Chapter 2: Agents and Environments

Dr Ammar Masood
Department of Cyber Security,
Air University Islamabad

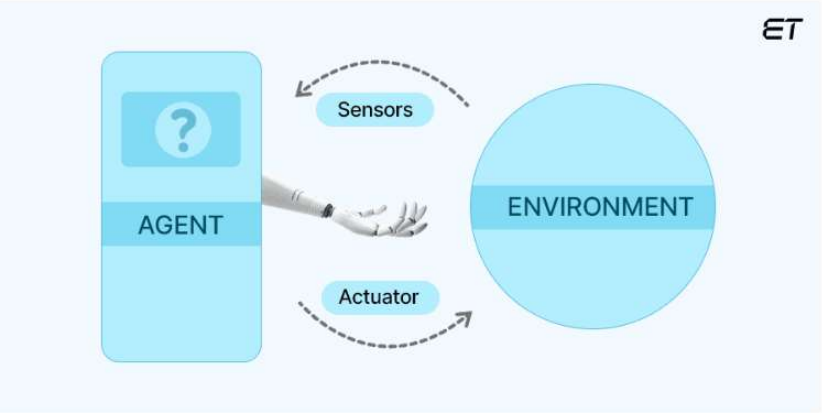
1

Contents

- Agents and environments
- Structure of intelligent agents
- Agent types
- Simple reflex agent
- Model-based reflex agent
- Goal based agent
- Utility based agent
- Learning agent

2

Agents and Environments



What is an intelligent agent?

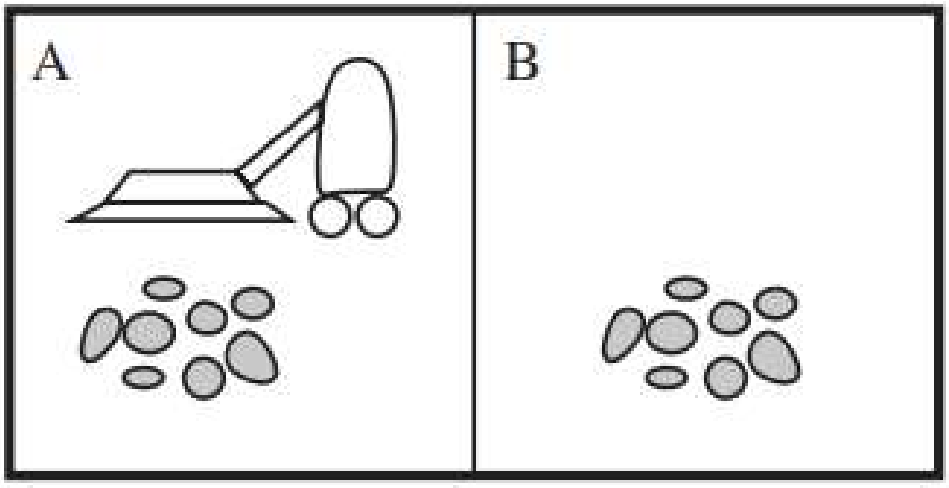
- An agent is anything that can be viewed as **perceiving its environment through sensors** **acting upon that environment through actuators**
- A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.
- A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators.
- A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets

Agent-Environment Cycle:

Perception: The agent receives input (percepts) from the environment.
Action: The agent performs actions based on its current state and its knowledge.

Example:
A robot vacuum cleaner that senses dirt (percept) and moves around to clean the floor (action).

5



A vacuum with just two locations

6

The vacuum cleaner example

- This particular world has **just two locations**: squares A and B. The vacuum agent **perceives which square it is in and whether there is dirt in the square.** It can **choose to move left, move right, suck up the dirt, or do nothing**. One very simple agent function is the following: if the current square is dirty, then suck; otherwise, move to the other square

