UE Apprentissage par renforcement - séance 5

Valentin Emiya

M2 IAAA

8 janvier 2020

Programme de l'UE

En 7 séances de 4h :

1. Bandits (1/2) : notions et strategies de base, UCB

   Exploration/exploitation
2. Bandits (2/2) : Thomson sampling, bandits contextuels

   contexte
3. Monte-Carlo Tree Search

   Environnement connu, simulations
4. Processus de décision de Markov

   Cas général d'environnements et de stratégies
5. **TD learning**
6. Miniprojet
7. Miniprojet

Programme de la séance : Temporal-Difference learning

Comment apprendre une bonne stratégie dans un MDP inconnu ?

- Deux algorithmes célèbres
  - SARSA : on-policy TD learning
  - Q-learning : off-policy TD learning
- Grâce à la théorie des MDP (définitions, fx d'éval., éq. de Bellman) :
  - Principe de Genéral Policy Iteration
  - Évaluation d'une stratégie
  - Amélioration d'une stratégie
- Et ça marche ? Quel est le meilleur ?

Références

- Chapitre 6 de *Reinforcement Learning : An Introduction*, R. S. Sutton et A. G. Barto, 2nd Edition, MIT Press, Cambridge, 2018.
- `https://github.com/mazzzystar/QLearningMouse`

Sommaire

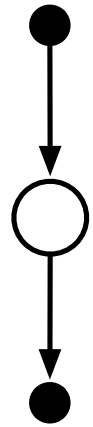
Sommaire

Algorithme SARSA

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal



Sarsa

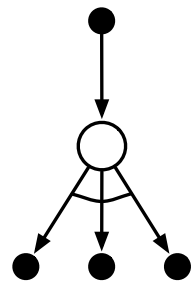
Sommaire

Algorithme Q-learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal



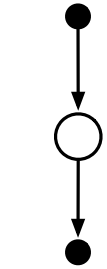
Q-learning

Sommaire

Principe de Temporal-Difference learning

SARSA :

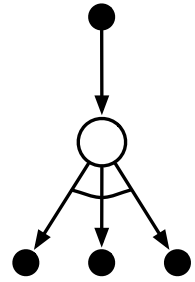
$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$$



Sarsa

Q-learning :

$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$$



Q-learning

On met à jour $Q(S_t, A_t)$

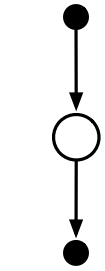
- après une étape $S_t, A_t, R_{t+1}, S_{t+1}$ (A_{t+1})
- sans attendre la fin de l'épisode
- sans rien connaitre de l'environnement ($p(s', r|s, a)$ inconnu)

$\rightarrow$ *Une avancée majeure apportée par l'**apprentissage** par renforcement !*

Principe de Temporal-Difference learning

SARSA :

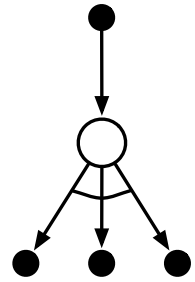
$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$$



Sarsa

Q-learning :

$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$$



Q-learning

On met à jour $Q(S_t, A_t)$

- après une étape $S_t, A_t, R_{t+1}, S_{t+1}$ (A_{t+1})
- sans attendre la fin de l'épisode
- sans rien connaitre de l'environnement ($p(s', r|s, a)$ inconnu)

$\rightarrow$ *Une avancée majeure apportée par l'**apprentissage** par renforcement !*

Pourquoi est-ce efficace ?

Sommaire

Processus de décision de Markov fini

Environnement

- un ensemble fini d'états de l'environnement $\mathcal{S} = \{s_i, 0 \leq i < S\}$
- un ensemble fini d'actions de l'agent $\mathcal{A} = \{a_i, 0 \leq i < A\}$
- un ensemble fini de récompenses immédiates $\mathcal{R} = \{r_i \in \mathbb{R}, 0 \leq i < R\}$
- la loi

$$p(s', r|s, a) = p(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a)$$

caractérisant la probabilité qu'ayant effectué l'action a dans l'état s de l'environnement à l'instant $t-1$, l'agent reçoive la récompense immédiate r et l'environnement se retrouve dans l'état s' à l'instant t.

