

Apprentissage par renforcement

MACHINE LEARNING

Le Casino de Monaco - un agriculteur Algérien - ChatGPT3

Vadim Lefevre

Valentine Julliard

Emilie Naux

21 novembre, 2025

A QUOI ÇA SERT ?

Apprendre à un agent comment se comporter dans un environnement

Pas de data set initial : l'agent construit petit à petit son data set

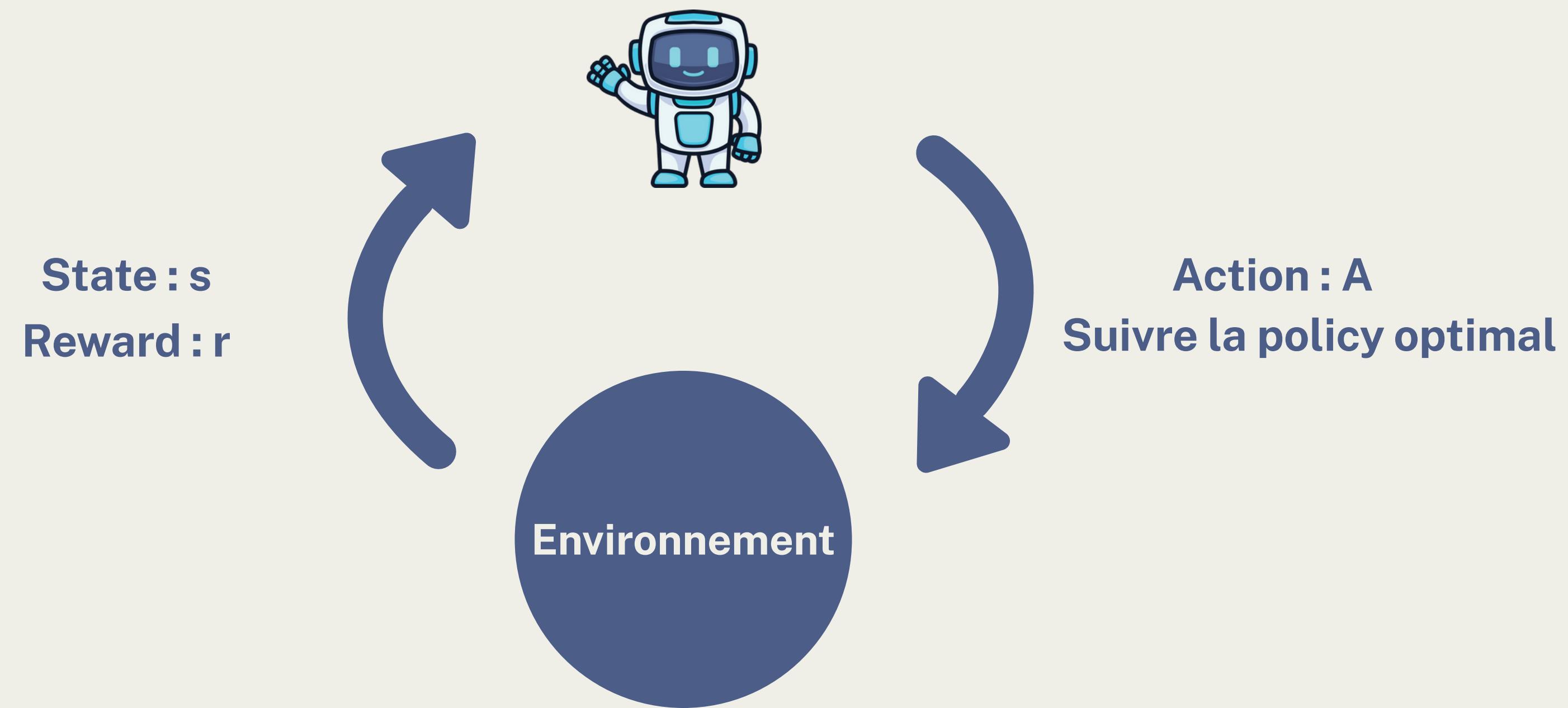
Environnement réel : robot (= agent) qui apprend à marcher dans le monde réel

Environnement virtuel : simulateur (ex : jeu de go), plus rapide et possibilité de transférer dans le monde réel

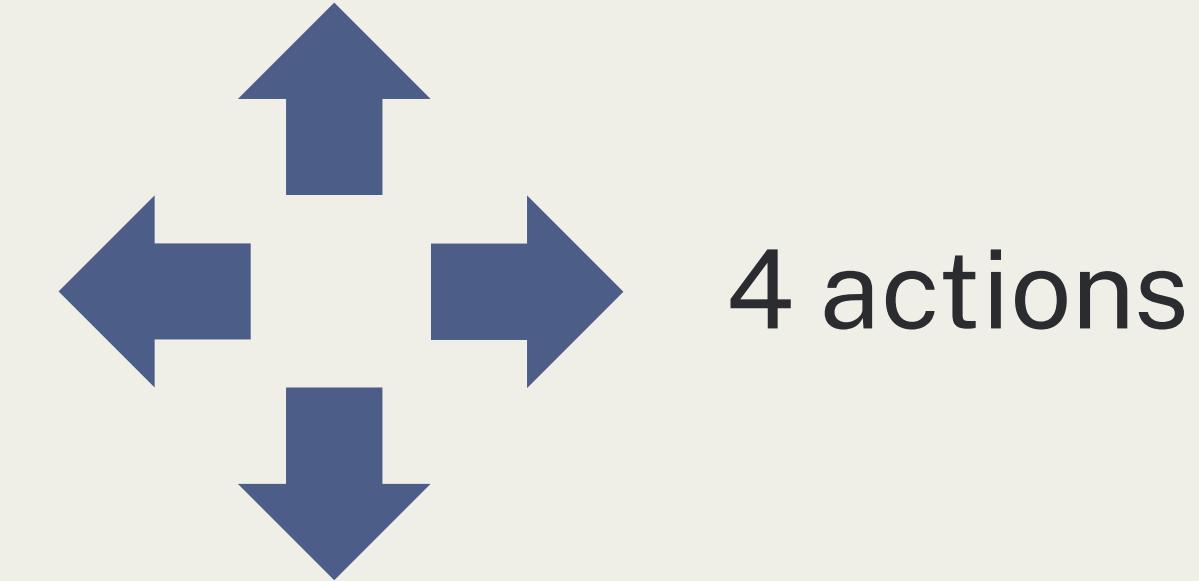
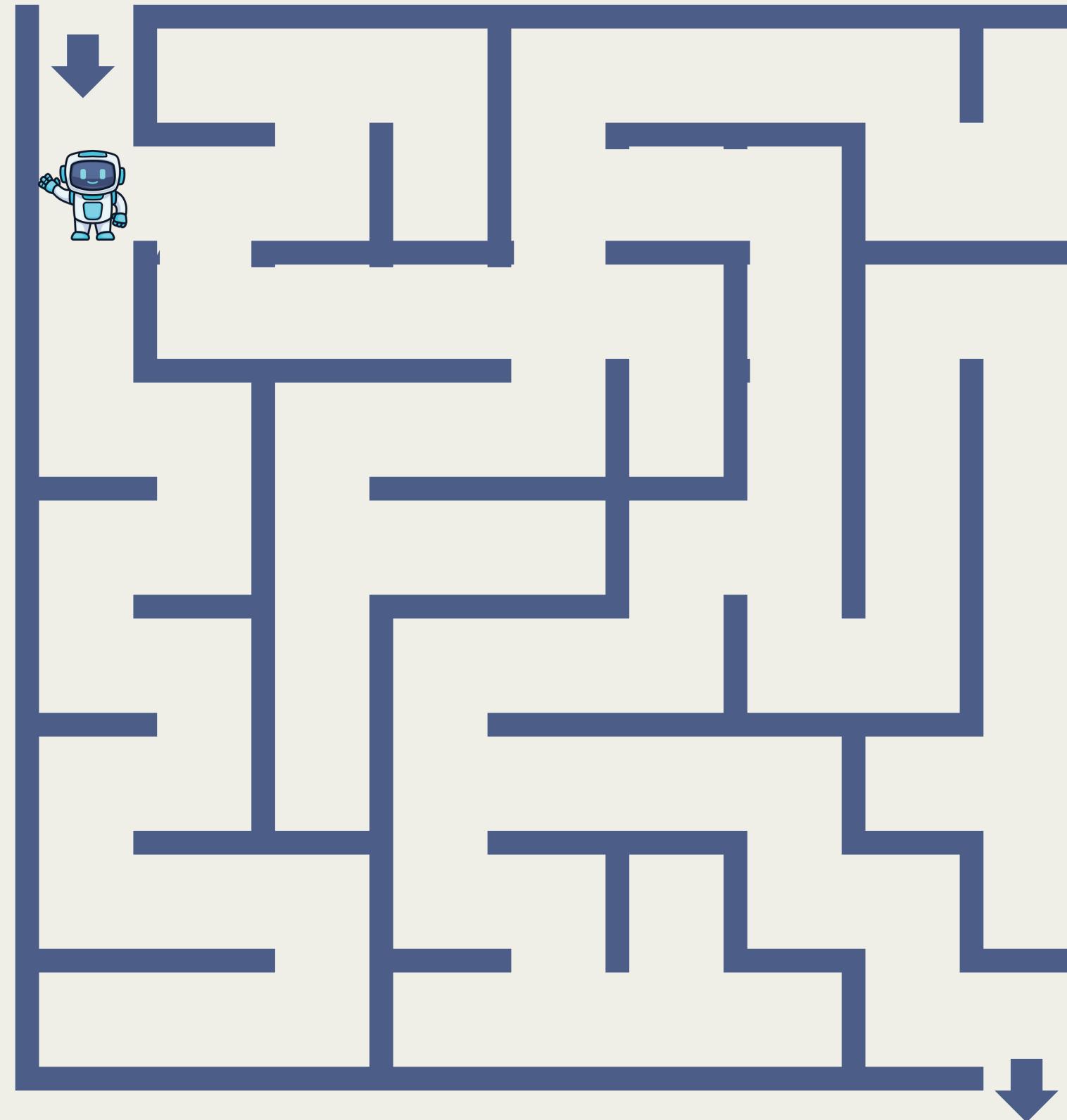


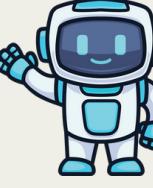
FONCTIONNEMENT GÉNÉRAL

OBJECTIF : Maximiser les récompenses, minimiser le coût (fonction d'erreur).

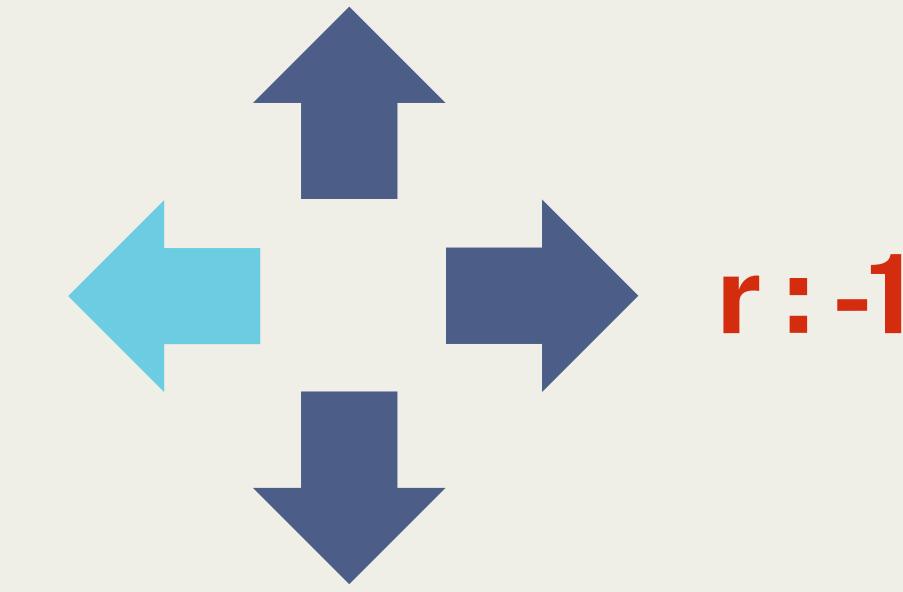
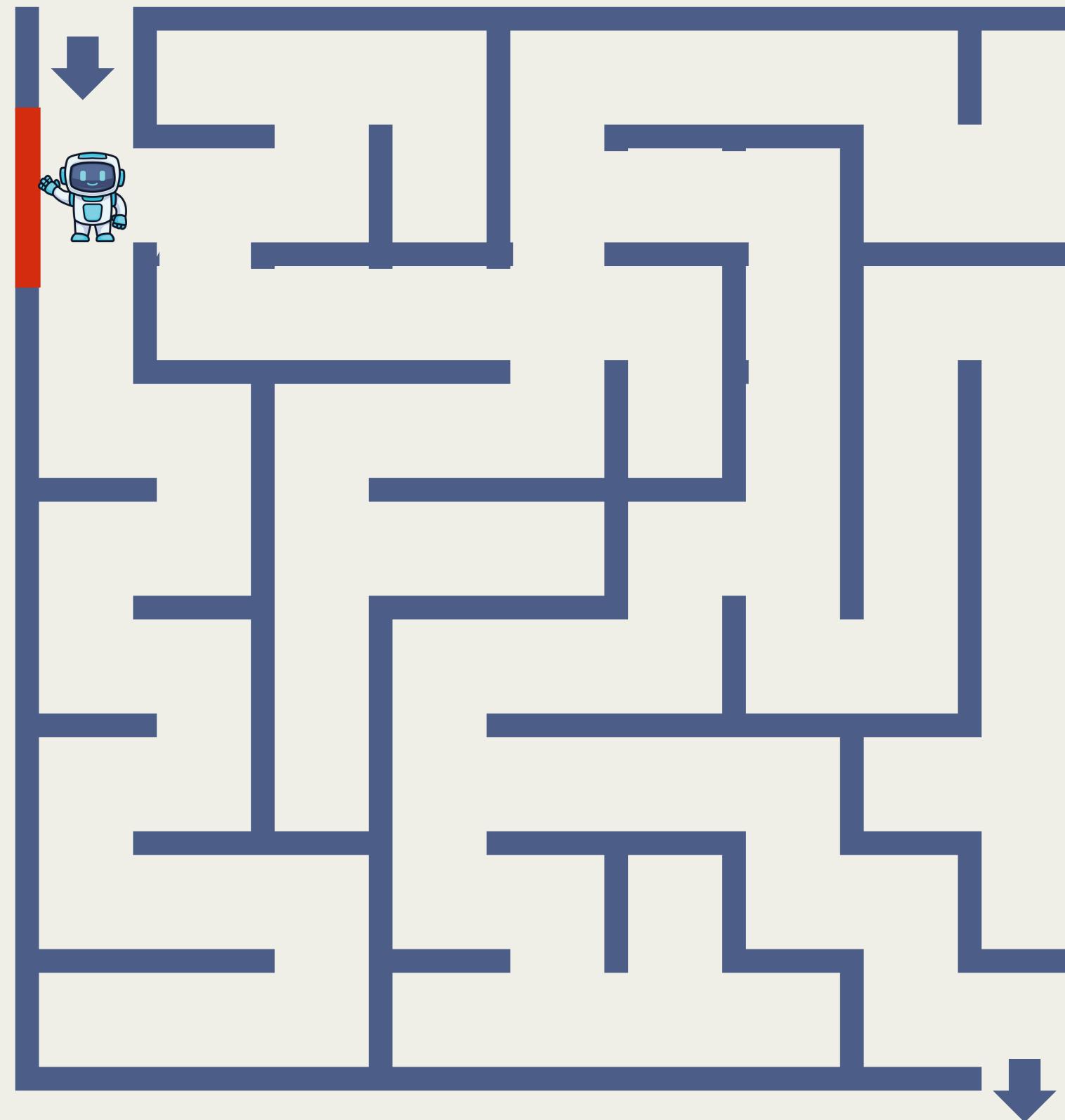