Sequence of actions

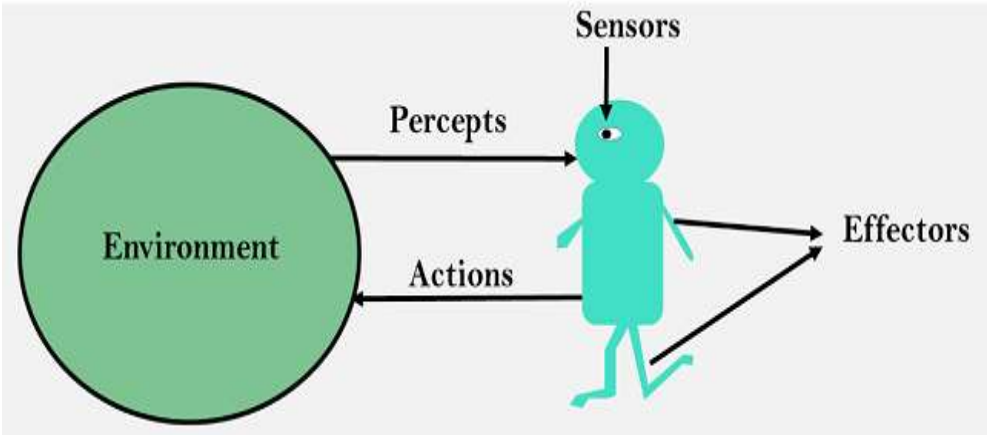
Percept Sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
...	...
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
...	...

Structure of intelligent agents

Key Components

- **Sensors:** Devices or processes that perceive the environment.
- **Actuators:** Mechanisms through which the agent acts (motors, speakers, etc.).
- **Agent Function:** Maps percept histories to actions.

9



Environment → [Sensors] → Agent → [Actuators] → Environment

10

Agent program

- The job of AI is to design an agent program **that implements the agent function— the mapping from percepts to actions.** We assume this program will run on some sort of computing device with physical sensors and actuators—we call this the architecture:

agent = architecture + program

Example

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*

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
…	…
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
…	…

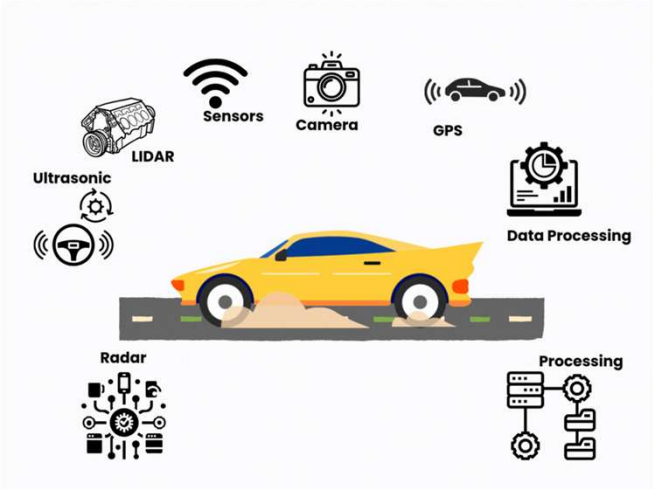
How it works?

1. The agent observes the environment and receives a **percept** (sensor input).

2. It stores this percept in a list of past percepts.

3. It checks the table to **find the action corresponding to the stored sequence of percepts.**

4. It performs the retrieved action.



In autonomous vehicles, sensors (cameras, radar, LiDAR) feed data into the vehicle's control system (the agent) that then makes driving decisions via actuators (steering, brakes).

How Agents should Act?

- An agent should be
  - a **rational agent** that does the **right thing**. (what is this?)
  - 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.

- Solution is;
  - **A fixed performance measure evaluates the sequence of observed action effects on the environment.**

15

How Agents should Act?

Example; Consider the case of an agent that is supposed to vacuum a dirty floor.

- In case of "*How*" work with the **evaluating performance measure.**
  - First, the performance measure would be the amount of dirt cleaned up in a single eight-hour shift.
  - Second, performance measure would factor in the amount of electricity consumed and the amount of noise generated as well.
  - Third, performance measure might give highest marks to an agent that not only cleans the floor quietly but also efficiently.
- The "*when*" of **evaluating performance measure** is also important.
  - If we should measure how much dirt the agent had cleaned up in the first hour of the day?
  - Either we query for appreciation or punishment to get positive results

16

PEAS Factors

Tasks Parameters for Agents to perform accurately

➢Major factors to evaluate the agents actions are;

- Use PEAS to describe task
  - Performance measure
  - Environment
  - Actuators
  - Sensors

Example: Autonomous Taxi

- Performance measure: safe, fast, comfortable (maximize profits).
- Environment: roads, other traffic, pedestrians, customers.
- Actuators: steering, accelerator, brake, signal, horn.
- Sensors: cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors.

