

MODULE -1

1.1 What is AI?

Thinking vs. Acting: Some people define AI by how it thinks like a human, while others define it by how it behaves.

Like Humans vs. Perfect Performance: Some measure AI by how similar it is to human actions, while others look at how well it performs, even if it's not like a human.

TWO MAIN METHODS

Human-Cantered: This method focuses on making AI act like humans and tests it through experiments.

E.g. Imagine you have a robot that can chat with you. You want it to talk like a friend, answer your questions, and even tell jokes. To make sure it acts like a real person, you test it by having it talk to different people and see how well it keeps up with the conversation. The goal is to make the robot interact in a way that feels natural and friendly.

Rationality-Cantered: This method uses math and engineering to make AI that makes the best decisions.

E.g. Let's say you have an AI that helps you shop online. You tell it that you want to buy a new phone, and it searches through hundreds of options to find the best one for your budget, needs, and reviews. It doesn't try to think like a human; it just uses data and calculations to make the smartest choice. The goal here is to get the best result, even if the process isn't human-like.

Both Methods Are Useful: Even though people might disagree, both methods have helped improve AI in different ways.

ACTING HUMANLY: THE TURING TEST APPROACH

The Turing Test: Alan Turing proposed this test in 1950 to check if a computer can show intelligence. The idea is to see if a computer can act so much like a human that a person can't tell whether they're talking to a computer or a human.

How It Works: In the test, a person asks questions to both a computer and a human, but only through text (like on a computer screen). If the person can't tell which is which, the computer passes the test.

What the Computer Needs: To pass the test, a computer needs:

- **Natural Language Processing:** To understand and respond in a human language.
- **Knowledge Representation:** To remember and use information from the conversation.
- **Automated Reasoning:** To think and answer questions based on the information it has.

- **Machine Learning:** To learn from new information and recognize patterns.

Real Time Example

You ask Google Assistant, "What time does the nearest coffee shop close?"

1. Natural Language Processing (NLP):

- **Understanding the Request:** Google Assistant uses NLP to interpret the question. It understands that you're asking about the closing time of a nearby coffee shop.
- **Extracting Information:** It breaks down your question into key parts: "what time" (a time-related question), "nearest coffee shop" (a location-based search), and "close" (refers to closing hours).

2. Knowledge Representation:

- **Storing and Using Information:** Google Assistant has access to stored information like your location, local businesses, and their operating hours. It uses this knowledge to find relevant coffee shops near you.
- **Context:** Google Assistant uses your current location to narrow down the search to coffee shops that are physically closest to you.

3. Automated Reasoning:

- **Answering the Question:** Google Assistant searches through its database of coffee shop information. It looks at all nearby coffee shops, checks their closing times, and determines which one is closest to you.
- **Reasoning:** It reasons that since you're asking about closing time, you likely want to know the nearest coffee shop that's still open. It might exclude shops that have already closed for the day.

4. Machine Learning:

- **Learning from Patterns:** Over time, Google Assistant learns your preferences. If you often visit a specific coffee shop, it might prioritize showing you the closing time of that shop first.
- **Improving Accuracy:** If you correct Google Assistant's results, like specifying a different coffee shop, it learns and improves future responses.

Total Turing Test: This is a more advanced version that also includes seeing and interacting with objects. To pass this test, the computer would need:

Computer Vision: To see and recognize objects.

Robotics: To move and handle objects.

AI and the Turing Test: Most AI work focuses on how computers interact with people rather than trying to pass the Turing Test. For example, AI might need to follow human-like rules to explain its decisions or have conversations, but it doesn't always try to mimic human behaviour perfectly.

THINKING HUMANLY: THE COGNITIVE MODELLING APPROACH

The Cognitive Modelling Approach in AI is about making computers think and learn like humans.

Mental Representations: It stores information the way we do in our minds. For example, if you think of a dog, the system will remember it as "four legs," "barks," and "furry."

Processes: It tries to copy how we make decisions or solve problems. For example, if you see a barking animal with four legs, you might decide it's a dog. The system does the same.

Learning: The system learns from experience, like how we learn from our mistakes and get better over time.

This approach helps create AI that can understand and react like a human, making it more useful and relatable.

THINKING RATIONALLY: THE LAWS OF THOUGHT APPROACH

Purpose: This approach focuses on reasoning based on formal rules or logic to make decisions or solve problems. It involves applying well-defined principles of logic to ensure that conclusions are valid given the premises.