FONCTIONNEMENT GÉNÉRAL

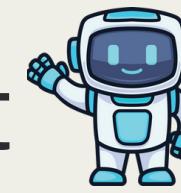


Etat de l'agent 
à l'instant t (x, y) : $(0, 1)$

FONCTIONNEMENT GÉNÉRAL



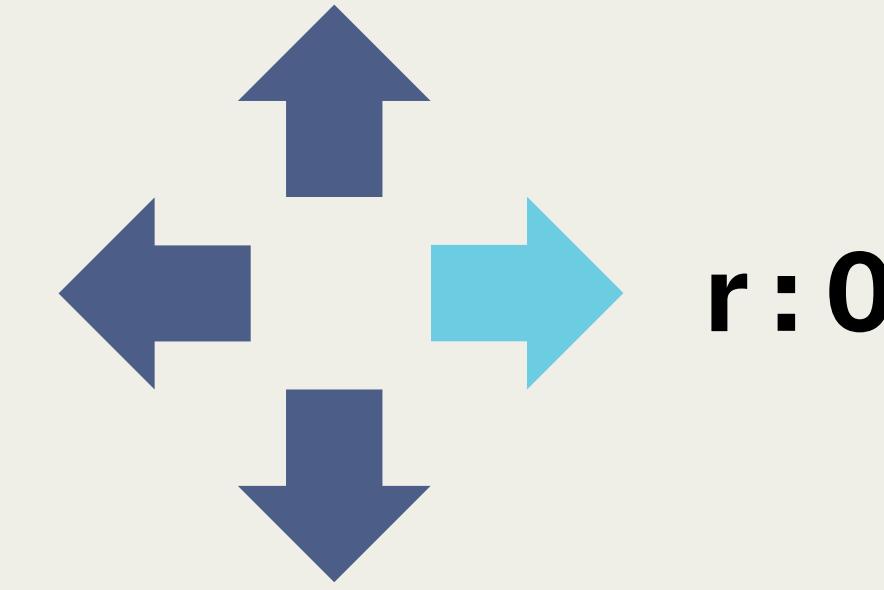
Etat de l'agent



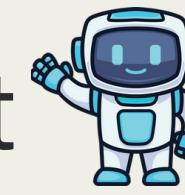
à l'instant t (x, y) : (0,1)



FONCTIONNEMENT GÉNÉRAL



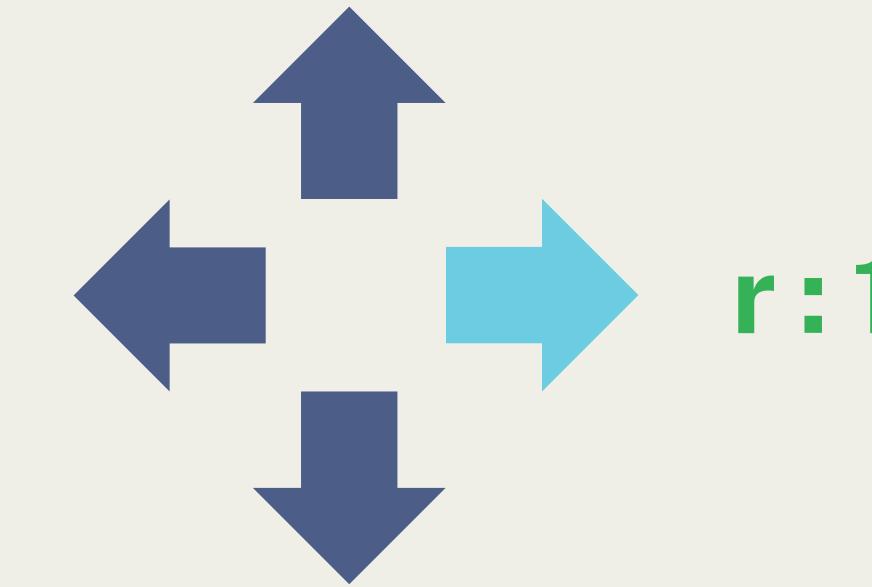
Etat de l'agent



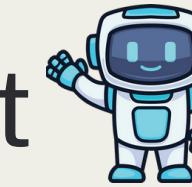
à l'instant t $(x, y) : (1, 1)$

(1,1)

FONCTIONNEMENT GÉNÉRAL



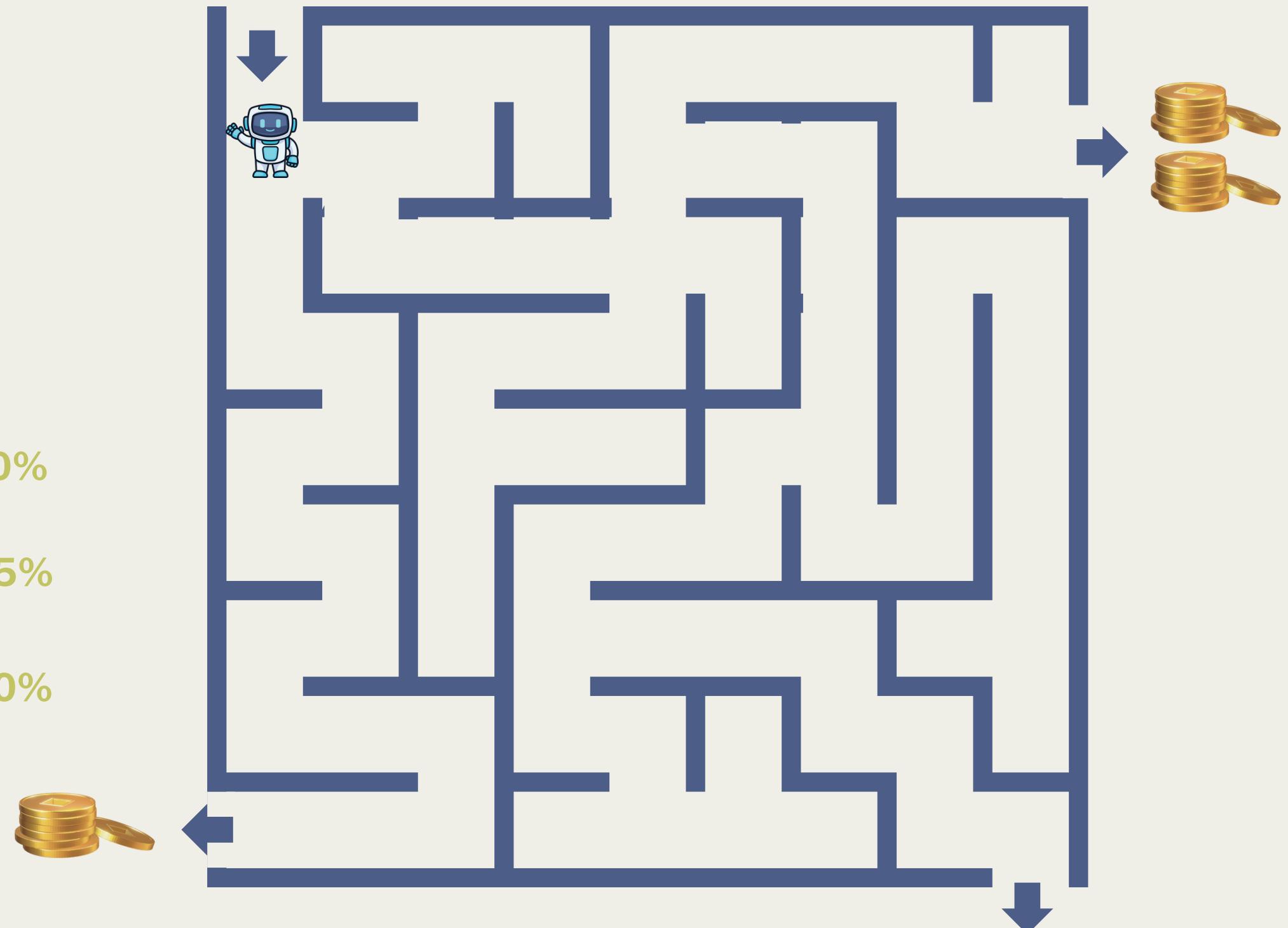
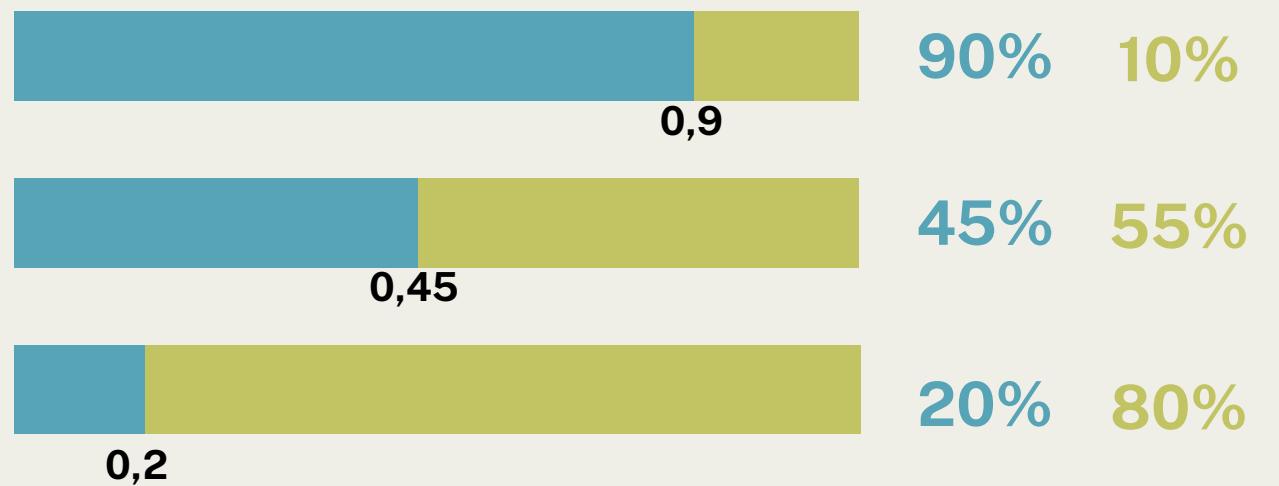
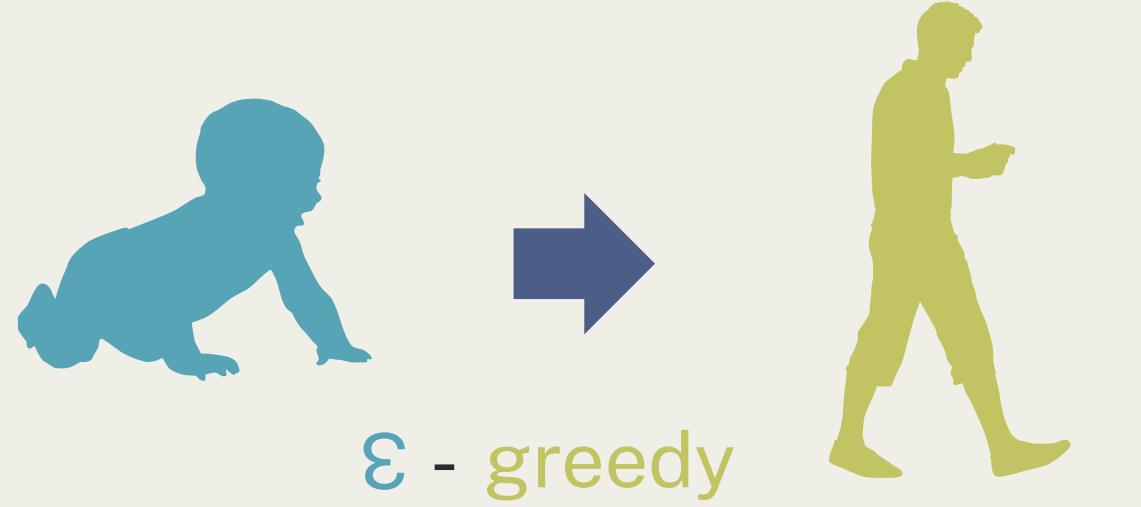
Etat de l'agent



à l'instant t (x, y) : (9,10)

