Reinforcement Learning

Introduction and preliminary concepts

Jordi Casas Roma

jordi.casas.roma@uab.cat

September 9, 2025

UAB
Universitat Autònoma
de Barcelona

Table of Contents

Table of Contents

Introduction

Definition

Reinforcement learning

Reinforcement learning is an area of machine learning concerned with how intelligent agents ought to **take actions** in an **environment** in order to **maximize** the notion of cumulative **reward**.

Introduction
Definition

Main properties (1/4):

- The first noteworthy characteristic to highlight is the **absence of a supervisor** who dictates the correct course of action in each instance.
- In each state or situation, the **agent** receives, following each **action** (interaction with the **environment**), a signal known as a **reward**.
- The **reward signal** offers an indication of the relative goodness or badness of something (be it an action or a state) when compared to something else, but it does not precisely instruct us on what to do. It does not provide us with an absolute truth as labels do in supervised learning.

Introduction
Definition

Main properties (2/4):

- The second characteristic of reinforcement learning that sets it apart from other machine learning paradigms is that the **reward received may not be instantaneous**; it can be delayed over time.
- Consequently, when selecting an action, it may not yield its **desired rewards immediately** but rather lead to the desired reward much later.
  - In other words, most of the time, we cannot ascertain whether the chosen action is good or bad until some time has passed.
  - In fact, it is possible that an action may yield immediate benefits or rewards, but over time, it could lead to a catastrophic situation, resulting in a negative ultimate outcome (also known as the return).
- Hence, the element of **time** and the **sequential** nature of **actions** are crucial, as once an action is taken, it may be impossible to undo it later.

Introduction
Definition

Main properties (3/4):

- This leads us to the third characteristic, which is that, due to its sequential decision-making nature, where the agent receives data (observations of the environment), makes a decision, executes an action, and the environment responds with new data, and so forth, these **sequential data cannot be considered independent and identically distributed (i.i.d.)**, as is the case in most problems studied in supervised and unsupervised learning.
- A typical example of this situation would be the data received by an agent playing a video game emulator. After each action (a console or computer controller input), the data received from the environment (the screen images) closely resemble what was received in the previous action, thus exhibiting a **high degree of correlation**.

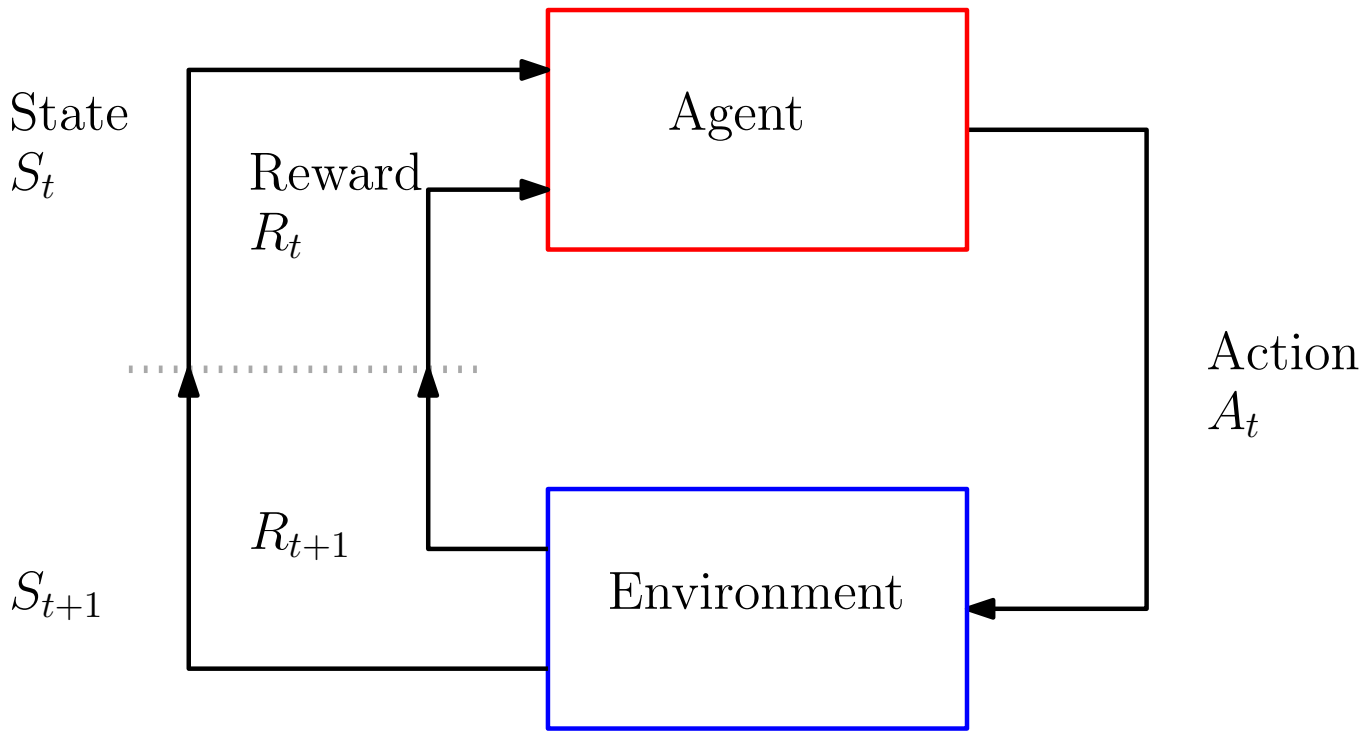
Introduction
Definition

Main properties (4/4):