Processus de décision de Markov fini

Environnement (suite)

On peut en déduire de $p(s', r|s, a)$ les quantités suivantes

- les probabilités de transition entre états : la probabilité de passer d'un état $s \in \mathcal{S}$ à un état $s' \in \mathcal{S}$ via une action $a \in \mathcal{A}$ est

$$p(s'|s, a) = \sum_{r \in \mathcal{R}} p(s', r|s, a)$$

- l'espérance de la récompense immédiate lorsque l'on choisit une action $a \in \mathcal{A}$ dans un état $s \in \mathcal{S}$ est

$$r(s, a) = \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a)$$

$\rightarrow$ $p(s', r|s, a)$ caractérise entièrement l'environnement.

Interaction agent/environnement

Étant donné un agent dont la stratégie/politique est caractérisée par une loi $\pi(a|s) = \mathbb{P}(A_t = a|S_t = s)$ [1]

- choisir un état initial : S_0
- répéter pour $t = 0, 1, \ldots$
  - choisir une action :

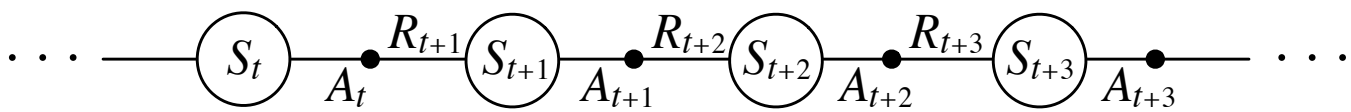
$$\text{tirer } A_t = a \text{ selon } \pi(a|S_t)$$

  - obtenir la récompense et le nouvel état :

$$\text{tirer } S_{t+1} = s', R_{t+1} = r \text{ selon } p(s', r|S_t, A_t)$$

  - mettre à jour l'état courant

$$s \leftarrow s'$$



1. ou une fonction déterministe $a = \pi(s)$

Récompense et retour

L'agent reçoit des **récompenses (immédiates)** R_t, R_{t+1}, R_{t+2}. Son objectif est de maximiser un **retour**, ou **récompense à long terme**, noté G_t que l'on peut définir de plusieurs façons :

- sur un horizon fini $T < +\infty$, par exemple pour le cas d'épisodes :

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

- avec une dévaluation $\gamma \in [0, 1[$, par exemple dans le cas perpétuel :

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=t+1}^{+\infty} \gamma^{k-t-1} R_k$$

Propriété : on a $G_t = R_{t+1} + \gamma G_{t+1}$ pour $t < T$ (avec $G_T = 0$).

Fonctions d'évaluation d'une politique π

Objectif : évaluer une politique π par le retour moyen depuis un état s.

Fonction d'évaluation de π depuis un état

Pour tout état s, on a

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{+\infty} \gamma^k R_{t+k+1}|S_t = s\right]$$
$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right]$$

Fonction d'évaluation de π depuis une paire (état, action)

Pour tout état s et action a, on a

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$
$$= \mathbb{E}_\pi\left[\sum_{k=0}^{+\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a\right]$$
$$= \sum_{s',r} p(s', r|s, a)\left[r + \gamma \sum_{a'} q_\pi(s', a')\pi(a'|s')\right]$$

Fonctions d'évaluations optimales

Fonction d'évaluation optimale depuis un état

Pour tout état s, on a

$$\begin{aligned} v_*(s) &\triangleq \max_\pi v_\pi(s) \\ &= \max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s')\right] \end{aligned}$$

Fonction d'évaluation optimale depuis une paire (état, action)

Pour tout état s et action a, on a

$$\begin{aligned} q_*(s, a) &\triangleq \max_\pi q_\pi(s, a) \\ &= \sum_{s',r} p(s', r|s, a) \left[r + \gamma \max_{a'} q_*(s', a')\right] \end{aligned}$$

Propriétés : $\forall s, v_*(s) = \max_a q_*(s, a)$ et $\pi_*(s) \in \operatorname{argmax}_a q_*(s, a)$.

Sommaire

Sommaire

General Policy Iteration (GPI)

Comment apprendre une stratégie optimale ?

