L'apprentissage par renforcement

Mélanie Raymond

Université du Québec à Montréal

Séminaire d'été 2023

Table des matières

Référence

Toutes les informations contenues dans cette présentation sont issues du livre « Reinforcement Learning, An Introduction » de Richard S. Sutton et Andrew G. Barto, 2e édition (2018).

Introduction

L'idée : on apprend en interagissant avec notre environnement.

On veut apprendre les actions à poser dans une situation donnée de manière à maximiser une récompense numérique.

Quelques exemples

- Apprendre le chemin le plus court pour sortir d'un labyrinthe.
- Apprendre à jouer (gagner) à un jeu (ex. : échecs, backgammon).
- Apprendre à un robot à ramasser le plus de déchets possible avant de retourner à sa charge.

Principaux éléments du système

- L'agent : celui qui apprend.

  Ex. : Robot, machine, etc.

- L'environnement : tous les éléments en interaction avec l'agent. L'agent va choisir des actions qui vont modifier l'environnement.

Sous-éléments du système

- La récompense : nombre que l'environnement envoie à l'agent. L'objectif de l'agent : maximiser ses récompenses.
- La valeur : somme des récompenses attendues en partant d'un état.
- La politique : plan des actions à prendre à chacun des états.
- Le modèle de l'environnement (optionnel) : permet de faire de l'inférence sur la manière dont l'environnement va se comporter.

Processus décisionnel markovien fini

Dans un processus décisionnel markovien fini, l'ensemble des actions, des états et des récompenses sont de dimension finie.

- $\mathcal{S}$, l'ensemble des états sans l'état final.
- $\mathcal{S}^+$, l'ensemble des états incluant l'état final.
- $\mathcal{A}(s)$, l'ensemble des actions possibles à l'état s.
- $\mathcal{R} \subset \mathbb{R}$, l'ensemble des récompenses.

Processus décisionnel markovien fini

À chaque temps t, l'agent reçoit une représentation de l'état de l'environnement, $S_t \in \mathcal{S}$.

L'agent choisit une action, $A_t \in \mathcal{A}(s)$.

Au temps suivant, $t+1$, l'agent reçoit une récompense, $R_{t+1} \in \mathcal{R}$, et se retrouve dans un nouvel état, S_{t+1}.

Le processus peut donc être représenté de la façon suivante :

$$S_0,\ A_0,\ R_1,\ S_1,\ A_1,\ R_2,\ S_2,\ A_2,\ R_3, \ldots$$

Dynamiques de l'environnement

Par la propriété markovienne, les variables aléatoires R_t et S_t dépendent seulement de l'état et l'action précédents.

On définit la probabilité de les observer au temps t :

$$p(s', r|s, a) \doteq P(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a),$$
$$s, s' \in \mathcal{S},\ r \in \mathcal{R},\ a \in \mathcal{A}(s).$$

On définit les probabilités de transition d'un état s à un état s' :

$$p(s'|s, a) \doteq P(S_t = s'|S_{t-1} = s, A_{t-1} = a) = \sum_{r \in \mathcal{R}} p(s', r|s, a).$$

Les récompenses et le retour

Le retour est défini comme la somme des récompenses. On le note G_t :

$$G_t \doteq R_{t+1} + R_{t+2} + \ldots + R_T = \sum_{k=t+1}^{T} R_k,$$

où T est l'étape finale. Remarquons que :

$$G_t = R_{t+1} + G_{t+1}.$$

La politique et la valeur

Si l'agent suit la politique π au temps t, on définit :

$$\pi(a|s) \doteq P(A_t = a|S_t = s).$$

La valeur d'un état s sous une politique π est :

$$v_\pi(s) \doteq E_\pi(G_t|S_t = s) = E_\pi\Big(\sum_{k=t+1}^{T} R_k|S_t = s\Big), \quad \forall s \in \mathcal{S}.$$

La valeur d'une action a à l'état s sous une politique π est :

$$q_\pi(s,a) \doteq E_\pi(G_t|S_t = s, A_t = a) = E_\pi\Big(\sum_{k=t+1}^{T} R_k|S_t = s, A_t = a\Big).$$

Les équations de Bellman

On peut définir ces équations récursivement.

$$
\begin{aligned}
v_\pi(s) &= E_\pi(G_t|S_t = s) \\
&= E_\pi(R_{t+1} + G_{t+1}|S_t = s) \\
&= \sum_a E_\pi(R_{t+1} + G_{t+1}|S_t = s, A_t = a) \cdot P(A_t = a|S_t = s) \\
&= \sum_a E_\pi(R_{t+1}|S_t = s, A_t = a) \cdot \pi(a|s) \\
&\quad + \sum_a E_\pi(G_{t+1}|S_t = s, A_t = a) \cdot \pi(a|s) \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r r \cdot p(s', r|s, a) \\
&\quad + \sum_a E_\pi(G_{t+1}|S_t = s, A_t = a) \cdot \pi(a|s)
\end{aligned}
$$

