
المدرسة الوطنية للذكاء الاصطناعي والرقمنة - بركان
ÉCOLE NATIONALE DE L'INTELLIGENCE ARTIFICIELLE ET DU DIGITAL - BERKANE

# MACHINE LEARNING 2

Pr. BOUTAHIR Mohamed Khalifa

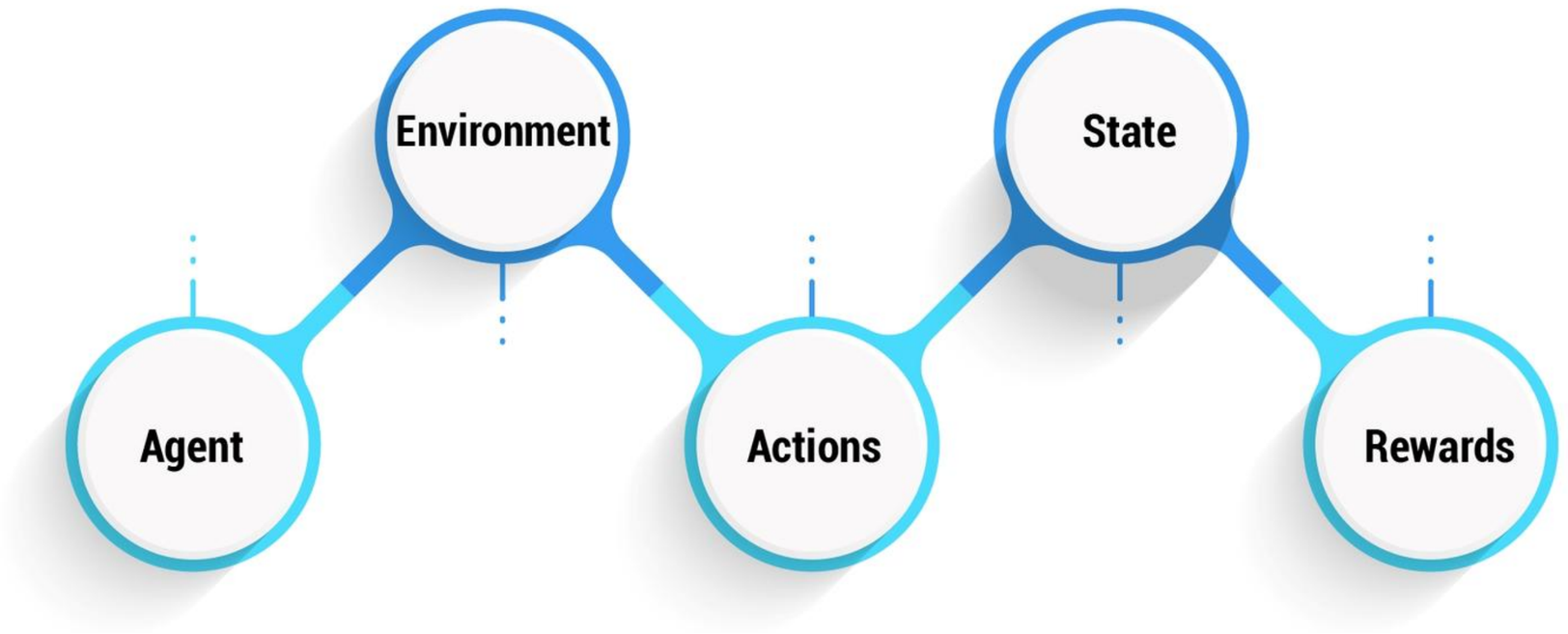
2024 – 2025

## OBJECTIFS DE LA SESSION

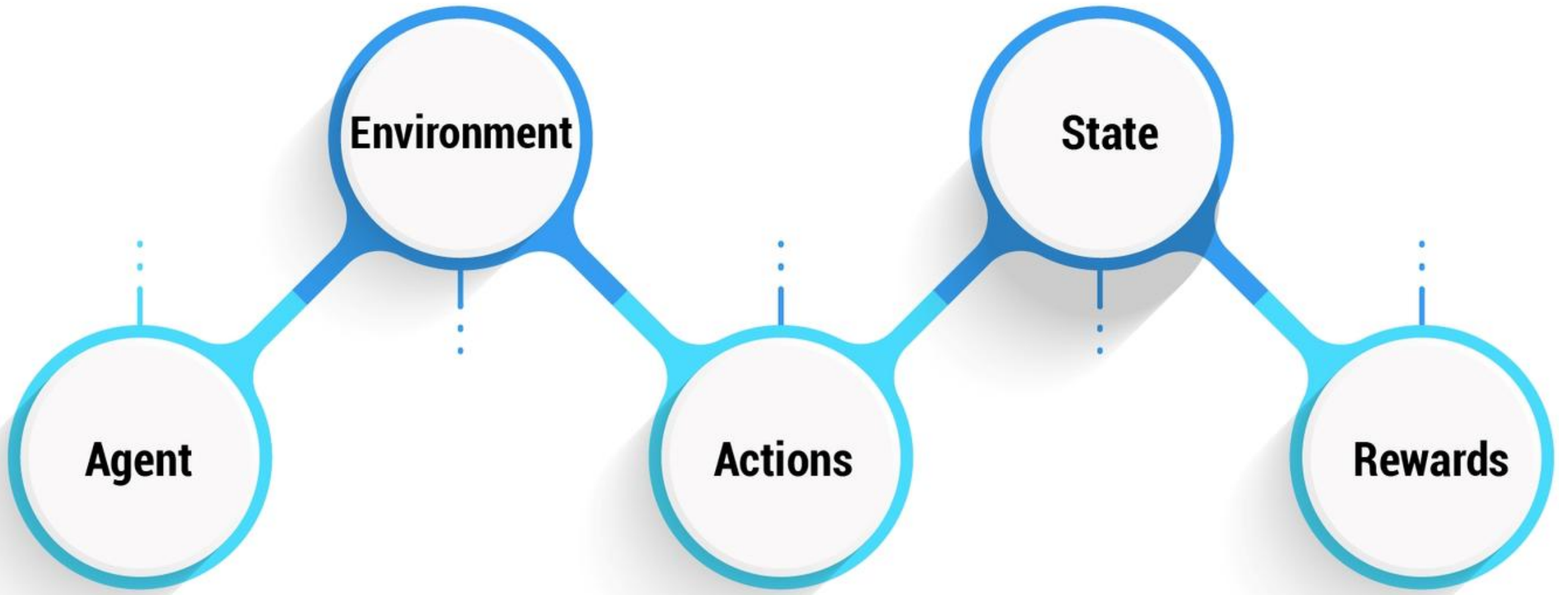
**L’apprentissage par renforcement (Reinforcement Learning - RL)** est une branche de l’intelligence artificielle où un agent apprend à interagir avec un environnement en recevant des récompenses pour ses bonnes actions et des pénalités pour ses erreurs. Après avoir découvert les bases du RL dans la session précédente, cette séance va approfondir ses composants fondamentaux et déterminer quand et comment l’utiliser efficacement.

