

Apprentissage par renforcement : cadre général et programmation dynamique

Clément Rambour

Plan

1 Introduction

- Motivations
- Caractéristiques de l'apprentissage par renforcement

2 Cadre théorique

- Processus Markovien
- Markov Reward Process
- Markov Decision Process

3 Vue d'ensemble du cours

4 Dynamic Programming

- Equations de Bellman
- Estimation de la politique optimale

Motivations

- Apprendre à un **agent** à atteindre un but à partir de son **expérience**
- Le but ou sa proximité est indiqué par une récompense
- L'agent apprend à définir une stratégie optimale



Exemples : contrôle



(a) Lower Walking Height



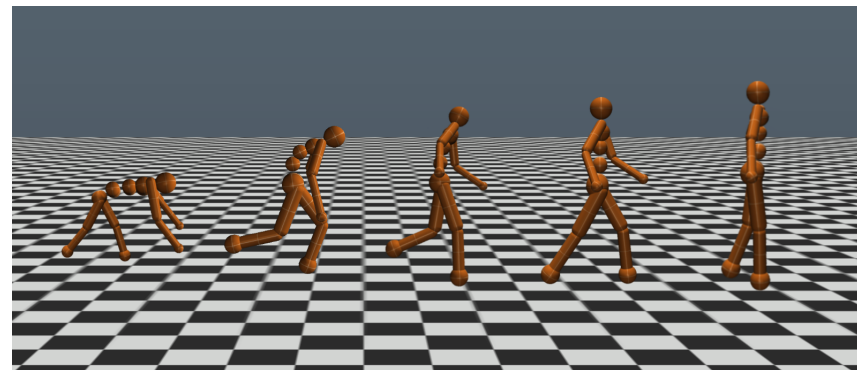
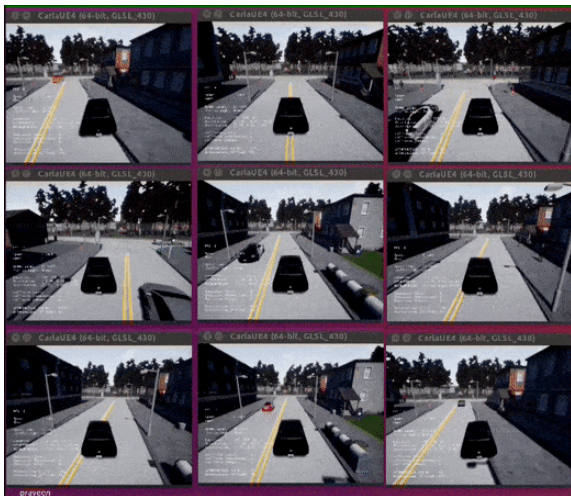
(b) Recover to Normal Height



(c) Push Recovery (Front)



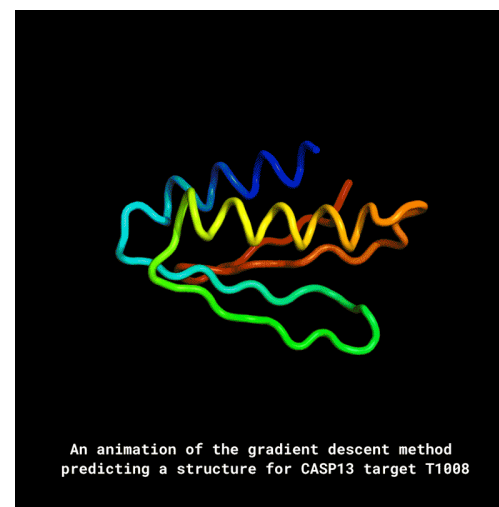
(d) Push Recovery (Back)



Exemples : jeux



Exemples : exploration



Caractéristiques de l'apprentissage par renforcement

- Pas de supervision, juste une **récompense**
- Le *feedback* est retardé
- Le processus est séquentiel : les données ne sont **pas *iid***
- Les actions de l'agent affectent l'environnement

Caractéristiques de l'apprentissage par renforcement

- Pas de supervision, juste une **récompense**
 - Le *feedback* est retardé
 - Le processus est séquentiel : les données ne sont **pas iid**
 - Les actions de l'agent affectent l'environnement
-
- La récompense (*reward*) $R_t \in \mathbb{R}$ mesure à quel point l'agent est performant à l'étape t

Exemples de rewards

- Manoeuvre de pilotage/conduite :
 - $+R$ / $-R$ si l'agent suit la bonne trajectoire/ dévie
 - $-\infty$ en cas d'accident
- Jeu de plateau :
 - $+R$ en cas de victoire, $-R$ en cas de défaite
- Gestion de portefeuille :
 - $+R/\text{€}$
- Jeu vidéo :
 - $+R$ / $-R$ par augmentation/diminution de score
 - $-\infty$ en cas de défaite

Environnement

- Complètement ou partiellement connu
- Conditionne les actions de l'agent
- Émet les récompenses pour l'agent



Agent

Définition

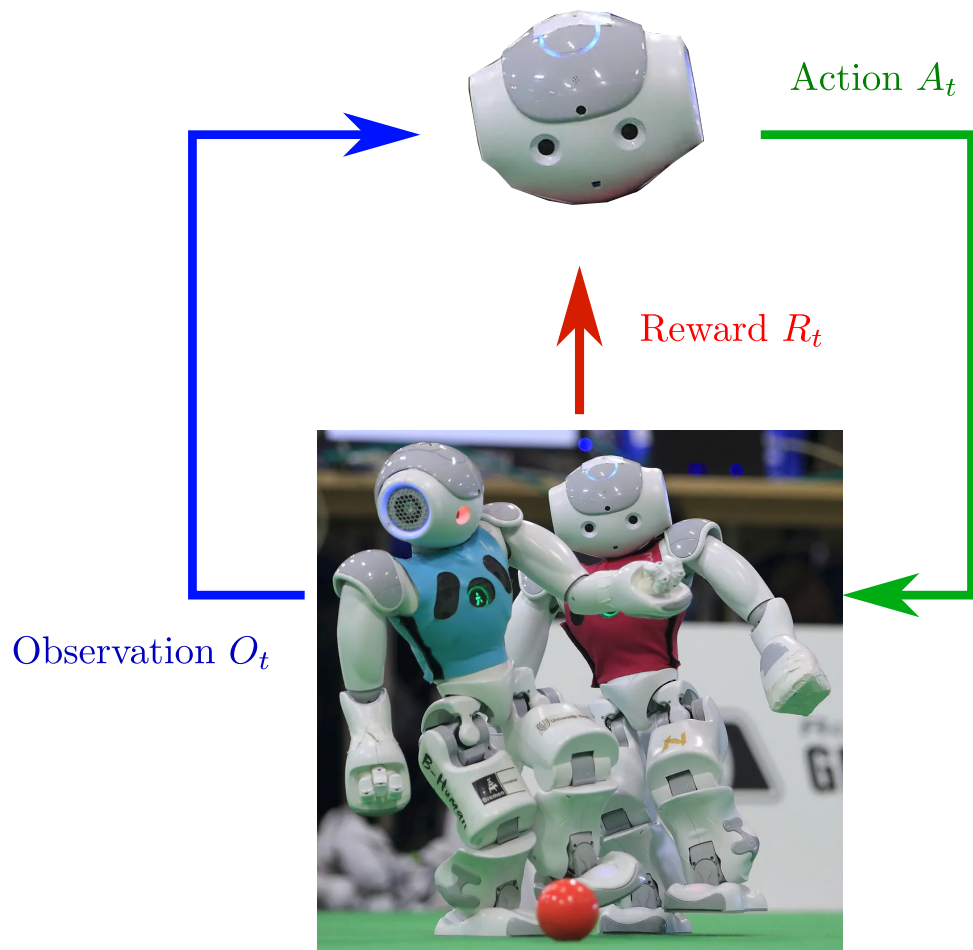
- L'agent désigne la machine, le robot ou l'algorithme qui doit apprendre la politique d'action optimale pour atteindre le but fixé.