Les équations de Bellman

$$
\begin{aligned}
&= \sum_a \pi(a|s) \sum_{s'} \sum_r r \cdot p(s', r|s, a) \\
&+ \sum_a \sum_{s'} E_\pi(G_{t+1}|S_{t+1} = s') \cdot p(s'|s, a) \cdot \pi(a|s) \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r r \cdot p(s', r|s, a) \\
&+ \sum_a \sum_{s'} \sum_r E_\pi(G_{t+1}|S_{t+1} = s') \cdot p(s', r|s, a) \cdot \pi(a|s) \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big(r + E_\pi(G_{t+1}|S_{t+1} = s') \Big) \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big(r + v_\pi(s') \Big).
\end{aligned}
$$

Les équations de Bellman

$$
\begin{aligned}
q_\pi(s,a) &= E_\pi(G_t \mid S_t = s, A_t = a) \\
&= E_\pi\big(R_{t+1} + G_{t+1} \mid S_t = s, A_t = a\big) \\
&= \sum_r r \sum_{s'} p(s', r|s, a) + E_\pi\big(G_{t+1} \mid S_t = s, A_t = a\big) \\
&= \sum_r r \sum_{s'} p(s', r|s, a) + \sum_{s'} E_\pi(G_{t+1} \mid S_{t+1} = s') \cdot p(s'|s, a) \\
&= \sum_r r \sum_{s'} p(s', r|s, a) + \sum_{s'} E_\pi(G_{t+1} \mid S_{t+1} = s') \sum_r p(s', r|s, a) \\
&= \sum_r r \sum_{s'} p(s', r|s, a) \\
&\quad + \sum_{s'} \sum_{a'} E_\pi(G_{t+1} \mid S_{t+1} = s', A_{t+1} = a') \cdot \pi(a'|s') \sum_r p(s', r|s, a) \\
&= \sum_r \sum_{s'} p\big(s', r|s, a\big)\Big(r + \sum_{a'} \pi(a'|s') \cdot q_\pi(s', a')\Big)
\end{aligned}
$$

Politique optimale

Une politique π est meilleure qu'une politique π' si :

$$\pi \geq \pi' \Leftrightarrow v_\pi(s) \geq v_{\pi'}(s), \quad \forall s \in \mathcal{S}.$$

Il existe toujours une politique meilleure ou égale aux autres, on l'appelle la politique optimale. On la note π_*.

Politique optimale

Quelle politique est la meilleure ?

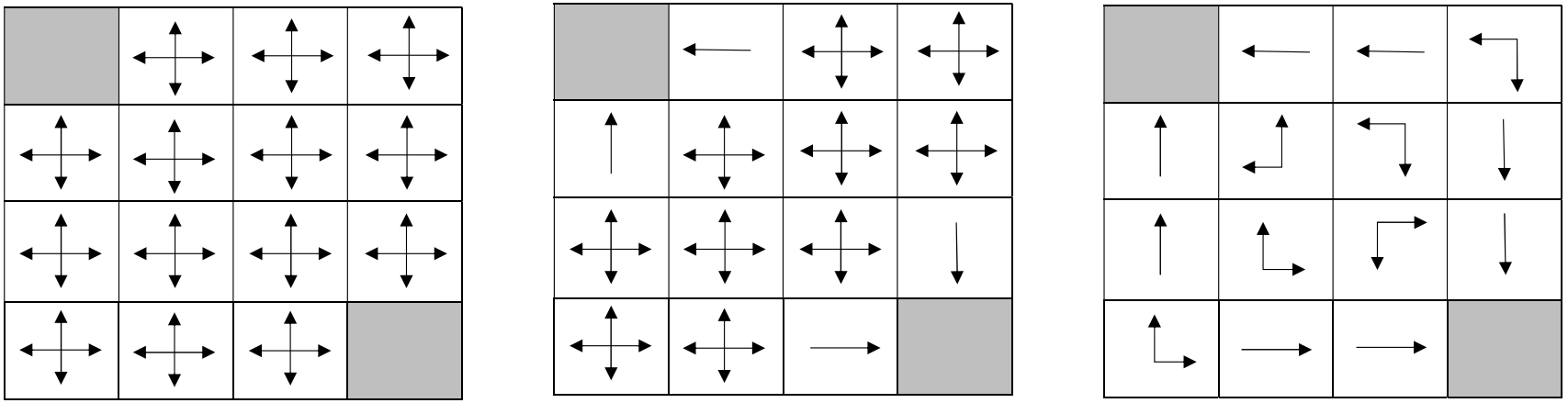


Figure – Exemple tiré de SUTTON, R. S., & BARTO, A. G. (2018). *Reinforcement learning : An introduction* (2nd ed.). The MIT Press.