- Comprendre les 5 éléments clés du RL : Agent, Environnement, États, Actions, Récompenses.
- Explorer le Processus de Décision de Markov (MDP) et son lien avec RL.
- Déterminer quand RL est applicable et quand d’autres approches sont plus adaptées.
- Illustrer des cas d’usage concrets et discuter de problèmes où RL peut ou ne peut pas être utilisé.

## LES CINQ ÉLÉMENTS DE L'APPRENTISSAGE PAR RENFORCEMENT



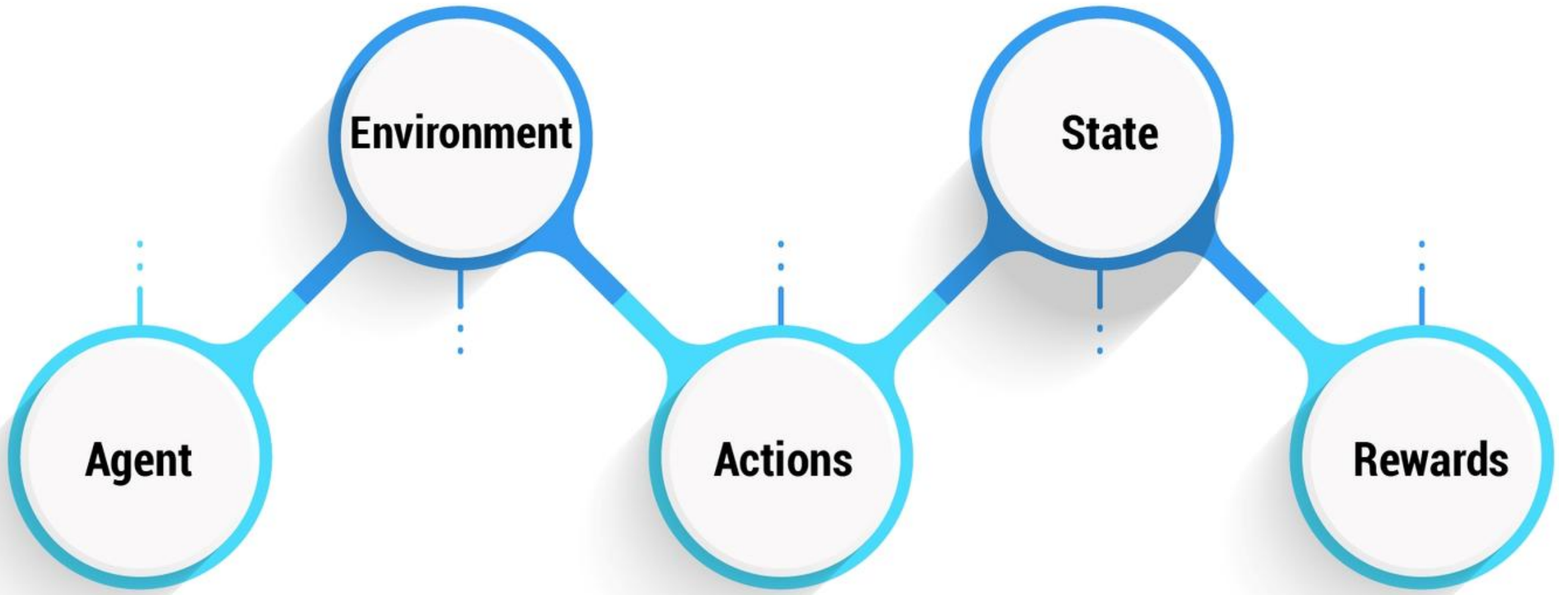
## LES CINQ ÉLÉMENTS DE L'APPRENTISSAGE PAR RENFORCEMENT




L'agent est l'intelligence artificielle qui apprend à prendre des décisions. Il explore l'environnement, choisit des actions et reçoit des récompenses.