- Lastly, the fourth characteristic that distinguishes reinforcement learning from other machine learning paradigms is that the **agent's actions affect subsequent interactions** and, consequently, the **data it receives from the environment**.
- In other words, the agent directly influences the data it receives from the environment.
- This feature, **absent in both supervised and unsupervised learning**, is perhaps the most distinctive and genuine aspect of reinforcement learning.
- Continuing with the example of the agent playing video games, depending on whether it moves the controller to one side or the other, the data it receives will be different. Therefore, the action (the controller movement in this case) completely conditions the data it receives (the screen image).

Introduction

Scenario



Introduction

Scenario

Main items in RL scenario:

- The agent
- The environment
- The reward signal
- The states
- The actions

Table of Contents

Introduction

Reward signal

The reward signal or function

The **reward** signal, denoted as R_t, is a **scalar value** that the agent receives at time t following each action it takes.

Introduction
Reward signal

Cumulative reward

The **cumulative reward** (also denoted as **return**) is defined as the accumulated value of all future rewards.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots$$

- Note that since these rewards are random, the return is also random and that is why the agent's objective is to **maximize its expected value**.

Introduction
Reward signal

- Given that the **number of episodes** (each time a reward is received after an action) may not be finite, the summation of rewards may not converge, resulting in an **infinite** return value.

Introduction
Reward signal

- Given that the **number of episodes** (each time a reward is received after an action) may not be finite, the summation of rewards may not converge, resulting in an **infinite** return value.

Cumulative reward

We redefine the **cumulative reward** (or **return**) as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

- *gamma* is the **discount factor**
- $0 \leq \gamma \leq 1$
  - $\lambda = 0 \rightarrow G_t = R_{t+1}$
  - $\lambda = 1 \rightarrow G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots$

Introduction

Reward signal

Example 1: **Maze**

How can we design an algorithm to find the exit path of a maze in the shortest possible time?

Introduction
Reward signal

Example 1: **Maze**

How can we design an algorithm to find the exit path of a maze in the shortest possible time?

A valid way to design the reward signal could be as follows:

$$\begin{cases} R_t = -1 & \text{for any movement except reaching the exit} \\ R_t = 0 & \text{if it reaches the exit} \end{cases}$$

- The **return value will always be negative**.
- Thus, the fewer movements the agent makes to exit the maze, the **greater its value will be**.

Introduction

Reward signal

Example 2: **Chess Player**

In the case of designing an agent to play chess, how could one define the reward signal to achieve this goal?

Introduction
Reward signal

Example 2: **Chess Player**

In the case of designing an agent to play chess, how could one define the reward signal to achieve this goal?

A valid way to design the reward signal could be as follows:

$$\begin{cases} R_t = 0 & \text{for any move except if the game is won} \\ R_t = 1 & \text{if the game is won} \end{cases}$$

- It is crucial to have a clear understanding of the ultimate goal and **avoid intermediate rewards** that could lead us astray from that objective.
  - For instance, if we establish a **reward for getting pawns to the end of the board and promoting them to queens**, the algorithm might focus on achieving that intermediate goal and leave the king unprotected, resulting in the eventual loss of the game, which was our ultimate objective.

Table of Contents

Introduction

Environment

The environment

The **environment** includes everything that is external to the agent.

Introduction
Environment

The environment

The **environment** includes everything that is external to the agent.

- In the case of the chess game it is ...
  - the board,
  - the pieces, and
  - the opposing player.

Introduction
Environment

The environment:

The **bidirectional communication** between the agent and the environment occurs through three signals:

- The **agent** can only influence the environment through the **actions** A_t chosen at each time step t.
- The **environment** responds to these actions by providing:
  - an **observation** O_{t+1} and
  - a **reward** signal R_{t+1}.

Introduction
Environment

The environment:

In each time step t the **agent**:

- Receives the observation O_t.
- Receives the reward R_t.
- Executes the action A_t.

Introduction
Environment

The environment:

In each time step t the **agent**:

- Receives the observation O_t.
- Receives the reward R_t.
- Executes the action A_t.

On the other hand, the **environment**:

- Receives the action A_t.
- Provides the observation O_{t+1}.
- Provides the reward R_{t+1}.

Table of Contents

Introduction

States

The agent's history

The **history** is the sequence of all actions, observations, and rewards from the beginning up to time step t:

$$H_t = \{O_0, A_0, R_1, O_1, A_1, R_2, O_2, A_2, \ldots, A_{t-1}, R_t, O_t\}$$

Introduction

States

The agent's history

The **history** is the sequence of all actions, observations, and rewards from the beginning up to time step t:

$$H_t = \{O_0, A_0, R_1, O_1, A_1, R_2, O_2, A_2, \ldots, A_{t-1}, R_t, O_t\}$$

- Hence, the history consists of **all the variables that can be observed**, both by the agent and the environment, over time.