Agent

Définition

- L'agent désigne la machine, le robot ou l'algorithme qui doit apprendre la politique d'action optimale pour atteindre le but fixé.
-
- Évolue dans l'environnement en se basant sur un ensemble d'action $\mathcal{A}$
 - Reçoit une récompense à chaque étape
 - Le but de l'agent est de maximiser l'espérance des récompenses cumulées

Schéma classique en RL



foreach **Time step**

t do

The agent :

- gets an observation O_t and a reward R_t
- Executes action A_t

The environment :

- Receives action A_t
- Emits observation O_{t+1} and reward R_{t+1}

end

État

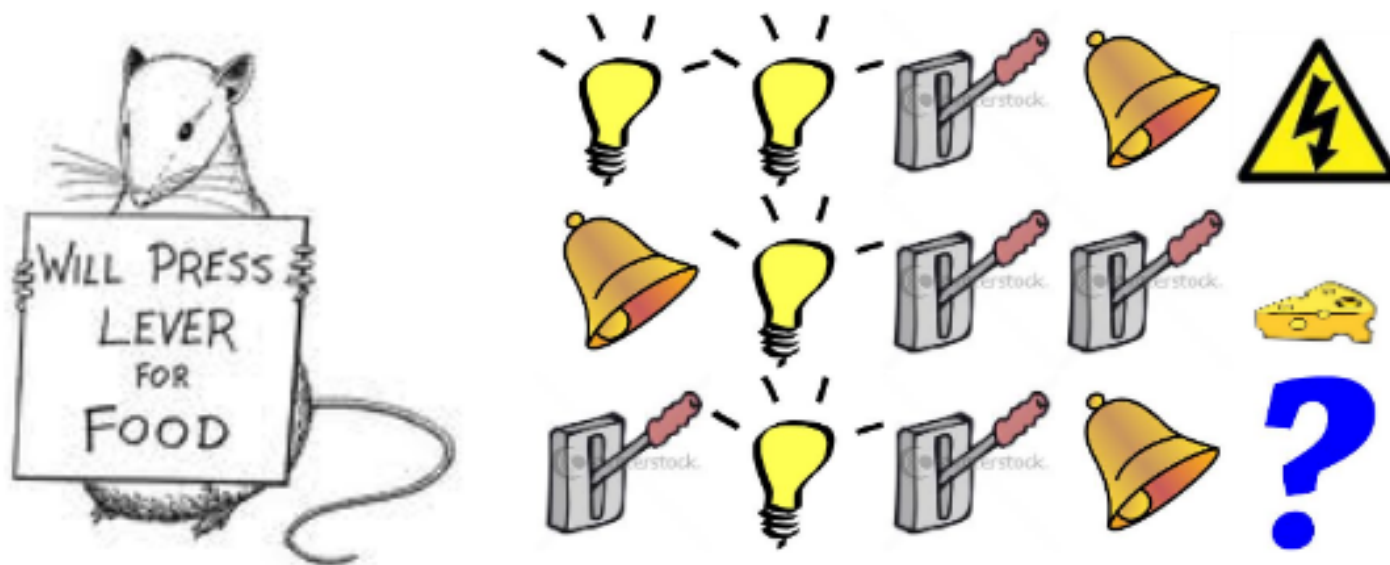
Définition

Un **état** s_t est une fonction de l'historique

$$H_t = \{O_0, R_0, A_0, \dots, A_{t-1}, O_t, R_t\} :$$

$$S_t = f(H_t)$$

Il contient toute l'information utilisée par l'agent pour décider son action prochaine.



Formalisation en RL

- En apprentissage par renforcement, la description formelle d'un environnement ainsi que les états $\mathcal{S}$ et actions $\mathcal{A}$ de l'agent se fait au travers d'un Processus de Décisions Markovien (MDP)
- Une vaste majorité des problèmes de RL peuvent être caractérisés par un MDP

Plan

1 Introduction

- Motivations
- Caractéristiques de l'apprentissage par renforcement

2 Cadre théorique

- Processus Markovien
- Markov Reward Process
- Markov Decision Process

3 Vue d'ensemble du cours

4 Dynamic Programming

- Equations de Bellman
- Estimation de la politique optimale

État Markovien

Un **état Markovien** contient toute l'information utile du passé pour prédire le future

Definition

Un état S_t est **Markovien** si et seulement si

$$p_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s] = \mathbb{P}[S_{t+1} | S_1, \dots, S_t]$$

État Markovien

Un **état Markovien** contient toute l'information utile du passé pour prédire le future

Definition

Un état S_t est **Markovien** si et seulement si

$$p_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s] = \mathbb{P}[S_{t+1} | S_1, \dots, S_t]$$

Definition : matrice de transition

La **matrice de transition** associée à un ensemble d'états $\mathcal{S}$ collecte toutes les probabilité de passer d'un état à un autre :

$$\mathcal{P} = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix}$$

Processus Markovien

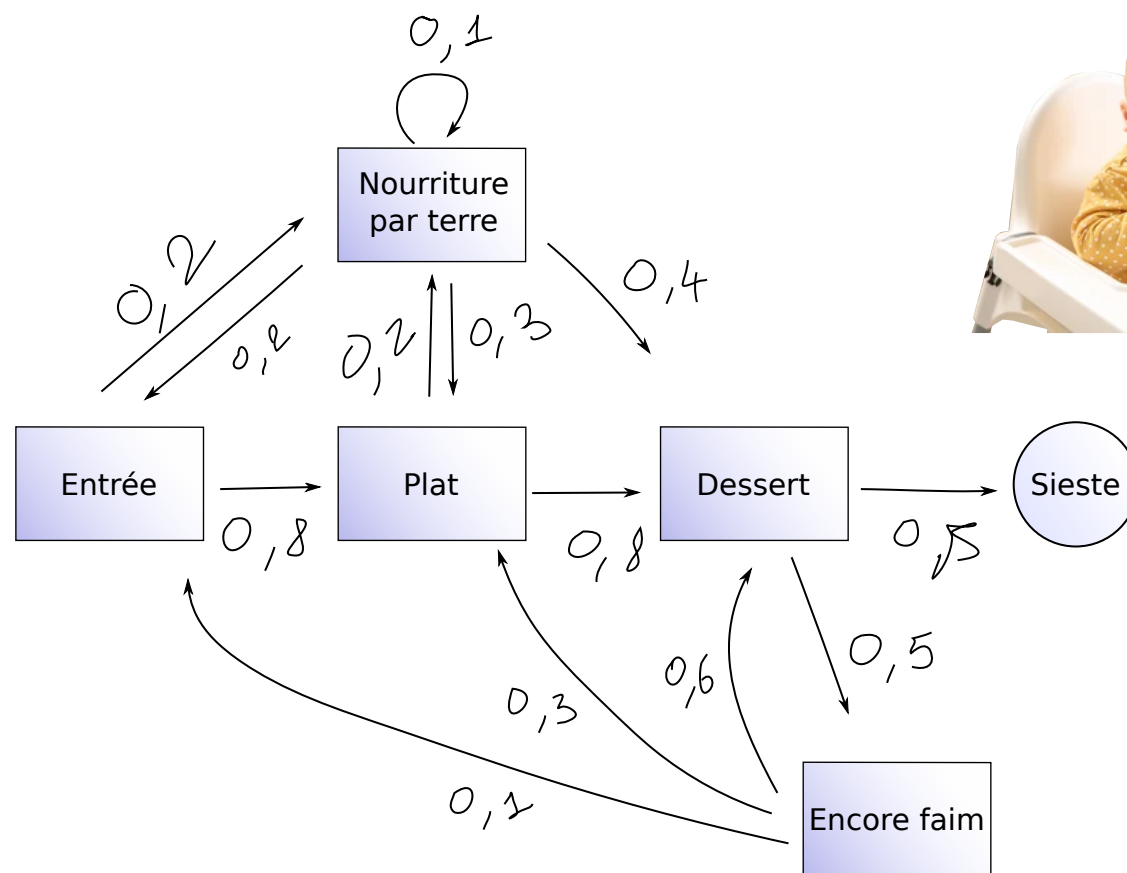
Un processus Markovien est un processus aléatoire sans mémoire *ie.* une succession d'états Markovien