## LES CINQ ÉLÉMENTS DE L'APPRENTISSAGE PAR RENFORCEMENT

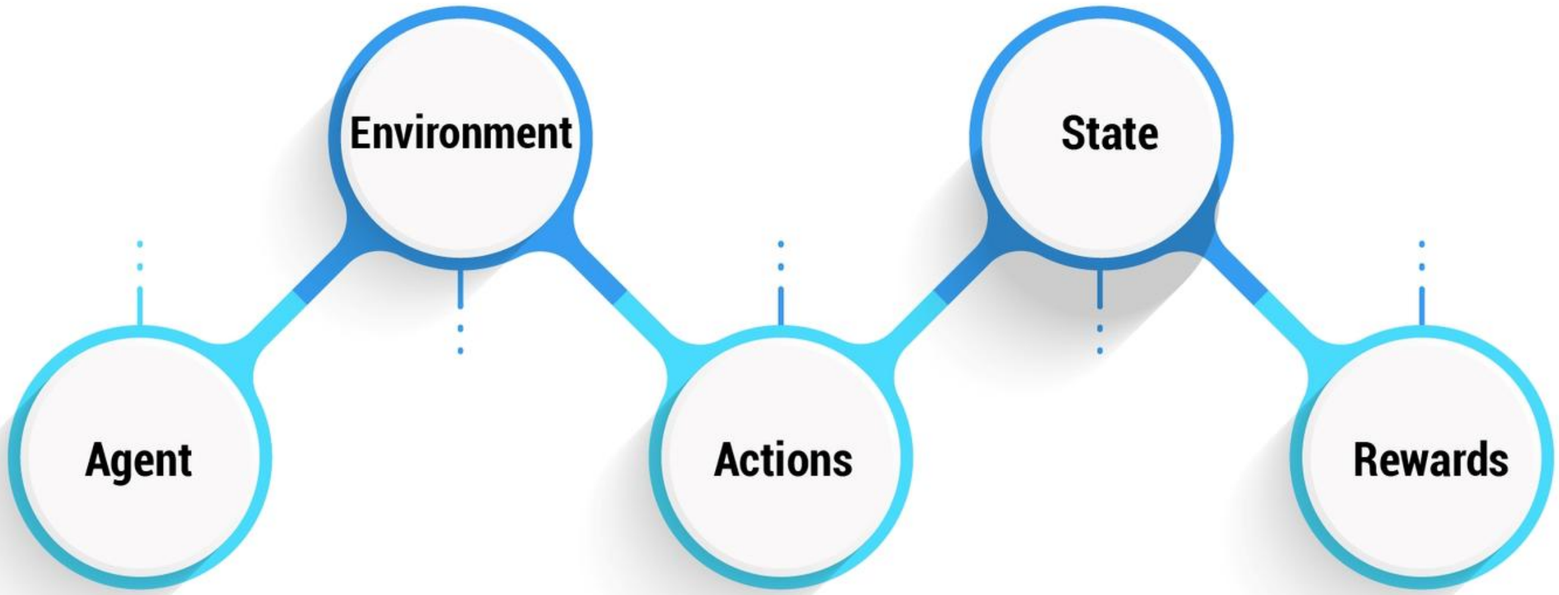
L'environnement est le monde dans lequel évolue l'agent. C'est lui qui définit les règles du jeu, les obstacles et les récompenses.




L'agent est l'intelligence artificielle qui apprend à prendre des décisions. Il explore l'environnement, choisit des actions et reçoit des récompenses pour s'améliorer.

## LES CINQ ÉLÉMENTS DE L'APPRENTISSAGE PAR RENFORCEMENT

L'environnement est le monde dans lequel évolue l'agent. C'est lui qui définit les règles du jeu, les obstacles et les récompenses.