Example: Imagine you have a basic email filtering system that uses logical rules to categorize incoming emails:

Rule 1: If an email contains the word "urgent," mark it as "High Priority."

Rule 2: If an email is from a known contact, mark it as "Safe."

Rule 3: If an email does not meet any of the above criteria, mark it as "General."

Scenario: You receive an email with the subject "Urgent: Please Review This Document." According to

Rule 1, the system recognizes the word "urgent" and applies the rule to mark the email as "High Priority," regardless of the sender or other content.

How It Demonstrates Thinking Rationally: The system uses predefined logical rules (laws of thought) to analyse the content of the email and categorize it. The approach is rational because it applies consistent rules to achieve the desired outcome.

ACTING RATIONALLY: THE RATIONAL AGENT APPROACH

Purpose: This approach focuses on taking actions that are expected to achieve the best outcome, considering the current knowledge and situation. It's more about making decisions that lead to effective results rather than strictly following predefined rules.

Example: Consider a navigation app like Google Maps:

Situation: You need to get to a meeting on time.

Information: You input your destination and current location into the app.

Rational Action: The app calculates the best route based on real-time traffic data, road conditions, and estimated travel time. It may suggest taking a different route if there is heavy traffic on the usual path.

Scenario: You start your drive, and halfway through, the app detects unexpected traffic congestion on your route. It reassesses the situation and suggests an alternative route that will get you to your destination faster, despite the original route being the most direct.

How It Demonstrates Acting Rationally: The app takes into account the current conditions and adjusts its recommendations to ensure you reach your destination efficiently. It acts rationally by adapting to new information and choosing actions that optimize the outcome (getting you to your meeting on time).

In summary:

Main Idea Behind These Approaches:

To explore different ways AI can replicate or simulate human intelligence and decision-making—either by copying human behavior, thinking like humans, following logic, or simply choosing the best actions.

Aim of These Concepts:

To develop intelligent systems that:

- Communicate and behave like humans,
- Think and learn from experiences,
- Use logical reasoning to solve problems,
- Make smart decisions in real-time environments.

Benefits of the Rational Agent Approach

Flexibility: This approach is more flexible than just focusing on logic because being rational is not only about following logical rules.

Easier to Study: It's easier to study because the idea of acting rationally can be applied in many different situations, unlike trying to understand complex human behaviour.

Example: Instead of only making the robot follow strict rules (like "always move straight ahead"), the rational agent approach allows the robot to adapt to different situations, like avoiding obstacles and learning new strategies.

Challenges with Perfect Rationality

The text points out that always making perfect decisions is not possible in complicated situations because it would require too much computation. But learning about perfect decision-making helps understand the basics of AI.

Example: If the robot tries to find the ball in a huge room with many obstacles, it can't always calculate the perfect path. Instead, it uses simpler rules to get close to the ball, which helps it learn and improve over time.

1.2 THE STATE OF THE ART

"state of the art" refers to the current highest level of knowledge or performance achieved in AI area.

1. Chess Grandmaster vs. Computer

A top chess player, Arnold Denker, plays against a computer named HITECH. Even though Denker is very skilled, the computer beats him. This is the first time a computer wins against a top human chess player.

2. Travel Booking with PEGASUS

A traveller wants to book a flight from Boston to San Francisco. She talks to a program called PEGASUS that understands her speech. Even though the program makes a few mistakes, it still books the cheapest flight correctly because it understands the conversation.

3. Spacecraft Monitoring with MARVEL

At a space lab, a system called MARVEL watches over the data from the Voyager spacecraft. When it finds a problem, it tells the scientists, who can then fix it. This helps catch and solve issues quickly.

4. Autonomous Vehicle Driving

A person is driving on a highway but hasn't touched the controls for 90 miles. An advanced robot inside the car is driving it by using cameras and sensors to understand the road.

5. Expert System for Medical Diagnosis

A doctor uses a computer program to help with a difficult medical case. At first, the doctor doubts the program's diagnosis, but when the program explains its reasoning, the doctor realizes it was correct.

6. Traffic Monitoring System

A camera above a busy street keeps track of traffic. It notices cars, reports accidents, and can even call for emergency help if there's a serious issue.

