

# Reinforcement Learning

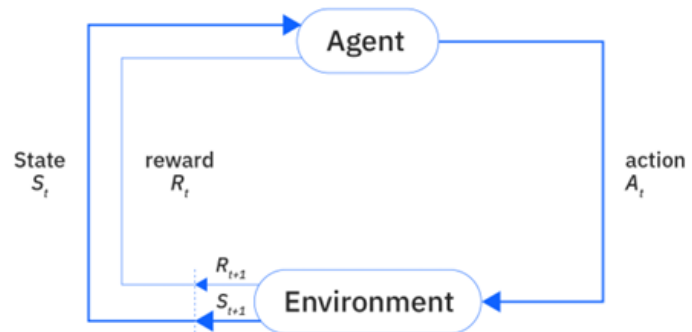


---

FATEME TAROODI

## Reinforcement Learning (RL)

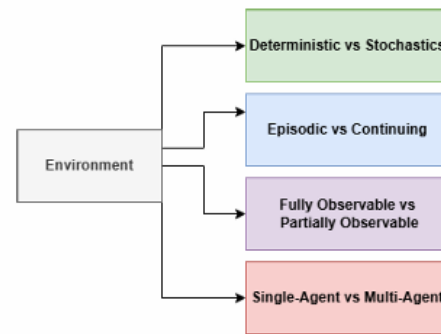
**Reinforcement Learning (RL)** is a branch of machine learning that focuses on how agents can learn to make decisions through trial and error to **maximize cumulative rewards**. RL allows machines to learn by interacting with an environment and receiving feedback based on their actions. This feedback comes in the form of rewards or penalties.



Fateme Taroodi

## Components of Reinforcement Learning

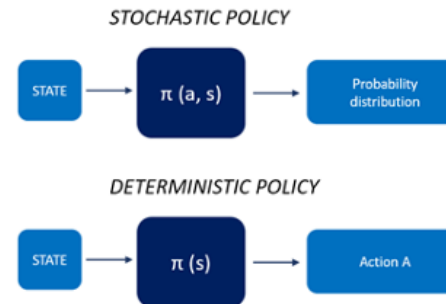
- **Agent:** The decision-maker that performs actions.
- **Environment:** The **external system** that the agent interacts with. It provides **feedback** to the agent based on its actions.



- **State(s)** : A representation of the **current situation** or configuration of the environment.
- **Action (a):** The **choices** or **decisions** made by the agent at each time step.  
Action Space : The Action space is the set of **all possible actions in an environment**.
  - *Discrete space:* the number of possible actions is **finite**.
  - *Continuous space:* the number of possible actions is **infinite**. (A Self Driving Car agent has an infinite number of possible actions since it can turn left 20°, 21,1°, 21,2°, honk, turn right 20°...)

Fateme Taroodi

- **Reward (r):** The **immediate feedback** the agent receives after taking an action in a state. It guides the agent towards maximizing its cumulative reward over time. The reward can be **positive** (to encourage behavior) or **negative** (to discourage behavior).
- **Policy( $\pi$ ):** A **strategy** used by the agent to determine which action to take based on the current state. It can be:
  - **Deterministic:** A specific action is taken for a given state.
  - **Stochastic:** The action is chosen probabilistically from a distribution.



Fateme Taroodi

## Deterministic Policy:

A **deterministic policy** provides a **single action** for each state, meaning that in any given state, the agent always takes the same action.

A deterministic policy is denoted as:  $\pi(s)=a$

where:

- $\pi(s)$  is the action chosen by the agent when in state  $s$ .
- $a$  is the action selected, and it is a **specific** action for state  $s$ .

**Intuition:** A deterministic policy means there is no randomness in action selection; the agent always follows a predefined path. Once the agent is in a state  $s$ , it knows exactly which action  $a$  to take.

**Example:** In a **chess game**, a deterministic policy could be: "When the board is in state  $s$  (e.g., the knight is positioned here), always move the knight to position  $a$  (a specific square)."

## Stochastic Policy:

A **stochastic policy** defines a **probability distribution** over actions for each state. In this case, the agent might take different actions with different probabilities depending on the state it is in.

A stochastic policy is denoted as:  $\pi(a|s) = \Pr \{A_t = a \mid S_t = s\}$  where  $\pi(a|s)$  is the probability of selecting action  $a$  when in state  $s$ .

**Intuition:** A stochastic policy introduces randomness in the decision-making process. Instead of always choosing the same action in a given state, the agent chooses an action based on a **probability distribution** that may vary over time or depending on other factors.

**Example:** In a **self-driving car**, if the car is in state  $s$  (e.g., driving at a certain speed on a road), it might choose actions like turning left, turning right, or going straight with probabilities, such as:  $\pi(\text{left}|s)=0.2$ ,  $\pi(\text{right}|s)=0.3$ ,  $\pi(\text{straight}|s)=0$

In this case, the policy doesn't force the car to always take the same action when in state  $s$ , but instead, it takes actions based on certain probabilities.

## Types of RL Tasks:

- **Episodic tasks:** In this case, we have a starting point and an ending point (**a terminal state**). **This creates an episode:** a list of States, Actions, Rewards, and new States. (For instance, think about Super Mario Bros: an episode begin at the launch of a new Mario Level and ending **when you're killed or you reached the end of the level.**)

**Episode:** An **episode** in **Reinforcement Learning (RL)** refers to a single, complete run of an agent interacting with an environment, starting from an initial state and ending in a terminal state. In an episode, the agent makes a series of decisions by selecting actions, receives rewards, transitions through states, and eventually reaches a terminal state or satisfies some stopping condition.

$$S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, \dots, S_T$$

- **Continuing tasks:** These are tasks that continue forever (no terminal state). In this case, the agent must learn how to choose the best actions and simultaneously interact with the environment. (For instance, an agent that does **automated stock trading**. For this task, there is no starting point and terminal state. The agent keeps running until we decide to stop them.