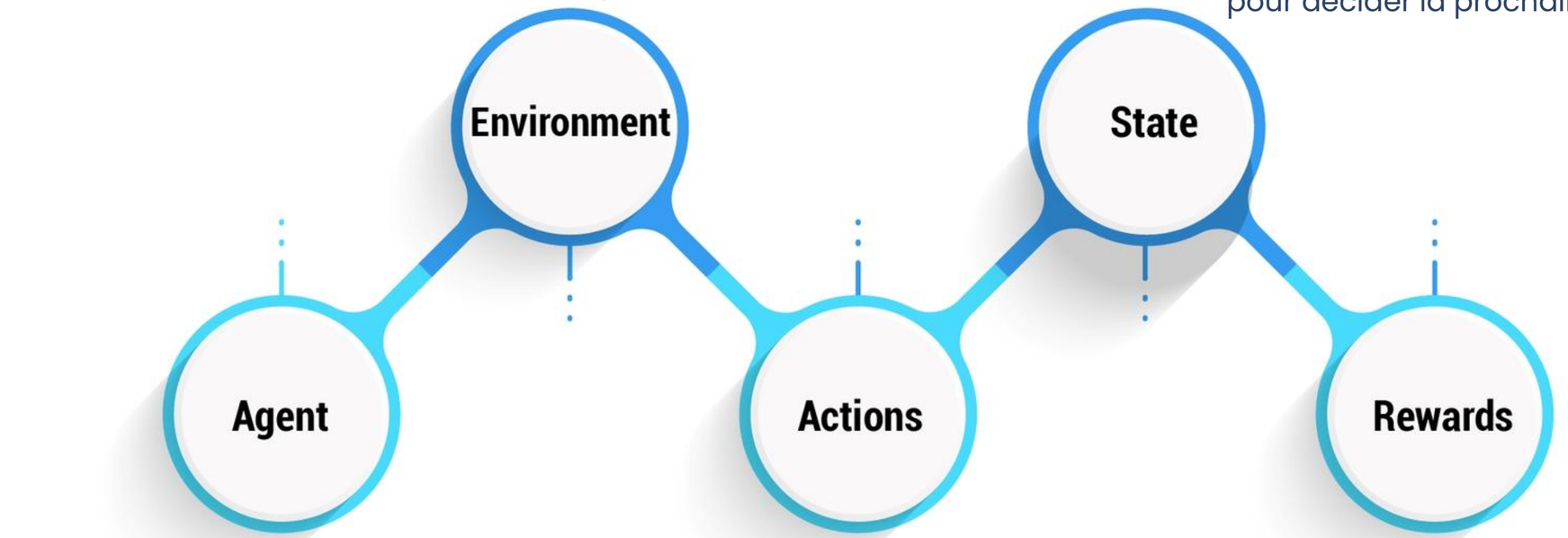

L'agent est l'intelligence artificielle qui apprend à prendre des décisions. Il explore l'environnement, choisit des actions et reçoit des récompenses pour s'améliorer.

Les actions sont les choix possibles pour l'agent. Il peut essayer différentes actions et observer leur effet sur l'environnement.

## LES CINQ ÉLÉMENTS DE L'APPRENTISSAGE PAR RENFORCEMENT

L'environnement est le monde dans lequel évolue l'agent. C'est lui qui définit les règles du jeu, les obstacles et les récompenses.

L'état représente la situation actuelle de l'agent dans l'environnement. Il contient toutes les informations utiles pour décider la prochaine action.



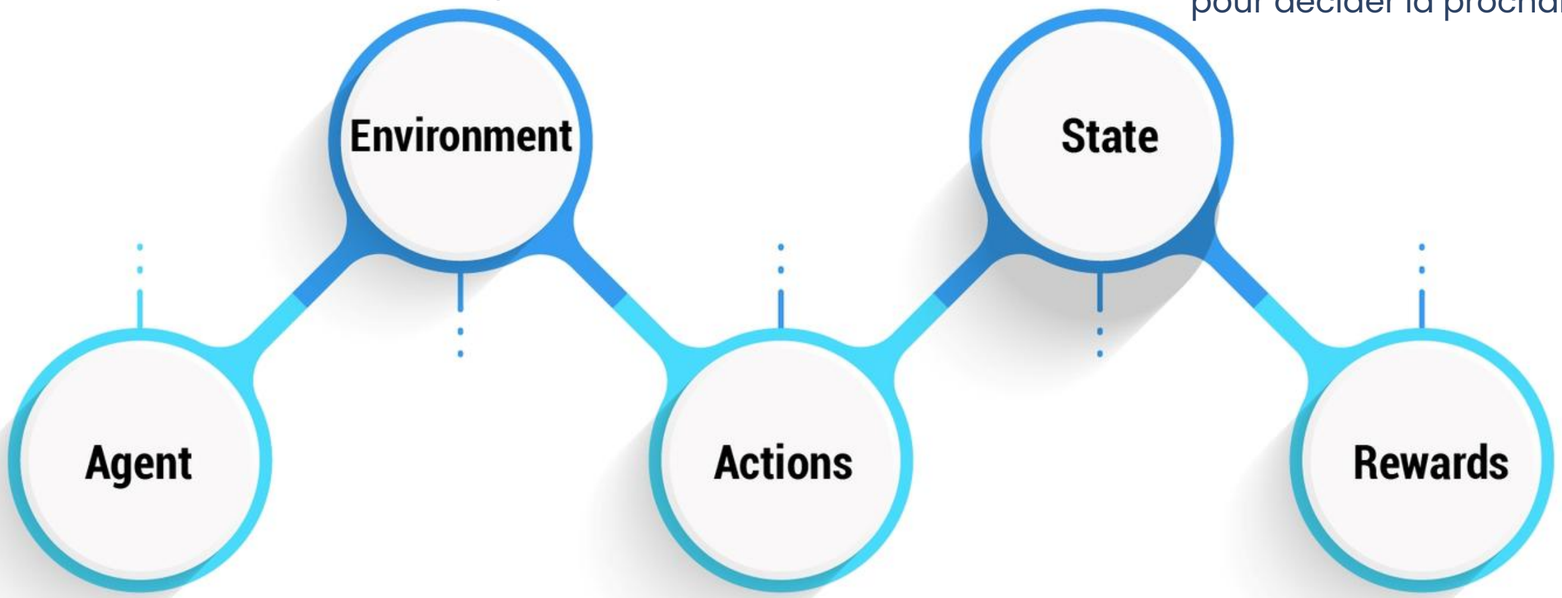

L'agent est l'intelligence artificielle qui apprend à prendre des décisions. Il explore l'environnement, choisit des actions et reçoit des récompenses pour s'améliorer.

Les actions sont les choix possibles pour l'agent. Il peut essayer différentes actions et observer leur effet sur l'environnement.

## LES CINQ ÉLÉMENTS DE L'APPRENTISSAGE PAR RENFORCEMENT

L'environnement est le monde dans lequel évolue l'agent. C'est lui qui définit les règles du jeu, les obstacles et les récompenses.