Valeurs optimales

Valeur optimale des états :

$$v_*(s) \doteq \max_\pi v_\pi(s), \quad \forall s \in \mathcal{S}.$$

Valeur optimale des couples état-action :

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a), \quad \forall s \in \mathcal{S},\ \forall a \in \mathcal{A}(s).$$

$$q_*(s,a) = E_{\pi_*}(G_t | S_t = s, A_t = a)$$

$$= E(R_{t+1} + v_*(S_{t+1}) | S_t = s, A_t = a).$$

Équations d'optimalité de Bellman

Valeurs optimales définies récursivement :

$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a E_{\pi_*}(G_t | S_t = s, A_t = a) \\
&= \max_a E_{\pi_*}(R_{t+1} + G_{t+1} | S_t = s, A_t = a) \\
&= \max_a E(R_{t+1} + v_*(S_{t+1}) | S_t = s, A_t = a) \\
&= \max_a \sum_{s'} \sum_r p(s', r | s, a) \big(r + v_*(s')\big).
\end{aligned}
$$

$$
\begin{aligned}
q_*(s,a) &= E(R_{t+1} + \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a) \\
&= \sum_{s'} \sum_r p(s', r | s, a) \big(r + \max_{a'} q_*(s', a')\big).
\end{aligned}
$$

Méthodes tabulaires

L'idée pour résoudre les équations d'optimalité de Bellman :

- Évaluer la politique : déterminer la valeur des états sous une certaine politique π.
- Améliorer la politique : vérifier s'il est préférable de suivre cette politique π ou une nouvelle politique π'.

Programmation dynamique

Algorithme d'itération de la valeur

1. On initialise $V(s)$ arbitrairement $\forall s \in \mathcal{S}$, $V(\mathit{final}) = 0$, $\theta > 0$, $\Delta \leftarrow 0$.
2. Faire
   - Pour chaque $s \in \mathcal{S}$,
     1. $v \leftarrow V(s)$
     2. $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\left(r + V(s')\right)$
     3. $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   Tant que $\Delta > \theta$.
3. Retourner $\pi \approx \pi_*$ telle que

   $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\left(r + V(s')\right)$

Problème du parieur

Une personne lance une pièce de monnaie.

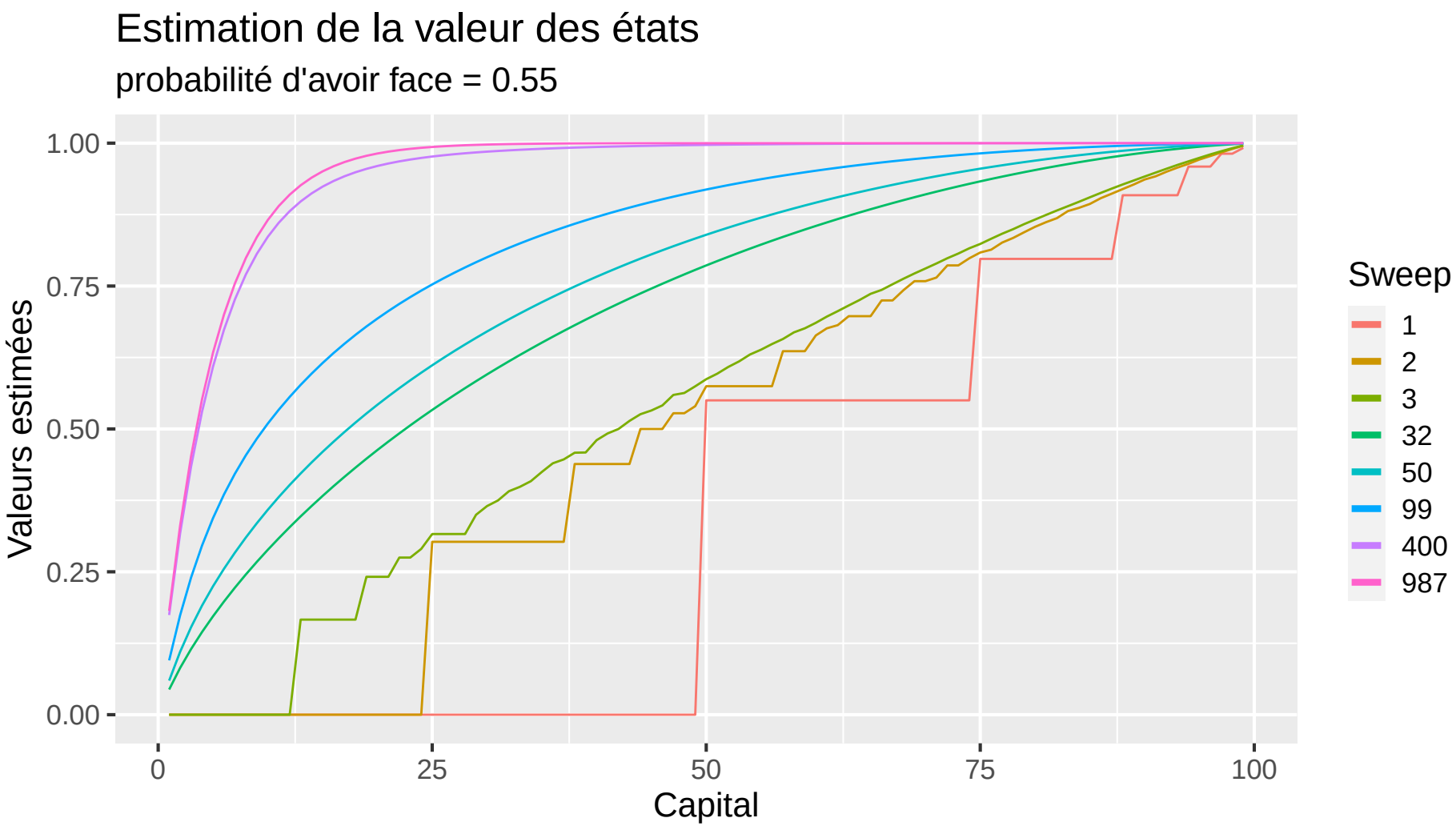
- Face : gagne sa mise.
- Pile : perd sa mise.