These examples show how artificial intelligence (AI) helps with different tasks. From winning at chess to booking flights, watching spacecraft, driving cars, helping doctors, and monitoring traffic, AI makes many things easier and smarter.

2.1 WHAT IS AN AGENT?

An agent is something that can sense what's around it and then do something in response.

- **Human Agent:** For people, the sensors are our eyes, ears, and other senses. The effectors are our hands, legs, and mouth.
- **Robotic Agent:** For robots, the sensors are cameras and infrared sensors. The effectors are the motors that help the robot move and act.
- **Software Agent:** For computer programs, the sensors are data inputs, and the effectors are the actions the program takes based on the data.

A rational agent is one that does the right thing. Obviously, this is better than doing the wrong thing, but what does it mean? As a first approximation, we will say that the right action is the one that will cause the agent to be most successful. That leaves us with the problem of deciding how and when to evaluate the agent's success

2.2 HOW AGENTS SHOULD ACT

1. **What is a Rational Agent?** A rational agent is something that makes the best decisions to achieve its goals based on what it knows.
2. **Measuring Success:** To know how well an agent is doing, we use a performance measure. This is a way to check how successful the agent is. For example, if the agent is a vacuum cleaner, we could measure how much dirt it picks up. We might also look at how much electricity it uses and how noisy it is.
3. **Timing for Evaluation:** We should measure the agent's performance over a longer period to see how it performs consistently. If we only looked at how well it works in the first hour, we might reward those that start quickly but don't work much later.
4. **Rationality vs. Perfect Knowledge:** Being rational means making the best choice based on the information you have. It doesn't mean knowing everything. For example, if you cross the street and a plane's cargo door falls off unexpectedly, it doesn't mean you were irrational for crossing the street. You made a reasonable decision based on what you could see and know at the time.

5. **Ideal Rationality:** An ideal rational agent should choose actions that are expected to give the best results based on what it knows. For instance, if crossing a busy road without looking is risky, a rational agent would first look both ways to avoid danger.

6. **Examples and Limits:** Not all agents are perfect. For example, a clock can be thought of as a simple agent that always shows the correct time. However, if you move to a different time zone, the clock won't adjust itself because it doesn't have the ability to sense time zone changes. This doesn't mean the clock is being irrational; it just has limited abilities.

Understanding Agent Behaviour and Design

1. **Behaviour and Percept Sequences:** An agent's actions are based on what it has perceived so far. This history of what it has sensed is called the "percept sequence." To understand how an agent behaves, we might want to list out every possible situation and what the agent would do. But this list can be very long or even endless.

2. **Mapping from Percept's to Actions:** Instead of listing every possibility, we create a "mapping" that tells us what actions the agent should take for different percept sequences. Knowing this mapping helps us understand how the agent should behave. For ideal agents, this mapping helps us design their actions.

3. **Simpler Examples:** Sometimes, we don't need a huge table. Take a calculator's square-root function, for example. The percept sequence is the number you enter, and the action is showing the square root of that number. We don't need to list every possible number and its square root. Instead, we use a formula to calculate it. This makes it simpler and more efficient.

4. **Compact and General Design:** We can design efficient agents that use mappings to handle many tasks. For instance, a calculator uses a formula to find square roots rather than having a massive table of results. Similarly, more complex agents can use general rules to solve various problems without needing an exhaustive list of actions for every possible situation.

Real Time Example

Example: A Basic Smart Thermostat

1. **Agent's Behaviour:** Imagine a smart thermostat in your home. The thermostat's job is to keep your room at a comfortable temperature.

2. **Percept Sequences:** The thermostat's "percept sequence" includes information it senses, like the current room temperature and your desired temperature.

3. Mapping Actions: The thermostat uses this information to decide what action to take. For example:

- If the room is too cold (say, below 68°F) and you want it to be 72°F, the thermostat will turn on the heater.
- If the room is too hot (above 75°F), it will turn on the air conditioner.

4. Creating a Simple Rule: Instead of listing every possible temperature and action, we use a simple rule:

- **Rule:** “If the room temperature is lower than the desired temperature, turn on the heater. If it’s higher, turn on the air conditioner.”

5. How It Works: You don’t need a huge table with every temperature scenario. The thermostat uses this rule to decide what to do based on the current temperature.

6. Efficiency: By using a simple rule, the thermostat handles many different situations efficiently. It doesn’t need to have a specific action listed for every single temperature; it just follows the rule to keep your room comfortable.