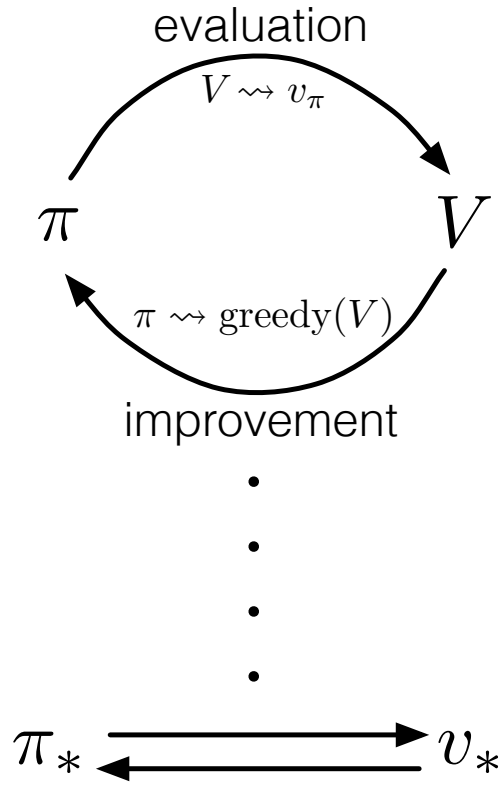
En partant d'une politique initiale π, répéter :

- évaluer la stratégie courante π

  en calculant v_π *ou* q_π

- améliorer π

  en trouvant π' *tel que* $v_{\pi'} \geq v_\pi$ *ou* $q_{\pi'} \geq q_\pi$



$\rightarrow$ Ce principe GPI est utilisé dans de nombreux algorithmes de renforcement, dont SARSA.

Sommaire

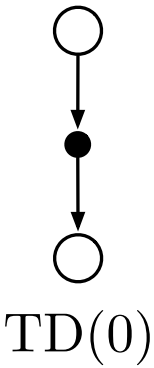
Évaluer une stratégie (policy evaluation)

Objectif : pour π fixée, estimer v_π (ou q_π).

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal



Estimation itérative à partir de tirages de A, R, S' avec 2 ingrédients :

- Bellman : $v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_t + 1) \,|\, S_t = s\right]$
- *Temporal-Difference error* : $R_{t+1} + \gamma V(S_t + 1) - V(S_t)$

Évaluation de la stratégie dans SARSA et Q-learning

$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$$

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal,\cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$$

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal,\cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

- Évaluation de q_π sur le même principe qu'avec v_π
- SARSA : équation de Bellman pour q_π

  $\pi \leftrightarrow q_\pi$: *On-Policy TD learning*
- Q-learning : équation d'optimalité de Bellman pour q_{π^*}

  $\pi \leftrightarrow q_{\pi^*}$: *Off-Policy TD learning*

Sommaire

Comment améliorer une stratégie π ?

Problème

Étant donné π et q_π évalué précédemment, trouver π' telle que $v_{\pi'} \geq v_\pi$.

Policy improvement

On définit $\pi'(s) \triangleq \text{argmax}_a \, q_\pi(s, a)$, i.e., π' est greedy par rapport à q_π.

Alors $q_\pi(s, a) \geq v_\pi(s)$ et $v_{\pi'}(s) = \sum_a \pi(a|s) q_\pi(s, a) \geq v_\pi(s)$.

Amélioration de la stratégie dans SARSA et Q-learning

Choose A from S using policy derived from Q (e.g., ε-greedy)

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

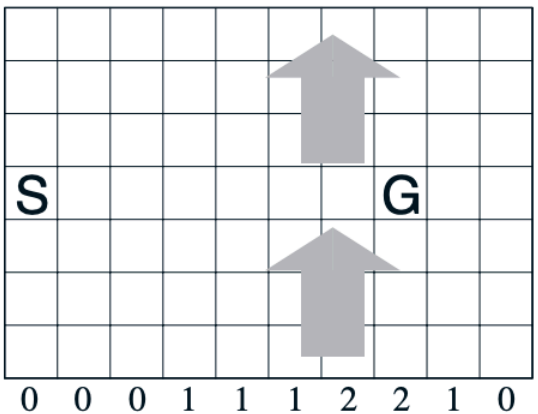
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Sommaire