Fin du jeu : le parieur a 100$ ou il a tout perdu.

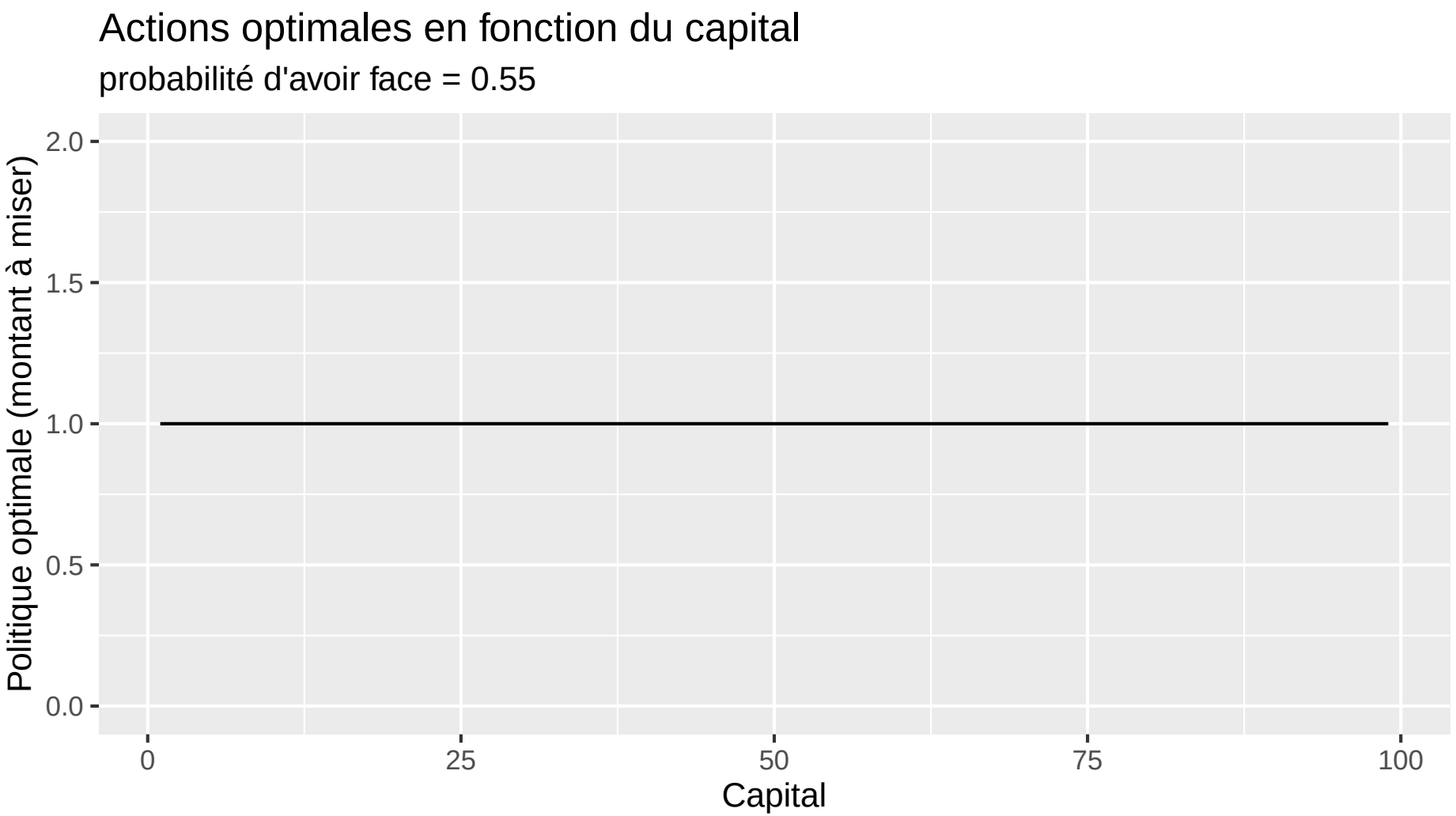
Objectif : déterminer le montant à miser.