Introduction

States

The agent's history

The **issue** with the history is that the **volume of variables** it stores increases over time, making it often unmanageable and therefore less useful for the learning algorithms we need to design.

Introduction
States

The agent's history

The **issue** with the history is that the **volume of variables** it stores increases over time, making it often unmanageable and therefore less useful for the learning algorithms we need to design.

- For instance, in the case of an **agent playing a video game emulator**, the history would need to store all the moves made by the agent, along with all the scores and game screens generated at each moment during gameplay.
- This would result in a **massive amount of data** to store.

Introduction

States

The state

The **state** S_t is the information used to determine what will happen in the next time step.

Formally, it is a function of the history:

$$S_t = f(H_t)$$

Introduction

States

The state

The **state** S_t is the information used to determine what will happen in the next time step.

Formally, it is a function of the history:

$$S_t = f(H_t)$$

- We can think of the state S_t as a **concise summary of the history** H_t but it must include all the information necessary to determine what will happen next, from both:
  - the agent's perspective (the agent needs to make decisions and take actions based on this state)
  - and the environment's perspective (the environment must generate the next observations and rewards based on the state).

Introduction

States

There are several definitions and classes of states. In addition to the definition we have already discussed, we will specify three more:

- **Environment state**: This refers to the information that captures the complete state of the environment, including everything that is not under the control of the agent. It encompasses all variables relevant to the agent-environment interaction.
- **Agent state**: This represents the internal state of the agent, including information that the agent has learned or derived from the environment. It may contain data like the agent's current strategy, beliefs, or any other information used for decision-making.
- **Information state**: Also known as the *belief state*, it includes all the information that the agent has about the environment. This state often combines the agent's observations and knowledge, reflecting the agent's beliefs or understanding of the current situation.

Introduction

States

Environment state

- The state of the environment, denoted as S_t^e, is a **private representation** of the environment.
- It consists of all the data that the environment uses **to generate the next observation and reward**.
- Typically, this information is **not visible** to the agent.
- The algorithms do not depend on this state, as most of the time, this state is hidden from the agent.
- The algorithm only observes the observations and rewards it receives from the environment, but not its state. Furthermore, even if the state S_t^e is observable by the agent, it may contain a lot of **irrelevant information for the agent**.

Introduction

States

Agent state

- The agent's state, denoted as S_t^a, is the **internal representation** that the agent forms based on the history.
- It encompasses any information that **the agent uses to choose its next action**, and therefore constitutes the information we will use to construct reinforcement learning algorithms.
- The agent's state, S_t^a, can be any **function of the history** ($S_t^a = f(H_t)$).
- It is up to us, as designers of the reinforcement learning algorithm, to determine what this state S_t^a should be, which observations and rewards should comprise it, which ones to discard, and how to combine them to condense all the necessary information that helps us select the best actions to achieve our goal.

Information state

Markov Property

A state S_t is called Markov (or Markovian) if and only if:

$$Pr\{S_{t+1}|S_t\} = Pr\{S_{t+1}|S_1, \ldots, S_t\}$$

- An **information state** (or **Markov state**) is one that contains **all the useful information** from the history H_t.
- The above definition states that the probability of the next state S_{t+1} conditioned on the current state is the same as if it is conditioned on the current state and all the previous states.
- This way, if we know the current state, we can disregard the entire past history to determine the next state.

Introduction
States

Fully observable and partially observable **environments**

- A **Fully observable** environment is the ideal case where the agent can observe the state of the environment and use it for choosing the action to take.
- In this case, the observation, the environment state, and the agent's state all coincide ($O_t = S_t^e = S_t^a$).
- When working with these types of environments, we can define one of the fundamental concepts that formally describe reinforcement learning problems: **Markov Decision Processes (MDPs)**.

Introduction
States

Fully observable and **partially observable environments**

- In **Partially observable** environments, the agent observes the environment indirectly and does not have a complete view of the environment and its state.
- For example,
  - A domino player cannot see their opponent's tiles.
  - An investor can see asset prices but may not know where those prices originate.
- In this case, the agent's state does not coincide with the environment's state ($S_t^e \neq S_t^a$). Mathematically, these types of problems are characterized through **Partially Observable Markov Decision Processes (POMDPs)**.

Introduction

States

Fully observable and **partially observable environments**

- In this case, it is up to the agent to construct a suitable representation of the state, denoted as S_t^a.
- There are several possibilities, among which we highlight:
  - **Using the Complete History**: This approach involves using the entire history as the state, i.e., $S_t^a = H_t$. The challenge with this solution, as mentioned earlier, is that it can often be impractical due to the length of the history.
  - **Statistical or Bayesian Approximation**: Here, the agent can make a statistical or Bayesian approximation to the state generation process and apply its own beliefs about what is happening in the environment by defining probabilities that characterize the different possibilities for the environment being in a specific state or another. In other words, $S_t^a = (Pr\{S_t^e = s_1\}, ..., Pr\{S_t^e = s_n\})$.
  - **Using a Recurrent Neural Network (RNN)**: An RNN can be employed to linearly combine both the agent's past actions and observations, followed by applying a non-linearity. The representation could be defined as $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$.