Définition

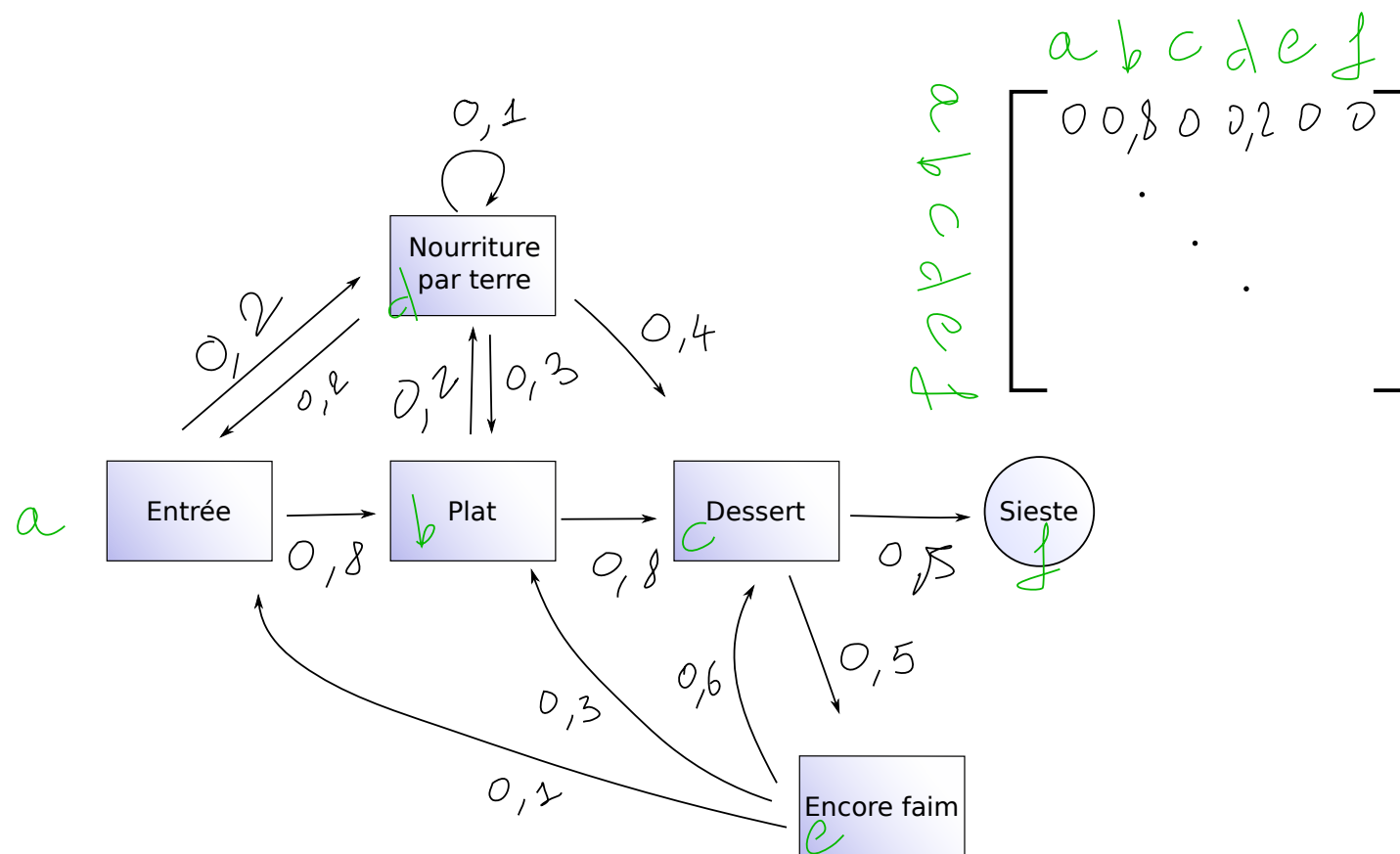
Un **processus Markovien** est défini par le couple $(\mathcal{S}, \mathcal{P})$ où :

- $\mathcal{S}$ est un ensemble fini d'états
- $\mathcal{P}$ est la matrice de transition associée à $\mathcal{S}$

Exemple



Exemple



Markov Reward Process

Définition

Un **Processus de Récompense Markovien (MRP)** est défini par :

- $\mathcal{S}$ un ensemble fini d'états
- $\mathcal{P}$ la matrice de transition associée
- $\mathcal{R}$ la récompense moyenne associée à chaque état :

$$\mathcal{R}_s = \mathbb{E} [\overset{R_{t+1}}{\cancel{R_t}} | S_t = s]$$

- γ un facteur de réduction temporel

Retour

Définition

Le retour G est l'espérance de la somme des récompenses obtenues par l'agent, pondérée par le facteur de réduction :

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots =$$

$$= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) =$$

$$= R_{t+1} + \gamma G_{t+1}$$

- Le facteur de réduction $\gamma \in [0, 1]$ représente le poids des futures récompenses
- $\gamma \sim 0 \rightarrow$ seul le épisode suivant importe
- $\gamma \sim 1 \rightarrow$ tous les épisodes futurs sont pris en compte

