Sutton Barto Learning

Chaptre 6: Temporal Difference Learning

Elkael MAXIME
Bin LIU

Master2 Informatique AMIS Paris Saclay
UVSQ 2019-2020

Sommaire :

- Méthode TD(0)
- Présentation du jeu du taxi
- Sarsa : On-Policy control.
- Q-learning : Off-Policy control

Equation de Bellman pour TD(0) (vue en cours) :

Equation

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + V(S_{t+1}) - V(S_t)\right]; \tag{2.1}$$

$\alpha \in [0, 1]$: le facteur d'apprentissage ;

Equation de Bellman pour TD(0) avec discount :

Equation

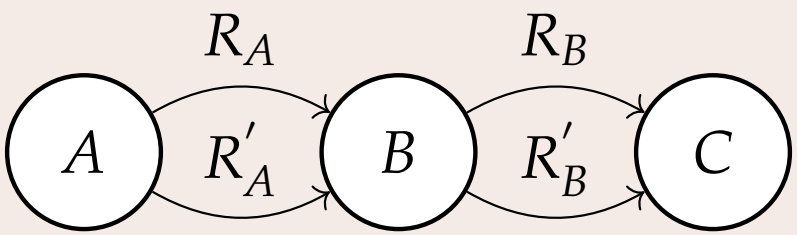
$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]; \tag{2.2}$$

$\alpha \in [0, 1]$: le facteur d'apprentissage
$\gamma \in [0, 1]$: le facteur de réduction

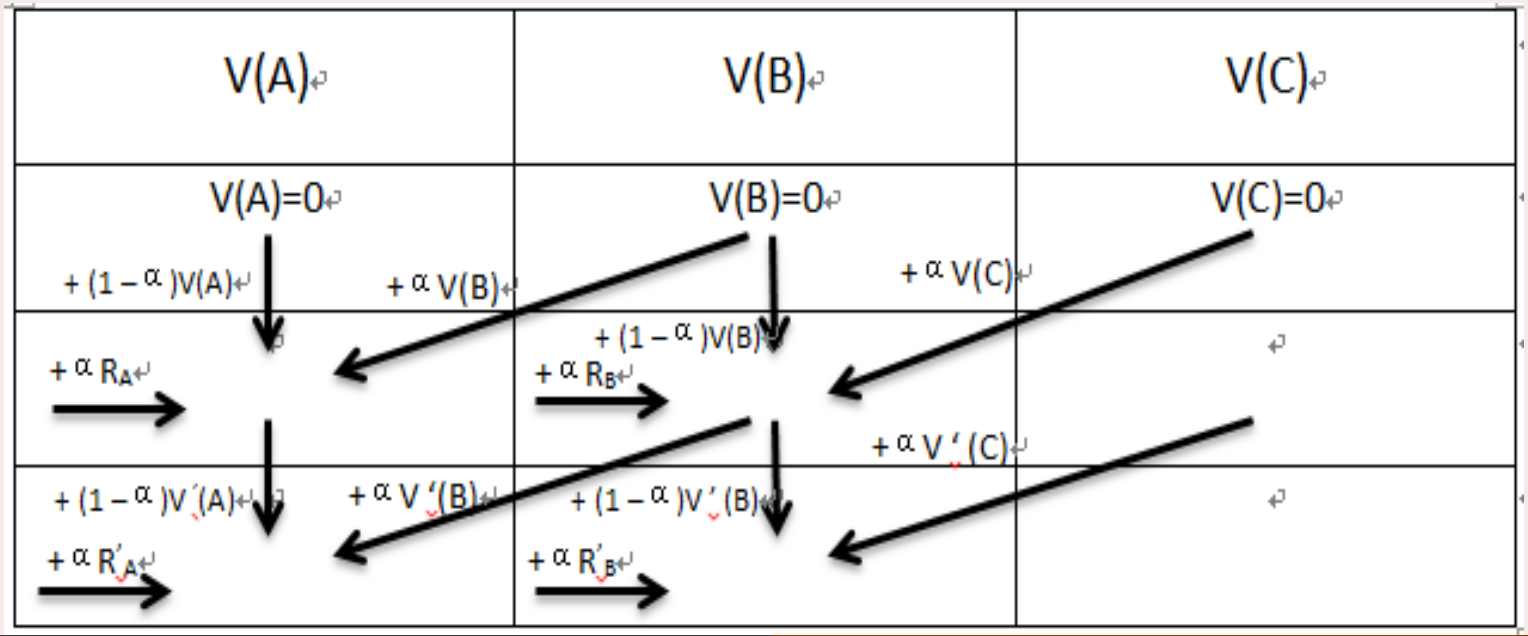
Les récompenses les plus lointaines dans le temps ont une probabilité plus faible d'arriver.