17

Environment Parameters

Environment Parameters for Agents to perform accurately:

i) Fully observable vs. partially observable
ii) Deterministic vs. stochastic / random
iii) Episodic vs. sequential
iv) Static vs. dynamic
v) Discrete vs. continuous
vi) Single agent vs. multiagent

i) Fully observable vs. partially observable

- monitoring the specific tasks (either Fully or partial).
- agent's sensory apparatus gives it access to the **complete/partial state** of the environment, then we say that the **environment is accessible (fully or partial)** to that agent.

Example :

- Chess is the fully observable scenario
- Autonomous Taxi driving is the partially observable scenario

18

Environment Parameters

ii) Deterministic vs Stochastic

- well-known knowledge of particular tasks.
- If the next state of the environment is completely determined by the current state **then deterministic.** (e.g., move an object in chess, next state is deterministic).
- If the next state of the environment is uncertain situation, determined by the current state **then Stochastic**. (e.g., driving car is acting as stochastic, as not sure about other objects motion).

iii) Episodic vs Sequential

- series of separate parts (either **sequential** or **episodic**).
- the agent's experience is divided into "episodes." Each **episode** consists of the agent perceiving and then acting.
- In each episode the agent receives a percept and then performs a single action. Crucially, **the next episode does not depend on the actions taken in previous episodes.**

(e.g., spotting defective parts on assembly line is episodic, playing chess or autonomous driving is a sequential task).

19

Environment Parameters

iv) Static vs dynamic

- knowledge about properties of environment of a system. (either static or dynamic).
- **Static environments** are easy to deal with because the agent need not keep looking at the world. (e.g., solving crossword puzzle).
- If the environment can change while an agent is deliberating (thinking), then we say the **environment is dynamic**. (e.g., driving a car is acting as dynamic).

v) Discrete vs Continuous

- individually separate or without interruption (either discrete or continuous).
- There are a fixed number of possible moves on each turn, then **Discrete**. (e.g., chess is acting as discrete).
- Agent receives numerous range of continuous values, then its Continuous. (e.g., driving car)

vi) Single agent vs. multiagent

- An agent solving a crossword puzzle by itself is clearly in a single-agent environment
- An agent playing chess is in a two agent environment (other agent could be another program or human being).

20

Examples

Environments	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock						

Fully observable vs. partially observable
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

Examples

Environments	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Deterministic	Sequential	Static	Discrete	Multi

Fully observable vs. partially observable
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

Examples

Environments	Obser vable	Determi nistic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Deterministic	Sequential	Static	Discrete	Multi
Taxi driving						

Fully observable vs. partially observable
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

Examples

Environments	Obser vable	Determi nistic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Deterministic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochastic	Sequential	Dynamic	Continuous	Single

Fully observable vs. partially observable
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

Examples

Environments	Obser vable	Determi nistic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Deterministic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochastic	Sequential	Dynamic	Continuous	Single
Robot part picking	Fully	Deterministic	Episodic	Static	Discrete	Single

Fully observable vs. partially observable
Deterministic vs. stochastic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

Agent types

Note:
Each type builds upon the previous, from basic condition–action rules to agents that learn and adapt.

Simple reflex agent

- Agents that act solely based on the current percept, following pre-defined condition–action rules.

if *car-in-front-is-braking* **then** *initiate-braking.*

How they work?

- They ignore history and internal state.
- Respond immediately to stimuli.

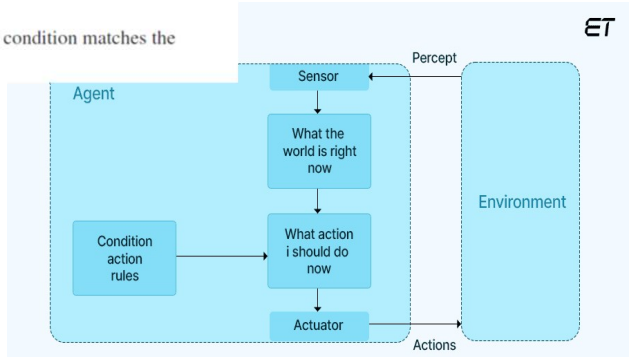
A basic thermostat that turns on the heater when the temperature drops below a set point.

Algorithm simple reflex agent

```
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
```

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.



Example

- A simple reflex agent can be used in a vending machine. The vending machine has a set of predefined rules that dictate what action to take when a user inserts a coin and selects a product. For example:
- If the user inserts a coin, then dispense the product.
- If the user selects a product, then dispense the change.
- The vending machine responds directly to the current state of the environment (i.e., the user inserting a coin or selecting a product) **without considering past experiences or future predictions.**

Strengths and Weaknesses

Strengths:

- They are simple and easy to implement
- Simple reflex agents are fast and efficient
- These AI agents are suitable for well-defined environments

Weaknesses:

- They have limited adaptability
- They cannot learn from past experiences
- These agents require a fully observable environment

Model based reflex agent

- Agents that maintain an internal model (state) of the world, enabling them to handle partially observable environments.