Fonction de valeur d'un MRP

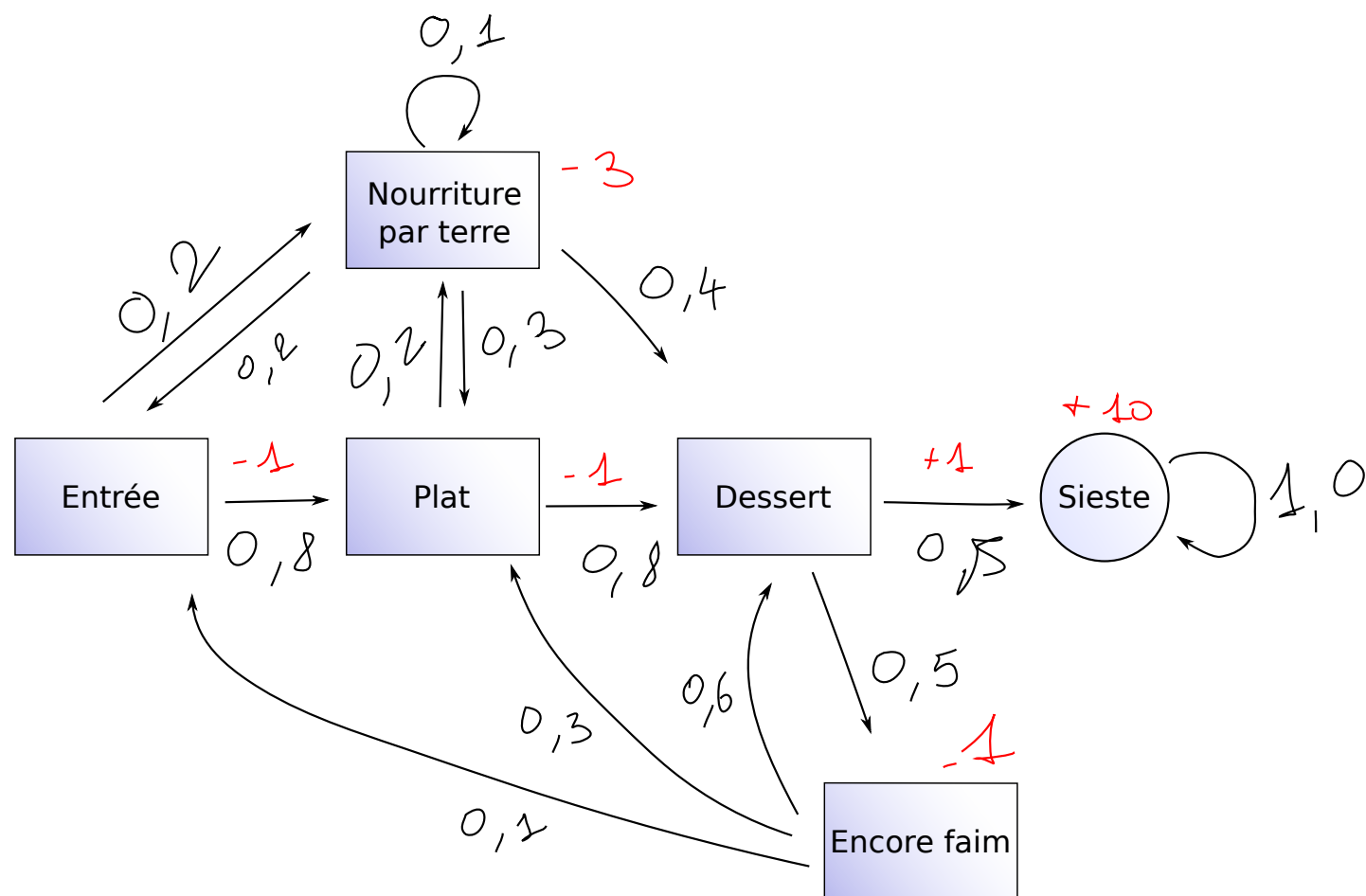
Définition

La fonction de valeur v donne le retour attendu d'un état :

$$v : \mathcal{S} \rightarrow \mathbb{R}$$

$$s \mapsto \mathbb{E}[G_t | S_t = s]$$

Exemple



Markov Decision Process

Un Processus de décision est un MRP auquel on rajoute les actions que peut prendre l'agent.

Définition

Un **Processus de Décision Markovien (MDP)** est défini par :

- $\mathcal{S}$ un ensemble fini d'états
- $\mathcal{A}$ un ensemble fini d'actions
- $\mathcal{P}$ la matrice de transition d'un couple $(s, a) \in \mathcal{S} \times \mathcal{A}$ vers $\mathcal{S}$:

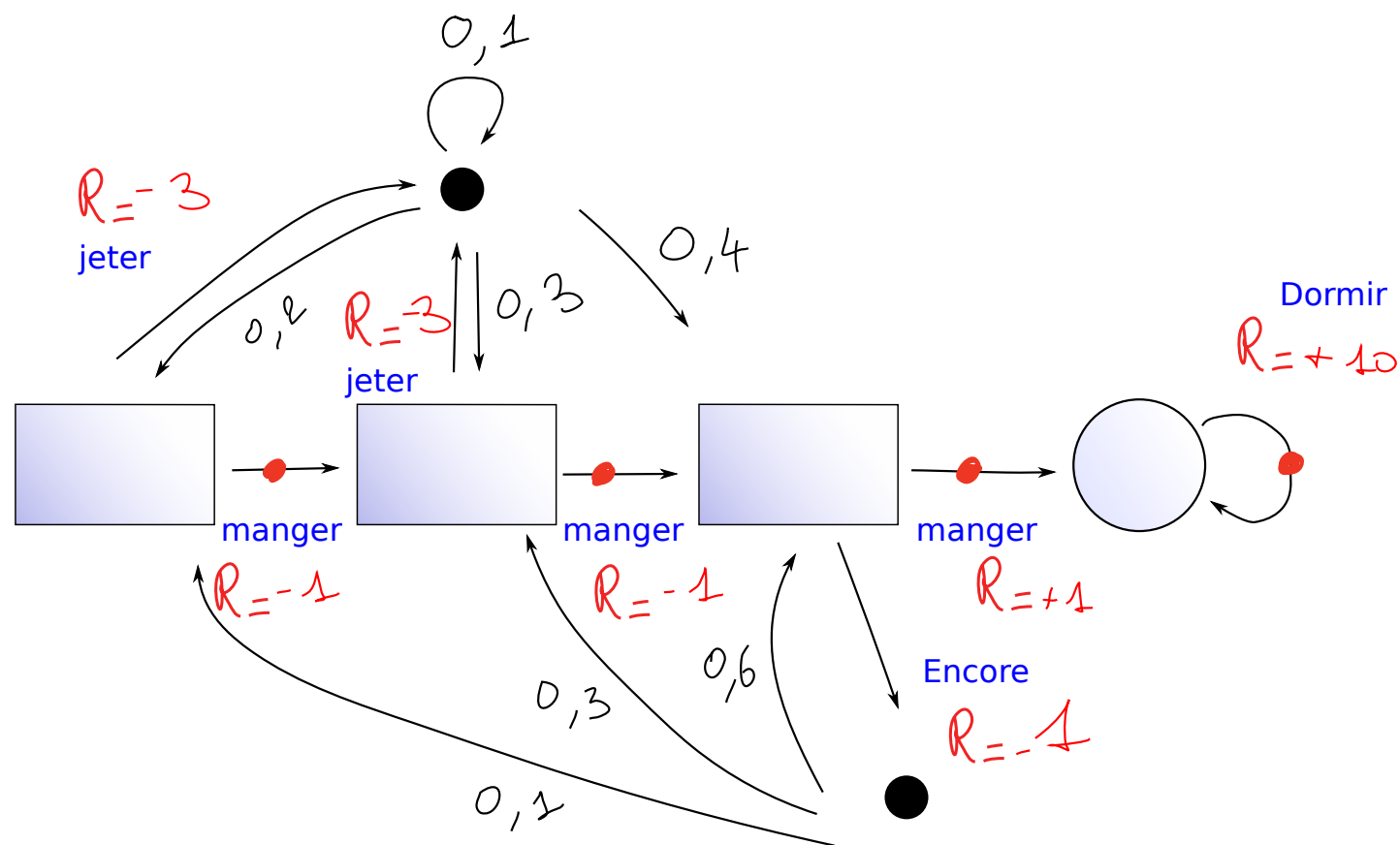
$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

- $\mathcal{R}$ la récompense moyenne associée à chaque couple état - action :

$$\mathcal{R}_s^a = \mathbb{E}[R_t | S_t = s, A_t = a]$$

- γ un facteur de réduction temporel

Exemple



Politique

Définition

Une politique π est la probabilité des actions étant donné l'état courant :

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

Politique

Définition

Une politique π est la probabilité des actions étant donné l'état courant :

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

Il est toujours possible de ramener un MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ à un MRP $(\mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma)$ en conditionnant les transitions selon la politique du MDP :

$$\mathcal{P}_{ss'}^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{ss'}^a$$

$$\mathcal{R}_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_s^a$$

Fonction de valeur d'un MDP

Fonction de valeur d'état

La fonction de valeur d'état v_π donne le retour attendu d'un état en suivant la politique π :

$$\begin{aligned} v_\pi : \mathcal{S} &\rightarrow \mathbb{R} \\ s &\mapsto \mathbb{E}[G_t | S_t = s] \end{aligned}$$