L'état représente la situation actuelle de l'agent dans l'environnement. Il contient toutes les informations utiles pour décider la prochaine action.




L'agent est l'intelligence artificielle qui apprend à prendre des décisions. Il explore l'environnement, choisit des actions et reçoit des récompenses pour s'améliorer.

Les actions sont les choix possibles pour l'agent. Il peut essayer différentes actions et observer leur effet sur l'environnement.

La récompense est un score donné à l'agent après chaque action. Son but est d'encourager les bonnes décisions et de décourager les mauvaises.

## PROCESSUS DE DÉCISION DE MARKOV (MDP)

**Un Processus de Décision de Markov (MDP)** est un modèle mathématique utilisé pour décrire comment un **agent** prend **des décisions** dans **un environnement**. Il est défini par **un ensemble d'états**, **d'actions**, de **probabilités de transition**, de **récompenses** et d'un **facteur de réduction**.

**L'apprentissage par renforcement (RL)** repose sur les **MDP** car ils fournissent un cadre structuré permettant à un **agent** d'apprendre à maximiser une **récompense** cumulative en explorant **l'environnement** et en ajustant ses **actions** en fonction des résultats obtenus.

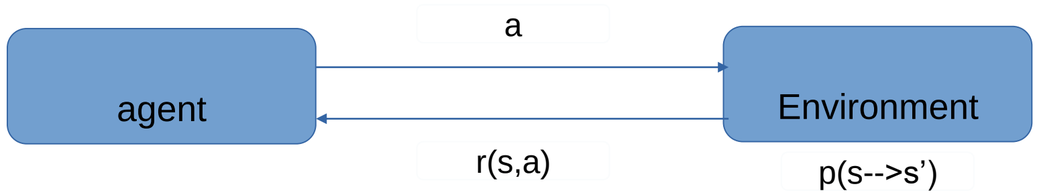
L'idée essentielle des **MDP** est que **l'état futur** dépend uniquement de **l'état actuel** et de **l'action** prise (et pas du passé).

- Presque tous les problèmes **d'apprentissage par renforcement (RL)** peuvent être modélisés comme un **Processus de Décision de Markov (MDP)**.
- Il existe certains problèmes **RL** qui ne rentrent pas parfaitement dans le cadre des **MDP**, notamment dans ces cas : **Processus de Décision de Markov Partiellement Observable (POMDP)** et **Problèmes avec Mémoire Long Terme**.

## PROCESSUS DE DÉCISION DE MARKOV (MDP)

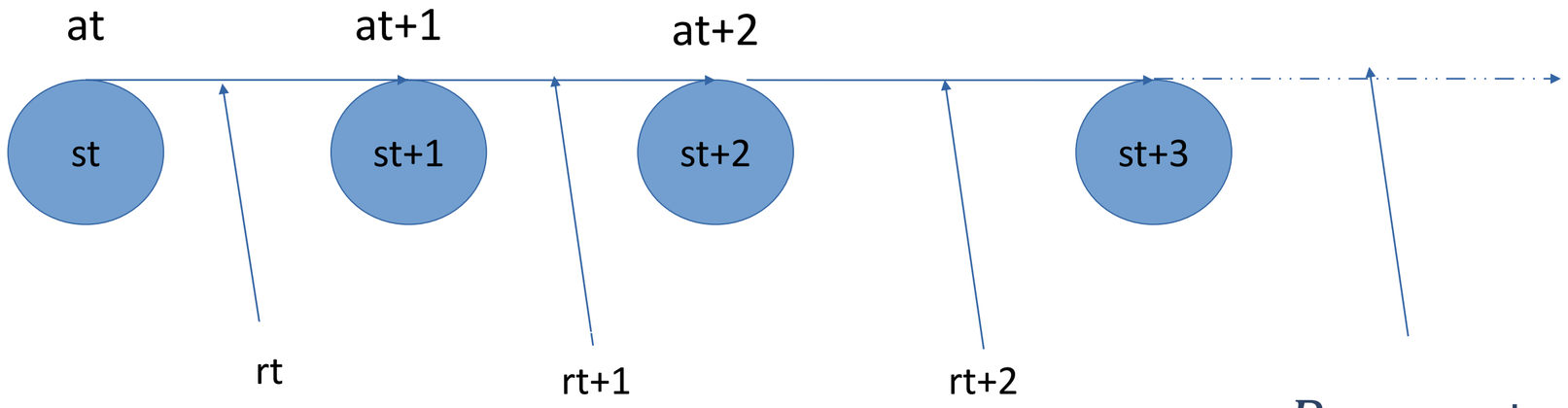
Un processus de Décision de Markov est défini par **M=⟨S,A,P,R,γ⟩ :**

- **S** - un ensemble d'États ;
- **A** - un ensemble d'actions ;
- **P** - fonction de probabilité de transition ;
- **R** - fonction de récompense ;
- **γ** - Facteur d'actualisation pour les récompenses futures. Dans un environnement inconnu, nous n'avons pas une connaissance parfaite de P et R.




**En résumé : Un MDP** aide à structurer un problème en définissant où **l'agent** est **(S)**, ce qu'il peut faire **(A)**, comment le monde réagit **(P)**, ce qu'il gagne ou perd **(R)**, et comment il anticipe l'avenir **(γ)**.

