INTELLIGENT AGENTS

Chapter 2

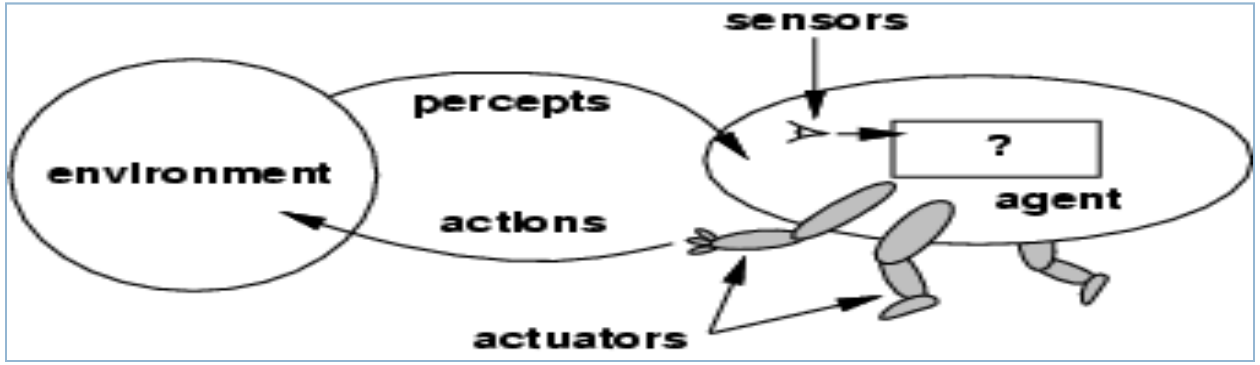
Outline

- 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:
  - sensors: eyes, ears, and other organs;
  - actuators: hands, legs, mouth, and other body parts;
- Robotic agent:
  - sensors: cameras and infrared range finders;
  - actuators: various motors

Agents and environments

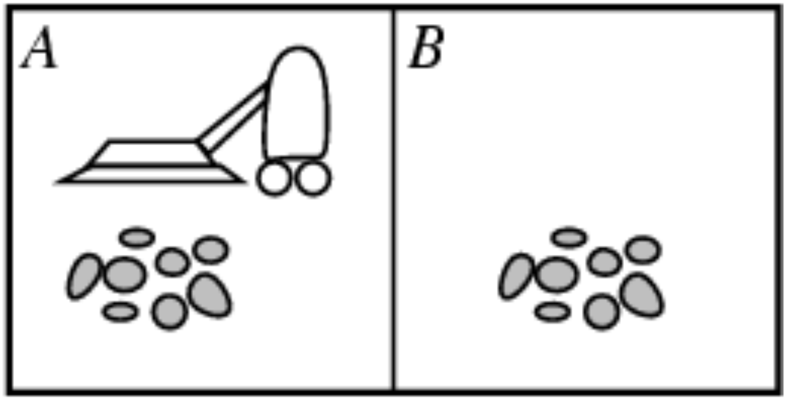


- The agent function maps percept histories to actions

$$[f: \mathcal{P}^{\star} \rightarrow \mathcal{A}]$$

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

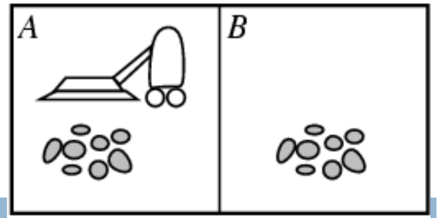
Vacuum-cleaner world



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

- A simple agent function
  - if the current square is dirty, then suck;
  - otherwise, move to the other square

A vacuum-cleaner agent



Partial tabulation of the previous **agent function** for the vacuum-cleaner world

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
$\vdots$	$\vdots$

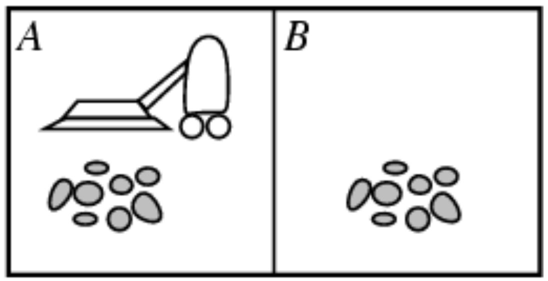
A vacuum-cleaner agent

- Various vacuum-world agents can be defined simply by filling in the right-hand column in various ways

- Obvious question: What is the right way to fill out the 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**: 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.



Rational agents

- Rational Agent:

  For each possible percept sequence,

  it should **select an action**

  that is **expected to maximize its performance measure**,

  - **given** the **evidence** provided by the **percept sequence**
  - **given** 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**)
- Ideally: equip an agent with **some prior knowledge** and **the ability to learn**

The task environments

- Task environments: the "problems" for which artificial agents are "solutions"
- A task environment is specified by 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?
- We will show examples of agent types and their PEAS descriptions

PEAS: taxi

- Agent: an automated taxi driver
  - **Performance measure:** Safe, fast, legal, comfortable trip, maximize profits
  - **Environment:** Roads, other cars in the traffic, pedestrians, customers
  - **Actuators:** Steering, accelerator, brake, signal, horn
  - **Sensors:** Cameras, sonar, speedometer, GPS, engine sensors, keyboard