A robot vacuum that maps out the layout of a room to avoid obstacles and cover the floor efficiently.

How they work?

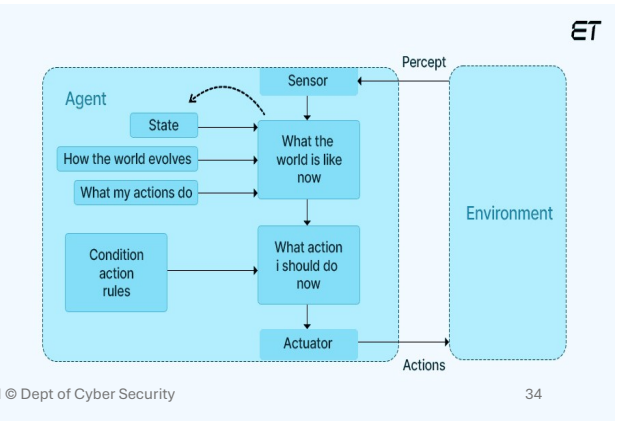
- Use a model to update their internal state based on past percepts
- Make decisions using both current percepts and stored state.

The interesting part is the function UPDATE-STATE

Algorithm for model based reflex agent

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



Example

- A model-based reflex agent can be used in a self-driving car. The self-driving car has an internal model of the environment that includes information about the road conditions, traffic patterns, and nearby obstacles. The car uses this internal model to predict the consequences of its actions and make decisions accordingly. For example:
- If the car detects a pedestrian crossing the road, then slow down and prepare to stop.
- If the car detects a traffic jam ahead, then change lanes to avoid the congestion.
- The self-driving car uses **its internal model to predict the consequences of its actions and make decisions accordingly, rather than simply responding to the current state of the environment.**

35

Benefits and drawbacks

Benefits:

- They can handle partially observable environments
- Model-based agents are more flexible
- They can use the internal model to make predictions about how the environment might react to their actions

Drawbacks:

- There is an increased level of complexity:
- The agent's performance relies heavily on the accuracy of its internal model
- There is always an issue of limited learning

36

Goal based agents

- Agents that make decisions by considering the future consequences of their actions and strive to achieve defined goals.

A navigation system that plans a route from point A to point B while avoiding traffic.

37

How they work?

- They have a goal (or set of goals) that guides decision-making.
- They use search or planning techniques to choose actions that lead to goal satisfaction.

the agent needs some sort of goal information that describes situations that are desirable

38

Algorithm of goal based agent

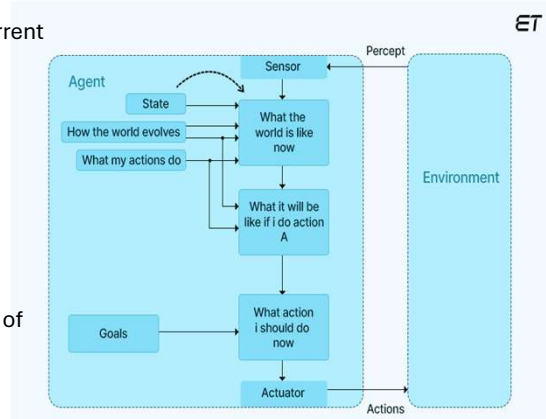
function GOAL-BASED-AGENT(percept) returns an **action**
  persistent: *state*, the agent's current conception of the world state
    goal, the desired goal state
    model, a description of how the next state depends on current state and action
    actions, a set of possible actions
    action, the most recent action, initially none

  state ← **UPDATE-STATE**(state, action, percept, model)

  if GOAL-REACHED(state, goal) **then**
    return STOP-ACTION()

  plan ← **SEARCH**(state, goal, actions, model) // Find a sequence of actions to reach goal
  action ← SELECT-ACTION(plan)

  return **action**



Example

A simple example of a goal-based agent is a robot that wants to navigate from point A to point B in a maze. **The robot's goal is to reach point B**.