AUTONOMY

Built-in Knowledge vs. Autonomy:

- **Built-in Knowledge:** This means the agent does what it's programmed to do without changing based on what’s happening around it.
- **Autonomy:** This means the agent can learn from its experiences and change its behaviour based on what it learns.

Why Autonomy Matters:

- An autonomous agent can adapt and change its behaviour based on new experiences. For example, animals learn from their environment and adjust their actions accordingly.
- The dung beetle example shows a lack of autonomy: it only works well when everything goes as expected. If something changes, it gets confused and doesn’t adapt.

Initial Knowledge and Learning:

- Agents might need some basic built-in knowledge to start with, so they can function initially. Over time, they should be able to learn from their experiences and improve how they work.

Built-in Knowledge:

- **Example:** Think of a coffee machine that’s programmed to make a cup of coffee at 8 AM every day. It doesn’t matter if you’re home or not; it just follows the schedule. If you’re late or want

coffee at a different time, the machine won't adjust on its own—it sticks to the programmed time.

Autonomy:

- **Example:** Now, imagine a smart coffee machine with a timer and a smartphone app. If you're running late, you can use the app to adjust the time or even set it to make coffee when you get home. Additionally, it can learn from your habits over time and suggest coffee times based on your schedule or even when you wake up.

2.3 STRUCTURE OF INTELLIGENT AGENTS

how intelligent agents work on the inside. It covers three main ideas:

1. What is an agent made of?

- An **agent** is something that can sense its environment and take action.
- To do this, it needs two things:
 - An **agent program** – this is like a brain that gives instructions based on what the agent senses.
 - An **architecture** – this is the hardware or system (like a computer or robot) that runs the program and performs actions.

◆ **Simple formula: Agent = Program + Architecture**

2. What should we think about before building an agent?

Before creating an agent, we need to decide:

- What kind of **information** will it receive?
 - What kind of **actions** can it take?
 - What are the **goals** it should achieve?
 - What kind of **environment** will it work in?
-

3. What are examples of environments?

The environment can be:

- **Simple:** Like a robot on a conveyor belt that just accepts or rejects parts.
- **Complex:** Like a software agent that works in a flight simulator or searches the internet—here, it has to make more decisions.

Sometimes, real and digital environments mix, like when a software agent reacts to a live camera feed of a room with a person in it.

The ****Turing Test**** is a famous example where computers and humans are tested equally, but it's really hard for computers to match human performance.

Agent Type	Percept's	Actions	Goals	Environment
Medical diagnosis system	Symptoms, findings, patient's answers	Questions, tests, treatments	Healthy patient, minimize costs	Patient, hospital
Satellite image analysis system	Pixels of varying intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite
Part-picking robot	Pixels of varying intensity	Pick up parts and sort into bins	Place parts in correct bins	Conveyor belt with parts
Refinery controller	Temperature, pressure readings	Open, close valves; adjust temperature	Maximize purity, yield, safety	Refinery
Interactive English tutor	Typed words	Print exercises, suggestions, corrections	Maximize student's score on test	Set of students

Figure 2.3 Examples of agent types and their PAGE descriptions.

1. What is a smart agent?

- It's a system (like a robot or software) that:
 - **Sees or senses** what's happening around it (this input is called a **percept**),
 - Then **decides what to do** based on that percept.

2. How does the smart agent work inside?

Even a simple smart agent will:

1. **Receive** new information (percept),
2. **Store** it in its **memory**,
3. **Use** what it knows (from memory) to decide the **best action**,
4. **Take the action**, and

5. **Remember** what it just did.

Then, it repeats this over and over.

3. Two Important Concepts:

◆ a) Percept and Memory

- The agent **only gets one piece of information at a time**, not everything at once.
- So, if it wants to use past information, it has to **save it in memory**.
- In **easy tasks**, the agent might not need memory.
- In **complex tasks**, remembering past inputs becomes important.

◆ b) Goal and Performance

- The **goal** (what success looks like) is usually **not inside the agent**.
- Instead, someone from the **outside** checks if the agent is doing a good job.
- The agent can still work well even if it doesn't know exactly what the goal is.