FONCTIONNEMENT GÉNÉRAL

EXPLORATION VS EXPLOITATION

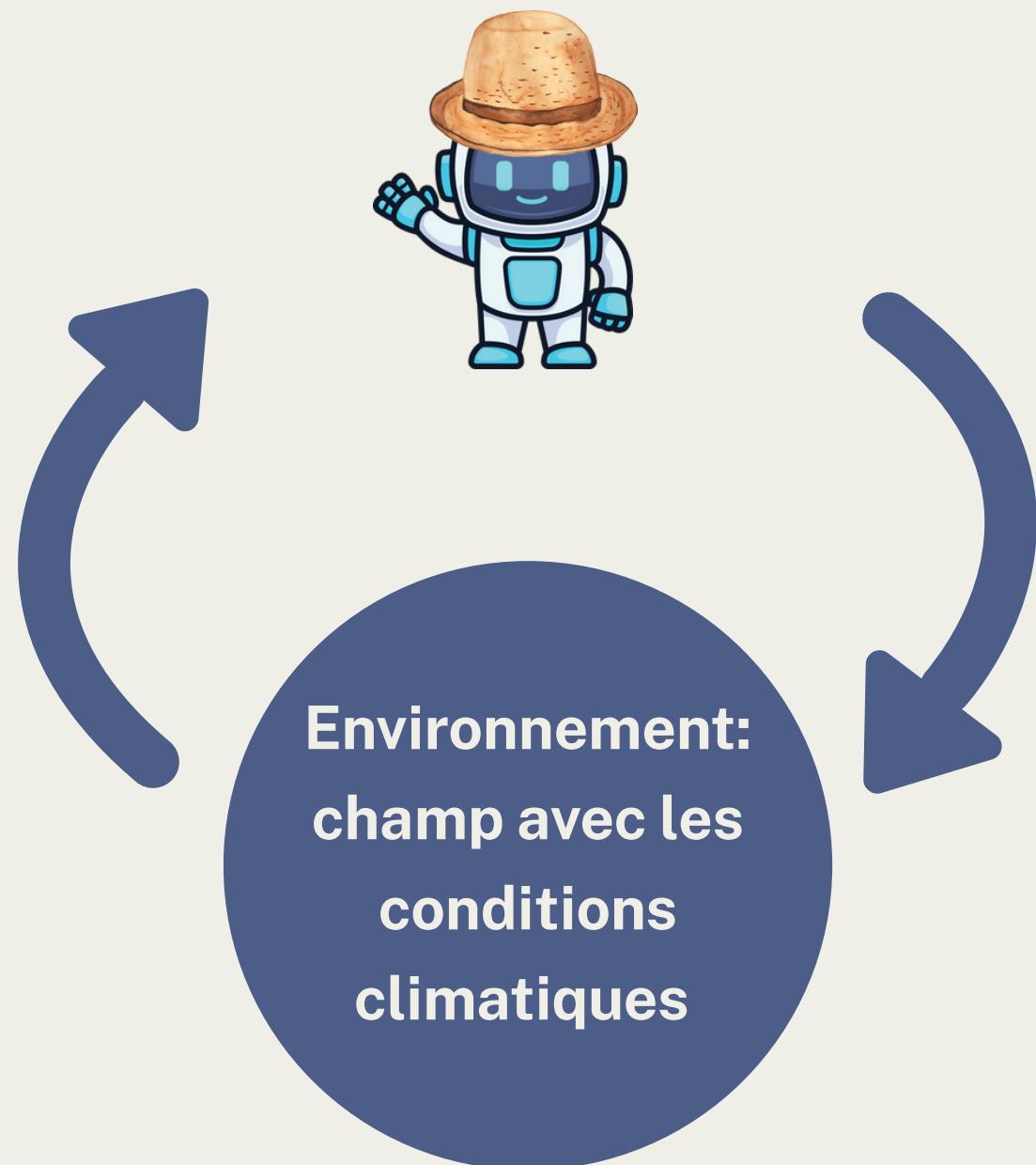


EXEMPLE

Problématique : Comment déterminer automatiquement le niveau d'irrigation optimal pour des plants de tomates en fonction des conditions environnementales et agronomiques, de manière à maximiser la croissance tout en minimisant le gaspillage d'eau ?

State : ce qui va influencer sur la décision d'arrosage et la croissance des plantes

Reward : équilibre entre rendement et coût en eau



Action : quantité d'eau à verser



EXEMPLE

Problématique : Comment déterminer automatiquement le niveau d'irrigation optimal pour des plants de tomates en fonction des conditions environnementales et agronomiques, de manière à maximiser la croissance tout en minimisant le gaspillage d'eau ?

Action : quantité d'eau à verser

4 actions possibles = 4 niveaux d'irrigation possibles

Actions	Interprétation	Objectif
0	pas d'irrigation	Economiser l'eau si le sol est déjà assez humide
1	faible irrigation	Ajustement léger de l'humidité du sol
2	irrigation moyenne	Maintenir un niveau d'humidité optimal
3	forte irrigation	Rattraper un sol trop sec

```
self.action_space = spaces.Discrete(4)
```

EXEMPLE

Reward :
équilibre entre
rendement et
coût en eau

```
def compute_reward(self, soil_moisture, action):
    irrigation_amount = action / 3      #Normalisation de l'action=0 -> 0, action=3 -> 1

    water_cost = 0.4 * irrigation_amount #coût de l'eau

    if soil_moisture < 0.5:
        reward = (soil_moisture + irrigation_amount) - water_cost # sol trop sec -> bonus si on irrigue
    elif soil_moisture > 0.65:
        reward = -(irrigation_amount * 2) - water_cost # sol trop humide -> pénalité si on irrigue
    else:
        reward = 0.45 - water_cost - abs(soil_moisture - 0.57) # zone optimale -> ne pas trop irriguer

    return reward
```

Water cost : plus on irrigue plus on paie un “coût” en eau

0.4 prix de l'eau

- sol sec (<0,5) : récompense positive si irrigation (action utile)

- sol trop humide (>0,65) : récompense négative si irrigation

2 pénalité

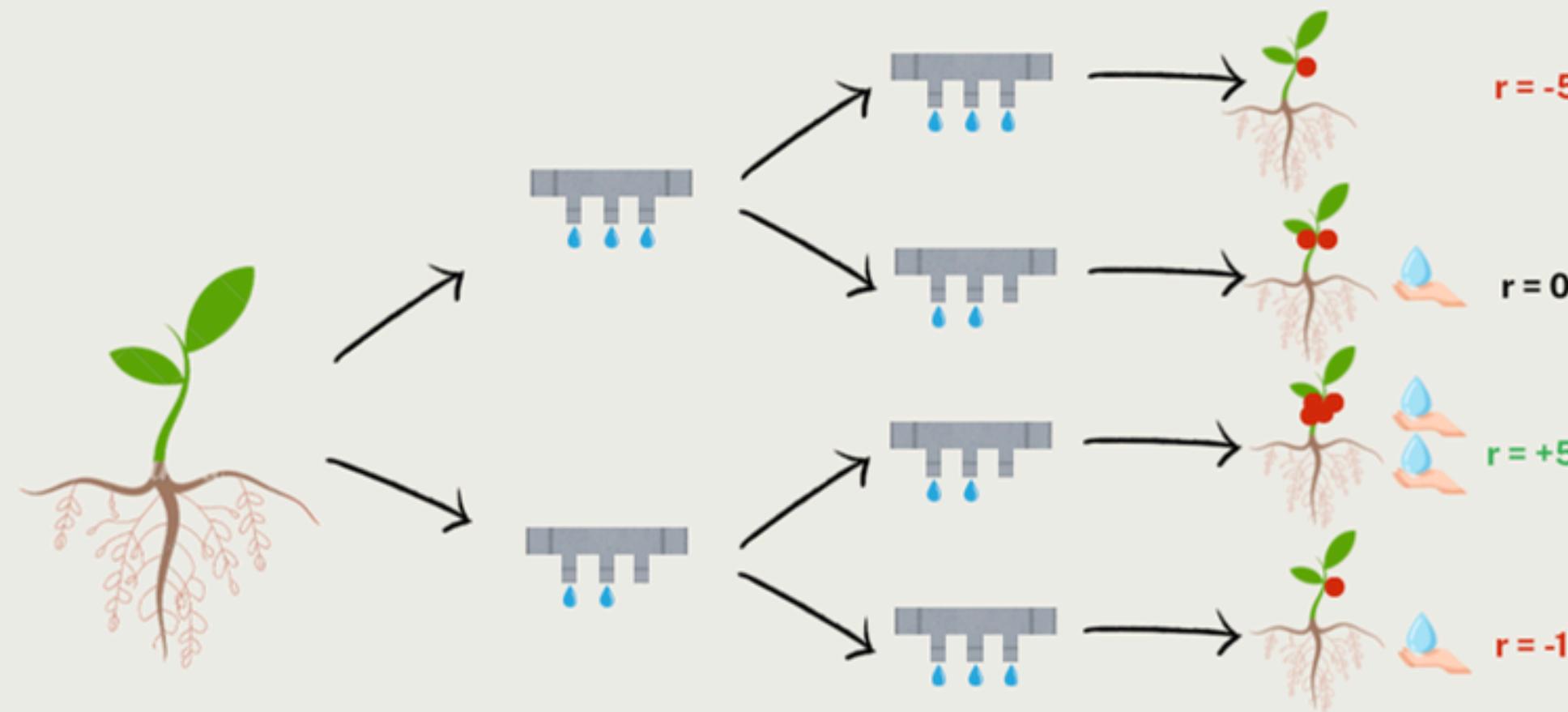
- humidité optimale (0,5 - 0,65) : récompense positive (0,45)

FONCTIONNEMENT GÉNÉRAL

VALUE - FUNCTION

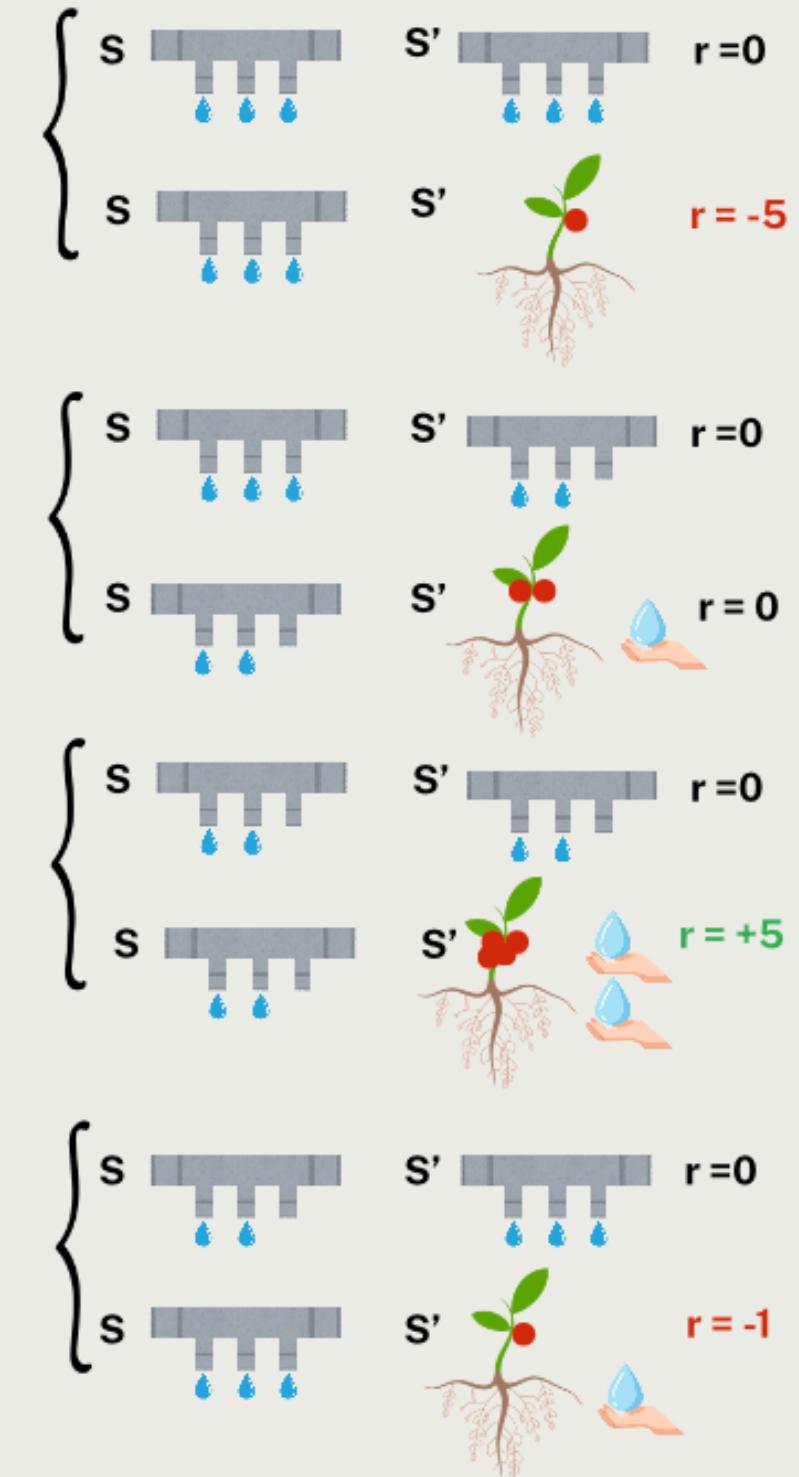
Moyenne de toutes les récompenses possibles obtenues dans un état fixe.

$$V(s) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right]$$



MDP : Markov decision process

L'environnement dans lequel l'agent évolue est une succession d'états et de récompenses (sans mémoire).



FONCTIONNEMENT GÉNÉRAL

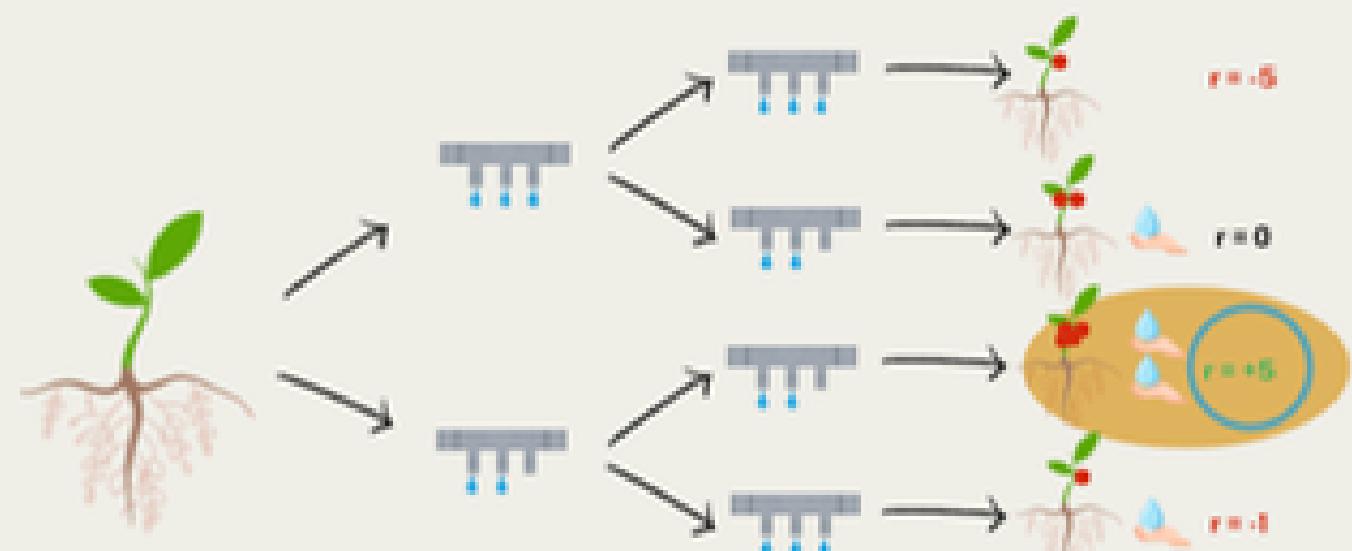
VALUE - FUNCTION

Moyenne de toutes les récompenses possibles obtenues dans un état fixe.