Fonction de valeur d'un MDP

Fonction de valeur d'état

La fonction de valeur d'état v_π donne le retour attendu d'un état en suivant la politique π :

$$\begin{aligned} v_\pi : \mathcal{S} &\rightarrow \mathbb{R} \\ s &\mapsto \mathbb{E}[G_t | S_t = s] \end{aligned}$$

Fonction de valeur d'état-action

La fonction de valeur d'état-action q_π donne le retour attendu d'un couple état - action en suivant la politique π :

$$\begin{aligned} q_\pi : \mathcal{S} \times \mathcal{A} &\rightarrow \mathbb{R} \\ (s, a) &\mapsto \mathbb{E}[G_t | S_t = s, A_t = a] \end{aligned}$$

Fonction de valeur optimale

Fonction de valeur d'état optimale

La fonction de valeur d'état optimale v^* est la fonction de valeur d'état maximale sur l'ensemble des politiques :

$$v^*(s) = \max_{\pi} v_{\pi}(s)$$

Fonction de valeur optimale

Fonction de valeur d'état optimale

La fonction de valeur d'état optimale v^* est la fonction de valeur d'état maximale sur l'ensemble des politiques :

$$v^*(s) = \max_{\pi} v_{\pi}(s)$$

Fonction de valeur d'état-action optimale

La fonction de valeur d'état-action optimale q^* est la fonction de valeur d'état-action maximale sur l'ensemble des politiques :

$$q^*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

Plan

1 Introduction

- Motivations
- Caractéristiques de l'apprentissage par renforcement

2 Cadre théorique

- Processus Markovien
- Markov Reward Process
- Markov Decision Process

3 Vue d'ensemble du cours

4 Dynamic Programming

- Equations de Bellman
- Estimation de la politique optimale

Grandes approches en RL

Un agent peut être optimisé en jouant sur les points suivant :

- La fonction de valeur d'état ou d'état-action
- La politique π
- Le modèle *ie.* la représentation des transition selon l'agent

Optimisation de la fonction de valeur

Le but d'un agent est de maximiser son retour attendu

- La fonction de valeur donne le retour attendu pour chaque état
- Pour une politique donnée, on peut donc prendre les meilleurs actions
- En tâtonnant, un agent peut également optimiser sa politique

Optimisation de la fonction de valeur

Le but d'un agent est de maximiser son retour attendu

- La fonction de valeur donne le retour attendu pour chaque état
 - Pour une politique donnée, on peut donc prendre les meilleurs actions
 - En tâtonnant, un agent peut également optimiser sa politique
-
- Environnement (MDP) parfaitement connu → résolu par programmation dynamique (*cf.* suite de ce cours)
 - Environnement partiellement observable ou trop vaste → résolu par échantillonnage (*cf.* cours 2 et 3)

Optimisation de la fonction de valeur

Le but d'un agent est de maximiser son retour attendu

- La fonction de valeur donne le retour attendu pour chaque état
- Pour une politique donnée, on peut donc prendre les meilleurs actions
- En tâtonnant, un agent peut également optimiser sa politique

Pros

- Convergent vers une solution optimale

Cons

- Convergence parfois difficile
- Complexité ↗ avec la taille du MDP

Optimisation de la politique

Le but d'un agent est de maximiser son retour attendu

- Bonne politique $\Rightarrow$ somme de récompenses élevée au cours du temps
- Optimisation d'une politique π_θ = trouver les paramètres θ qui maximisent la somme des récompenses
- Peut être combiné à une optimisation de la fonction de valeur

Optimisation de la politique

Le but d'un agent est de maximiser son retour attendu

- Bonne politique $\Rightarrow$ somme de récompenses élevée au cours du temps
 - Optimisation d'une politique π_θ = trouver les paramètres θ qui maximisent la somme des récompenses
 - Peut être combiné à une optimisation de la fonction de valeur
-
- Optimisation de la politique $\rightarrow$ policy gradient (*cf.* cours 4)
 - Optimisation conjoint de la politique et de la fonction de valeur $\rightarrow$ Actor Critic (*cf.* cours 4)

Optimisation de la politique

Le but d'un agent est de maximiser son retour attendu

- Bonne politique $\Rightarrow$ somme de récompenses élevée au cours du temps
- Optimisation d'une politique $\pi_\theta =$ trouver les paramètres θ qui maximisent la somme des récompenses
- Peut être combiné à une optimisation de la fonction de valeur

Pros

- Meilleures garanties de convergence
- Convient pour des environnement très vastes voire ∞

Cons

- Convergence plus lente
- Optimalité non garantie

Plan

1 Introduction

- Motivations
- Caractéristiques de l'apprentissage par renforcement

2 Cadre théorique

- Processus Markovien
- Markov Reward Process
- Markov Decision Process

3 Vue d'ensemble du cours

4 Dynamic Programming

- Equations de Bellman
- Estimation de la politique optimale

Équations de Bellman

$$\begin{aligned}
 v_{\pi}(s) &= \mathbb{E}[G_t | S_t = s] \\
 &= \mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right] \\
 &= \mathbb{E}\left[R_{t+1} + \sum_{k=2}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right] \\
 &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
 &= \mathbb{E}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s]
 \end{aligned}$$

Équations de Bellman

Les fonctions de valeurs peuvent se décomposer de façon récurrentes :

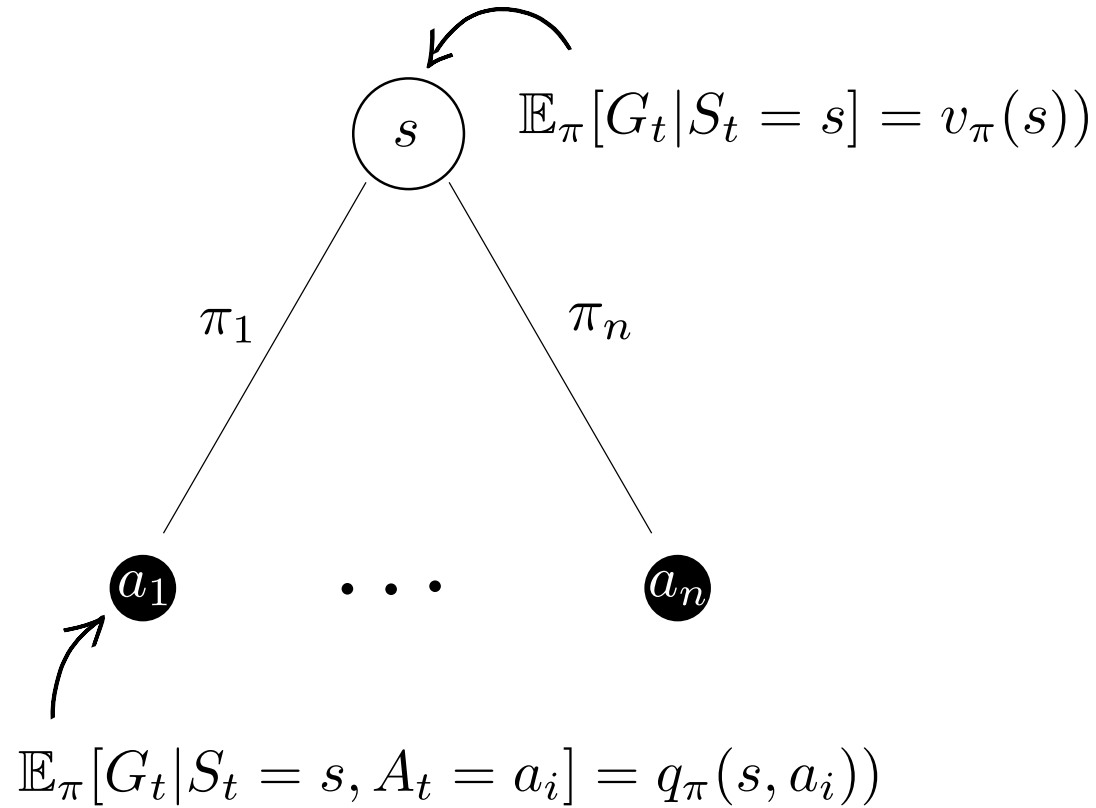
- Valeur d'état (*state value function*) :