How does this robot work?
Step-by-step:
1. It senses the floor (percept):
○ It moves around and senses: "Is this spot dirty or clean?"
2. It stores this info in memory:
○ It remembers: "I already cleaned this area," or "This corner was dirty last time."
3. It decides what to do:
○ If it sees dirt → clean it.
○ If it's already clean → move to a new spot.
4. It performs the action:
○ It vacuums the dirty spot.
5. It updates memory:
○ "This spot is now clean."
Then it repeats this process again and again.

What is a table-driven agent?

Imagine you are building a robot. Instead of thinking for itself, the robot:

- **Remembers everything it sees (called percepts),**
- Looks up the **exact match** of what it has seen before in a **big table**, and
- **Finds the action** from that table.

So, it works like:

👉 "If I see this, do that."

✓ How it works (step-by-step):

1. The agent **collects all its past inputs** (percepts).
2. It **checks the table**: "Have I seen this exact sequence before?"
3. If yes, it **picks the action** listed there.
4. Then it **performs that action**.

✗ Why this doesn't work well:

1. **Too Big to Handle**
 - The table becomes **huge**. For example, a chess-playing agent would need **3.5 billion entries** to cover every situation.
2. **Takes Too Long to Create**
 - You would have to list **every possible situation** and **every correct action** one by one.
3. **No Thinking Ability**
 - The agent can't **adapt or make decisions** on its own. If something new happens, it will **fail** because that case is **not in the table**.
4. **Learning is Slow**
 - Even if the agent is allowed to **learn and fill the table**, it will take a **very long time**.

Real time Example

Automated Customer Service Chatbot

How the Lookup Table Works:

Percept Sequence: The chatbot remembers everything it receives from the user, such as questions and responses. For example, a user might type:

"What are your business hours?"

"Do you have a refund policy?"

"How can I contact customer support?"

Using the Lookup Table: The chatbot uses this memory to look up the right responses in a predefined table:

If the user types “What are your business hours?”, the chatbot looks up the table and finds the answer: “Our business hours are from 9 AM to 5 PM.”

If the user types “Do you have a refund policy?”, it finds the response: “Yes, we offer a 30-day refund policy.”

It continues this for every query based on its memory.

Why This Approach Won't Work:

Huge Table Size:

If the chatbot tries to cover all possible customer inquiries and responses, it would need a massive table. Imagine a company with hundreds of products and countless variations of customer questions; the table could become unmanageable.

Time to Create the Table:

Building this table would take a long time. The team would need to anticipate every possible question and create appropriate responses, which is practically impossible.

No Independence:

The chatbot cannot think for itself. If a user asks a question that isn't in the table, like “What should I do if my product is damaged?”, it won't know how to respond and will likely provide an unhelpful answer or say it doesn't understand.

Learning Problems:

If the chatbot had a learning feature, it could improve over time by adding new questions and answers. However, updating the table with new entries based on user interactions would be slow and cumbersome. It could take a long time to fill in new responses correctly and comprehensively.

Now we need to figure out how to create a real program that can turn what an agent senses (percepts) into actions. Different parts of driving suggest using different kinds of agent programs. We will look at four types:

1. Simple reflex agents
2. Agents that keep track of the world
3. Goal-based agents
4. Utility-based agents

Simple Reflex Agents

- **How they work:**

- They **react immediately** to what they sense using fixed rules.
 - They do **not remember** anything from the past.
 - **Example:**
 - A self-driving car sees an obstacle and stops right away.
 - **Limitation:**
 - They **can't handle complex or changing situations**.
 - They work only for **very basic tasks**.
-

Agents That Keep Track of the World

- **How they work:**
 - They **remember past information** and **track changes** in the environment using an **internal memory (internal state)**.
 - **Example:**
 - A self-driving car that remembers where other cars were, even if they are no longer visible.
 - **Benefit:**
 - Can make **smarter decisions** and handle **more complex situations** than reflex agents.
-

Goal-Based Agents


- **How they work:**
 - These agents choose actions that **help them reach a goal**.
 - They ask: **"What should I do now to reach my goal?"**
- **Example:**
 - A self-driving car that decides how to **reach a destination safely and quickly**.
- **Benefit:**

- They can **plan ahead** and are more **flexible**.

Utility-Based Agents

- **How they work:**
 - These agents look at **different options**, give each one a **value (called utility)**, and choose the best one.
 - They try to **maximize total performance**, not just reach a goal.
- **Example:**
 - A self-driving car that tries to:
 - Reach its destination,
 - Use less fuel,
 - Avoid traffic violations,
 - Give passengers a smooth ride.
- **Benefit:**
 - These agents are the **smartest** because they **balance multiple goals** and make **the best overall decisions**.