$$V(s) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right]$$

$$V(S) = V(S) + LR^*(V(S') - V(S))$$

Modifier la valeur de l'état actuel en fonction des valeurs des états futurs, effectuer un backtracking : prendre les récompenses de fin d'expérience et les faire remonter jusqu'aux états initiaux afin de maximiser la récompense finale.



$$V(\text{幼苗}) = 0$$

$$V(\pi) = 0$$

$$V(\text{---}) = 0$$

$$V(\overline{\text{TT}}\text{TT}) = 0$$

$$\nabla(\frac{\partial f}{\partial x} \frac{\partial f}{\partial y}) = 0$$

$$V(\overline{\pi}, \overline{\pi}) = 0$$

$$\nabla \left(\text{---} \right) = 0$$

$$\nabla \left(\frac{\partial L}{\partial u} \right) = 0$$

$$V(\textcolor{red}{\pm}) = 0$$

$$\nabla \left(\frac{\partial}{\partial x} \right) = 0$$

$$V(s) = 0 + 0,5 \cdot (5-0) = 2,5$$



VII

$$V\left(\frac{\pi}{3}\right) = 0$$

$$\nabla \left(\frac{\cdot}{\cdot} \right) = 0$$

$$\nabla \left(\frac{\partial}{\partial x} \frac{\partial}{\partial y} \right) =$$

$\nabla \left(\begin{array}{c} \text{TTT} \\ \text{TTF} \end{array} \right) =$

$\nabla(\Pi\Pi) =$

$$\nabla \cdot (\text{grad } f) = 0$$

$$\nabla \cdot \mathbf{v} = 0$$

$$\sqrt{1 + \frac{1}{4}} = \frac{\sqrt{5}}{2}$$

$$V(\cdot) = 0$$

FONCTIONNEMENT GÉNÉRAL

VALUE - FUNCTION

Moyenne de toutes les récompenses possibles obtenues dans un état fixe.

$$V(S) = V(S) + LR * (V(S') - V(S))$$

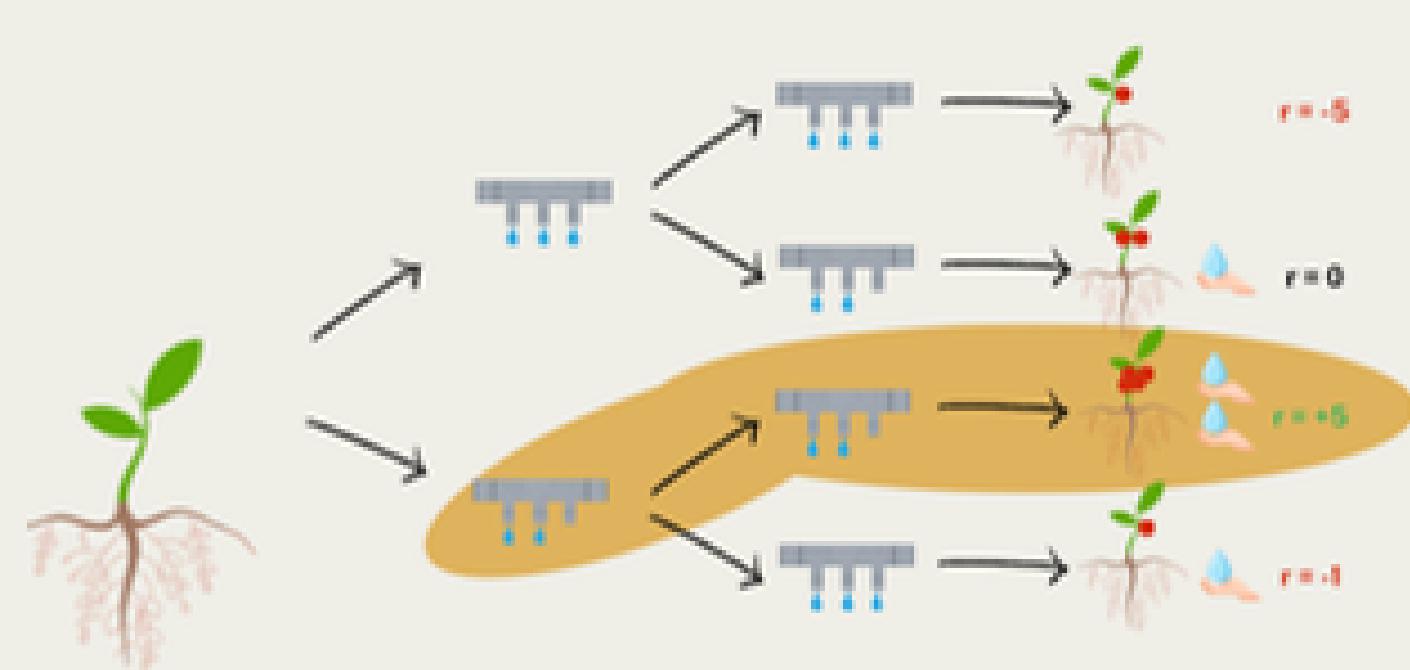
FONCTIONNEMENT GÉNÉRAL

VALUE - FUNCTION

Moyenne de toutes les récompenses possibles obtenues dans un état fixe.

$$V(s) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right]$$

$$V(S) = V(S) + LR * (V(S') - V(S))$$



$$\begin{aligned} V(\text{Seedling}) &= 0 \\ V(\text{Grey}) &= 0 \\ V(\text{Grey}) &= 0 \\ V(\text{Grey Grey}) &= 0 \\ V(\text{Grey Grey}) &= 1.25 \\ V(\text{Grey Grey}) &= 0 \\ V(\text{Red}) &= 0 \\ V(\text{Red Blue}) &= 0 \\ V(\text{Red Blue}) &= 2.5 \\ V(\text{Red Blue}) &= 0 \end{aligned}$$

$$V(s) = 0 + 0.5(1.25 - 0) = 0.625$$

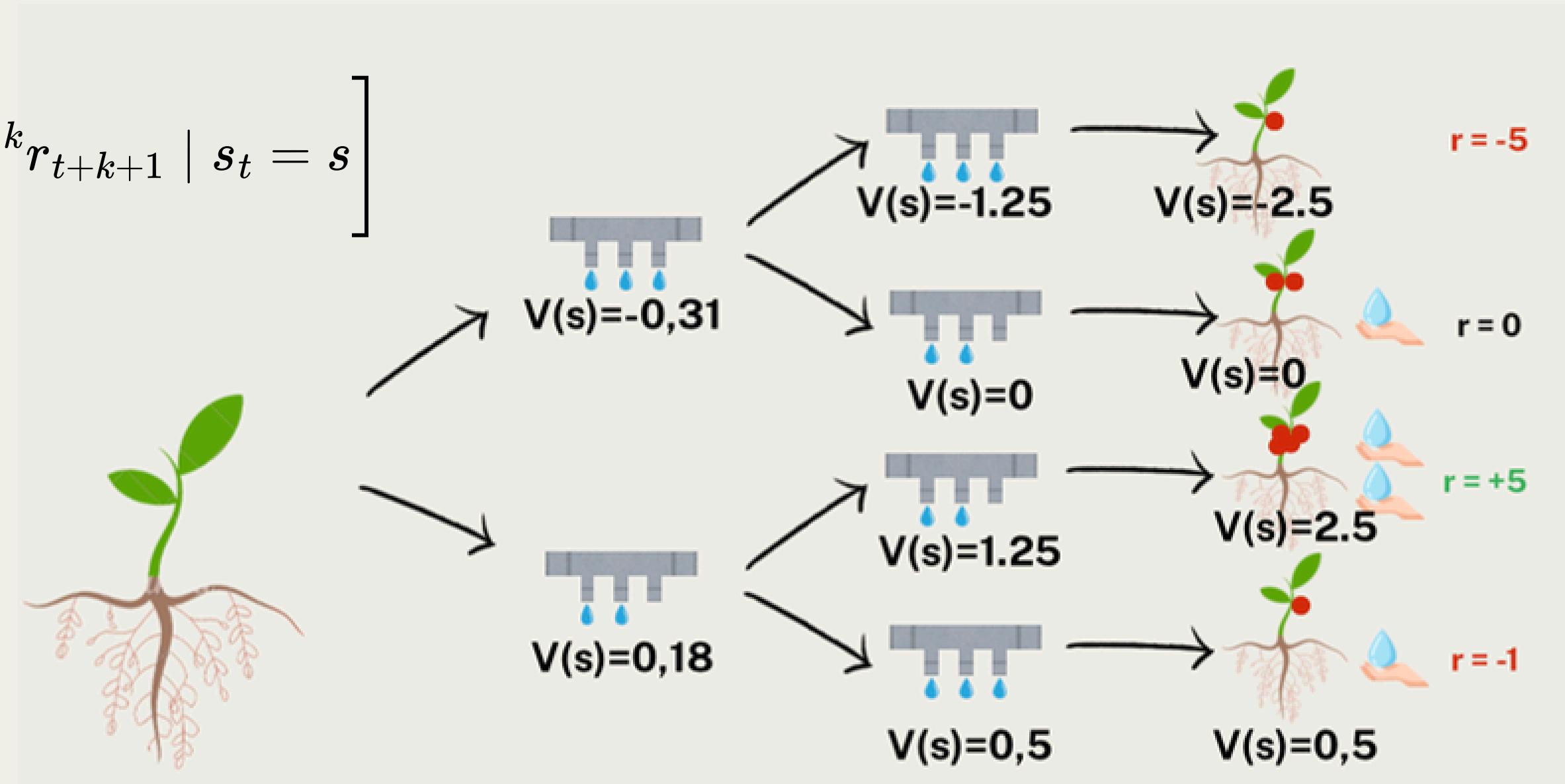
$$\begin{aligned} V(\text{Seedling}) &= 0 \\ V(\text{Grey}) &= 0 \\ V(\text{Grey}) &= 0.625 \\ V(\text{Grey Grey}) &= 0 \\ V(\text{Grey Grey}) &= 0 \\ V(\text{Grey Grey}) &= 1.25 \\ V(\text{Grey Grey}) &= 0 \\ V(\text{Red}) &= 0 \\ V(\text{Red Blue}) &= 0 \end{aligned}$$

FONCTIONNEMENT GÉNÉRAL

VALUE - FUNCTION

Moyenne de toutes les récompenses possibles obtenues

$$V(s) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right]$$

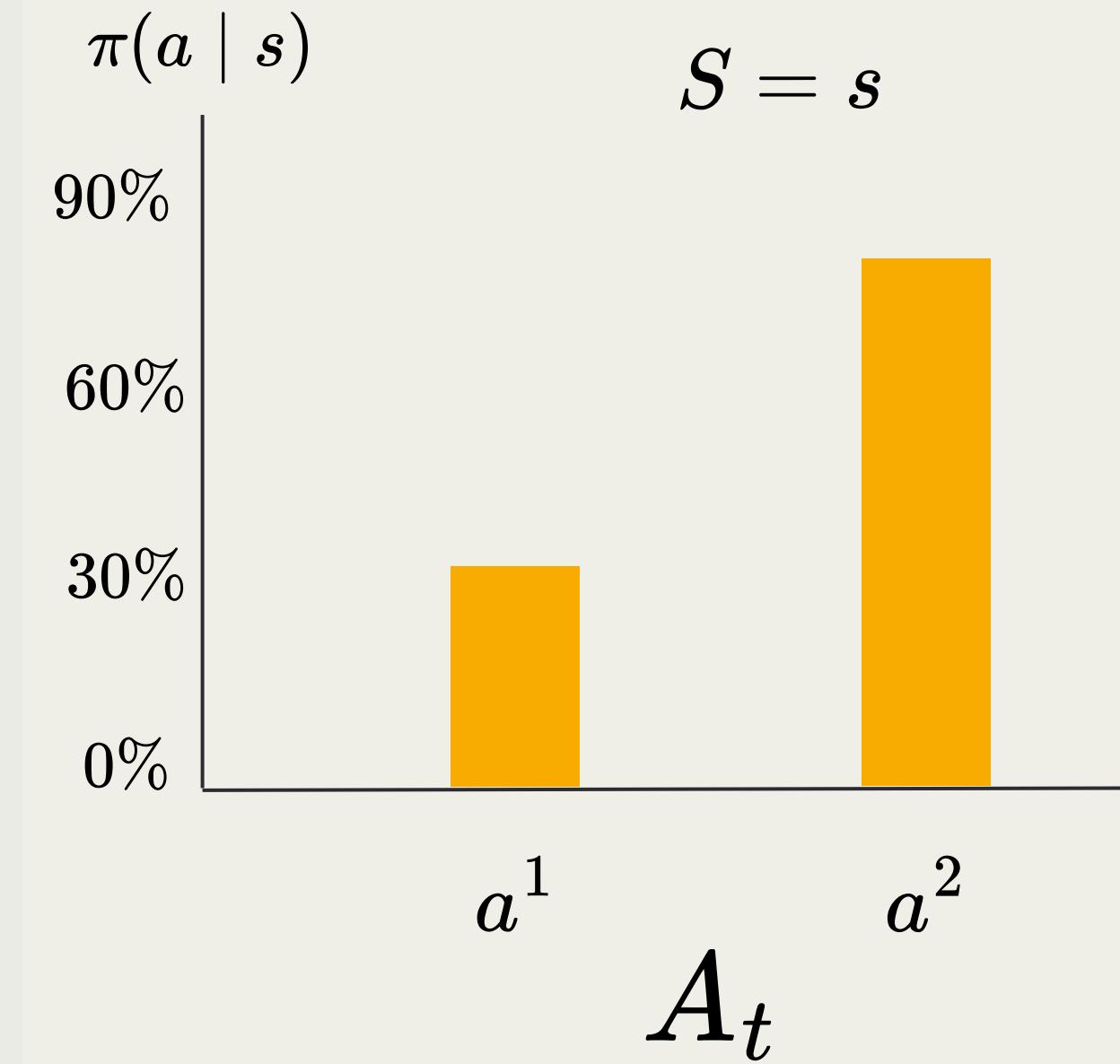
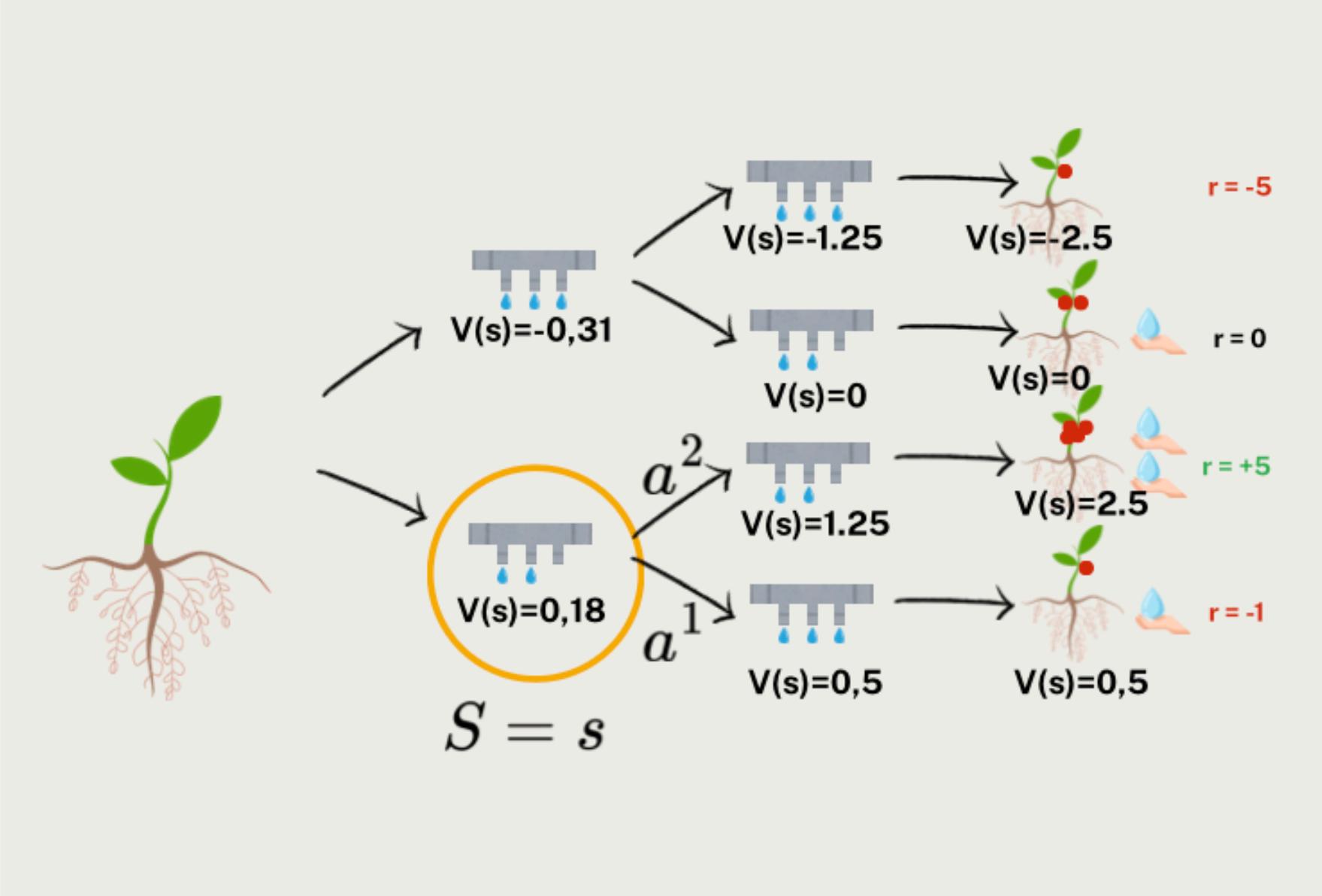