$$v_{\pi_t}(s) = \mathbb{E}[R_{t+1} + \cancel{v_{\pi}}(S_{t+1}) | S_t = s]$$

- Valeur d'état-action (*action-state value function*) :

$$q_{\pi_t}(s, a) = \mathbb{E}[R_{t+1} + \cancel{q_{\pi}}(s, a) | S_t = s, A_t = a]$$

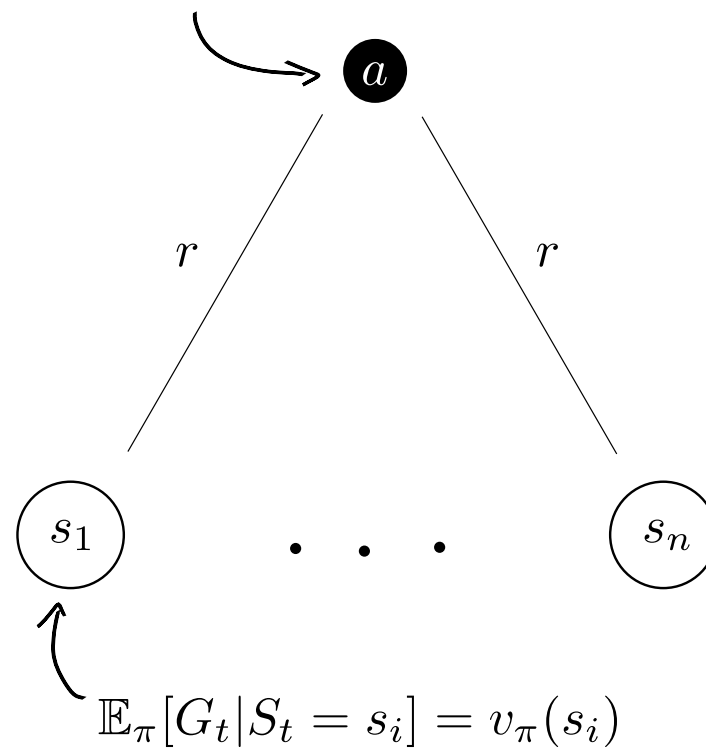
Fonction de valeur d'état



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$

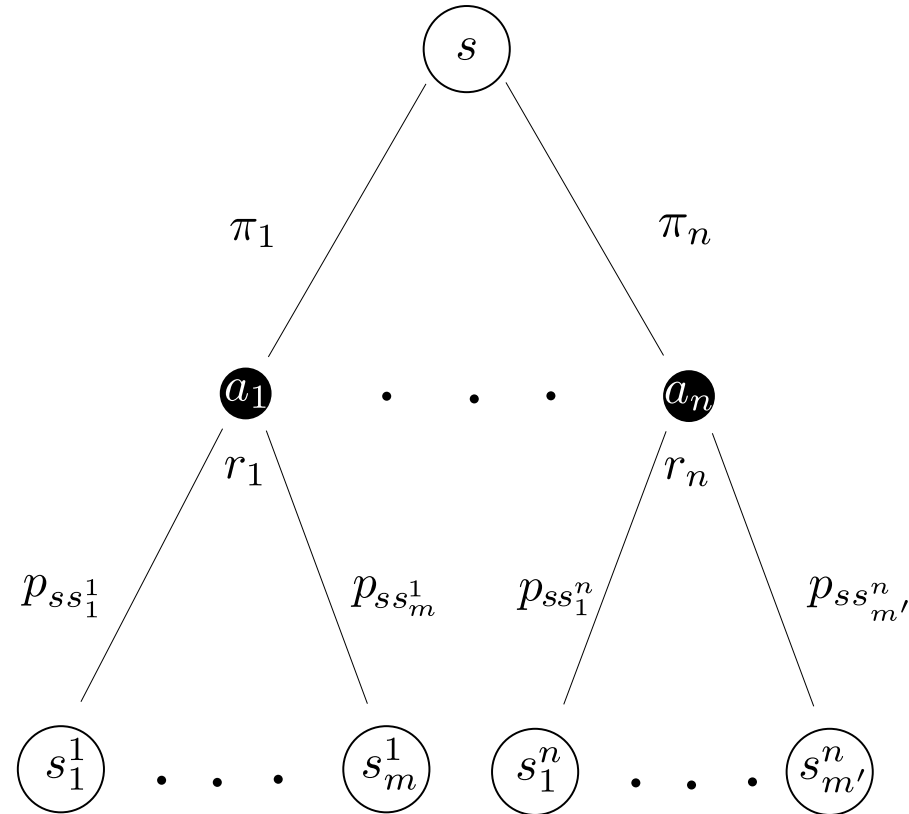
Fonction de valeur d'action-état

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



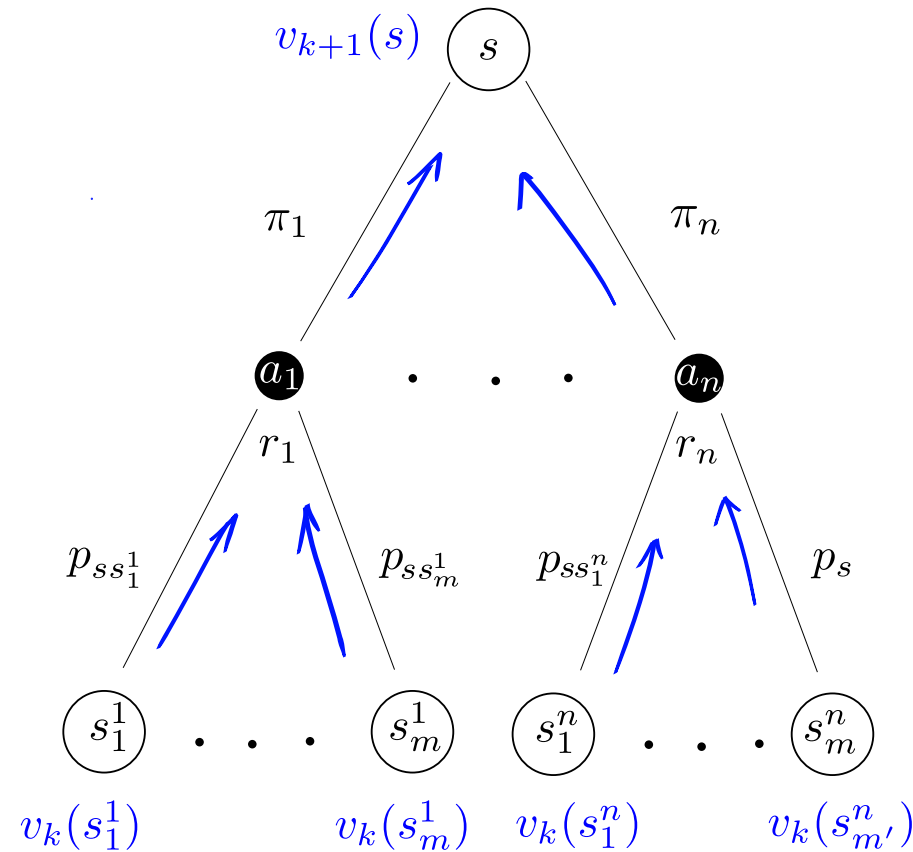
$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Fonction de valeur d'état