SARSA en action

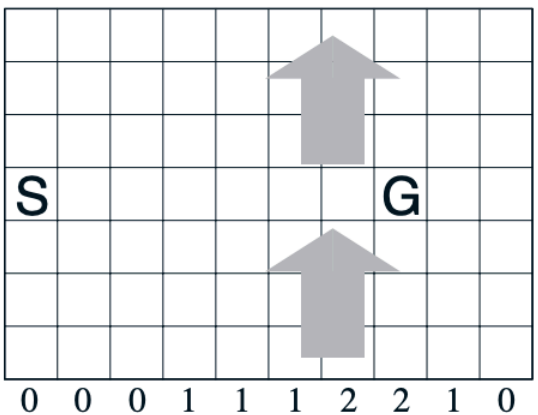
[Sutton, exemple 6.5]

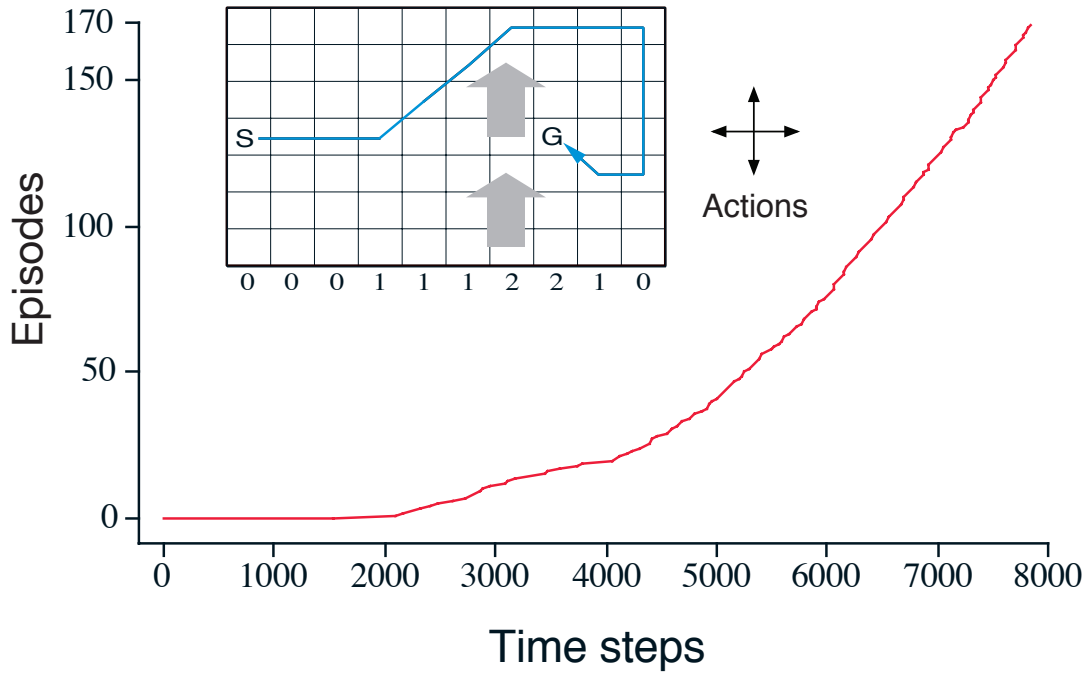


À programmer en TP

SARSA en action

[Sutton, exemple 6.5]