Capable d'estimer à quel point il est bon d'être dans un certain état

FONCTIONNEMENT GÉNÉRAL

POLICY

Donne le comportement de l'agent, probabilité que l'agent prenne chaque action dans un état particulier

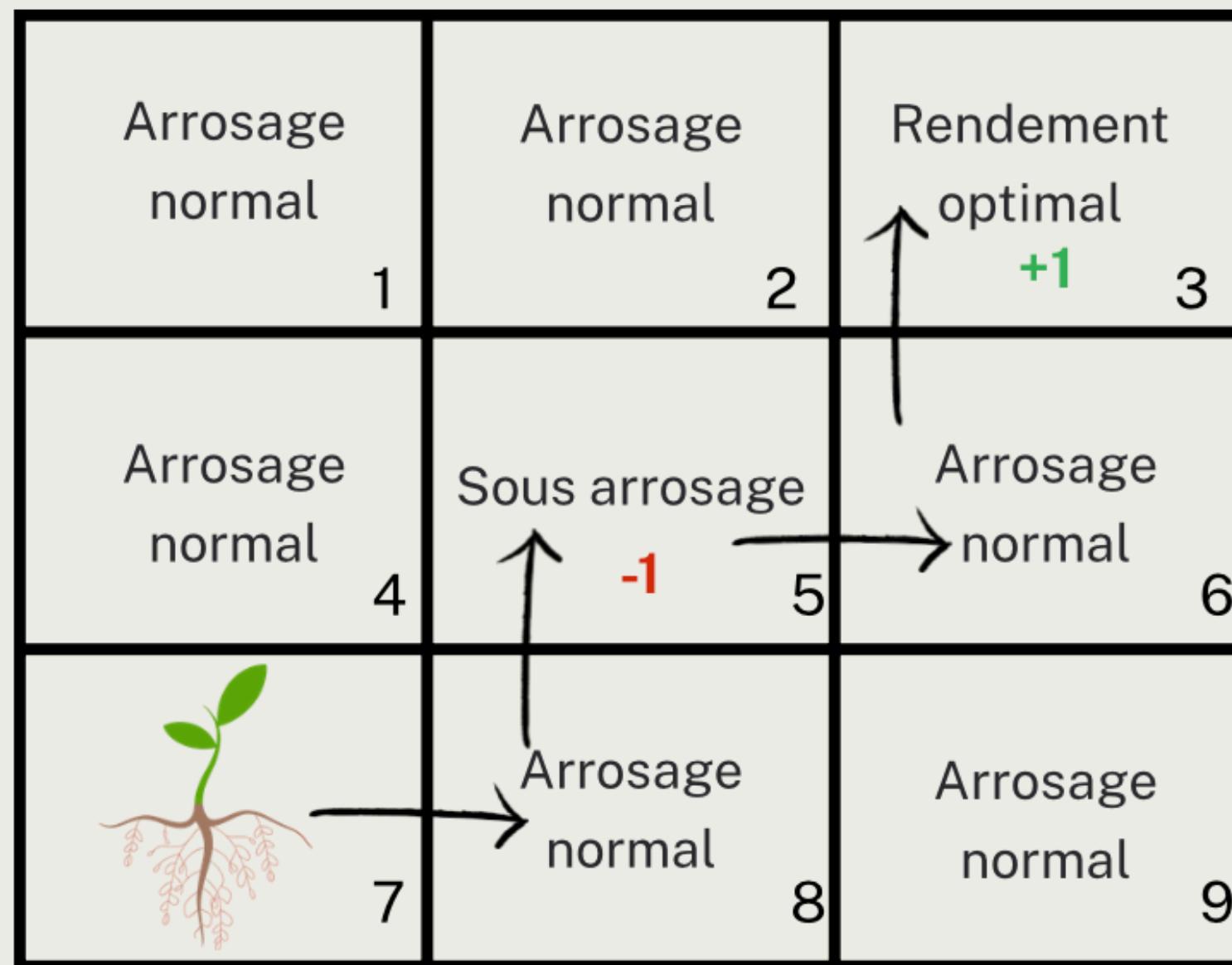


On voit que la policy prend en compte l'action en plus de l'état ...

Q - LEARNING

$$Q(s_t, a_t)$$

State Action



Objectif : rendement optimal

$$R_{t+1} + R_{t+2} + R_{t+3} + R_{t+4}$$

$$0 + (-1) + 0 + 1$$

γ (ex : 0,9), erreur arbitraire : à quel point on apporte de l'importance aux prochaines récompenses.

$$R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4}$$

$$0 + (-0,9) + 0 + 0,72$$

Q - LEARNING

L'objectif est de maximiser la somme des récompenses obtenues au cours de la trajectoire :

La policy π cherche à maximiser G_t

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

Ainsi, appliquer la policy à 100 % revient à effectuer uniquement de l'exploitation.

$$Q(s_t, a_t)^\pi = \mathbb{E} [R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | s_t, a_t]$$

Une politique optimale revient donc à maximiser la q-function (moyenne des récompenses prenant en compte *action* et *état*)

$$\pi^*(s) = \arg \max_a q^*(s, a)$$

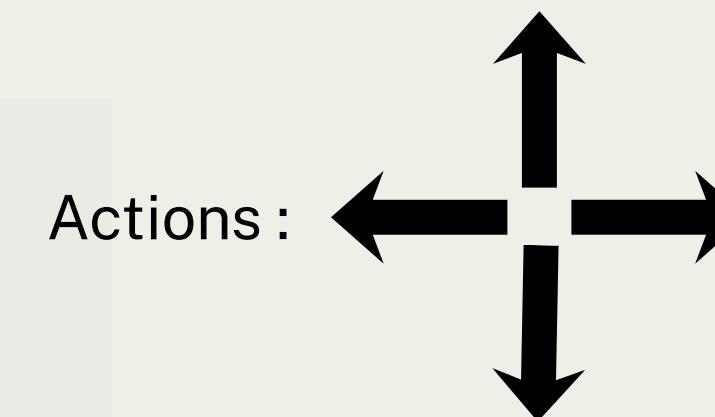
Redéfinie de manière recursive avec l'**équation de Bellman** :

$$Q(s_t, a_t)^\pi = r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})^\pi$$

γ : La récompense qui vient d'être obtenue. Ensuite, l'action a_{t+1} est sélectionnée de manière à maximiser la valeur de la fonction Q à l'instant $t+1$, tout en respectant la policy.

Q - LEARNING

Arrosage normal 1	Arrosage normal 2	Rendement optimal +1 3
Arrosage normal 4	Sous arrosage -1 5	Arrosage normal 6
	Arrosage normal 7	Arrosage normal 8

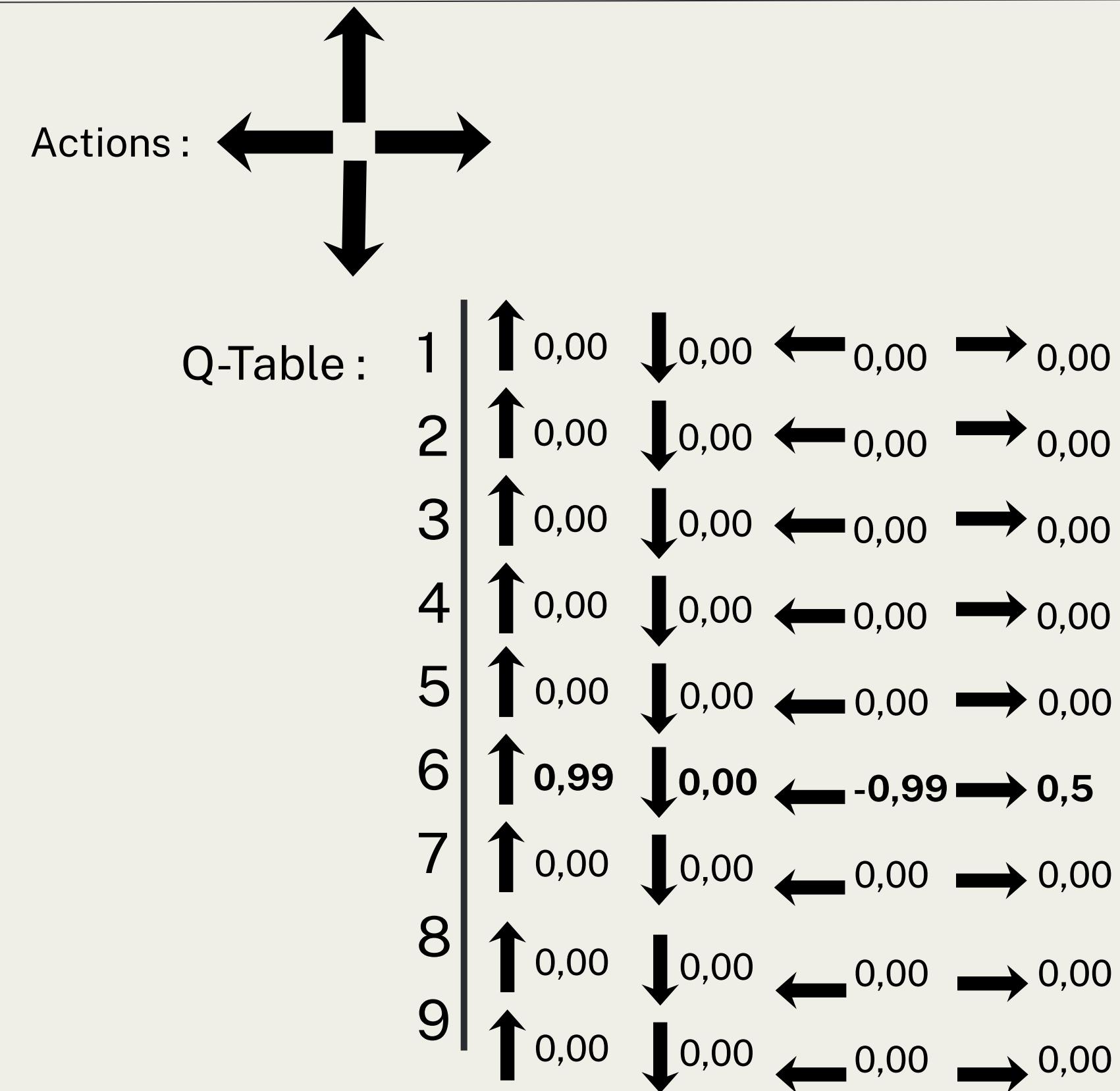


Q-Table :

	↑ 0,00	↓ 0,00	← 0,00	→ 0,00
1	↑ 0,00	↓ 0,00	← 0,00	→ 0,00
2	↑ 0,00	↓ 0,00	← 0,00	→ 0,00
3	↑ 0,00	↓ 0,00	← 0,00	→ 0,00
4	↑ 0,00	↓ 0,00	← 0,00	→ 0,00
5	↑ 0,00	↓ 0,00	← 0,00	→ 0,00
6	↑ 0,00	↓ 0,00	← 0,00	→ 0,00
7	↑ 0,00	↓ 0,00	← 0,00	→ 0,00
8	↑ 0,00	↓ 0,00	← 0,00	→ 0,00
9	↑ 0,00	↓ 0,00	← 0,00	→ 0,00

Q - LEARNING

Arrosage normal 1	Arrosage normal 2	Rendement optimal +1 3
Arrosage normal 4	Sous arrosage -1 5	 6
Arrosage normal 7	Arrosage normal 8	Arrosage normal 9



Q - FUNCTION : UPDATE

Permet la propagation des récompenses dans les états précédents.

$$Q(s_t, a_t)_{\text{new}} = Q(s_t, a_t)_{\text{old}} + \alpha \left[r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)_{\text{old}} \right]$$

$r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)_{\text{old}}$: La différence entre la valeur actuelle et la valeur de l'action optimale à l'instant $t+1$, le facteur γ (qui réduit l'importance accordée aux récompenses futures) ainsi que la récompense immédiate r .