## PROCESSUS DE DÉCISION DE MARKOV (MDP)




$$R_t = r_t + \gamma r_{t+1} + \gamma r_{t+2} + \ldots..$$

- Fonction de recompense ( Reward function ) :

On cherche à maximiser la récompense cumulative espérée (somme des récompenses futures, souvent pondérée par un facteur $\gamma$ (gamma) pour privilégier les récompenses proches).

$$R_t = r_t + \gamma R_{t+1}$$

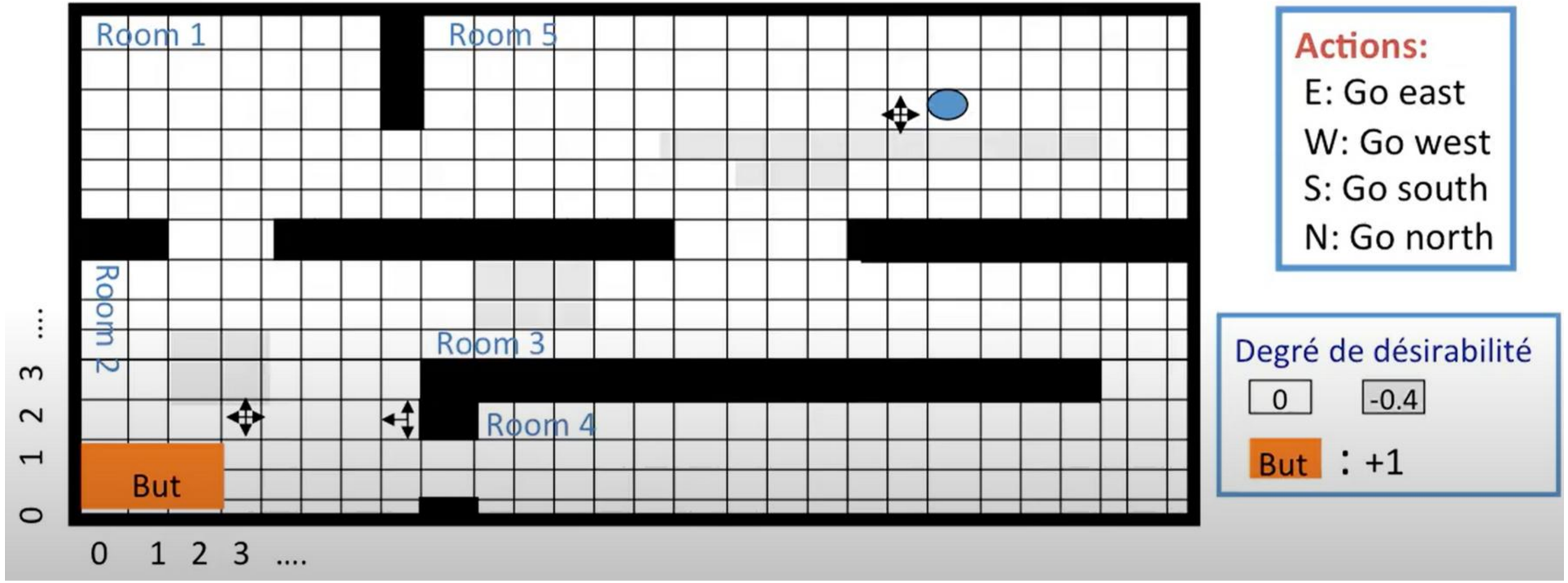
- rt est la récompense immédiate obtenue après être dans l'état s.
- $\gamma$ (gamma) est le facteur d'actualisation ($0 \leq \gamma \leq 1$ )

- Le facteur $\gamma$ permet de :

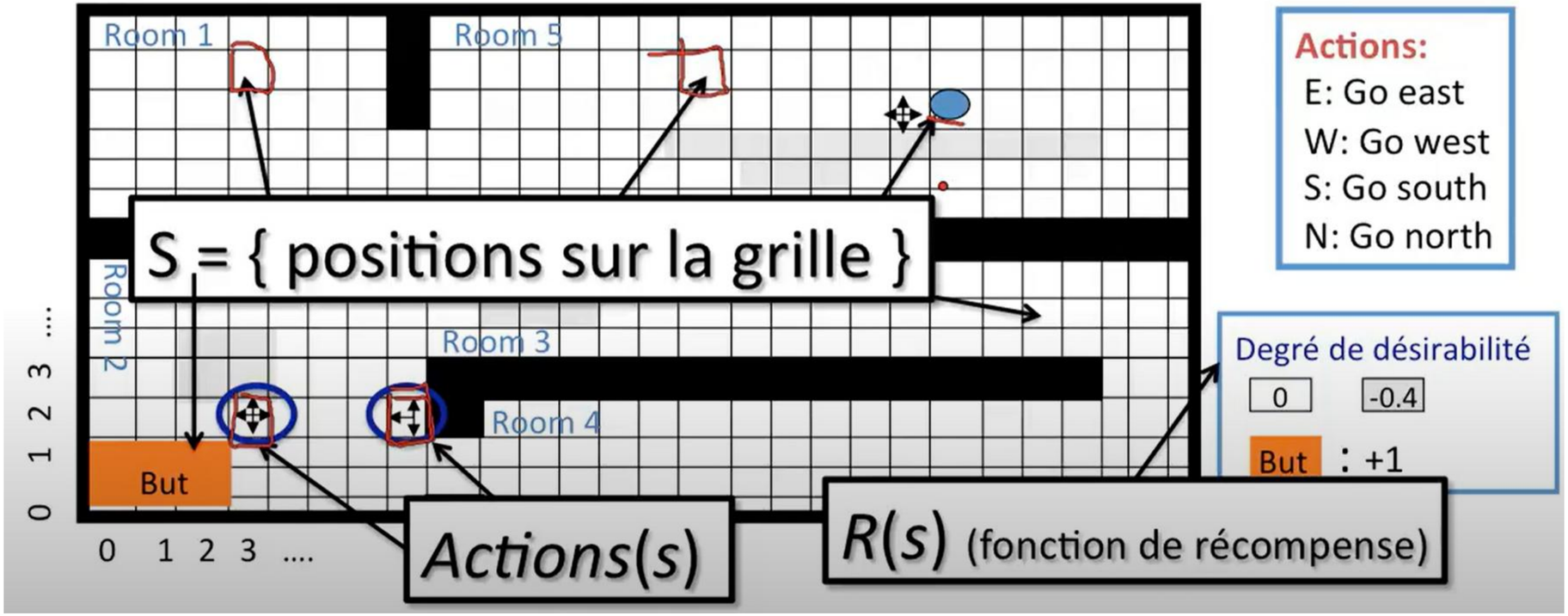
✓ Privilégier les récompenses à court terme si $\gamma$ est proche de 0.