Table of Contents

Agents

Main components of an agent

A reinforcement learning agent may have one or more of the following elements:

- **Policy**, which governs how the agent selects actions.
- **Value function**, which is a function that tells us how good (in terms of the possible return) it is to be in a certain state or perform a certain action.
- **Model**, which is the agent's view of the environment, how the agent thinks the environment evolves.

Table of Contents

Agents
Policy

The **policy** $\pi(\bullet)$ is a function that indicates how the agent selects an action. It is a function that maps states S_t and actions A_t. Therefore, it explains the **agent's behaviour**. There are **two types of policies**:

Agents
Policy

The **policy** $\pi(\bullet)$ is a function that indicates how the agent selects an action. It is a function that maps states S_t and actions A_t. Therefore, it explains the **agent's behaviour**. There are **two types of policies**:

- **Deterministic**. The action to be taken 'a' is uniquely determined by the state 's' in which the agent is. Consequently, it is a direct mapping between state and action:

$$a = \pi(s)$$

Agents
Policy

The **policy** $\pi(\bullet)$ is a function that indicates how the agent selects an action. It is a function that maps states S_t and actions A_t. Therefore, it explains the **agent's behaviour**. There are **two types of policies**:

- **Deterministic**. The action to be taken 'a' is uniquely determined by the state 's' in which the agent is. Consequently, it is a direct mapping between state and action:

$$a = \pi(s)$$

- **Stochastic**. In this case, there can be more than one action to choose for each state, and what the policy does is assign a probability to each of those options:

$$\pi(a|s) = Pr\{A_t = a|S_t = s\}$$

  - where $\pi(a|s)$ is the probability of selecting the action 'a' in state 's'.
  - This is usually the case in **most real-life problems**.
  - A stochastic policy allows exploration of the environment, since it means that if we go through the same state twice, the agent can take different actions.

Table of Contents

Agents
Value function

There are two types of value functions, which are:

- The **state value function** $v_\pi(s)$ is a function that tells us how good (in terms of the possible return) it is to be in a certain state.
- The **action value function** (or a **state-action value function**) $q_\pi(s, a)$ is a function that tells us how good (in terms of the possible return) it is to be in a certain state and select a specific action.

Agents
Value function

The **state value function** $v_\pi(s)$ is a function that tells us how good (in terms of the possible return) it is to be in a certain state.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots|S_t = s] \end{aligned}$$

- This function calculates the expected value of all possible future accumulated rewards (the future return) if the agent is in state 's' and follows a policy $\pi(\bullet)$ from this instant on.
- It is, therefore, a **prediction of future rewards**.

Agents
Value function

The **action value function** (or a **state-action value function**) $q_\pi(s, a)$ is a function that tells us how good (in terms of the possible return) it is to be in a certain state and select a specific action.

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a] \end{aligned}$$

- This equation calculates the expected value of all possible future cumulative rewards (the future return) if the agent is in state 's', takes action 'a', and follows a policy $\pi(\bullet)$.
- This function predicts future rewards based on the action the agent selects.
- Therefore, a possible policy would consist of selecting that action that maximizes this function. This is the **basis of some of the most relevant reinforcement learning algorithms**.

Table of Contents

Agents
Model

A **model** is the agent's view of the environment, i.e. how the agent thinks the environment evolves. It is not the environment itself, but is intended to be a prediction of its behaviour.

Formally, there are **two types of models** or predictions:

- **Transition model** $\mathcal{P}$. It is a prediction of the next state that the environment will generate, of its dynamics. For example:

$$\begin{aligned} \mathcal{P}^a_{ss'} &= p(s'|s,a) \\ &\approx Pr\{S_{t+1} = s' \mid S_t = s, A_t = a\} \end{aligned}$$

- **Reward model** $\mathcal{R}$. It is a prediction of the expected value of the next immediate reward that the environment will generate.

$$\begin{aligned} \mathcal{R}^a_s &= r(s,a) \\ &\approx \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] \end{aligned} \tag{1}$$

Agents
Model

- When an agent has a model, it can predict what the environment will do at the next moment and act accordingly by selecting the appropriate actions.
- It is important to note that a good model does not provide us with a good policy in itself, but it makes it easier for us to develop it. To do this, based on the model we must predict the future behaviour of the environment to incorporate it into our policy, this is what is known as **planning**.
- An agent may or may not have a model (it is optional).
- The majority of agents that we will see **lack a model**, but it is important to keep in mind that this possibility exists.