$Q(s_t, a_t)_{\text{old}} + \alpha \left[r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)_{\text{old}} \right]$: Le learning rate α permet de moyenner cette mise à jour sur un grand nombre de transitions et empêche une modification immédiate.

Q - LEARNING

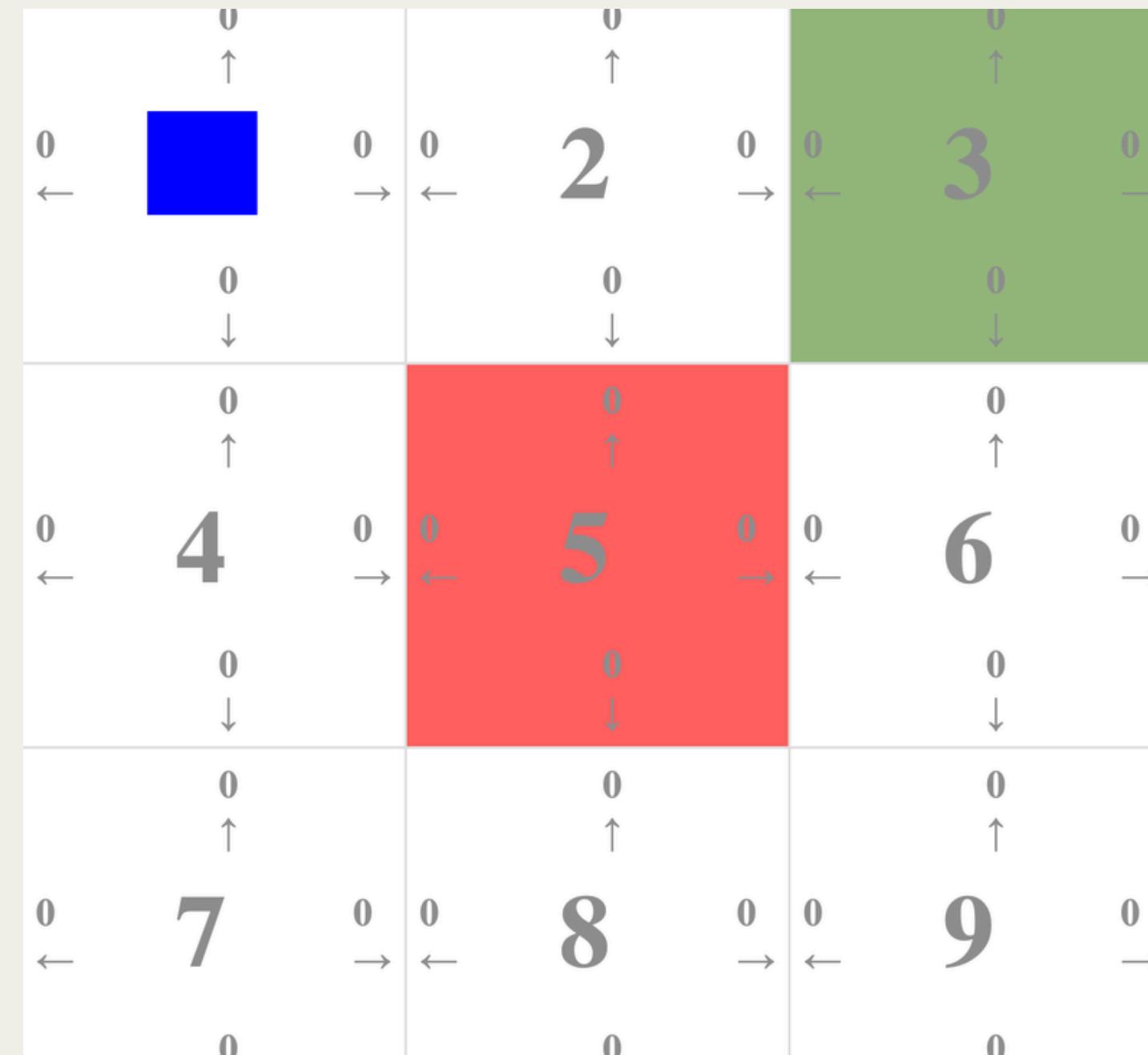
Q - FUNCTION : UPDATE

Permet la propagation des récompenses dans les états précédents.

ϵ - greedy = 0,4

40% exploration

60% exploitation



Q - TABLE :

Cette approche n'est pas extensible à des environnements plus complexes, la Q-table devenant alors beaucoup trop volumineuse.

E X E M P L E

Kumar Kasera, Rohit; Acharjee, Tapodhir (2024), "Dataset on irrigation for Tomato", Mendeley Data

Jeu de données contenant des mesures environnementales et agronomiques pour des plants de tomates cultivés dans une zone aride en Algérie. Il sert de base à un agent d'apprentissage par renforcement afin d'optimiser les décisions d'irrigation.

3000 observations

14 variables

Conditions climatiques :

définissent l'état climatique à chaque instant

Sol : caractérisent l'état du sol et la santé de la plante

Rendement / Irrigation : Indicateurs d'arrosage et de besoins en eau de la plante

Temps : permet d'avoir une évolution temporelle



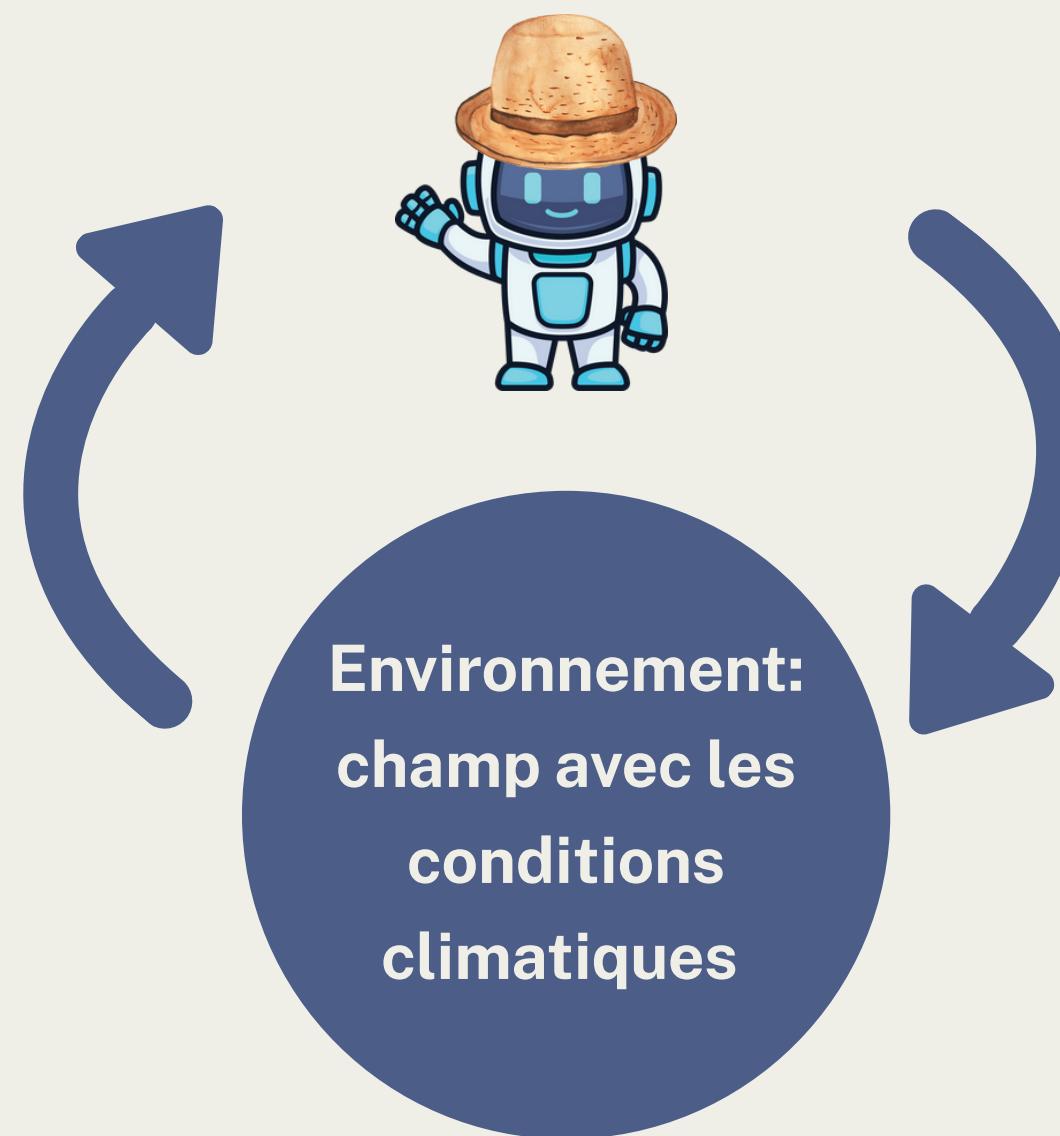
Temperature...C.	: num	31.2 31.2 30.5 30.4 30.4 30.3 30.3 30.3 ...
Humidity....	: num	93.6 93.6 74.6 76.6 76.6 77.1 76.5 78.1 ...
Soil.moisture	: num	567 567 307 308 308 307 307 307 307 307 ...
Reference.evapotranspiration	: num	563 561 561 559 559 559 559 559 559 559 ...
Evapotranspiration	: num	236 236 236 235 235 235 235 235 235 235 ...
Crop.Coefficient	: num	0.42 0.42 0.42 0.42 0.42 0.42 0.42 0.42 0.42 0.42 ...
Crop.Coefficient.stage	: chr	"Initial Stage" ...
Nitrogen..mg.kg.	: int	107 107 107 107 107 107 107 107 107 107 ...
Phosphorus..mg.kg.	: int	38 38 38 38 38 38 38 38 38 38 ...
Potassium	: int	53 53 53 53 53 53 53 53 53 53 ...
Solar.Radiation.ghi	: num	622 622 622 622 622 622 622 622 622 622 ...
Wind.Speed	: num	2.09 2.09 2.09 2.09 2.09 2.09 2.09 2.09 2.09 2.09 ...
Days.of.planted	: int	1 1 3 3 3 3 3 3 3 3 ...
pH	: num	3.32 3.77 2.9 7.71 5.11 4.48 4.32 3.77 ...

EXEMPLE

Problématique : Comment déterminer automatiquement le niveau d'irrigation optimal pour des plants de tomates en fonction des conditions environnementales et agronomiques, de manière à maximiser la croissance tout en minimisant le gaspillage d'eau ?

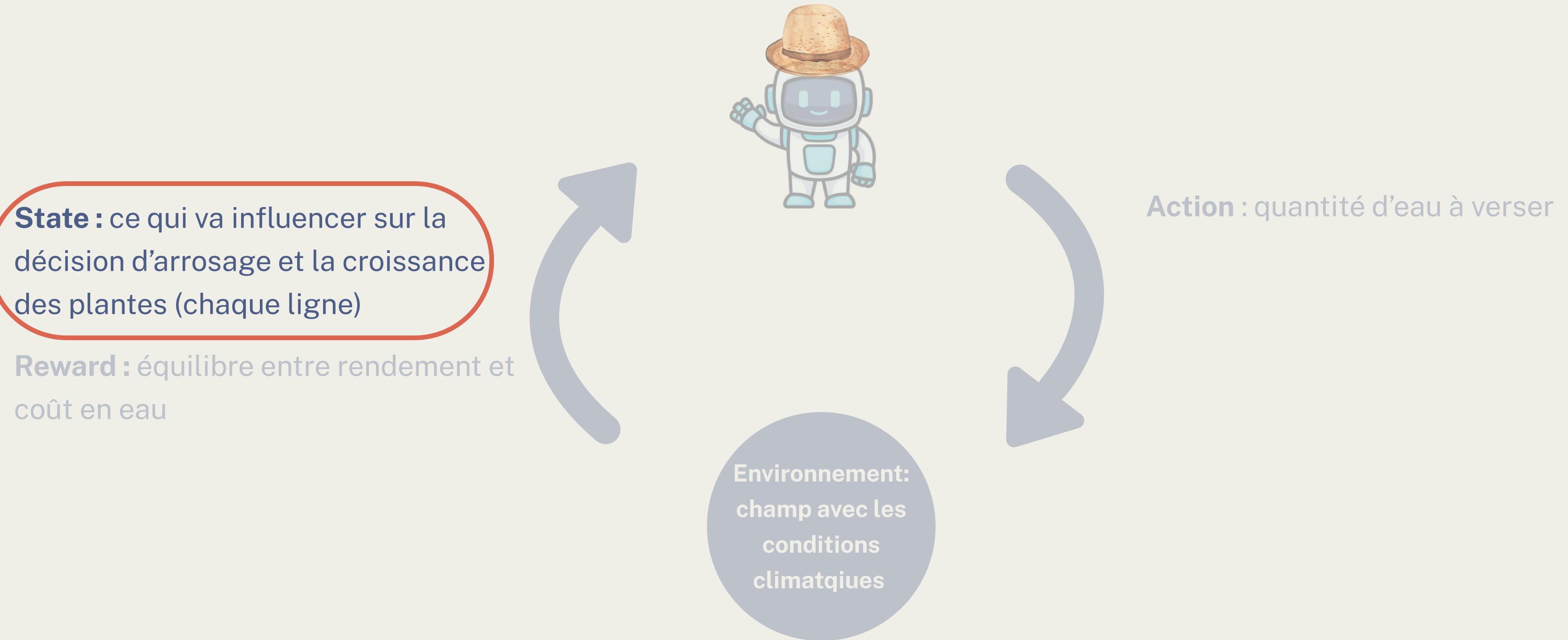
State : ce qui va influencer sur la décision d'arrosage et la croissance des plantes (chaque ligne)