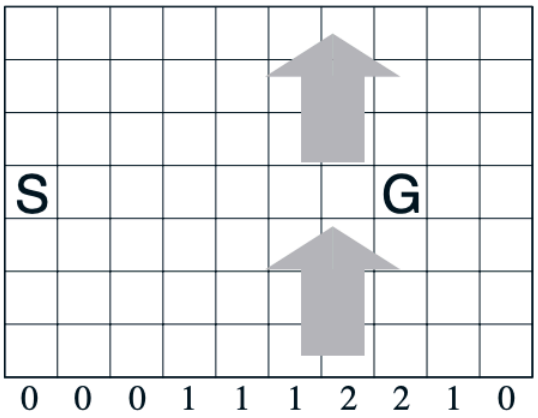


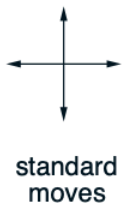


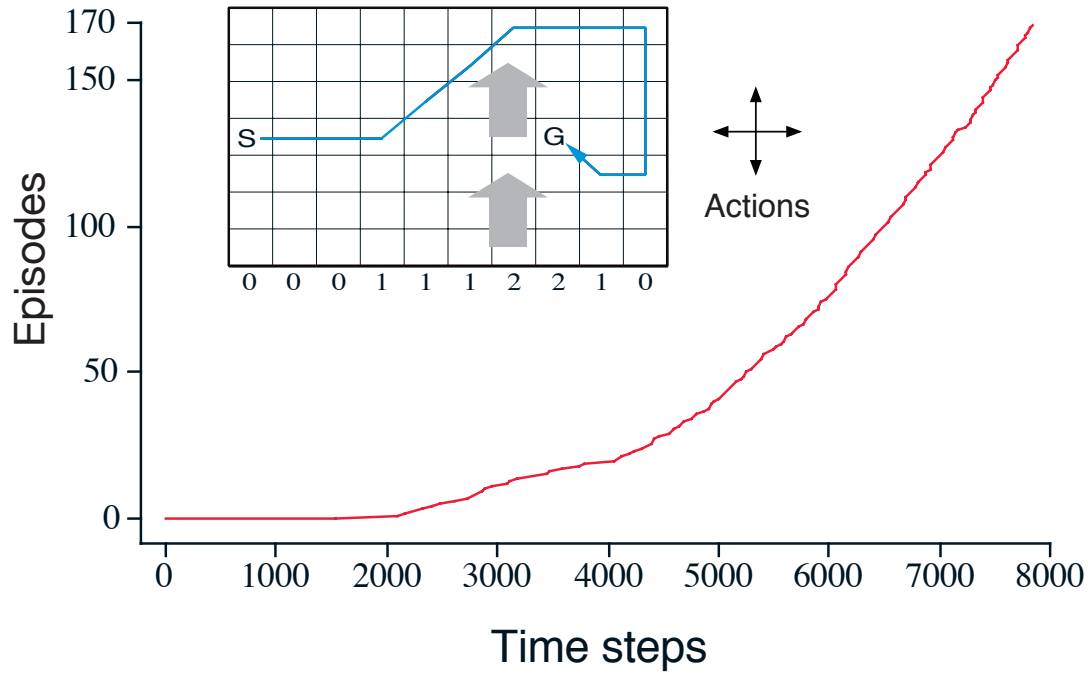
À programmer en TP

SARSA en action

[Sutton, exemple 6.5]







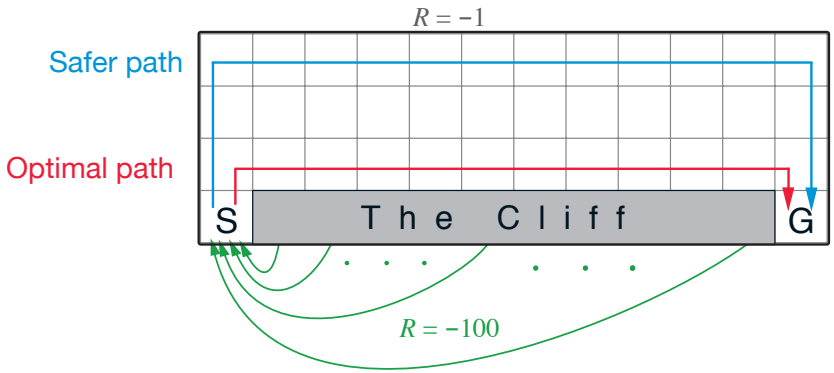
À programmer en TP

Episodes vs. time steps

1. partie plate de la courbe : amélioration
2. partie convexe de la courbe : amélioration
3. partie linéaire de la courbe : convergence