- États : le capital détenu par l'agent.
- Actions : la mise faite par l'agent.
- Récompense : 1 si l'agent gagne, 0 sinon.

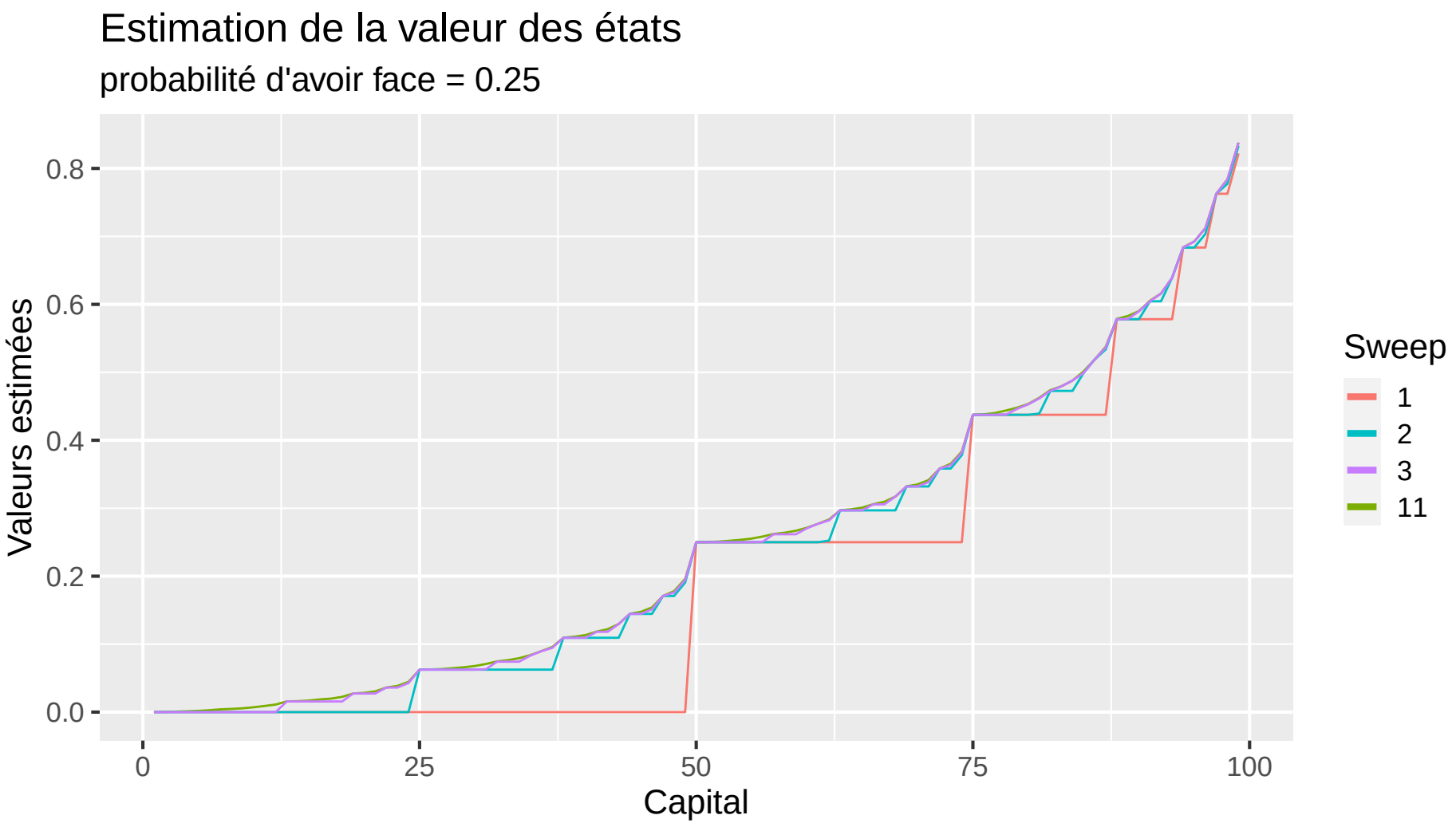
Estimation de la valeur des états - 1er cas