## Markov Decision Process

A Markov decision process (MDP) is a Markov reward process with decisions. It is an *environment* in which all states are Markov.

### Definition

A *Markov Decision Process* is a tuple  $\langle S, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $S$  is a finite set of states
- $\mathcal{A}$  is a finite set of actions
- $\mathcal{P}$  is a state transition probability matrix,  
 $\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$  is a reward function,  $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- $\gamma$  is a discount factor  $\gamma \in [0, 1]$ .

### Model of the Environment:

A **model of the environment** in RL refers to a function or system that **predicts the next state and reward** given the current state and action. More formally, it consists of two key components:

- **Transition Function:**  $P(s'|s,a) \rightarrow$  Probability of reaching state  $s'$  after taking action  $a$  in state  $s$ .
- **Reward Function:**  $R(s,a) \rightarrow$  Reward received after taking action  $a$  in state  $s$ .

If an RL algorithm learns or uses this model, it is **Model-Based**. If it does not learn or use this model and directly learns from experience, it is **Model-Free**.

Fateme Taroodi

## Return

### Definition

The *return*  $G_t$  is the total discounted reward from time-step  $t$ .

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

- $\gamma$  is the **discount factor** ( $0 \leq \gamma \leq 1$ )
- The discount factor adjusts the value of future rewards to make them less impactful than immediate rewards.
- When  $\gamma=1$ , future rewards are valued the same as immediate rewards. The agent treats all rewards equally, regardless of when they occur.
- When  $\gamma=0$ , only the immediate reward matters. The agent ignores all future rewards and only cares about what happens immediately.

Fateme Taroodi



## Value Functions:

### Definition

The *state-value function*  $v_{\pi}(s)$  of an MDP is the expected return starting from state  $s$ , and then following policy  $\pi$

$$v_{\pi}(s) = \mathbb{E}_{\pi} [G_t \mid S_t = s]$$

### Definition

The *action-value function*  $q_{\pi}(s, a)$  is the expected return starting from state  $s$ , taking action  $a$ , and then following policy  $\pi$

$$q_{\pi}(s, a) = \mathbb{E}_{\pi} [G_t \mid S_t = s, A_t = a]$$

**State-Value Function** represents how good a particular state is for an agent, considering the long-term rewards the agent can expect starting from that state. It evaluates the expected return (or future rewards) when an agent starts in a given state and follows a particular policy.

**Action-Value Function** evaluates the quality of taking a specific action in a given state and then following a particular policy thereafter. It tells the agent how good a particular action is in a particular state in terms of expected future rewards.

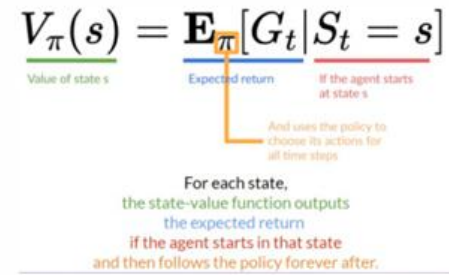
### State - Value Function:

$$V_{\pi}(s) = \mathbf{E}_{\pi}[G_t | S_t = s]$$

Value of state  $s$       Expected return      If the agent starts at state  $s$

And uses the policy to choose its actions for all time steps

For each state,  
the state-value function outputs  
the expected return  
if the agent starts in that state  
and then follows the policy forever after.

The diagram shows the equation  $V_{\pi}(s) = \mathbf{E}_{\pi}[G_t | S_t = s]$ . The terms are color-coded:  $V_{\pi}(s)$  is green,  $\mathbf{E}_{\pi}$  is blue,  $G_t$  is orange, and  $S_t = s$  is red. An orange line connects the  $\pi$  in the expectation operator to the text 'And uses the policy to choose its actions for all time steps'. Below the equation, a paragraph explains the function's purpose for each state.

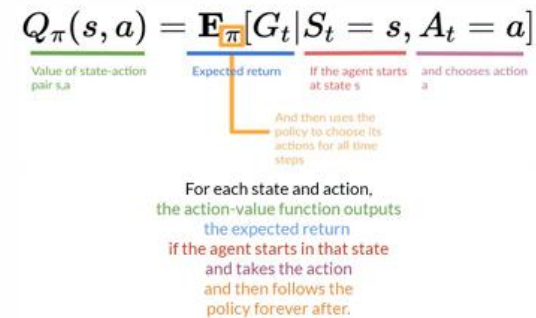
### Action - Value Function:

$$Q_{\pi}(s, a) = \mathbf{E}_{\pi}[G_t | S_t = s, A_t = a]$$

Value of state-action pair  $s, a$       Expected return      If the agent starts at state  $s$  and chooses action  $a$

And then uses the policy to choose its actions for all time steps

For each state and action,  
the action-value function outputs  
the expected return  
if the agent starts in that state  
and takes the action  
and then follows the policy forever after.

The diagram shows the equation  $Q_{\pi}(s, a) = \mathbf{E}_{\pi}[G_t | S_t = s, A_t = a]$ . The terms are color-coded:  $Q_{\pi}(s, a)$  is green,  $\mathbf{E}_{\pi}$  is blue,  $G_t$  is orange,  $S_t = s$  is red, and  $A_t = a$  is purple. An orange line connects the  $\pi$  in the expectation operator to the text 'And then uses the policy to choose its actions for all time steps'. Below the equation, a paragraph explains the function's purpose for each state and action.

Fateme Taroodi