$$v_{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_{\pi}(s') \right)$$

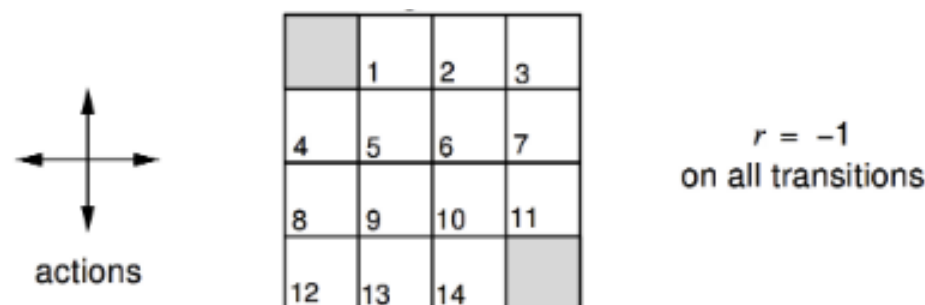
Estimation de la valeur : Iterative policy evaluation



$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right)$$

$$v_{k+1} = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v^k \rightarrow \text{Can be seen as a matrix equality}$$

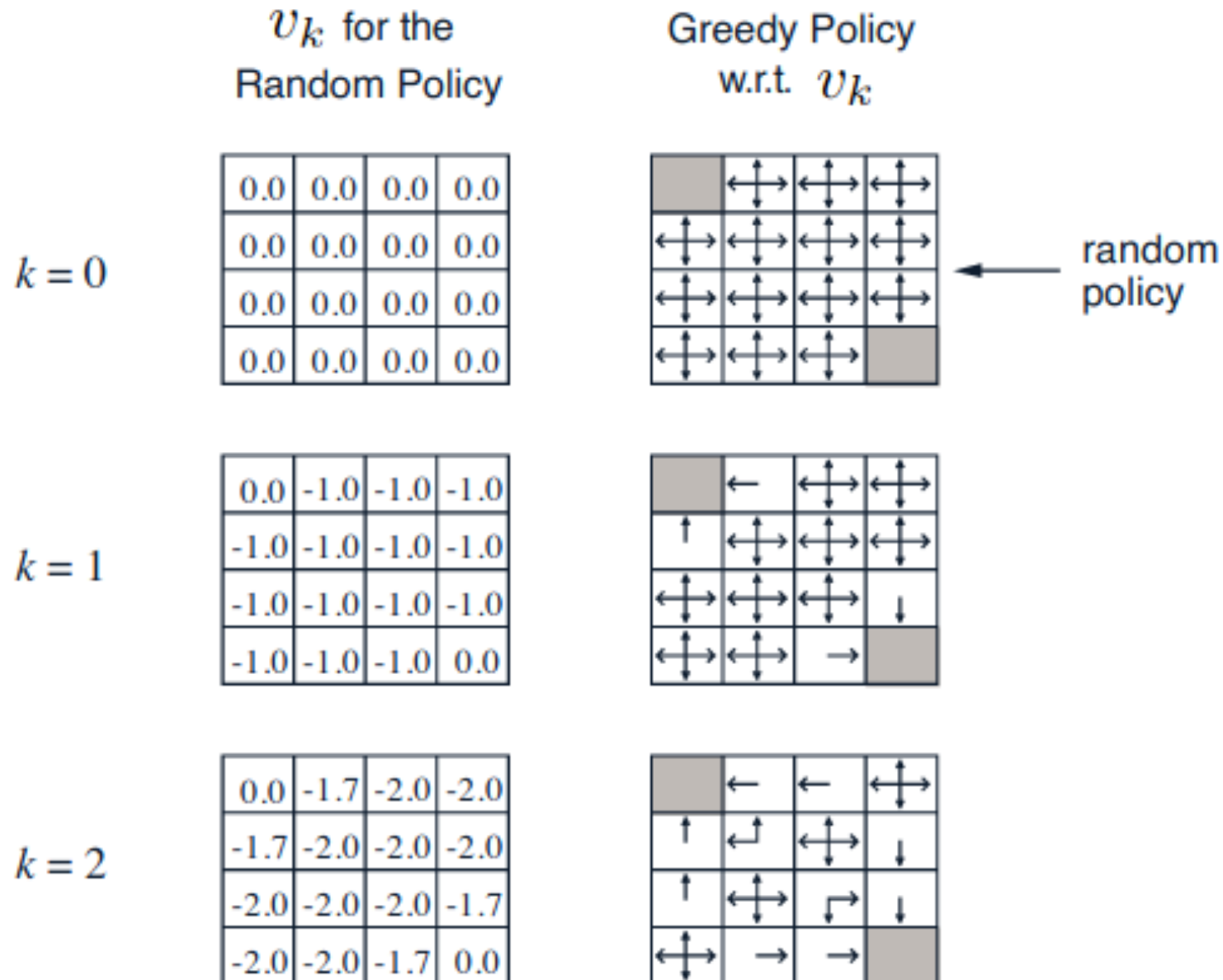
Exemple



- Undiscounted episodic MDP ($\gamma = 1$)
- Nonterminal states $1, \dots, 14$
- One terminal state (shown twice as shaded squares)
- Actions leading out of the grid leave state unchanged
- Reward is -1 until the terminal state is reached
- Agent follows uniform random policy

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

Exemple



Exemple

 $k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

	←	←	↙
↑	↖	↙	↓
↑	↖	↘	↓
↖	→	→	

 $k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

	←	←	↙
↑	↖	↙	↓
↑	↖	↘	↓
↖	→	→	

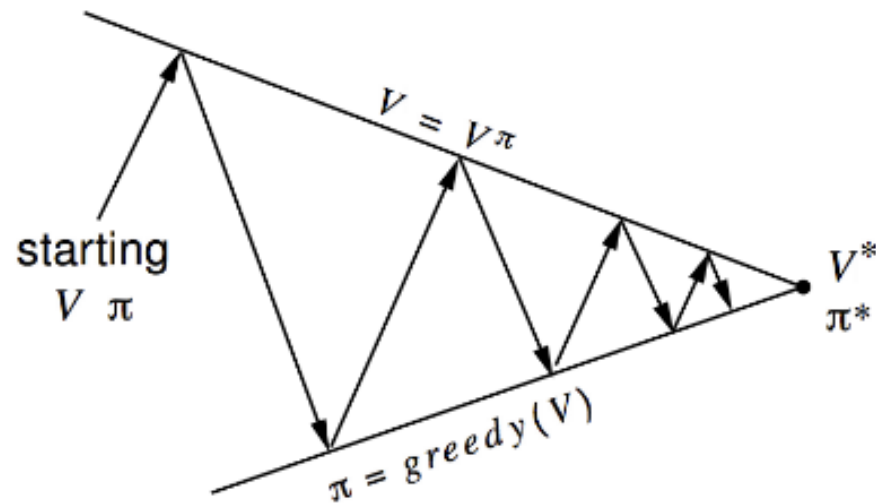
 $k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