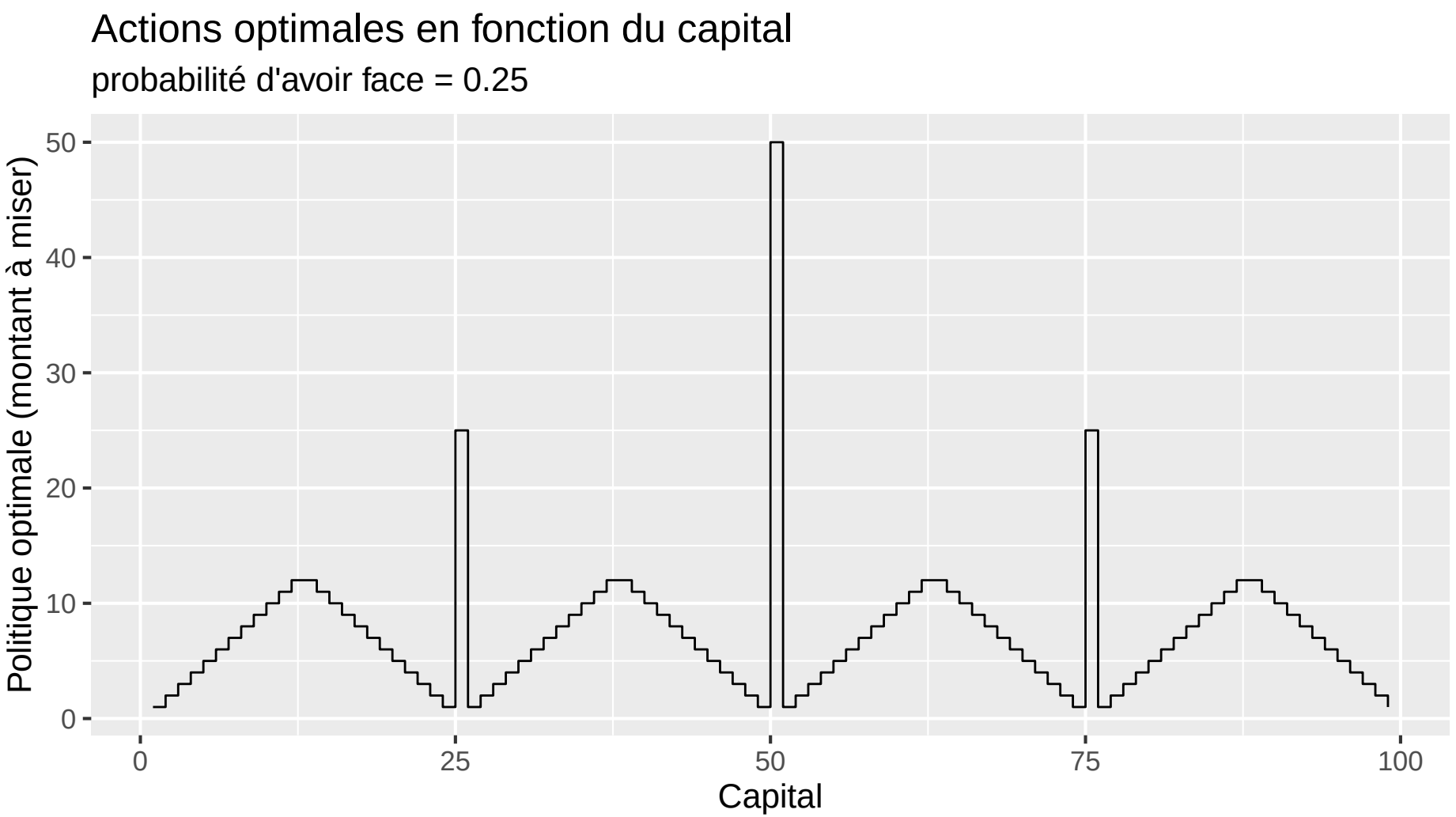
Politique optimale



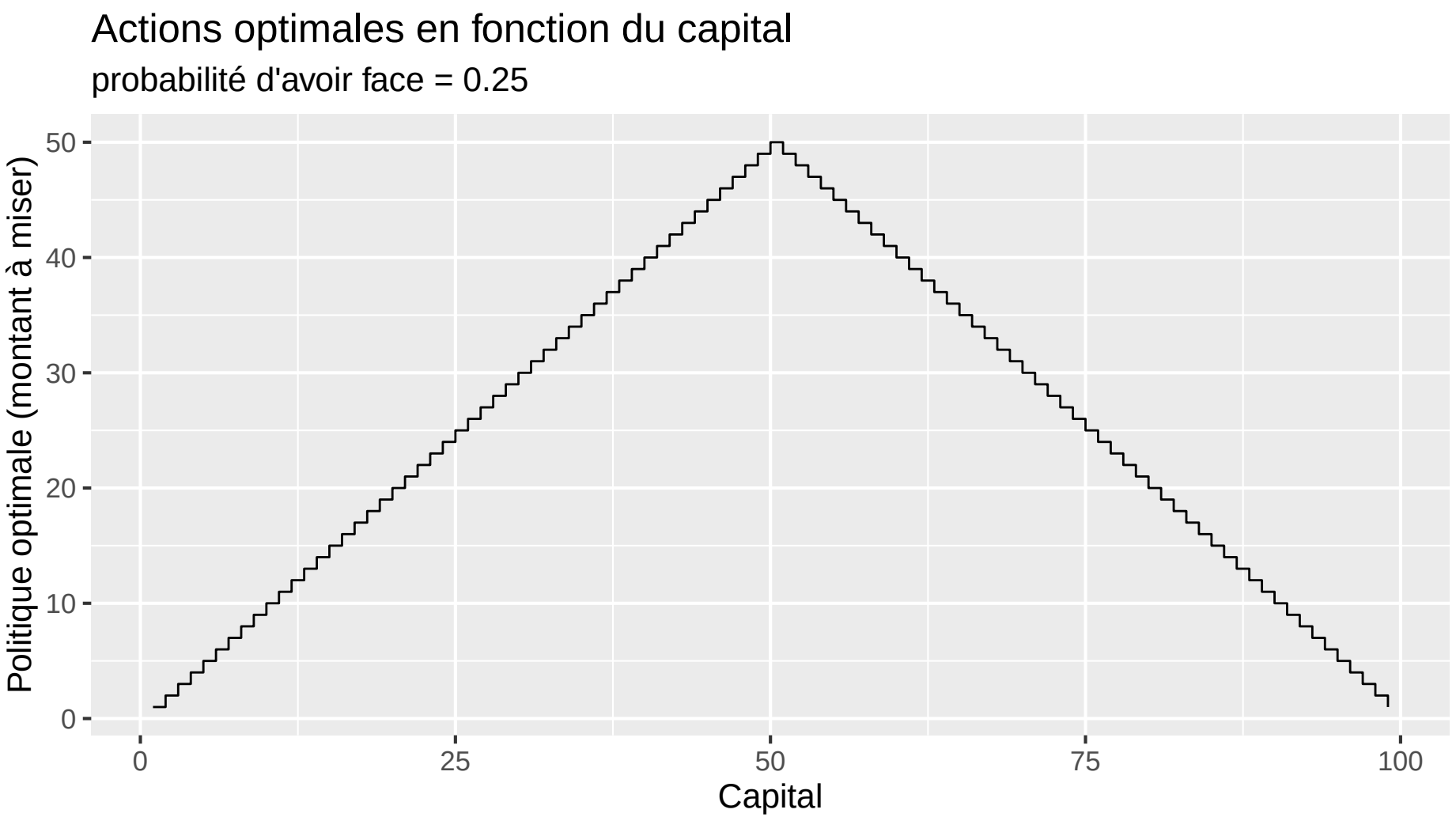
Estimation de la valeur des états - 2e cas



Politique optimale - 1re version



Politique optimale - 2e version



Quelques problèmes...

Pour résoudre les équations d'optimalité de Bellman, il faut :

1. Connaître les dynamiques de l'environnement : $p(s', r|s, a)$.

   Solutions :

   - méthodes Monte-Carlo,
   - méthodes avec la différence temporelle (TD).

Quelques problèmes...

Pour résoudre les équations d'optimalité de Bellman, il faut :

2. Avoir les ressources informatiques pour faire les calculs.

   Solutions :

   - méthodes d'estimation.

Les méthodes d'estimation

Idée : On veut généraliser les apprentissages.

On veut prendre une bonne décision dans un état jamais visité, en se basant sur ceux déjà rencontrés.

Estimer la valeur des états

On suppose qu'il existe :

$$\hat{v}(s, \boldsymbol{w}) \approx v_\pi(s),$$

une fonction dérivable en $\boldsymbol{w} \in \mathbb{R}^d$, $\forall s \in \mathcal{S}$, avec $d << |\mathcal{S}|$.

Méthodes linéaires

Par exemple, pour chaque état s, on suppose que :

$$\hat{v}(s, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}(s) = \sum_{i=1}^{d} w_i x_i(s),$$

où $\boldsymbol{x}(s) = \Big(x_1(s), x_2(s), \ldots x_d(s)\Big)^T$ est un vecteur d'attributs qui représente l'état s.

Estimer la valeur des états

On cherche à minimiser :

$$\overline{VE} = \sum_{s \in \mathcal{S}} \mu(s) \left[v_\pi(s) - \hat{v}(s, \boldsymbol{w}) \right]^2,$$

où $\mu(s) \geq 0, \sum_s \mu(s) = 1$, représente la distribution des états.