Table of Contents

Taxonomy

Taxonomy

There are **two main ways to classify agent types** based on the key elements they contain:

$$\textit{Agents} \begin{cases} \text{Based on value function, policy or both} \\ \text{Model-based or model-free} \end{cases}$$

Taxonomy

Taxonomy - **Based on value function, policy or both**

The first classification refers to whether **the agent uses the policy, the value function or both** to determine the actions to choose at each moment:

- **Value function-based agents**: They explicitly contain a value function, while the policy is implicit.
- **Policy-based agents**: They explicitly contain a representation of the policy to be followed, but not of the value function (if present, it is implicit).
- **Actor-Critic**: It is a combination of the two previous types of agents.
  - It is a type of agent that explicitly stores both the value function and the policy, and uses both (combines them) to determine the action to choose in each situation.

Taxonomy

Taxonomy - **Model-based or model-free**

Another of the main distinctions in reinforcement learning is whether the agents have a model of the environment:

- **Model-free**: The agent does not try to explicitly understand the environment, nor does it try to learn a representation of the environment (a model). Learning from it is based, solely and exclusively, on a policy and/or a value function.
- **Model-based**: The agent tries to learn a model of the environment, and then make predictions based on this model. In this case, there may also be a policy and/or a value function, but what is essential is the existence of the model.

Taxonomy

Taxonomy

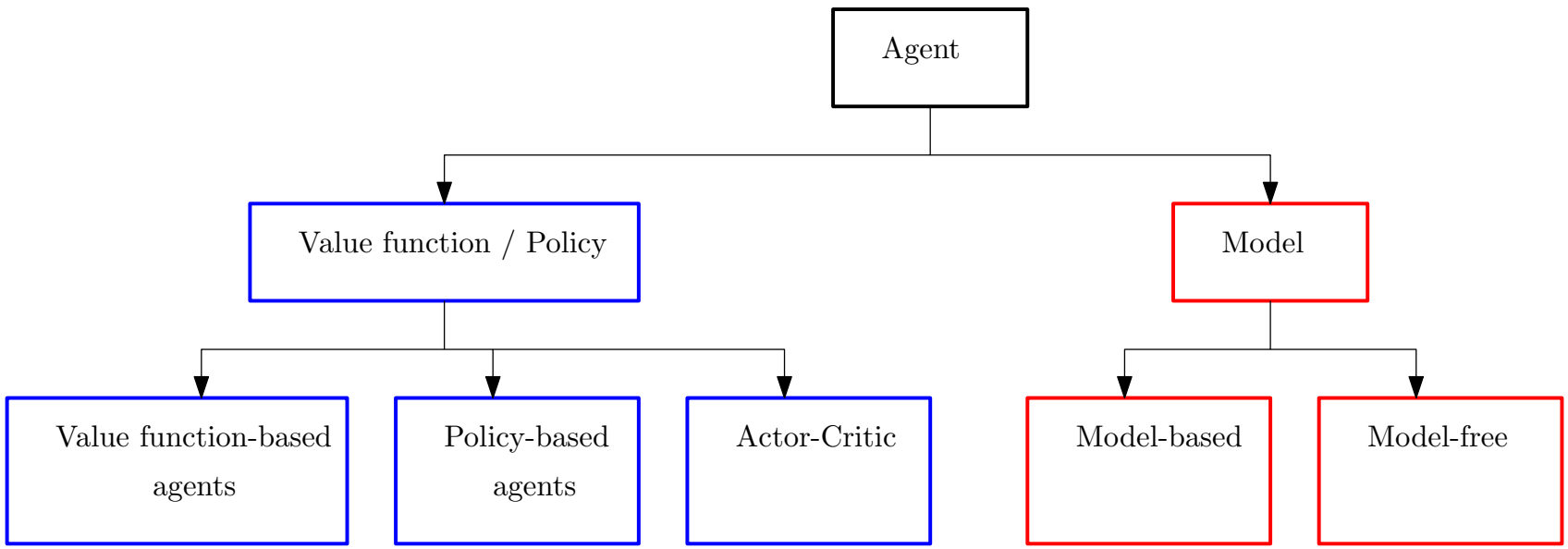
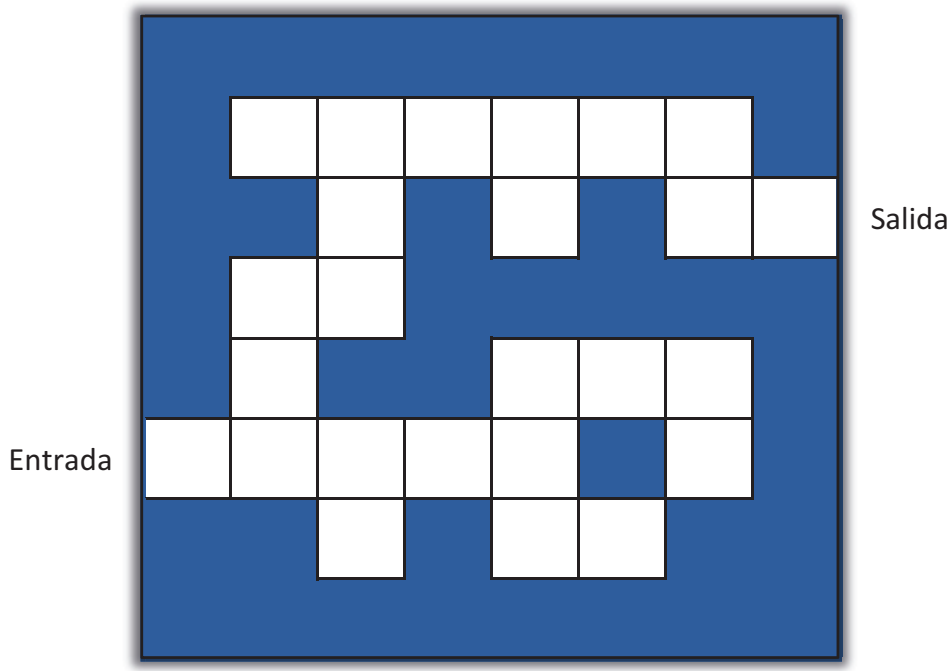