Reward : équilibre entre rendement et coût en eau



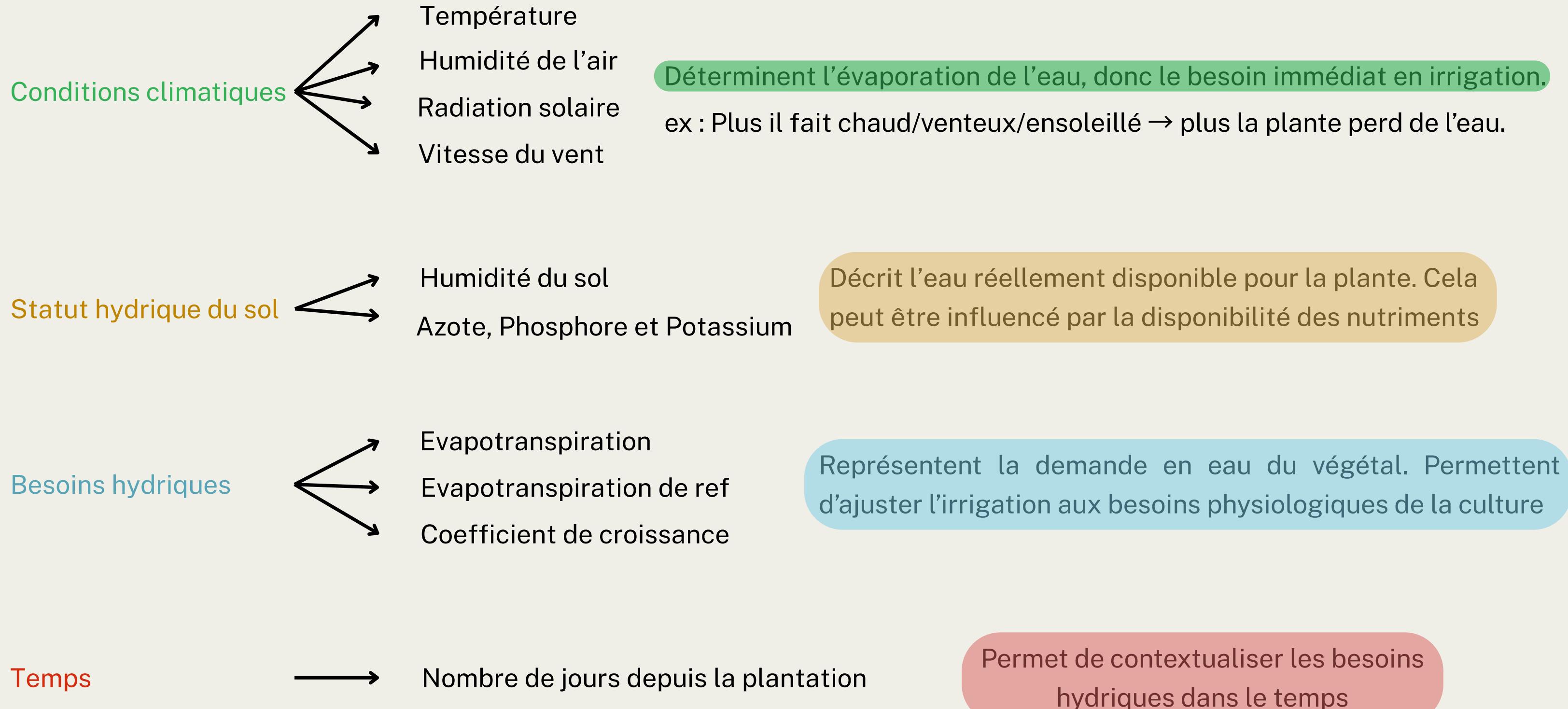
Action : quantité d'eau à verser

EXEMPLE



EXEMPLE

Pourquoi ces variables définissent l'état ?



PROXIMAL POLICY OPTIMIZATION

(Schulman et al., 2017)

Optimisation de politique proximale : algorithme d'optimisation de la politique de notre agent, introduit par OpenAI en août 2017. Le PPO utilise un gradient de fonction pour minimiser la fonction de Loss propre à cet algorithme.

Dans le RL présenté, on optimise la politique : $\pi(s) = \text{argmax}_a Q(s, a)$ directement; tandis qu'avec PPO : optimisation de la politique via un objectif *surrogate* (proxy, agent intermédiaire/substitut) sans expliciter la maximisation de Q.

Le **PPO utilise un objectif “clipped”** pour limiter le changement de politique :

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]$$

→ Avec $r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)}$: le rapport entre la politique considérée et l'ancienne.

→ La fonction Lclip encourage donc le changement de pi à 1+/-epsilon près, pondéré par l'avantage .

“Avec ce système, nous ignorons uniquement la variation du rapport de probabilité lorsqu'elle améliore l'objectif, et nous la prenons en compte lorsqu'elle le détériore.”

PROXIMAL POLICY OPTIMIZATION

(Schulman et al., 2017)

La fonction de Loss L^{CLIP} mesure à quel point la politique $\pi_t(\theta)$ au temps t est mauvaise par rapport à la politique $\pi_{t-1}(\theta)$ au temps t-1, pondéré par l'avantage \hat{A} .

On a donc la fonction L_{clip} qui met à jour θ (paramètres de la politique) :

- Si $r_t(\theta)$ est trop grand (ex. : $> 1+\epsilon$), le gradient est "clippé" pour éviter une mise à jour excessive.
- Si $r_t(\theta)$ est trop petit (ex. : $< 1-\epsilon$), le gradient est aussi clippé.

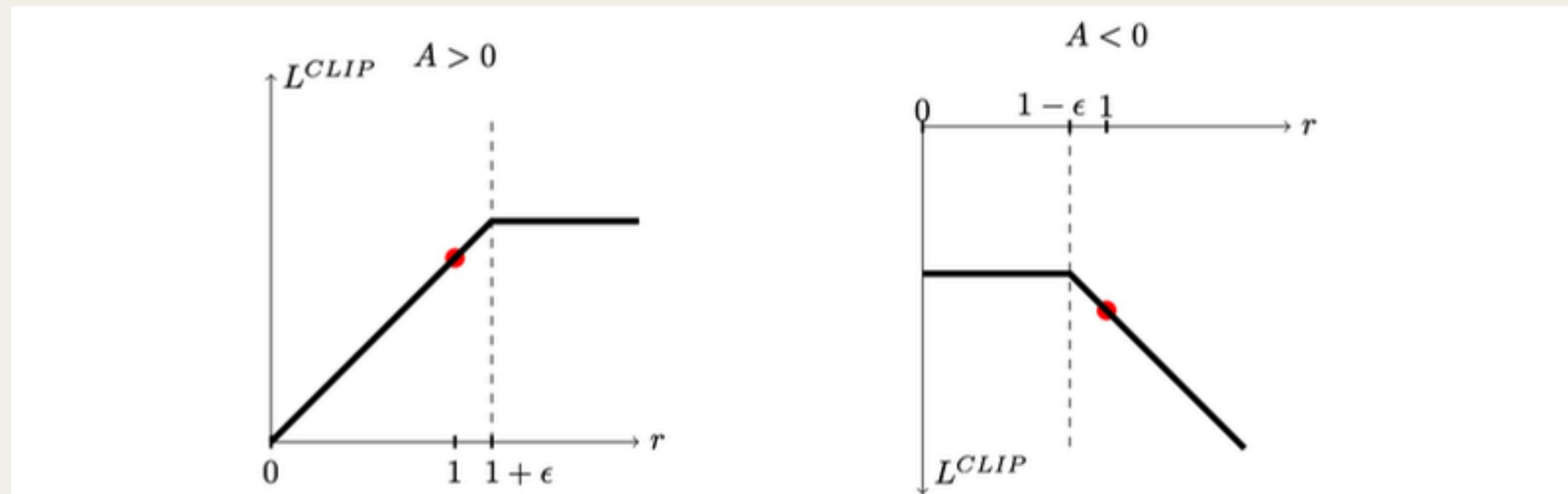


Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function L^{CLIP} as a function of the probability ratio r , for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP} sums many of these terms.

PROXIMAL POLICY OPTIMIZATION

(Schulman et al., 2017)

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]$$

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t-1} r_{T-1} + \gamma^{T-t} V(s_T)$$

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t-1}\delta_{T-1},$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

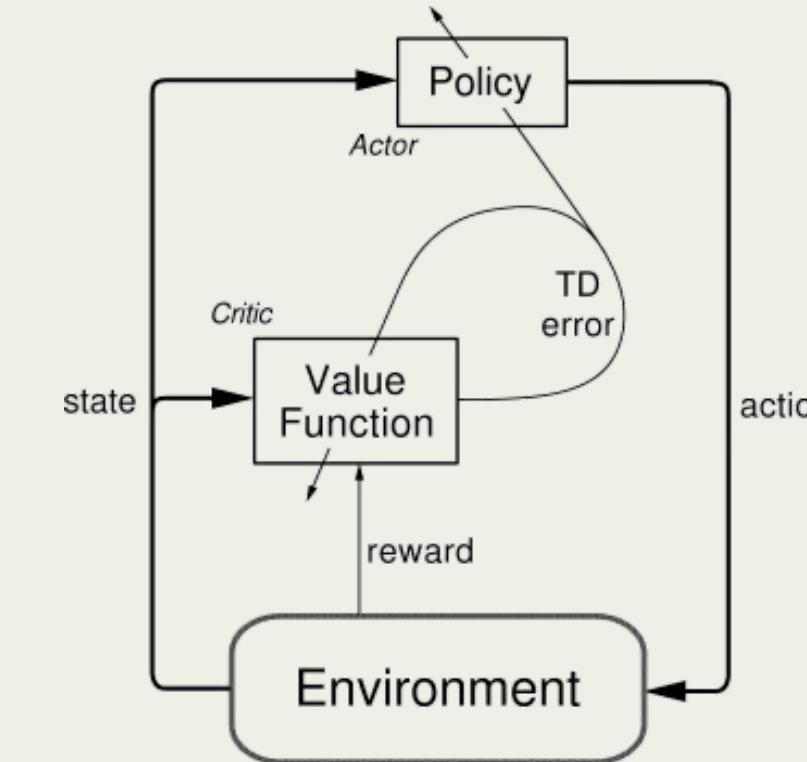
- Indique si une action est meilleure ou pire que la moyenne
- Cet avantage \hat{A} induit la “direction” du changement, **est ce que l’ancienne ou la récente version de la politique est meilleure.** Ce qui pénalise les changements trop importants de la politique ($\pi(s)$).

$\theta \leftarrow \theta - \alpha \nabla_\theta L^{\text{CLIP}}(\theta)$: les paramètres théta de la politique sont mis à jour
 Avec α : taux d'apprentissage (learning rate).

PROXIMAL POLICY OPTIMIZATION

(Schulman et al., 2017)

On appelle cet algorithme “**actor-critic**” (politique = actor et value function = critic) : la politique est constamment “**critiquée**” par la value function. Pas de maximisation explicite de Q : la politique est mise à jour en maximisant un objectif surrogate (encourage des changements modérés).

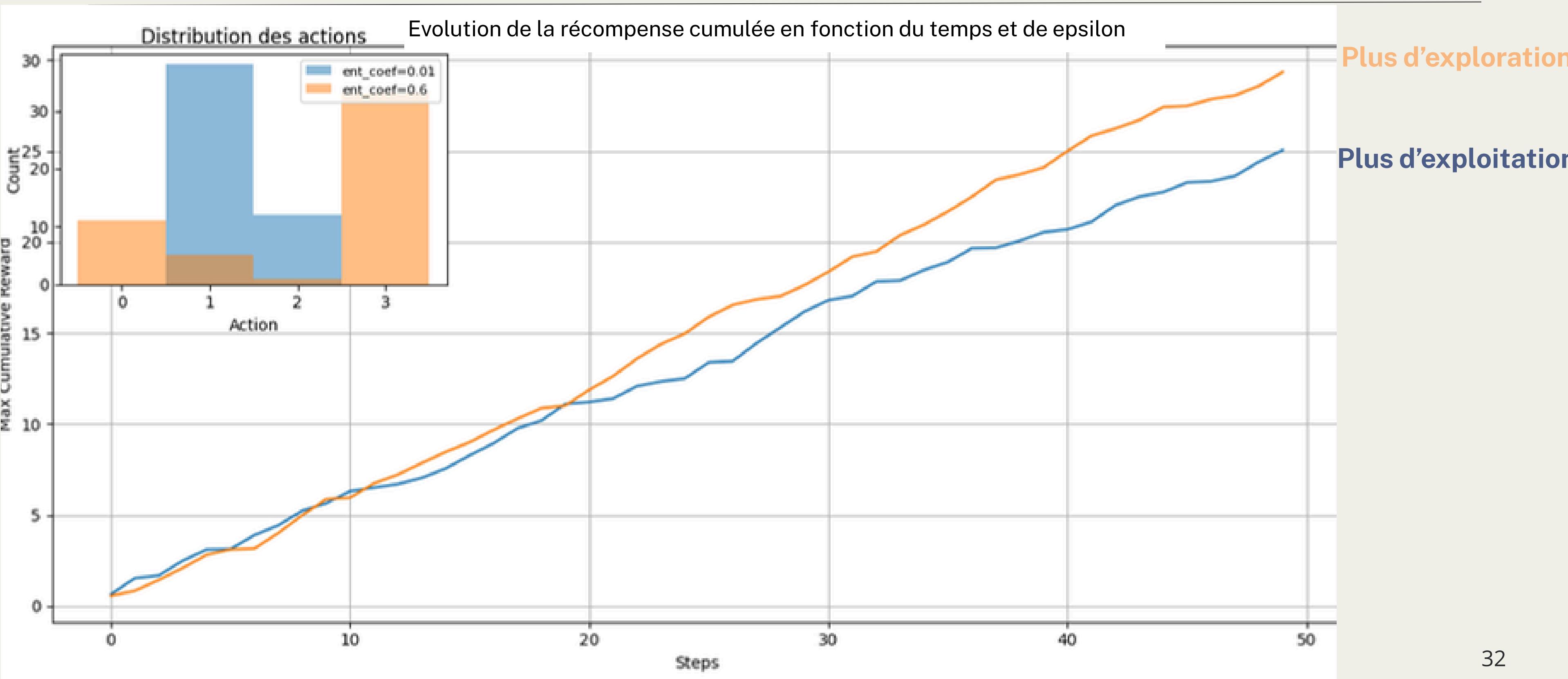


Analogie : Un acteur de théâtre (politique) qui apprend quel geste faire (action) dans une scène donnée (état) pour recevoir des applaudissements (récompense); et un critique de théâtre (value function) qui donne un feedback à l'acteur sur la qualité de son jeu, en disant "cette action était bonne/mauvaise" sans trop le brusquer et prioriser les changements doux (clipped loss).

Précisions :

- La State Value Function $V(s)$ est apprise pour estimer l'avantage \hat{A} , mais pas pour dériver directement la politique (contrairement à Q-learning).
- L'avantage \hat{A} est utilisé pour pondérer les mises à jour de la politique, pas pour sélectionner les actions.