Summary Table:

Type of Agent	Memory	Goal Awareness	Decision Quality	Example
Simple Reflex Agent	✗ No	✗ No	 Basic	Stop when obstacle is detected

Type of Agent	Memory	Goal Awareness	Decision Quality	Example
Model-Based Agent (Track World)	☑ Yes	✗ No	☐ Smarter	Remembers where cars are
Goal-Based Agent	☑ Yes	☑ Yes	🎯 Goal-driven	Plans best route to destination
Utility-Based Agent	☑ Yes	☑ Yes	💡 Best choice	Balances safety, speed, comfort

2.4 ENVIRONMENT

In this section, and in the exercises at the end of the chapter, you'll learn how to connect an agent to an environment. Earlier, we talked about different types of agents and environments. No matter the type, the connection between them works the same way: the agent does something in the environment, and the environment sends information (called percepts) back to the agent.

First, we'll look at the different kinds of environments and how they affect the design of agents. Then, we'll talk about creating environment programs that act as testing areas for agent programs to see how well they perform.

Properties of Environments

Different environments can be classified based on several factors:

1. Accessible Environment

🔑 Definition:


An accessible environment is one where the agent can get all the information it needs to make the best possible decision.

- The agent doesn't need to guess or assume.
- Everything it needs to sense (see, hear, read, etc.) is available.

Real-life Example:

Self-checkout machine in a supermarket

- It knows what items you're scanning.
- It sees the price, quantity, and payment.
- All the necessary data is available through its sensors and scanners.

 **Conclusion:** The self-checkout machine doesn't need to guess anything. It has complete info. So, it's in an accessible environment.

2. Inaccessible Environment

Definition:


An inaccessible environment is one where the agent cannot see or sense everything it needs to make the best decision.

- It may have to guess, remember past events, or predict missing info.
- This is harder and needs smarter decision-making.

Real-life Example:

Self-driving car during heavy fog

- It can't clearly see the road, pedestrians, or traffic lights.
- Sensors are limited due to fog.
- It has to make safe guesses using partial data and past knowledge.

 **Conclusion:** The car doesn't have complete information. So, it's working in an inaccessible environment.

2. Deterministic vs. Nondeterministic:

1. Deterministic Environment = "No Surprises"

Meaning: The environment always behaves the **same way** when you do something. You can **predict the result** every time.

□ Simple Example:

You press a button on a **vending machine** to get a juice.

- You press "B2" → you **always** get the juice.
- The machine does **exactly** what you ask.
- There's **no randomness**.

□ Same input → same output

□ For the agent: It doesn't need to guess. It **knows** what will happen.

✗ 2. Nondeterministic Environment = "Things Might Not Go as Planned"

Meaning: Even if the agent does the same thing, the result **might change** every time. The environment is **uncertain** or has **randomness**.

🚗 Simple Example:

A robot is trying to **cross the road**.

- It waits for no cars and starts walking.
- Suddenly, a car speeds up or someone runs across.
- Even though the robot did the right thing, it got a different result.

□ Same input → different output

□ For the agent: It must be **prepared for surprises** and **adjust its plan**.

3. Episodic vs. No episodic:

1. Episodic Environment = “One-time tasks”

Meaning: Each task or situation is independent.

What the agent does now has no impact on what happens next.

□ The agent:

- Doesn't need to remember the past
 - Doesn't need to plan for the future
 - Just focuses on one episode at a time
-

Real-life Example:

Image recognition system (like Google Photos)

- You give it one photo.
- It checks and says: “This is a dog.”
- Then you give another photo — it starts fresh.
- It doesn't care what it said about the last photo.

 Each photo = one episode, fully independent.

✗ 2. Non-Episodic Environment = “Actions build up over time”

Meaning: The agent's current actions affect the future.

□ The agent:

- Needs to remember past events
 - Must think about how actions today affect tomorrow
 - Needs to plan ahead
-

Real-life Example:

Playing a chess game

- Your move now affects how the game unfolds.
- If you make a bad move now, you might lose later.
- Every move is connected.

🔄 One move affects the next — so it's non-episodic.