Rappel sur l'éxecution de TD(0) :

Example



$V(A) = V(B) = V(C) = 0$ au début de l'estimation



Les récompenses les plus lointaines dans le temps ont une probabilité plus faible d'arriver.

Equation

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]; \tag{2.3}$$

Gain : $G_t = R_t + \gamma R_{t+1} + ... + \gamma^n R_{t+n}$
$V(S_t) = E[G_t]$

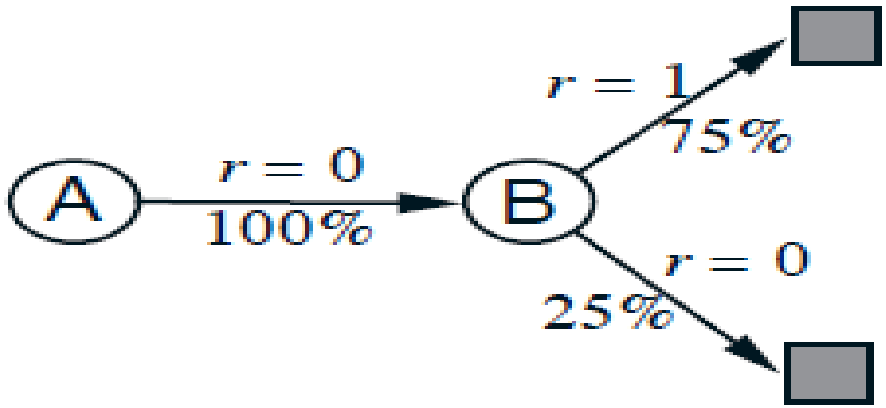


FIGURE – Le modèle de chaîne de markov

On considère 8 épisodes :

B ; 1	A ; 0 ; B ; 0
B ; 1	B ; 1
B ; 1	B ; 1
B ; 1	B ; 1
B ; 1	B ; 0

Pour B, $V(B) = 3/4$, une seule solution !
Pour A, 2 choix possibles :

- Méthode de Monte Carlo : $V(A) = 0$ en moyenne.
- Méthode TD(0) : $V(A) = 3/4$ car A mène toujours à B.

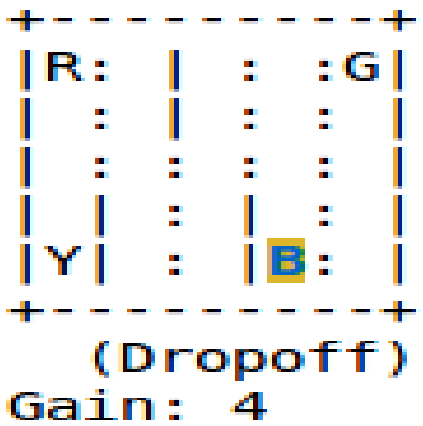
Intuition :

- TD (0) : le résultat correspond au modèle qui a le plus probablement généré ces données.
- Monte Carlo : le résultat correspond exactement aux données.

Policy Evaluation

Dans un jeu à 1 joueur, en fixant la stratégie π, on se ramène à un jeu à 0 joueur et on peut appliquer TD(0) pour évaluer cette stratégie

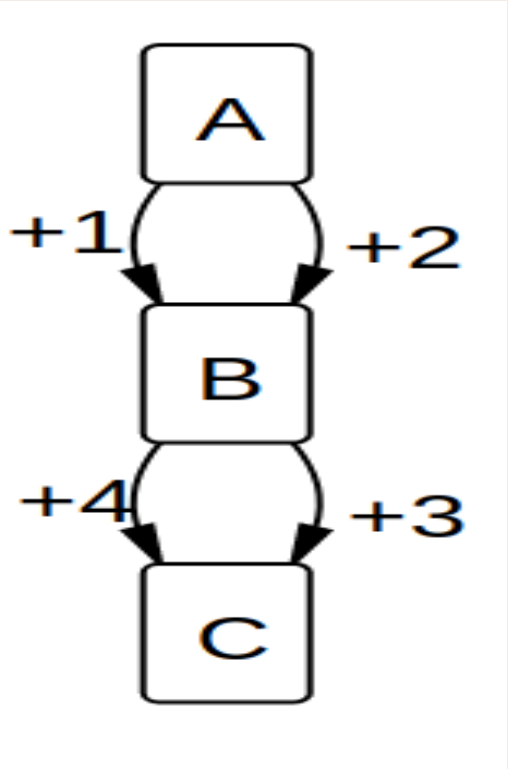
Un exemple d'éxecution dans un jeu à 1 joueur : le jeu du taxi