✓ Penser au long terme si $\gamma$ est proche de 1.

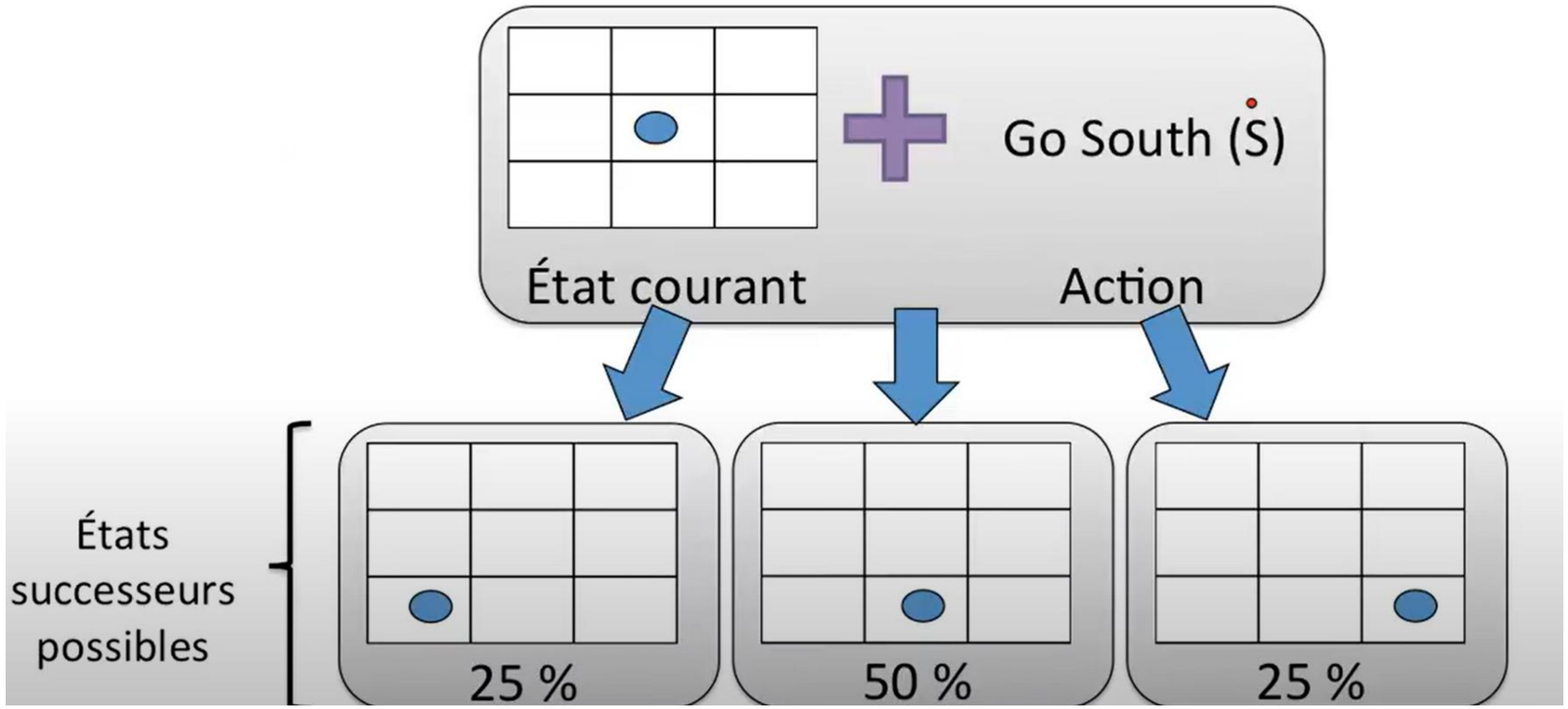
## PROCESSUS DE DÉCISION DE MARKOV (MDP)



## PROCESSUS DE DÉCISION DE MARKOV (MDP)



## PROCESSUS DE DÉCISION DE MARKOV (MDP)



## EXEMPLE PRATIQUE

Un robot aspirateur intelligent qui apprend à nettoyer une pièce efficacement en évitant les obstacles et en minimisant sa consommation d’énergie.

| Composant RL | Explication |
|---|---|
| Agent | |
| Environnement | |
| État | |
| Actions | |
| Récompense | |

