Chapter 1: Introduction

(2) Intelligent Agents

Nguyễn Hải Minh, Ph.D
nhminh@fit.hcmus.edu.vn
Outline

1. Agents and environments
2. Rationality
3. The Nature of Environment
4. The Structure of Agents
1. Agents and Environments

- Agent
- Percept Sequence
- Agent Function
- Agent Program
- The Vaccum-Cleaner World
What is Agents?

Artificial intelligence is the study of how to make computers do things that people are better at if:
- they could extend what they do to huge data sets
- do it fast, in near real-time
- not make mistakes

We call such systems, Agents
What is Agents?

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

- **Environment**
- **Percepts** → **Agent**
- **Sensors** → **Agent**
- **Agent** → **Actions**
- **Actions** → **Effectors**
What is Agents?

- **Human agent:**
  - **Sensors:** eyes, ears, and other organs
  - **Actuators:** hands, legs, and some body parts

- **Robotic agent:**
  - **Sensors:** camera, infrared range finders, etc.
  - **Actuators:** levels, motors, etc.

- **Software agent:**
  - **Sensors:** keystrokes, file contents, network packets
  - **Actuators:** displaying on the screen, writing files, sending network packets
What is Agents?

Diagram of an agent:

Agent → Sensors → ? → Actuators

Percepts

Actions

What AI should fill

Nguyễn Hải Minh @ FIT

2018/05/11

CuuDuongThanCong.com

https://fb.com/tailieudientucntt
Percept Sequence

Percept:
- the agent’s perceptual inputs at any given instant.

Percept sequence:
- The complete history of everything the agent has ever perceived
Describe Agent’s Behavior

☑ Agent function:
  ○ maps from percept sequence to an action:

\[ f: P \rightarrow A \]

☑ Agent program:
  ○ the implementation of an agent function.

agent = architecture + program

mathematical \hspace{2cm} \text{practical}
The Vacuum-cleaner world

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

- **Actions:**
  - Left, Right, Suck, Do Nothing
# The Vacuum-cleaner world

<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>

## Simple Agent Function Table
The Vacuum-cleaner world

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

An example of Agent Program in the two-state vacuum environment
Why do we need Agents?

- A tool for **analyze systems**.
- All areas of engineering can be seen as **designing artifacts** that *interact with the world*.
  - AI designs artifacts that have significant *computational resources* and the task environment requires *nontrivial decision making*. 
2. Rationality

- Rational Agent
- Performance Measure
- Rationality
- Definition of Rational Agent
- Omniscience, learning, and autonomy
Rational Agents

- Rational agent: one that does the right thing
  \[ \text{\textit{Fill out every entry in the table correctly (rationally)}} \]

- What is “right” thing?
  - The actions that cause the agent to be most successful
  \[ \text{\textit{We need ways to measure success}} \]

Performance Measure
Performance Measure

☐ An agent, based on its percepts → generates actions sequence → environment goes to sequence of states
  - If this sequence of states is desirable → the agent performed well

☐ Performance measure
  - Evaluates any given sequence of environment states.
  - An objective function that determines how the agent does successfully
  - 90%? 30%?
Performance Measure

- A general rule: Design performance measures according to
  - What one actually **wants** in the environment
  - Not how one **thinks** the agent should behave

- E.g., in vacuum-cleaner world
  - We want the floor clean, no matter how the agent behaves
  - We don’t restrict how the agent behaves

→ *Give some examples of performance measure of a vaccum-cleaner*
Rationality

What is rational depends on:

1. The *performance measure* that defines the criterion of 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 expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has

E.g., an exam
  - Maximize marks, based on
  - the questions on the paper & your knowledge
Vaccum-cleaner agent

1. **Performance measure:**
   - Awards one point for each clean square
   - at each time step, over 10000 time steps

2. **Prior knowledge about the environment**
   - The geography of the environment (2 squares)
   - The *effect* of the actions

3. **Actions that can perform**
   - Left, Right, Suck and Do Nothing

4. **Percept sequences**
   - Where is the agent?
   - Whether the location contains dirt?

→ *Under this circumstance, the agent is rational.*
Omniscience, learning, and autonomy
Omniscience

Omniscient agent
• Knows the actual outcome of its actions in advance
• No other possible outcomes
• However, impossible in real world
• Example?

Rational agent
• Maximize performance measure given the percepts sequence to date and prior knowledge

Rationality is not perfection
Learning

Does a rational agent depend on only current percept?
- No, the past percept sequence should also be used
- This is called learning
- After experiencing an episode, the agent
  - should adjust its behaviors to perform better for the same job next time.
Autonomy

- If an agent just relies on the prior knowledge of its designer rather than its own percepts then the agent lacks **autonomy**

_A rational agent should be autonomous— it should learn what it can to compensate for partial or incorrect prior knowledge._

- E.g., a clock
  - No input (percepts)
  - Run only but its own algorithm (prior knowledge)
  - No learning, no experience, etc.
3. The Nature of Environments

- The task environment
- Automated Taxi Driver
- Software Agents
- Properties of task environments
The task environment

- **Include:**
  - Performance measure
  - Environment
  - Agent’s Actuators
  - Agent’s Sensors.

- **First step in designing an agent**
  - Describe **PEAS** as fully as possible
Automated Taxi Driver

Performance measure:
- How can we judge the automated driver?
- Which factors are considered?
  - getting to the correct destination
  - minimizing fuel consumption
  - minimizing the trip time and/or cost
  - minimizing the violations of traffic laws
  - maximizing the safety and comfort, etc.
Automated Taxi Driver

**Environment:**
- A taxi must deal with a variety of roads
- Traffic lights, other vehicles, pedestrians, stray animals, road works, police cars, etc.
- Interact with the customer
Automated Taxi Driver

- **Actuators (for outputs)**
  - Control over the accelerator, steering, gear shifting and braking
  - A display to communicate with the customers

- **Sensors (for inputs)**
  - Detect other vehicles, road situations
  - GPS (Global Positioning System) to know where the taxi is
  - Many more devices are necessary
## Automated Taxi Driver

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Performance Measure</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxi driver</td>
<td>Safe, fast, legal, comfortable trip, maximize profits</td>
<td>Roads, other traffic, pedestrians, customers</td>
<td>Steering, accelerator, brake, signal, horn, display</td>
<td>Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard</td>
</tr>
</tbody>
</table>

**PEAS description of the task environment for an automated taxi**
Software Agents

- Sometimes, the environment may not be the real world
  - E.g., flight simulator, video games, Internet
  - They are all artificial but very complex environments
  - Those agents working in these environments are called
    - Software agent (softbots)
    - Because all parts of the agent are software
## Properties of Task Environment

<table>
<thead>
<tr>
<th>Fully observable</th>
<th>Partially observable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single agent</td>
<td>Multiagent</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Stochastic</td>
</tr>
<tr>
<td>Episodic</td>
<td>Sequential</td>
</tr>
<tr>
<td>Static</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Discrete</td>
<td>Continuous</td>
</tr>
<tr>
<td>Known</td>
<td>Unknown</td>
</tr>
</tbody>
</table>
Properties of Task Environment

- **Observable**: The agent’s sensory gives it access to the complete state of the environment.

- **Single agent**: An agent operating by itself in an environment.

- **Deterministic**: The next state of the environment is completely determined by the current state and the actions selected by the agent.
Properties of Task Environment

**Episodic:** The agent’s experience is divided into independent “episodes,” each episode consisting of agent perceiving and then acting.

- Quality of action depends just on the episode itself, because subsequent episodes do not depend on what actions occur in previous episodes.

→ Do not need to think ahead

**Static:** The environment is unchanged while an agent is deliberating
Properties of Task Environment

- **Discrete**: A limited number of distinct, clearly defined percepts and actions.

- **Known**:
  - In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given.
  - Obviously, if the environment is unknown, the agent will have to learn how it works in order to make good decisions.
<table>
<thead>
<tr>
<th>Environment</th>
<th>Accessible</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chess with a clock</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Semi</td>
<td>Yes</td>
</tr>
<tr>
<td>Chess without a clock</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Poker</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Backgammon</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Taxi driving</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Image-analysis system</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Semi</td>
<td>No</td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Refinery controller</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Interactive English tutor</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Properties of Task Environment

- The simplest environment is
  - Fully observable, deterministic, episodic, static, discrete and single-agent.

- Most real situations are:
  - Partially observable, stochastic, sequential, dynamic, continuous and multi-agent.
4. The Structure of Agents

- Agent programs
- Types of Agent Programs
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
- Learning agents
Agent Programs

- **Input for Agent Program**
  - Only the current percept

- **Input for Agent Function**
  - The entire percept sequence
  - The agent must remember all of them

- **Implement the agent program as**
  - A look up table (agent function)
Agent Programs

```plaintext
function TABLE-DRIVEN-AGENT(\textit{percept}) returns an action
  
  \textbf{persistent}: \textit{percepts}, a sequence, initially empty
  \textit{table}, a table of actions, indexed by percept sequences, initially fully specified

  append \textit{percept} to the end of \textit{percepts}

  \textit{action} ← LOOKUP(\textit{percepts}, \textit{table})

  return \textit{action}
```

Skeleton design of an agent program
Agent Programs

- $P =$ the set of possible percepts
- $T =$ lifetime of the agent
  - The total number of percepts it receives
- Size of the look up table
  - Consider playing chess $\sum_{t=1}^{T} |P|^t$
    - $P = 10, T = 150$
    - Will require a table of at least $10^{150}$ entries
Agent programs

- Despite of huge size, look up table does what we want.

- The key challenge of AI
  - Find out how to write programs that, to the extent possible, produce rational behavior
    - From a small amount of code
    - Rather than a large amount of table entries
  - E.g., a five-line program of Newton’s Method
  - V.s. huge tables of square roots, sine, cosine, ...
Types of agent programs

- Five types
  1. Simple reflex agents
  2. Model-based reflex agents
  3. Goal-based agents
  4. Utility-based agents
  5. Learning agents
1. Simple reflex agents

- It uses just *condition-action rules*
  - The rules are like the form “if ... then ...”
  - Efficient but have narrow range of applicability
  - Because knowledge sometimes cannot be stated explicitly
- Work only
  - if the environment is *fully observable*
1. Simple reflex agents

```plaintext
function SIMPLE-REFLEX-AGENT(percept) returns an action
  persistent: rules, a set of condition–action rules

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```
1. Simple reflex agents
percepts
(size, motion)

**RULES:**

(1) If small moving object, then activate SNAP

(2) If large moving object, then activate AVOID and inhibit SNAP

ELSE (not moving) then NOOP

*Action*: SNAP or AVOID or NOOP

needed for completeness
2. Model-based Reflex Agents

- For the world that is partially observable
  - the agent has to keep track of an internal state
  - That depends on the percept history
  - Reflecting some of the unobserved aspects
  - E.g., driving a car and changing lane

- Requiring two types of knowledge
  - How the world evolves independently of the agent
  - How the agent’s actions affect the world
2. 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
```
2. Model-based Reflex Agents
## Example Table Agent With Internal State

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saw an object ahead, and turned right, and it’s now clear ahead</td>
<td>Go straight</td>
</tr>
<tr>
<td>Saw an object Ahead, turned right, and object ahead again</td>
<td>Halt</td>
</tr>
<tr>
<td>See no objects ahead</td>
<td>Go straight</td>
</tr>
<tr>
<td>See an object ahead</td>
<td>Turn randomly</td>
</tr>
</tbody>
</table>
3. Goal-based agents

- Current state of the environment is always not enough
- The goal is another issue to achieve
  - Judgment of rationality / correctness
- Actions chosen $\rightarrow$ goals, based on
  - the current state
  - the current percept
3. Goal-based agents

CONCLUSION

- Goal-based agents are less efficient
- but more flexible
  - Agent ⇐ Different goals ⇐ different tasks

Search and planning

- two other sub-fields in AI
- to find out the action sequences to achieve its goal
3. Goal-based agents

![Diagram of goal-based agents](image)

- **Agent**
  - **Goals**
  - **State**
  - **How the world evolves**
  - **What my actions do**
  - **What the world is like now**
  - **What it will be like if I do action A**
  - **What action I should do now**

- **Environment**
  - **Sensors**
  - **Actuators**
4. Utility-based agents

- Goals alone are not enough
  - to generate **high-quality** behavior
  - E.g. meals in Canteen, good or not?

- Many action sequences $\rightarrow$ the goals
  - some are better and some worse
  - If goal means success,
  - then **utility** means the degree of success (how successful it is)
4. Utility-based agents
4. Utility-based agents

- It is said state A has higher utility
  - If state A is more preferred than others
- Utility is therefore a function
  - that maps a state onto a real number
  - the degree of success
4. Utility-based agents

Utility has several advantages:

- When there are conflicting goals,
  - Only some of the goals but not all can be achieved
  - Utility describes the appropriate trade-off
- When there are several goals
  - None of them are achieved certainly
  - Utility provides a way for the decision-making
Learning Agents

- After an agent is programmed, can it work immediately?
  - No, it still need teaching

- In AI,
  - Once an agent is done, we teach it by giving it a set of examples
  - Test it by using another set of examples

- We then say the agent learns
  - A learning agent
Learning Agents

Four conceptual components

1. Learning element → Making improvement
2. Performance element → Selecting external actions
3. Critic → Tells the Learning element how well the agent is doing with respect to fixed performance standard. (Feedback from user or examples, good or not?)
4. Problem generator → Suggest actions that will lead to new and informative experiences.
Learning Agents
Individual Assignment 1 (10 mins)

For each of the following activities, give a PEAS description of the task environment in your opinion: (Choose as much activities as you like, minimum is 2)

a) Playing soccer
b) Shopping for used AI books on the Internet.
c) Playing a tennis match.
d) Practicing tennis against a wall.
e) Performing a high jump.
f) Knitting a sweater.
g) Bidding on an item at an auction.
Homework #1

- Read chapter 1 (page 1-29) and 2 (page 34-59) in the textbook (3rd edition)

- Answer the questions
Next class

- Individual Assignment 1
- Chapter 2: Solving Problems by Searching