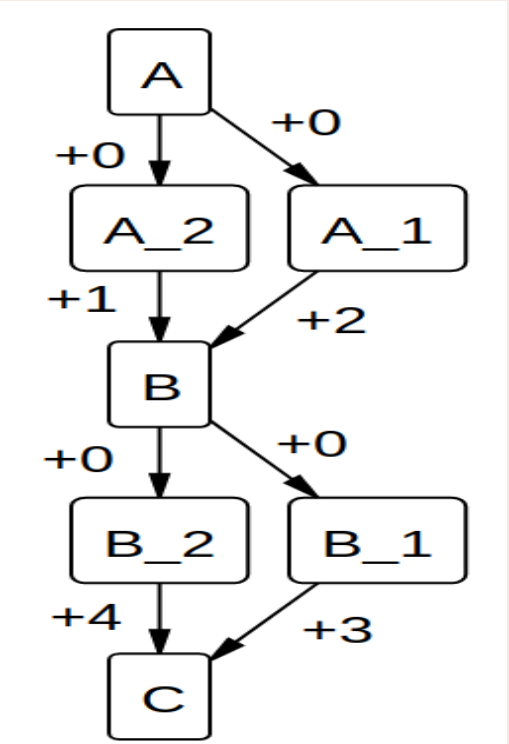


- 500 états (grille 5 * 5, 5 positions possibles pour le passager, 4 destinations possibles)
- 6 actions : haut, bas, gauche, droite, pickup, drop-off.
- Récompenses : -1 par tour de jeu, +20 pour déposer le passager, -10 pour pickup/dropoff illégal.
- Jeu fini : dans tous les cas au bout de 200 étapes on stoppe le jeu

Q-learning et SARSA : trouver une stratégie optimale dans les jeux à 1 joueur à horizon fini

Example





- On définit la fonction Q(S, A) comme la fonction qui donne la valeur d'une action A dans l'état S
- La stratégie optimale est la stratégie qui maximise toutes les Q values pour toutes les actions qu'elle choisit
- Donc on cherche à calculer Q pour tous les couples (S, A) pour en déduire la stratégie optimale

- On cherche à améliorer la stratégie qu'on a déja découverte (exploitation) donc on va jouer les actions qui ont la meilleure Q-value (stratégie gloutonne)
- Si on joue uniquement nos meilleures actions on rate peut etre des opportunités d'innover pour s'améliorer
- Pour aider l'exploration on joue donc une stratégie ϵ-gloutonne, c'est à dire qu'on joue aléatoirement avec proba ϵ

Sarsa : Choisir une stratégie optimale dans un jeu à un joueur

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$

Repeat (for each episode):

  Initialize S

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

  Repeat (for each step of episode):

    Take action A, observe R, S'

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

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

    $S \leftarrow S'; A \leftarrow A';$

  until S is terminal

Q-learning : Choisir une stratégie optimale dans un jeu à un joueur

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$

Repeat (for each episode):

  Initialize S

  Repeat (for each step of episode):

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

    Take action A, observe R, S'

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

    $S \leftarrow S'$;

  until S is terminal

Sarsa : on-policy, Q-learning : off-policy

- Sarsa :
  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$

- Q-learning :
  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$

La mise à jour donnée par Q-learning ne dépend pas de l'action effectuée, donc Q-learning peut apprendre la stratégie optimale sans suivre la stratégie qu'il est en train d'apprendre.

Conclusion

Limites de ces algos :

- Si l'espace d'états/actions est très grand : peu réaliste
- Si l'espace d'actions ou d'états est continu, on ne peut pas directement représenter le jeu par un tableau

Questions ?