PEAS: medical diagnosis system

- Agent: Medical diagnosis system
  - **P**erformance measure: Healthy patient, minimize costs, lawsuits
  - **E**nvironment: Patient, hospital, medical staff
  - **A**ctuators: Screen display (questions, tests, diagnoses, treatments, referrals)
  - **S**ensors: Keyboard (entry of symptoms, findings, patient's answers)

PEAS: Interactive English tutor

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

Task 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
- Partially observable: because of noisy, inaccurate sensors ..
- No sensors → the environment is unobservable

Task Environment types

- **Deterministic**: The **next state** of the environment is **completely determined by the current state** and the **action** executed by the agent
- **Stochastic**: there is **uncertainty** on **the next state** and
  it is expressed with probabilities
- **Non-deterministic**: there is **uncertainty** on the **next state**
  but no probabilities are available
- **Uncertain**: **not fully observable** and **non-deterministic**
- **Examples**
  - Vacuum example → deterministic
  - Taxi example → stochastic

Task Environment types

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

Sequential: a **current decision** may affect **future decisions**

- **Examples**
  - Classification tasks are usually episodic
  - Chess and taxi are sequential

Task Environment types

- **Static** (vs. dynamic): The environment is **unchanged while an agent is deliberating**
  - No need to keep looking at the world while deliberating the next action
  - nor worrying about time

- **Dynamic**: **continuously asking the agent** what it wants to do

- The environment is **semidynamic** if
  - the environment itself **does not change** with the passage of time
  - but the **agent's performance score** does

- **Examples**
  - Crossword puzzles → static
  - Chess with clock → semidynamic
  - Taxi → dynamic

Task Environment types

- **Discrete** (vs. continuous): A **finite number** of distinct, clearly defined **states**, **percepts** and **actions.** Applies also to **time**
  - Chess → discrete
  - Taxi driving → continuous

Task Environment types

- **Single agent** (vs. multi-agent): An agent operating by itself in an environment

  The **multi-agent** case can be
  - <u>competitive</u>: two agents playing chess
  - <u>cooperative</u>: taxis trying to avoid each other

Task Environment types

- **Known** (vs. unknown): depends on the **knowledge of the agent** or the designer of the agent of the **rules governing** the environment

  In a known environment for each action there is
  - an outcome (if deterministic) or
  - a probability distribution over the possible outcomes (if stochastic)

  An environment can also be
  - **known** and **partially observable** (for example, a card game)
  - **unknown** and **fully observable** (a new videogame)

Task Environment types

	Chess w clock	Chess no clock	Taxi driving
Fully/partially observable	Fully observable	Fully observable	Partially observable
Deterministic/Stochastic	Deterministic	Deterministic	Stochastic
Episodic/Sequential	Sequential	Sequential	Sequential
Static/Semi/Dynamic	Semi	Static	Dynamic
Discrete/Continuous	Discrete	Discrete	Continuous
Single/Multi-Agent	Multi-agent	Multi-agent	Multi-agent

- 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
- Aim of AI: design **agent programs** that **implement the agent functions** concisely
- **Agent = architecture + program**
- **Architecture:**
  - feeds the percepts from the sensors to the program
  - runs the program
  - feeds the actions to the actuators

Table-lookup agent

- **Agent programs** we design have the same skeleton
  - They **take** the current percept as input **from the sensors**
  - They **return** an action to the **actuators**

- **Agent program ≠ Agent function**
  - Agent **program**: takes the **current percept** as input
  - Agent **function**: takes the **entire percept history**
  - If the agent's **actions** need to depend on the entire percept sequence, the agent will have to remember the percepts

Table-Driven Agent

The TABLE-DRIVEN-AGENT program
- invoked **for each new percept**
- retains the complete percept sequence in memory
- returns **an action** each time

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*

Table-Driven Agent

The TABLE-DRIVEN-AGENT program
- invoked for each new percept
- retains the complete percept sequence in memory
- returns an action each time

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*

- Drawbacks:
  - Invoked for each new percept
  - Huge table (chess: 10^{150} entries)
  - Takes a long time to build the table
  - Even with learning, need a long time to learn the table entries
  - Who knows how to build it?!

Agent types

- Four basic types of agents in order of increasing generality:
  - **Simple reflex** agents
  - **Model-based reflex** agents
  - **Goal-based** agents
  - **Utility-based** agents

Agent types

- **Simple reflex agents** choice of the action depends only on the current percept
- **Model-based reflex agents** maintain internal state to track aspects of the world that are not evident in the current percept
- **Goal-based agents** act to achieve their goals
- **Utility-based agents** try to maximize their own expected "happiness"

How the components of agent programs work

- **Agent programs** consist of various <u>**components**</u>
- The **components** can **represent** the **environment** in three ways (with <u>increasing</u> complexity and <u>expressive power</u>)
  - **Atomic**
  - **Factored**
  - **Structured**

- A <u>**more expressive**</u> representation can capture at least as concisely
  - **everything a less expressive** one can capture
  - **plus some more**

How the components of agent programs work

- **Atomic representation**
  - **<u>Each state</u>** of the world is **indivisible** – it has **no internal structure**
    - **Example:** the algorithms underlying search
- **Factored representation**
  - **<u>Each state</u>** contains a fixed **set of variables** (or attributes)
  - Each variable can have a **value**
  - Two different factored states can <u>share some variables</u>
    - **Example:** constraint satisfaction algorithms, planning
- **Structured representation**
  - **<u>Each state</u>** contains **objects** (with variables with values) and **relations with other objects,**
    - **Example:** knowledge-based learning, natural language understanding