Table of Contents

Example

The maze

The problem: The problem is to **design an agent** that is capable of learning how to cross the maze as quickly as possible.



Example
The maze

- **Reward**: The agent learns to cross the maze as quickly as possible

$$\begin{cases} R_t = -1 & \text{for any movement, except if it reaches the exit} \\ R_t = 0 & \text{if it reaches the exit} \end{cases}$$

- **Actions**: The possible actions to select consist of making a movement in the direction of the four cardinal points:

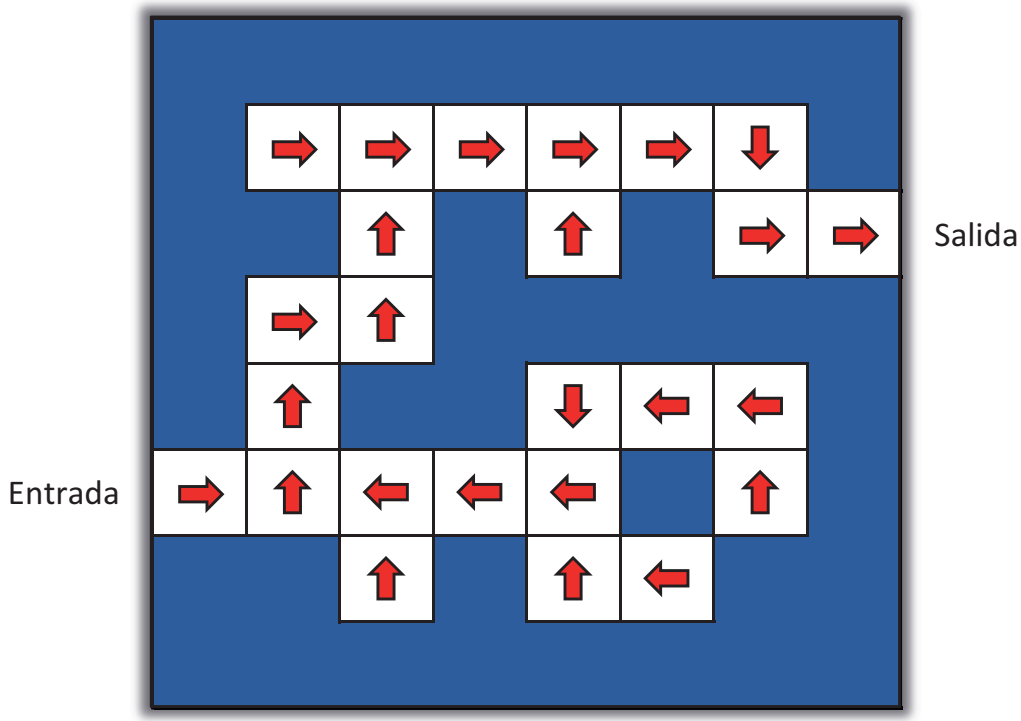
$$A_t \in \mathcal{A} = \{N = north, S = south, E = est, W = west\}$$

- **States** S_t: Include only the position of the agent within the maze (the cells of the maze).

Example

The maze - **Policy**

There is only one path that allows us to traverse the maze in the shortest possible time, the policy to be used is **deterministic** ($a = \pi(s)$).