1. **Goal Definition**: The robot defines its goal as reaching point B.
2. **Planning**: The robot uses a planning algorithm to determine the best course of action to reach point B. The algorithm considers the robot's current location, the location of point B, and the obstacles in the maze.
3. **Action Selection**: The robot selects the best action to take based on the planning algorithm. For example, the robot might decide to move north to reach a point where it can see point B.
4. **Action Execution**: The robot executes the selected action by moving north.
5. **Percept**: The robot receives feedback from the environment about the outcome of its action. For example, the robot might detect a wall in front of it and realize that it needs to change direction.
6. **Goal Achievement**: The robot checks if its goal has been achieved. If not, it repeats the planning and action selection process.

Advantages and Weaknesses

Advantages:

- They can adapt their behavior depending on the current situation
- These AI agents function in environments with multiple possible outcomes
- They have solid reasoning capability

Weaknesses:

- The planning algorithms can be computationally expensive
- Defining clear goals is crucial for the agent's success
- If the agent doesn't have complete information about the environment, its planning might be flawed

41

Utility based agents

- Agents that not only pursue goals but also evaluate how "good" a state is by assigning a utility value.

An investment advisory system that recommends portfolios based on risk (utility) and return.

42

How they work?

- They balance multiple goals or preferences.
- They choose actions that maximize overall utility.

An agent's utility function is essentially an internalization of the performance measure. If the internal utility function and the external performance measure are in agreement, then an agent that chooses actions to maximize its utility will be rational according to the external performance measure

Algorithm of utility based agent

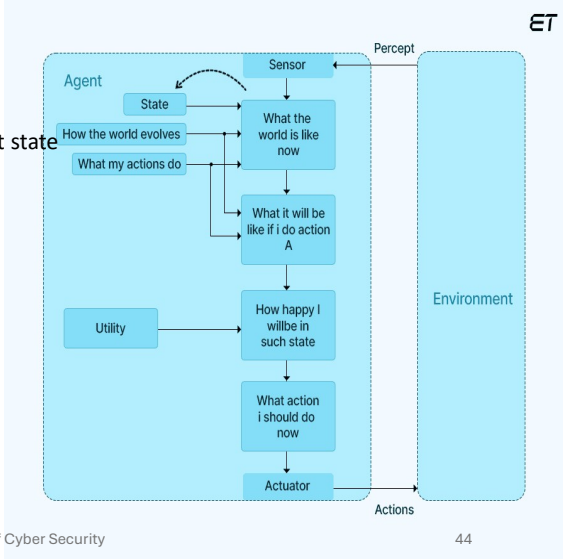
function UTILITY-BASED-AGENT(percept) returns an **action**
  persistent: *state*, the agent's current conception of the world state
    utility, a function that maps states to a measure of desirability
    model, a description of how the next state depends on current state and action
    actions, a set of possible actions
    action, the most recent action, initially none

  state ← **UPDATE-STATE**(state, action, percept, model)

  if GOAL-REACHED(state) then
    return STOP-ACTION()

  best_action ← argmax a ∈ actions UTILITY(RESULT(state, a))

  return **best_action**



Example

A simple example of a utility-based agent is a financial advisor app **who wants to maximize the returns on a client's investment portfolio.**

1. **Goal Definition**: The financial advisor defines its goal as maximizing the returns on the client's investment portfolio.
2. **Utility Function**: The financial advisor defines a utility function that evaluates the desirability of different investment options based on their expected returns and risk levels.
3. **Action Selection**: The financial advisor selects the best investment option based on the utility function. For example, the financial advisor might decide to invest in a high-risk, high-return stock if the utility function indicates that the expected returns outweigh the risk.
4. **Action Execution**: The financial advisor executes the selected investment option by buying or selling the stock.
5. **Percepts**: The financial advisor receives feedback from the market about the outcome of its investment decision.
6. **Goal Achievement**: The financial advisor checks if its goal has been achieved. If not, it repeats the process of evaluating investment options and selecting the best course of action.

Benefits and Limitations

Benefits:

- These AI agents are flexible and adaptive
- They can incorporate the agent's preferences and priorities into their decision-making process
- Utility-based agents can consider factors like risk, time, and effort when evaluating different options

Limitations:

- Designing the utility function is complex
- Evaluating the utility of all possible outcomes can be computationally expensive
- There is a degree of uncertainty about the outcomes

Learning agents

- Agents that can improve their performance over time by learning from their experiences.

An AI-based game-playing agent that learns strategies by playing many rounds.

47

How they work?

