g., 1000 time steps)

Design Principle

Design performance measures according to what you *actually want* achieved in the environment, not according to how you think the agent should behave.

What is Rationality?

Rationality depends on four things:

1. The **performance measure** that defines success
2. The agent's **prior knowledge** of the environment
3. The **actions** that the agent can perform
4. The agent's **percept sequence** to date

Definition of a 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.

Rationality vs. Omniscience

- **Omniscience:** Knowing the *actual* outcome of actions
  - Impossible in reality!
- **Rationality:** Maximizing *expected* performance
  - Based on percept sequence to date
  - Does not require knowing the future

Example: Crossing the street

- Rational: Look both ways, then cross if safe
- Not rational: Cross without looking (even if you get lucky)
- Omniscient: Know exactly when the cargo door will fall from the plane

Rationality $\neq$ *Perfection*

Information Gathering and Learning

A rational agent should:

- **Gather information** to improve future decisions
  - Example: Looking before crossing the street
  - Example: Exploring an unknown environment
- **Learn** from experience
  - Modify and augment initial knowledge
  - Adapt to changing environments

Autonomy

An agent is **autonomous** if its behavior is determined by its own experience (with ability to learn and adapt), rather than solely by its initial programming.

The Nature of Environments

Task Environment

The "problem" to which rational agents are the "solution"

PEAS Description:

- **P**erformance measure
- **E**nvironment
- **A**ctuators
- **S**ensors

Design Principle

The first step in designing an agent must always be to specify the task environment as fully as possible.

PEAS Example: Automated Taxi

Performance	Safe, fast, legal, comfortable trip; maximize profits
Environment	Roads, traffic, pedestrians, weather, customers
Actuators	Steering, accelerator, brake, horn, display
Sensors	Cameras, GPS, speedometer, accelerometer

Key insight: The more restricted the environment, the easier the design problem!

Properties of Task Environments (1)

Fully Observable vs. Partially Observable

- **Fully observable:** Sensors detect all aspects relevant to action choice
- **Partially observable:** Noisy sensors or missing information
- **Unobservable:** No sensors at all

Single-agent vs. Multiagent

- **Single-agent:** Crossword puzzle
- **Multiagent:** Chess (competitive), taxi driving (partially cooperative)
- Key: Does entity B's behavior depend on maximizing a performance measure that depends on agent A's behavior?

Properties of Task Environments (2)

Deterministic vs. Nondeterministic

- **Deterministic:** Next state completely determined by current state and action
- **Nondeterministic:** Uncertainty in outcomes
- **Stochastic:** Nondeterministic with explicit probabilities

Episodic vs. Sequential

- **Episodic:** Each episode is independent (e.g., image classification)
- **Sequential:** Current decision affects future decisions (e.g., chess, taxi driving)

Properties of Task Environments (3)

Static vs. Dynamic

- **Static:** Environment doesn't change while agent is deliberating
- **Dynamic:** Environment changes (e.g., taxi driving)
- **Semidynamic:** Environment doesn't change, but performance score does

Discrete vs. Continuous

- **Discrete:** Finite number of distinct states/actions (e.g., chess)
- **Continuous:** Continuous range of states/actions (e.g., taxi driving)

Known vs. Unknown

- Refers to agent's knowledge of the "laws of physics" of the environment
- Not strictly a property of the environment itself

Environment Properties: Examples

Task	Obs.	Agents	Det.	Ep.	Static	Disc.
Crossword	Fully	Single	Det.	Seq.	Static	Discrete
Chess	Fully	Multi	Det.	Seq.	Semi	Discrete
Poker	Partial	Multi	Stoch.	Seq.	Static	Discrete
Taxi	Partial	Multi	Stoch.	Seq.	Dynamic	Cont.
Medical Dx	Partial	Single	Stoch.	Seq.	Dynamic	Cont.

Hardest case: Partially observable, multiagent, nondeterministic, sequential, dynamic, continuous, and unknown

Taxi driving is hard in all these senses except that it's mostly known!

The Structure of Agents

Agent = Architecture + Program

Four basic agent types:

1. Simple reflex agents
2. Model-based reflex agents
3. Goal-based agents
4. Utility-based agents

All can be turned into **learning agents**

Challenge

Write programs that produce rational behavior from a *small* program rather than from a vast lookup table

Table-Driven Agent (Don't Do This!)

Idea: Store a lookup table mapping every possible percept sequence to an action

Why this fails:

- Table size: $\sum_{t=1}^{T} |P|^t$ entries
- For automated taxi with one camera over 1 hour: $> 10^{600,000,000,000}$ entries
- For chess: $\geq 10^{150}$ entries
- Number of atoms in observable universe: $< 10^{80}$

Problems:

1. No physical agent has space to store the table
2. Designer doesn't have time to create it
3. Agent can't learn all entries from experience

Simple Reflex Agents

Principle: Select actions based on *current percept only*, ignoring percept history

Condition-Action Rules:

`if` *condition* `then` *action*

Example: Driving

- `if` car-in-front-is-braking `then` initiate-braking

Advantages:

- Simple and fast

Limitations:

- Only works in *fully observable* environments
- Can get stuck in infinite loops in partially observable environments