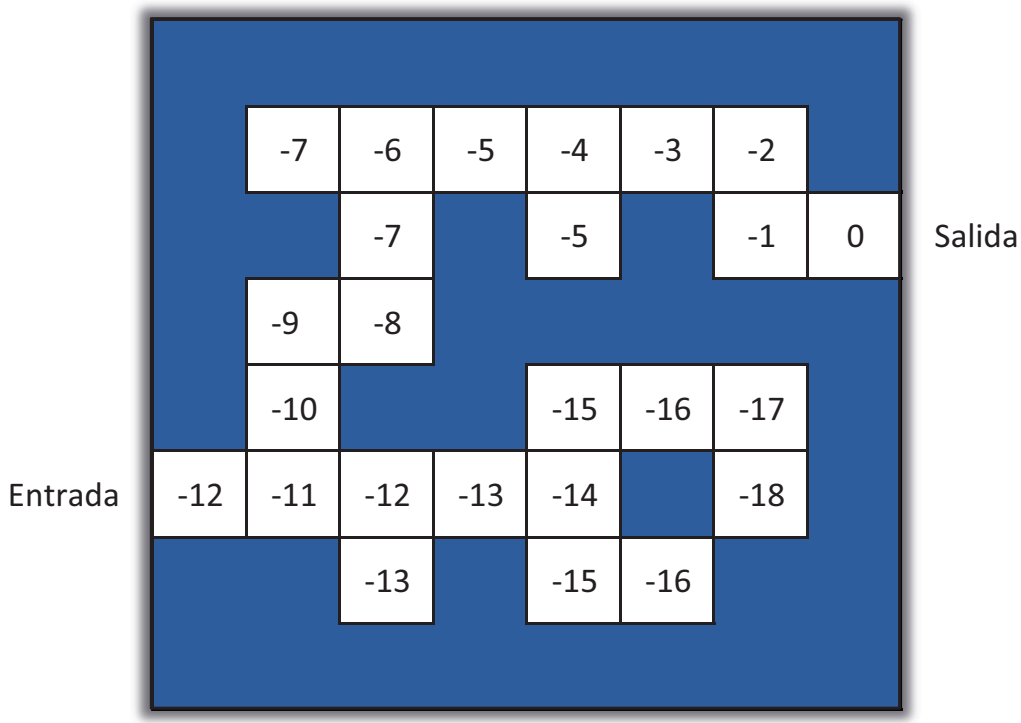


- Where the arrows represent the policy ($a = \pi(s)$) for each state 's'.
- It is a mapping between states and actions, since it tells us what action 'a' we should take in each state 's'.

Example

The maze - **State value function**

Each cell represents the expected value of the return $v_\pi(s)$ in that state 's' following the deterministic policy explained in the previous section.

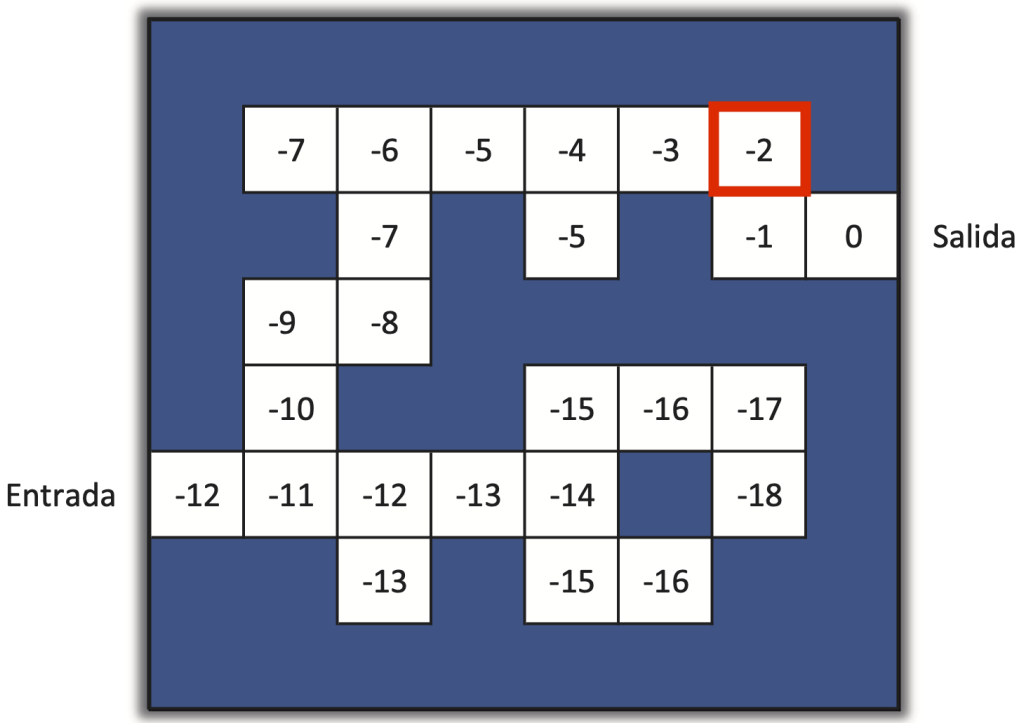


- The values have been calculated with a discount factor $\gamma = 1$.