- They include components for learning and adapting.
- Often use techniques from reinforcement learning, supervised learning, or unsupervised learning.

A learning agent can be divided into four conceptual components. The most important distinction is between the **learning element**, which is responsible for making improvements, and the **performance element**, which is responsible for selecting external actions. The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions. The learning element uses **CRITIC** feedback from the critic on how the agent is doing and determines how the performance element should be modified to do better in the future.

48

Learning agents

1.Critic

a) **Role:** Evaluates the agent's performance by comparing it to a predefined standard and provides feedback.

b) **Example:** In a **self-driving car**, the critic monitors how smoothly and safely the car follows traffic rules and gives feedback on any deviations.

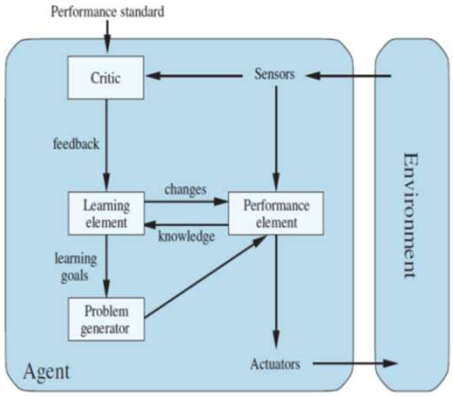
2.Learning Element

a) **Role:** Improves the agent's performance by using feedback from the critic to update knowledge.

b) **Example:** The **self-driving car** refines its driving decisions over time by learning from past mistakes, such as adjusting braking distance based on road conditions.

3.Problem Generator

1. **Role:** Introduces new situations for the agent to explore, encouraging learning and innovation.
2. **Example:** The **self-driving car** might deliberately test different routes or braking strategies in a simulated environment to improve decision-making.



Learning agents

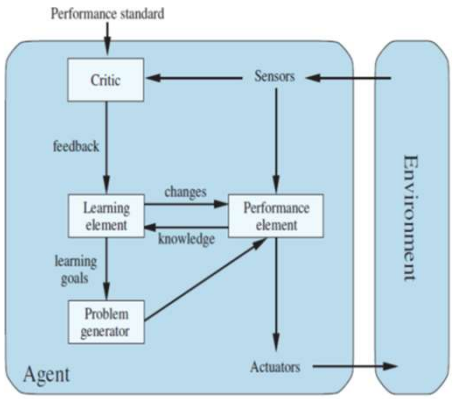
4. Performance Element

a) **Role:** Makes decisions and takes actions based on the learned knowledge.

b) **Example:** The **self-driving car** detects traffic lights using **sensors**, decides when to stop or go, and sends commands to **actuators** like the steering wheel and brakes.

How It Works Together

1.Sensors detect the environment (e.g., road conditions, obstacles).
2.Performance Element decides the best action (e.g., slow down at a red light).
3.Critic evaluates the decision (e.g., Was the stop smooth? Was it too late?).
4.Learning Element adjusts behavior based on feedback.
5.Problem Generator suggests new learning scenarios (e.g., What if it's raining?).



Benefits and drawbacks

Benefits:

- Learning agents can adjust to new situations and environments by continuously improving their performance
- They can handle complex tasks
- These AI agents have real-world applicability

Drawbacks:

- Learning agents require a significant amount of data and time
- The agent needs to balance exploring new options for learning with exploiting its current knowledge for good performance
- Understanding how a learning agent arrived at a particular decision can be challenging

51

Summary and key takeaways

- **Agents** interact with their environments via sensors and actuators.
- **Structure:** The design of an agent (from simple reflex to learning) influences its ability to handle complexity.
- **Agent Types:** Each type offers a trade-off between simplicity and capability—choosing the right type depends on the problem.
- **Real-World Implications:** Understanding these types helps in designing systems for robotics, autonomous vehicles, and intelligent software.

52

Conclusive summary between agents

Agent Type	Key Characteristics	Example
Simple Reflex Agent	- Reacts solely to current percepts - Uses fixed condition–action rules	Basic thermostat
Model-Based Reflex	- Maintains internal state - Updates state based on percept history	Robot vacuum with a room map
Goal-Based Agent	- Considers future consequences - Uses search/planning for decision-making	Navigation system planning routes
Utility-Based Agent	- Evaluates multiple outcomes - Maximizes a utility function for decision-making	Investment advisory system
Learning Agent	- Adapts and improves from experience - Uses learning algorithms	Game-playing AI (reinforcement learning in chess or Go)