4. Static vs. Dynamic:

Static Environment = “Nothing changes while thinking”

Meaning:

The environment stays unchanged while the agent is deciding or acting.

□ **The agent:**

- Has more time to think
 - Doesn't need to hurry
 - Doesn't need to watch for sudden changes
-

🔗 **Real-life Example:**

Solving a crossword puzzle on paper

- The clues don't change while you're solving it.
- The puzzle sits still until you finish it.
- You can think for 5 minutes — and the puzzle will still be the same.

🔄 So, it's a static environment.

Dynamic Environment = “Things change while you're deciding”

Meaning:

The environment may change even while the agent is still thinking or deciding.


□ **The agent:**

- Has to act fast
 - Must keep checking for changes
 - Needs to handle unpredictability
-

Real-life Example:

Driving a car in traffic

- While you're deciding to turn right, a car may suddenly move or a person may cross.
- You can't wait too long — things are changing around you.

 So, driving is a dynamic environment.

5. Discrete vs. Continuous:

Discrete: The environment has a limited number of choices or actions. For example, in chess, there are a fixed number of moves.

Continuous: The environment has an infinite range of possible actions or states, like driving a car, where speed and position constantly change.

Discrete: **Tic-tac-toe** is a discrete environment. There are only a limited number of moves you can make, and each one is clearly defined.

Continuous: **Playing soccer** is a continuous environment. The ball can be anywhere on the field, and the players' positions and actions change fluidly, without fixed steps.

Each type of environment poses different challenges for an agent to operate successfully.

Environment programs

Environment Program and Simulator:

Basic Idea: This is about how agents (like robots or virtual assistants) interact with their surroundings, and how we can simulate that using a program.

Environment Simulator:

What it does:

1. It gives the agent information about the surroundings (called "percepts").
2. It receives the agent's response (like a chosen action).
3. It updates the environment based on the agent's action and other factors (like weather or traffic).

Initial State and Update:

The environment starts with a setup, and the agent's actions change it according to certain rules (called the "update function"). For example, if a robot vacuum moves, the environment updates to show the vacuum in a new location.

Real vs. Simulated Environment:

- The agent should be able to work both in a simulated world (inside a computer) and the real world, as long as it gets the same information and responds in the same way.

Running the Environment:

RUN-ENVIRONMENT: This is the process that keeps the simulation going. The agents interact with the environment, make decisions, and act over and over.

Measuring Agent Performance:

Performance Measure: To see how well the agent is doing, we need to measure its performance. For simple agents (like chatbots), you can just observe them. For more complex ones (like self-driving cars), special code is used to track their performance.

RUN-EVAL-ENVIRONMENT: These checks how well the agent performs its tasks and gives a score based on its performance.

Example: Imagine testing a ****self-driving car**** in a simulator. The simulator gives the car info like the road layout, traffic lights, and nearby cars. The car makes decisions (like when to speed up or stop). The simulator updates the environment (like the road conditions) based on the car's actions. Then, it checks how well the car performed—did it avoid accidents? Follow traffic laws? This information is used to score the car's performance, helping to test how it would behave in the real world.

Real time Example

Environment Program and Simulator (Robot Vacuum Cleaner Example):

Basic Idea: A robot vacuum cleaner interacts with the environment (your home) and can also be tested in a virtual simulation of that environment.

Environment Simulator:

- The **simulator** would act like your home's layout, giving the robot vacuum cleaner information about the rooms, furniture, walls, and dirt spots.
- The vacuum receives this info, decides how to clean, and moves around.
- The simulator then **updates** the environment: if the robot moves, the simulator shows its new position. If the robot cleans a dirty spot, the dirt disappears in the simulator.

Initial State and Update:

- When the robot vacuum starts, it has a map of the house. As it moves, the map updates to show which parts are cleaned and where the vacuum is currently located.

Real vs. Simulated Environment:

- The robot can work in the **real world** (your home) or in a **simulator** (a virtual version of your home). In both cases, it gets the same type of information—like obstacles and dirt—and makes decisions based on that.

Running the Environment:

- In the simulation, the vacuum interacts with the environment continuously. It moves, cleans, avoids obstacles, and updates its map until the task is complete.

Measuring Agent Performance:

- In the simulation, you measure how well the vacuum did by checking:

- How much dirt was cleaned.
- How efficiently it moved around (did it avoid bumping into furniture?).
- Did it miss any areas?