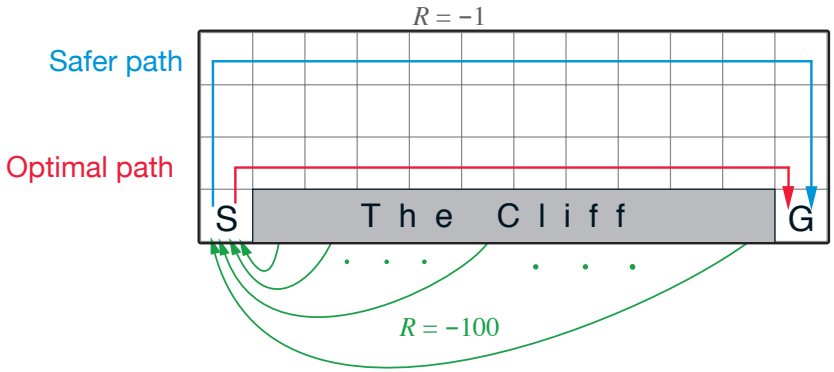
SARSA vs Q-learning

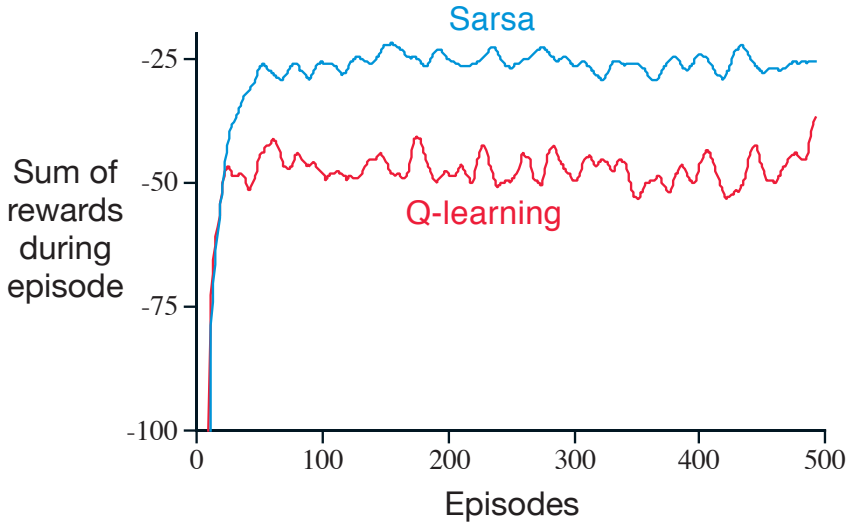
[Sutton, exemple 6.6]



SARSA vs Q-learning

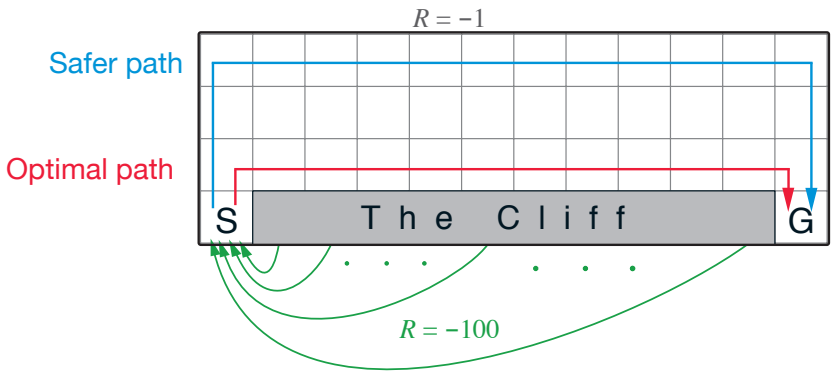
[Sutton, exemple 6.6]

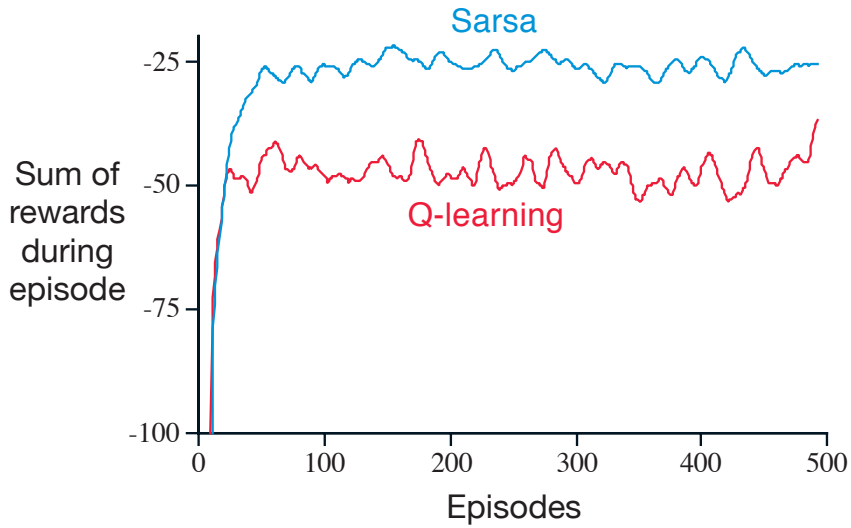




SARSA vs Q-learning

[Sutton, exemple 6.6]