Example

The maze - **State value function**

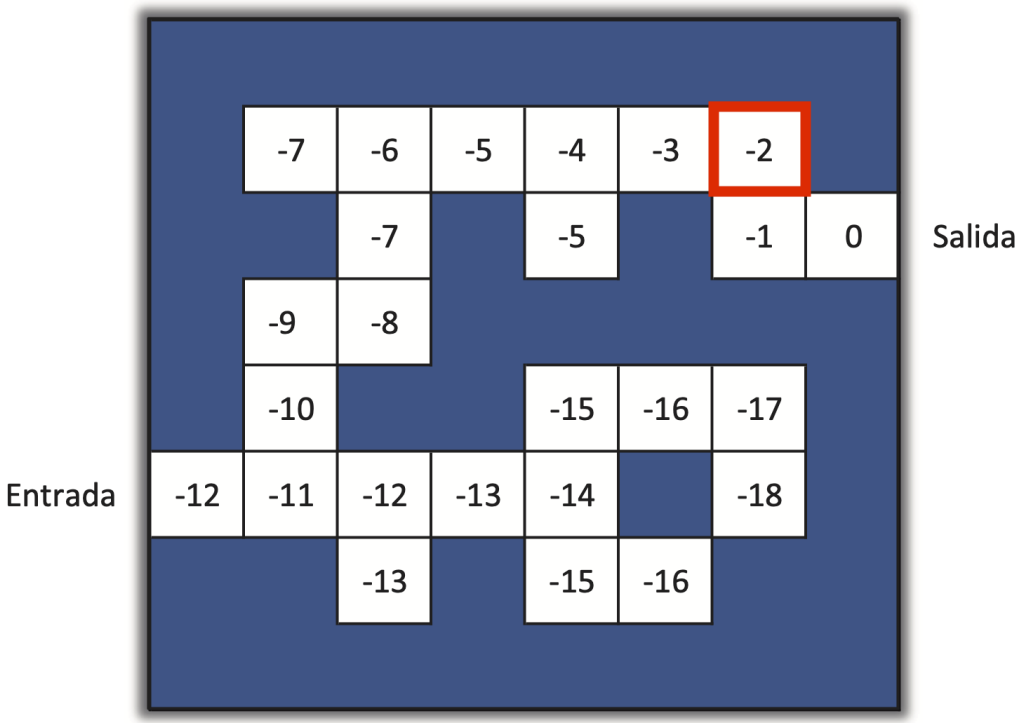
How do we compute the state value function of the selected state?



Example

The maze - **State value function**

How do we compute the state value function of the selected state?

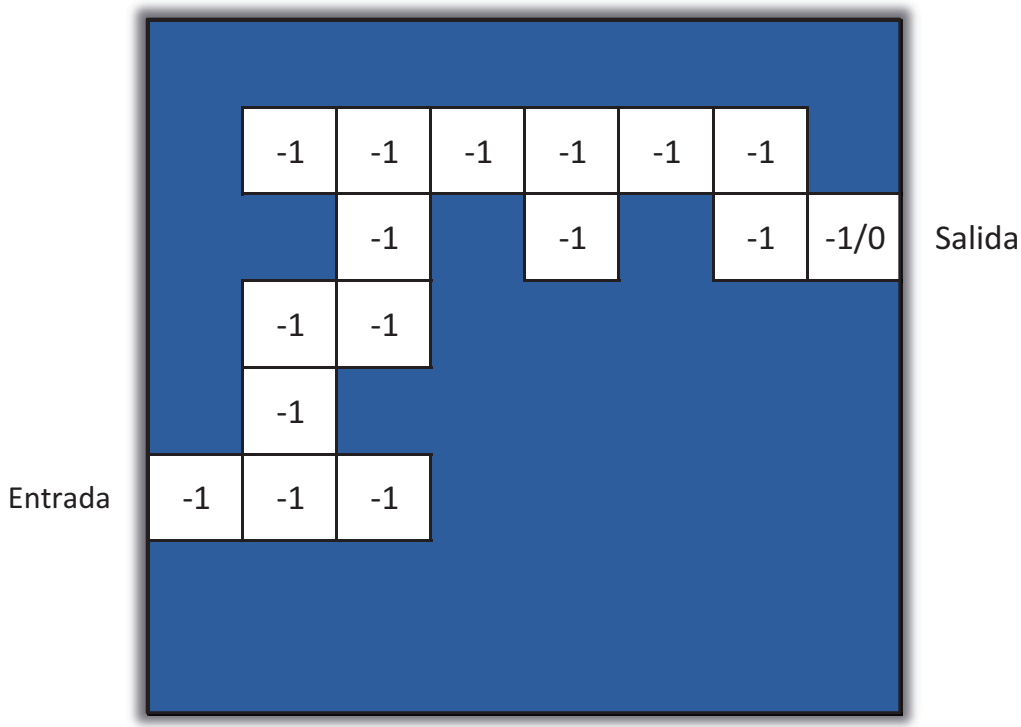


- $v_\pi(s) = R_{t+1} + R_{t+2} + R_{t+3} = (-1) + (-1) + 0 = -2$

Example

The maze - **Model**

Let's imagine that the agent has observed a trajectory through the maze (the path in the figure), has reached the exit, and intends to build a model.



- The grid partially represents the transition pattern $\mathcal{P}^a_{ss'}$.
- The numbers in each cell represent the immediate reward pattern $\mathcal{R}^a_s$ for each state 's'.

Table of Contents

Biliography

References

Some relevant references:

1. **R. S. Sutton, A. G. Barto**. (2018). *Reinforcement Learning: An Introduction (Second Edition)*. MIT Press, Cambridge, MA.
2. **M. Lapan**. (2024). *Deep Reinforcement Learning Hands-On (Third Edition)*. Packt Publishing.