	←	←	↙
↑	↖	↙	↓
↑	↖	↘	↓
↖	→	→	

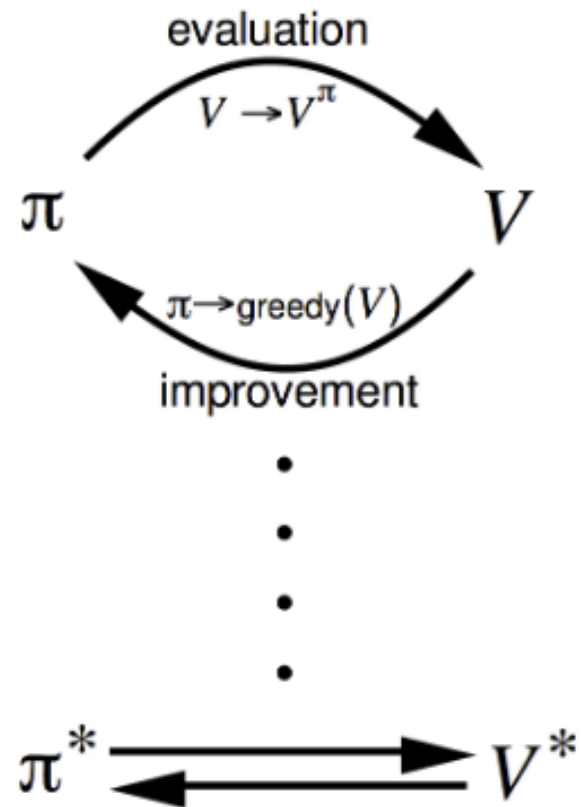
optimal
policy

Amélioration de la politique (*Policy Improvement*)



Policy evaluation Estimate v_π
 Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
 Greedy policy improvement



Amélioration de la politique (*Policy Improvement*)

Policy Improvement

Soit une politique π

- **Évaluation** de la politique π par l'estimation de la fonction de valeur associée v_π
- **Amélioration** gloutonne (*greedy*) de la politique par rapport à v_π

$$\pi' = \text{greedy}(v_\pi) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} q_\pi(s, a)$$



Dans l'exemple précédent, l'algorithme converge en une seule itération (pas le cas généralement)

Policy Improvement Theorem

Améliorer la politique augmente la fonction de valeur

- $\pi'(s) = \operatorname{argmax}_{a \in \mathcal{A}} q_\pi(s, a)$ *$\pi'(s)$ is an action*
- $q_\pi(s, \underline{\pi'(s)}) = \max_{a \in \mathcal{A}} q_\pi(s, a) \geq q_\pi(s, \underline{\pi(s)}) = v_\pi(s)$
- Et par récurrence :

$$\begin{aligned}
 v_\pi(s) &\leq q_\pi(s, \pi'(s)) = \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
 &\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
 &\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 q_\pi(S_{t+2}, \pi'(S_{t+2})) | S_t = s] \\
 &\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+2} + \cdots | S_t = s] = v_{\pi'}(s)
 \end{aligned}$$

Policy Improvement Theorem

- Convergence en cas d'égalité :

$$q_{\pi}(s, \pi'(s)) = \max_{a \in \mathcal{A}} q_{\pi}(s, a) = q_{\pi}(s, \pi(s)) = v_{\pi}(s)$$

- On a alors l'optimalité (au sens de Bellman) :

$$v_{\pi}(s) = \max_{a \in \mathcal{A}} q_{\pi}(s, a) = v_{*}(s)$$

- $\pi = \pi^{*}$ est la politique optimale

Policy iteration

- Dans l'exemple griddworld, la politique optimale π^* est obtenue avant d'avoir fini d'estimer v_π .
- Pas nécessaire d'estimer complètement la fonction de valeur pour améliorer la politique
- ϵ -convergence ou un nombre fixé d'itération permet également de converger

Itération sur la fonction de valeur (*Value Iteration*)

Si on peut subdiviser le problème, il peut être plus intéressant d'itérer sur la fonction de valeur

Principe d'optimalité

Une politique $\pi(s|a)$ est optimale pour l'état s *ie.* $v_\pi(s) = v_{\pi^*}(s)$, si et seulement si,

π est optimale pour chaque état s' atteignable depuis

s *ie.* $v_\pi(s') = v_{\pi^*}(s')$

Intuition : Si on connaît la solution à partir d'un état, on peut la *back-propager* aux états précédents

Exemple

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

 V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

 V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

 V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

 V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

 V_5

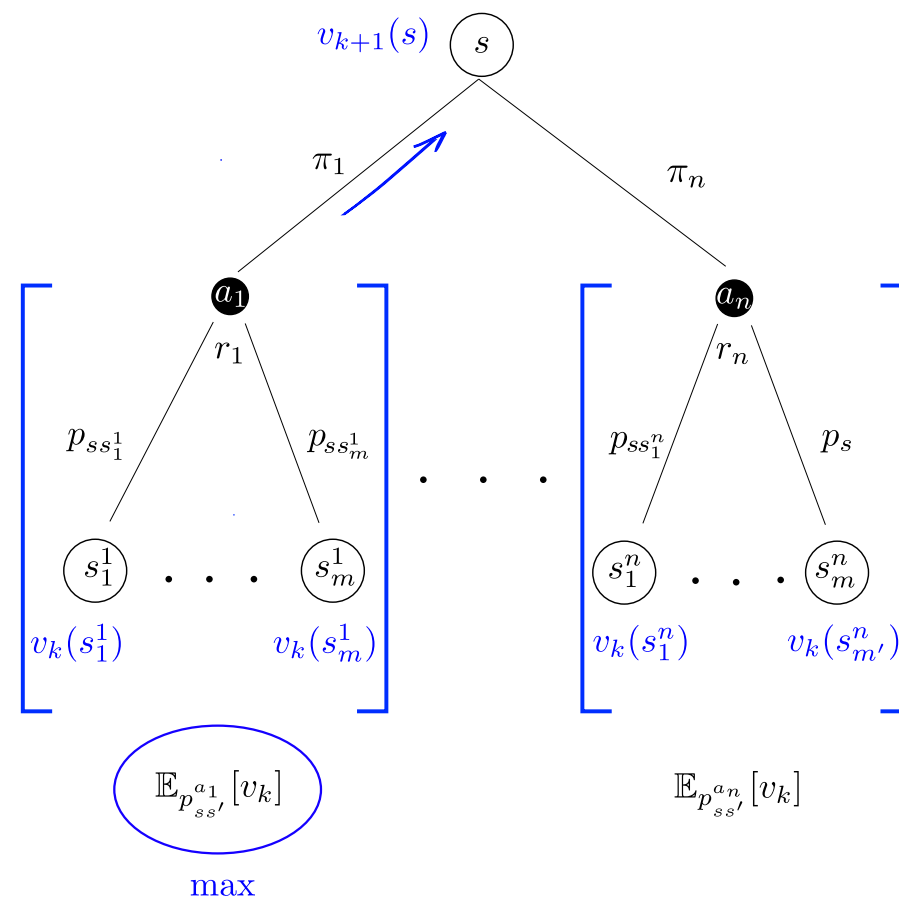
0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

 V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

 V_7

Itération sur la fonction de valeur (*Value Iteration*)



$$v_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right)$$

$$v_{k+1} = \max(\mathcal{R}^\pi + \gamma \mathcal{P}^\pi v^k)$$

Lien entre *Policy improvement* et *Value Iteration*

- *Policy Improvement* :

$$\mathbf{v}_{k+1} = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi \mathbf{v}^k$$

- *Value Iteration* :

$$\mathbf{v}_{k+1} = \max_{a \in \mathcal{A}} (\mathcal{R}^a + \gamma \mathcal{P}^a \mathbf{v}^k)$$

- Équivalent si un seul échantillon est utilisé pour estimer la valeur