Algorithme du gradient stochastique

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{1}{2}\alpha\nabla\left[v_\pi(S_t) - \hat{v}(S_t, \boldsymbol{w}_t)\right]^2$$

$$= \boldsymbol{w}_t + \alpha\left[v_\pi(S_t) - \hat{v}(S_t, \boldsymbol{w}_t)\right]\nabla\hat{v}(S_t, \boldsymbol{w}_t),$$

où $\alpha > 0$, contrôle le taux d'apprentissage et

$$\nabla\hat{v}(S_t, \boldsymbol{w}_t) = \left(\frac{\partial\hat{v}(S_t, \boldsymbol{w}_t)}{\partial w_{t_1}}, \frac{\partial\hat{v}(S_t, \boldsymbol{w}_t)}{\partial w_{t_2}}, \ldots, \frac{\partial\hat{v}(S_t, \boldsymbol{w}_t)}{\partial w_{t_d}}\right)^T.$$

Algorithme du gradient stochastique

Dans le cas linéaire, on a :

$$\hat{v}(s, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}(s),$$

donc, on a :

$$\nabla \hat{v}(S_t, \boldsymbol{w}_t) = \boldsymbol{x}(s).$$

Algorithme du gradient stochastique

Comme on ne connaît pas $v_\pi(S_t)$, on utilise :

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha\big[U_t - \hat{v}(S_t, \boldsymbol{w}_t)\big]\nabla\hat{v}(S_t, \boldsymbol{w}_t),$$

où U_t est un estimateur non biaisé de $v_\pi(s)$.

Avec une méthode Monte-Carlo, on utilise $U_t = G_t$, car $E(G_t|S_t = s) = v_\pi(s)$.

Estimer la valeur des états sous π, $\hat{v} \approx v_\pi$

Algorithme du gradient Monte Carlo

1. Entrée : une politique π à évaluer
2. Entrée : une fonction dérivable $\hat{v} : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}$
3. Paramètre : $\alpha > 0$
4. Initialisation : $\boldsymbol{w} \in \mathbb{R}^d$, arbitrairement (p. ex. $\boldsymbol{w} = 0$)
5. Boucle infinie (pour chaque épisode) :
   1. Générer un épisode $S_0, A_0, R_1, S_1, A_1, ..., R_T, S_T$ en suivant π
   2. Pour $t = 0, 1, 2, ..., T-1$, faire :
      - $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \big[G_t - \hat{v}(S_t, \boldsymbol{w})\big] \nabla \hat{v}(S_t, \boldsymbol{w})$

Exploitation vs exploration

On **exploite** les apprentissages déjà faits

versus

on **explore** les états jamais visités.

Estimer la valeur optimale des états

On suit une politique $\epsilon - greedy$.

- Exploitation : on choisit les actions maximisant les récompenses $(1-\epsilon)\%$ du temps.
- Exploration : on choisit une autre action $\epsilon\%$ du temps.

On cherche alors $\hat{q}(s, \boldsymbol{w}, a) \approx q_*(s, a)$. On utilise :

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \Big[U_t - \hat{q}(S_t, A_t, \boldsymbol{w}_t) \Big] \nabla \hat{q}(S_t, A_t, \boldsymbol{w}_t).$$

Estimer $\hat{q}(s, \boldsymbol{w}, a) \approx q_*(s, a)$

Algorithme du semi-gradient avec Sarsa

1. Entrée : une fonction dérivable $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$.
2. Paramètre : $\alpha > 0$, un petit $\epsilon > 0$.
3. Initialisation : poids $\boldsymbol{w} \in \mathbb{R}^d$ arbitrairement (p. ex. $\boldsymbol{w} = 0$).

Estimer $\hat{q}(s, \boldsymbol{w}, a) \approx q_*(s, a)$

Algorithme du semi-gradient avec Sarsa - suite

Boucle infinie (pour chaque épisode) :

1. Initialiser S, A $(\epsilon - greedy)$.
2. Pour chaque temps t de l'épisode :
   1. Faire l'action A, observer R, S'
   2. Si S' est l'état final :
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\left[R - \hat{q}(S, A, \boldsymbol{w})\right]\nabla\hat{q}(S, A, \boldsymbol{w})$.
      Aller à l'épisode suivant.
   3. Sinon :
      Choisir A' en utilisant $\hat{q}(S, A, \boldsymbol{w})$ $(\epsilon - greedy)$.
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\left[R + \hat{q}(S', A', \boldsymbol{w}) - \hat{q}(S, A, \boldsymbol{w})\right]\nabla\hat{q}(S, A, \boldsymbol{w})$.
      $S \leftarrow S'$.
      $A \leftarrow A'$.

Conclusion

- On cherche les actions pour maximiser les récompenses.
- L'agent apprend en interagissant avec son environnement.
- Pour résoudre les problèmes d'apprentissage par renforcement :
  - méthodes tabulaires (programmation dynamique, méthodes Monte Carlo, différence temporelle),
  - méthodes d'estimation (apprentissage supervisé).

Conclusion

Merci !

Des questions ?