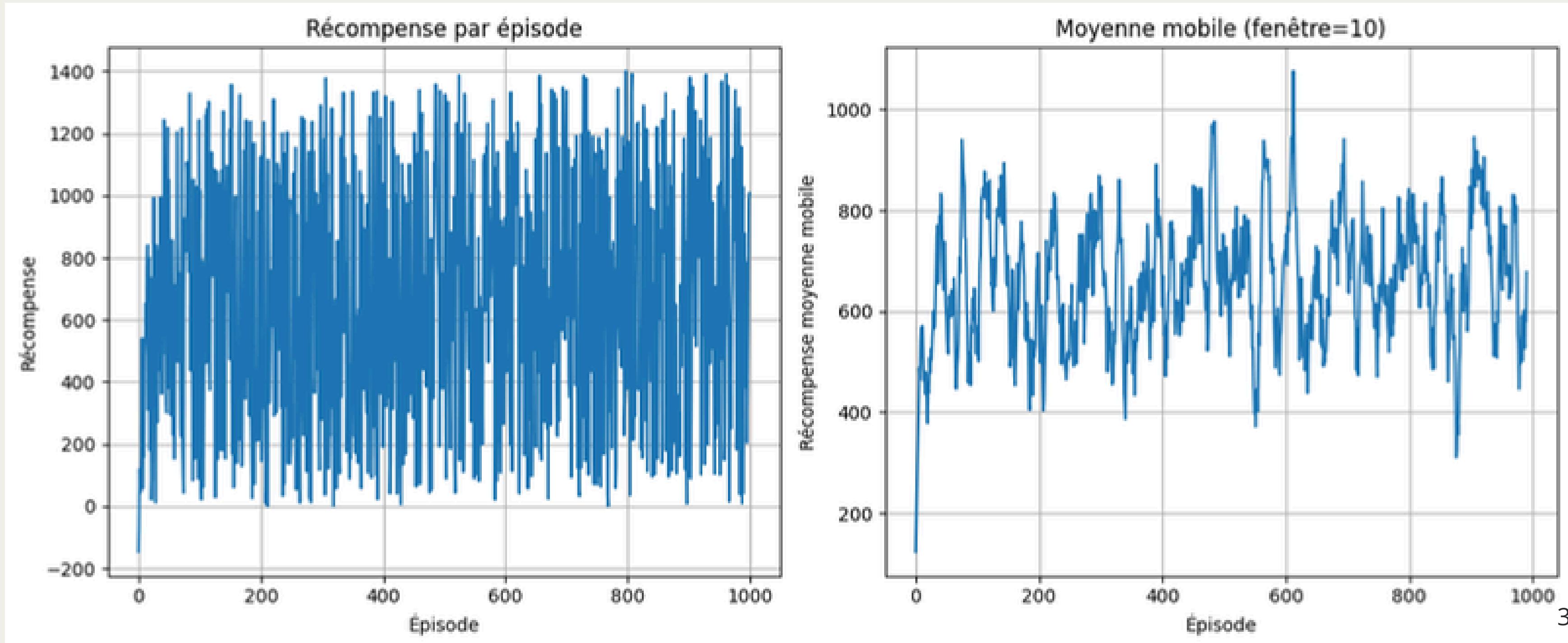
VISUALISATIONS



Apprentissage par renforcement

VISUALISATIONS

Evolution de la récompense moyenne en fonction des épisodes (“session d’entraînement”)



RÉSULTATS

L'apprentissage par renforcement démontre qu'il peut devenir un **outil efficace** pour une **irrigation intelligente**, capable d'adapter les apports d'eau aux besoins réels de la plante et de réduire les consommations hydriques.

Les résultats montrent que l'agent apprend à arroser davantage lorsque l'humidité du sol est faible, et à réduire l'irrigation lorsqu'elle est suffisante. La stratégie obtenue équilibre efficacement rendement potentiel et coût en eau, répondant ainsi à notre problématique d'optimisation

Croisement de nos résultats avec ceux d'OpenAI (Schulman et al., 2017).

Chaque graphique montre l'évolution de la récompense cumulée moyenne en fonction du nombre de timesteps.

(X) : Nombre de timesteps (de 0 à 1 000 000).

(Y) : Récompense cumulée moyenne obtenue par les algorithmes.

PPO (Clip) et TRPO montrent non seulement de meilleures performances, mais aussi une plus grande stabilité (moins de variabilité).

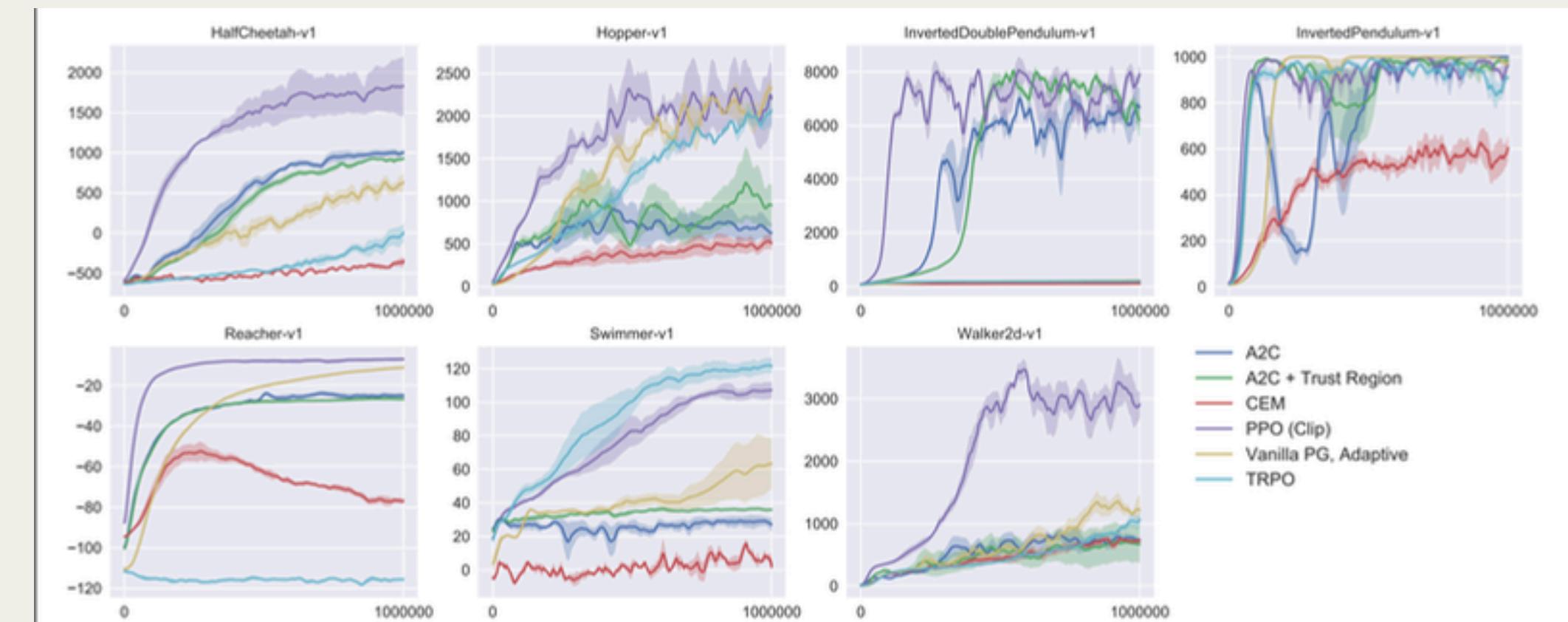


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps. 34

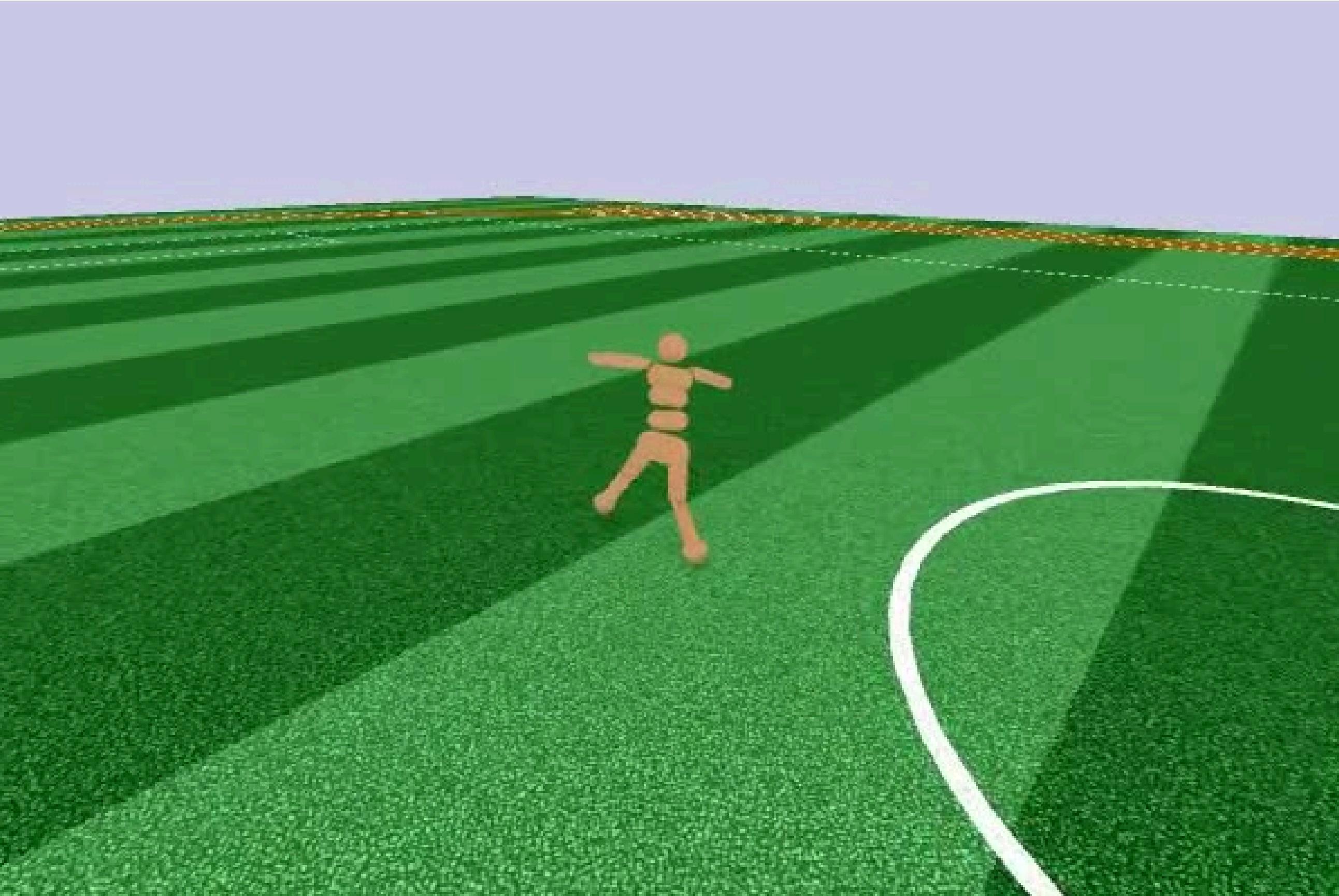
OUVERTURE

Cet algorithme n'a pas été développé par hasard:

- aout 2017 : publication de l'article ***Proximal Policy Optimization Algorithms*** (Schulman et al., 2017) par OpenAI
- février 2018 : **multi goal reinforcement learning** (fondé sur PPO)
- sortie de **chatGPT-3**, premier modèle d'OpenAI, avec les paramètres du réseau de neurones tunés par PPO (Ray, 2023)

DANGERS (Duan & Zou, 2025):

- **Certains robots “physiques” boosté par RL peuvent maximiser leur Reward à outrance s'ils ne sont pas cadrés par des gardes-fous** (The death of Elaine Herzberg, perte de 460M\$ par Knight Capital : “flash crash” pour le bénéfice).
- **Des systèmes de RL peuvent dénicher des failles ou des raccourcis dangereux dans le but de maximiser leur Reward** (pas de notion “d'éthique”)



“PPO lets us train AI policies in challenging environments, like the Roboschool one shown above where an agent tries to reach a target (the pink sphere), learning to walk, run, turn, use its momentum to recover from minor hits, and how to stand up from the ground when it is knocked over.”

Sources et bibliographie

- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017, 20 juillet). Proximal Policy optimization Algorithms. arXiv.org. <https://arxiv.org/abs/1707.06347>
- Treloar, N. J., Fedorec, A. J. H., Ingalls, B., & Barnes, C. P. (2020). Deep reinforcement learning for the control of microbial co-cultures in bioreactors. PLoS Computational Biology, 16(4), e1007783. <https://doi.org/10.1371/journal.pcbi.1007783>
- Andrew, A. M. (1999). REINFORCEMENT LEARNING : AN INTRODUCTION by Richard S. Sutton and Andrew G. Barto, Adaptive Computation and Machine Learning series, MIT Press (Bradford Book), Cambridge, Mass., 1998, xviii + 322 pp, ISBN 0-262-19398-1, <https://doi.org/10.1017/s026357479921174>
- Mutual Information. (2022, 24 octobre). Reinforcement Learning, by the Book [Vidéo]. YouTube. https://www.youtube.com/watch?v=NFo9v_yKQXA
- Julia Turc. (2025, 7 mars). Proximal Policy Optimization (PPO) for LLMs Explained Intuitively [Vidéo]. YouTube. <https://www.youtube.com/watch?v=8jtAzxUwDj0>
- freeCodeCamp. (2018, 3 septembre). An introduction to Q-Learning : reinforcement learning. freeCodeCamp.org. <https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/>
- Thibo. (s. d.). aihub/rl/q_learning_visu.html at master · thibo73800/aihub. GitHub. https://github.com/thibo73800/aihub/blob/master/rl/q_learning_visu.html
- Thibault Neveu. (2018, 8 octobre). Apprentissage par renforcement # 1 : Introduction [Vidéo]. YouTube. <https://www.youtube.com/watch?v=PKNxF9CGn8>
- Kumar Kasera, Rohit; Acharjee, Tapodhir (2024), “Dataset on irrigation for Tomato”, Mendeley Data, V2, doi: 10.17632/33cngpcrmx.2
- Ray, P. P. (2023). ChatGPT : A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. Internet Of Things And Cyber-Physical Systems, 3, 121-154. <https://doi.org/10.1016/j.iotcps.2023.04.003>
- Duan, K., & Zou, Z. (2025). Safety-constrained Deep Reinforcement Learning control for human–robot collaboration in construction. Automation In Construction, 174, 106130. <https://doi.org/10.1016/j.autcon.2025.106130>