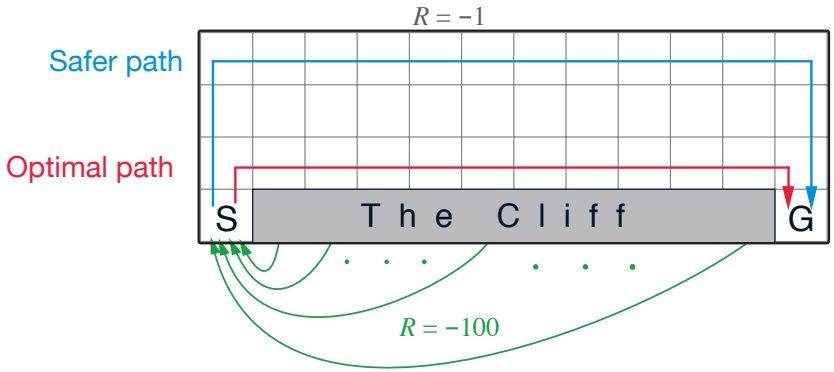


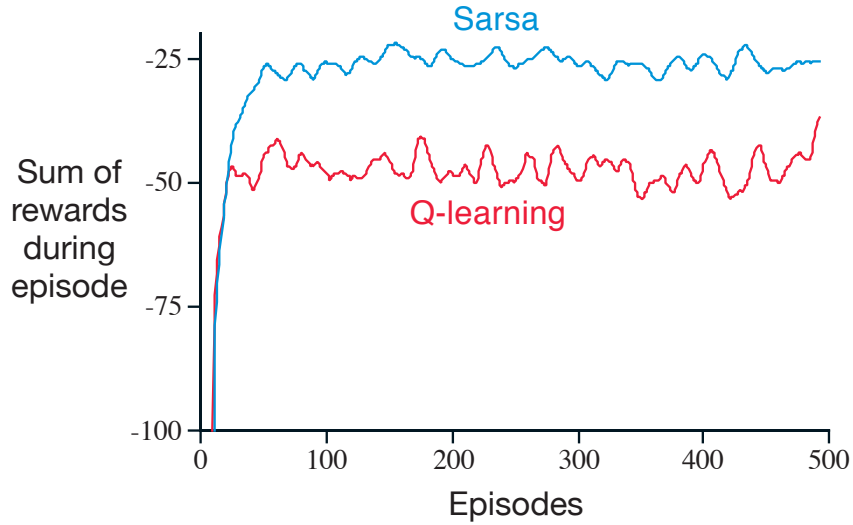
SARSA

- converge vers une stratégie sous-optimale
+ mais moins risquée et qui **récolte une meilleure récompense**

SARSA vs Q-learning

[Sutton, exemple 6.6]





SARSA

- converge vers une stratégie sous-optimale
+ mais moins risquée et qui **récolte une meilleure récompense**

Q-learning

+ **trouve la stratégie optimale**
- mais ne l'utilise pas (ϵ-greedy, off-policy)

Show time !

Sommaire

Conclusion

- vous connaissez les deux plus célèbres algorithmes d'AR
- vous connaissez les principes sous-jacents

  TD-learning, General Policy Iteration
- ils sont appropriés quand l'environnement est inconnu
- ils ont chacun leurs qualités et leurs défauts
- le principe TD-learning s'étend à un horizon temporel plus élevé

  TD(λ), eligibility traces
- plus généralement, les principes vus donnent lieu à de nombreux algorithmes que vous pouvez aborder.

Sommaire

TP

Exercice 1 – Windy Grid World

1. Programmez l'environnement, avec les changements d'états et les récompenses, et un affichage dans la console.
2. Programmez les agents Q-learning et Sarsa sur ce problème.
3. Faites évoluer en boucle les agents dans l'environnement et affichez la meilleure stratégie de chacun.
4. Faites l'exercice 6.9 du livre de Sutton.
5. Optionnel : faites l'exercice 6.10 du livre de Sutton.

Exercice 2 – QLearningMouse

1. Lancez et étudier le code fourni
2. Identifiez la définition des états que la souris voit. À quoi correspond chaque état ? Combien y a-t-il d'états ?
3. Quelle est la taille de l'ensemble des paramètres (s, a) sur lequel est définie la fonction q ?
4. Programmez Sarsa dans ce contexte