## EXEMPLE PRATIQUE

Un robot aspirateur intelligent qui apprend à nettoyer une pièce efficacement en évitant les obstacles et en minimisant sa consommation d'énergie.

| Composant RL | Explication |
|---|---|
| Agent | Le robot aspirateur qui décide où aller et comment nettoyer. |
| Environnement | L'appartement avec des meubles, des tapis, et des obstacles. |
| État | Position actuelle du robot, niveau de batterie, saleté détectée. |
| Actions | Avancer, tourner, aspirer, recharger la batterie. |
| Récompense | Points positifs pour nettoyage efficace, pénalité pour collision ou batterie vide. |

## EXEMPLE PRATIQUE

Un robot aspirateur intelligent qui apprend à nettoyer une pièce efficacement en évitant les obstacles et en minimisant sa consommation d'énergie.

Le robot évolue selon :

- **États :** $s_t$ représente la position du robot, la saleté détectée, et le niveau de batterie.
- **Actions :** $a_t$ inclut tourner, avancer, aspirer, recharger.
- **Transition :** $P(s_{t+1}|s_t, a_t)$ définit l'effet d'une action sur l'état futur.
- **Récompense :**

$$r_t = \begin{cases} +10, & \text{si la zone est nettoyée} \\ -5, & \text{si le robot heurte un mur} \\ -10, & \text{si la batterie tombe à } 0\% \\ +5, & \text{si le robot se recharge à temps} \end{cases}$$

Le but est de **maximiser la somme des récompenses** sur le long terme.

## QUAND RL EST-IL APPLICABLE ?

Avant de choisir RL, posez-vous **ces questions** :

1. **Problème séquentiel :** Est-ce que les décisions prises affectent les états futurs ?
2. **Exploration nécessaire :** Le problème contient-il des choix inconnus que l'agent doit découvrir par essai-erreur ?
3. **Récompense mesurable :** Pouvez-vous quantifier le succès avec une récompense (positive ou négative) ?
4. **Incertain et dynamique :** L'environnement change-t-il de façon non déterministe (ex : météo, marchés) ?
5. **Pas de données labellisées :** Manquez-vous de données pour entraîner un modèle supervisé ?
6. **Long horizon temporel :** Le problème nécessite-t-il de maximiser des gains sur le long terme ?

## ◦ QUAND RL EST-IL APPLICABLE ?

Avant de choisir RL, posez-vous **ces questions** :

1. **Problème séquentiel :** Est-ce que les décisions prises affectent les états futurs ?
2. **Exploration nécessaire :** Le problème contient-il des choix inconnus que l'agent doit découvrir par essai-erreur ?
3. **Récompense mesurable :** Pouvez-vous quantifier le succès avec une récompense (positive ou négative) ?
4. **Incertain et dynamique :** L'environnement change-t-il de façon non déterministe (ex : météo, marchés) ?
5. **Pas de données labellisées :** Manquez-vous de données pour entraîner un modèle supervisé ?
6. **Long horizon temporel :** Le problème nécessite-t-il de maximiser des gains sur le long terme ?

✓ **Si vous répondez "oui" à la majorité de ces questions, RL est probablement adapté !**

## EXEMPLES DE PROBLÈMES ADAPTÉS AU RL

| Problème | Type d'IA Recommandé |
|---|---|
| Jouer à un jeu comme Pac-Man | |
| Classer des images de chats et de chiens | |
| Optimiser le trafic des feux de circulation | |
| Prédire la météo demain | |
| Apprendre à un robot à marcher | |
| Segmenter des tumeurs sur des IRM | |
| Découverte de segments clients | |

## EXEMPLES DE PROBLÈMES ADAPTÉS AU RL

| Problème | Type d'IA Recommandé |
|---|---|
| **Jouer à un jeu comme Pac-Man** | Reinforcement Learning (RL) |
| **Classer des images de chats et de chiens** | Apprentissage supervisé (CNN) |
| **Optimiser le trafic des feux de circulation** | Reinforcement Learning (RL) |
| **Prédire la météo demain** | Apprentissage supervisé (RNN) |
| **Apprendre à un robot à marcher** | Reinforcement Learning (RL) |
| **Segmenter des tumeurs sur des IRM** | Apprentissage supervisé (UNet) |
| **Découverte de segments clients** | Apprentissage non supervisé ( K-means ) |

# Devoir : Implémentation d'un Agent d'Apprentissage par Renforcement

**Objectif :** Dans cet exercice, vous allez programmer un agent intelligent pour trouver un trésor 🏆 dans une grille tout en évitant les pièges 💀.

Vous devez écrire le code ( en python ) de l'agent et lui permettre d'apprendre à trouver le chemin optimal jusqu'au trésor.

## Règles du Jeu

- L'agent commence en haut à gauche de la grille (case (0,0)).
- Il peut se déplacer : HAUT, BAS, GAUCHE, DROITE.
- Chaque déplacement a un coût de -1 (pour encourager le chemin le plus court).
- S'il atteint un piège (💀), il perd immédiatement (-10 points).
- S'il atteint le trésor (🏆), il gagne la partie (+10 points).
- L'agent doit apprendre par lui-même en jouant plusieurs parties.

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 1 | 🟩 | 💀 | 🟩 | 💀 | 🟩 |
| 2 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 3 | 🟩 | 🟩 | 🏆 | 🟩 | 🟩 |
| 4 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |