TRAORÉ
Madina

Compte-rendu

TME sur l'apprentissage par renforcement

MU5IN259 - Intelligence Artificielle pour la Robotique

Master d'Informatique

Année universitaire 2018-2019

Sommaire

1 Dynamic Programming

Comparaison des algorithmes Value Iteration et Policy Iteration

Pour le labyrinthe de grande taille (cf Figure 7), les algorithmes Value Iteration et Policy Iteration convergent en autant d'itérations l'un que l'autre (17 itérations). Cependant, Value Iteration semble avoir une complexité temporelle meilleure que celle de Policy Iteration (0.23s en moyenne pour Value Iteration contre 0.29s pour Policy Iteration). Cela peut s'expliquer par le fait que les mises à jour de Policy Iteration sont beaucoup plus coûteuses (évaluation de la politique actuelle puis amélioration de celle-ci) que celles de Value Iteration (amélioration directe de la politique à l'aide des valeurs venant d'être calculées).

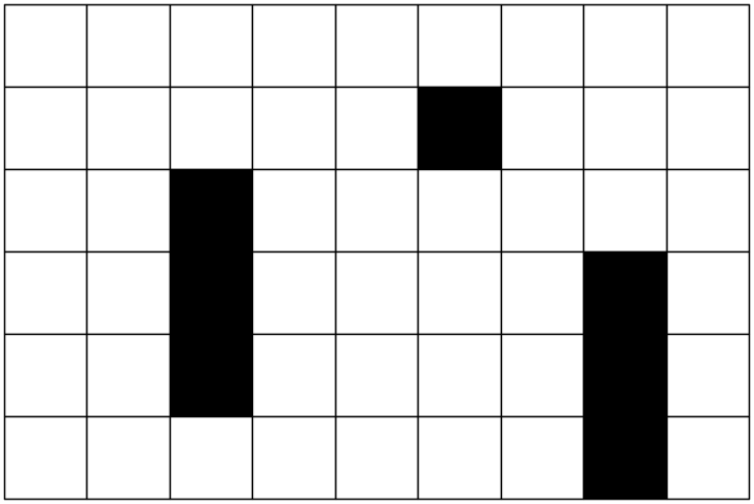


Figure 1 – Labyrinthe utilisé pour l'analyse des algorithmes Value Iteration et Policy Iteration

2 (Model-Free) Reinforcement Learning

Efficacité de l'algorithme Q-learning en fonction des hyper-paramètres

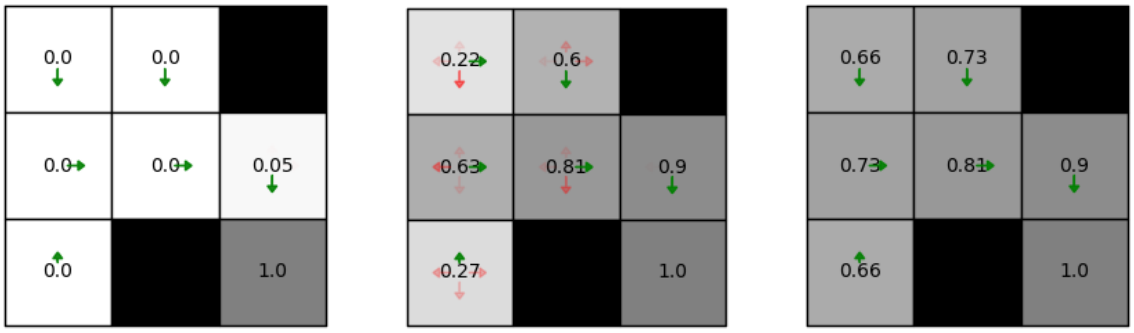
Afin de tester l'efficacité de l'algorithme Q-learning en fonction des hyper-paramètres τ (*softmax temperature*), α (*learning rate*) et γ (*discount factor*), on fait varier chacun de ceux-ci en maintenant les autres fixés.

Softmax temperature τ

En faisant varier l'hyper-paramètre τ de 0.1 à 1, on s'aperçoit que la Q-table est davantage mise à jour lorsque τ est grand. En effet, plus τ est grand, plus l'agent explore son environnement : il se retrouve donc régulièrement sur des cases qu'il n'a jamais ou peu visité et effectue de nouvelles actions, cela lui permet de mieux comprendre comment est constitué le labyrinthe et quels déplacements il doit effectuer pour atteindre la case finale.

Learning rate α

Après 100 épisodes, pour τ et γ respectivement fixés à 0.1 et 0.9, on obtient les résultats suivants :



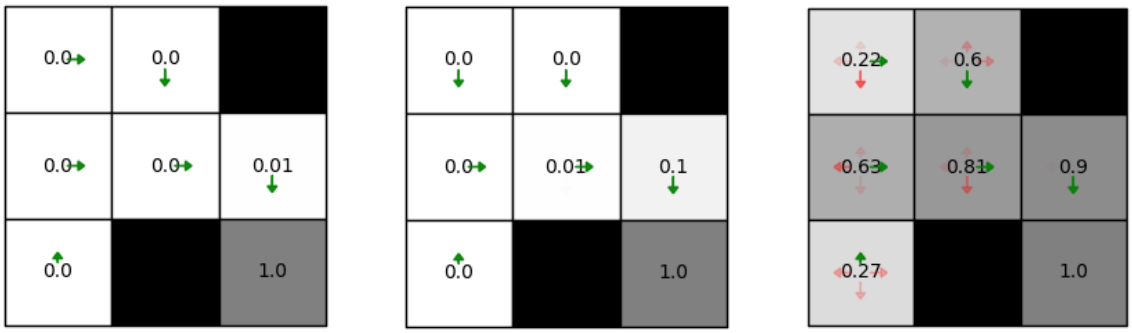
(a) $\alpha = 0.001$ (b) $\alpha = 0.1$ (c) $\alpha = 1$

FIGURE 2 – Valeurs et politiques associées à chaque cases après avoir exécuté l'algorithme Q-learning sur un labyrinthe de petite taille avec différentes valeurs de α

On constate que plus l'hyper-paramètre α est petit plus le temps mis par l'algorithme pour trouver les véritables valeurs de la Q-table (celles qui permettent à tout agent la connaissant de trouver la case finale en effectuant le moins de déplacements possibles à partir de n'importe quelle position) est long.

Discount factor γ

Le discount factor influe de la même manière sur l'apprentissage du labyrinthe par l'agent : plus il est petit plus l'agent met de temps à trouver une politique complète et optimale. Lorsque celui-ci est trop faible, l'erreur de prédiction faite à chaque étape sur la valeur qui aurait du être attribuée à chaque case (*TD error*) est faible. Ainsi, les mises à jour des valeurs de la Q-table se font très lentement. Ici, les meilleurs résultats ont été obtenus pour $\gamma = 0.9$.

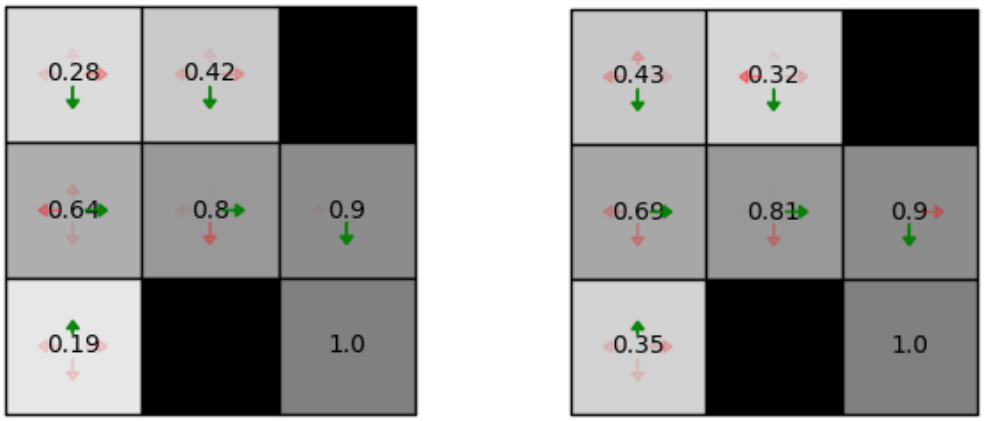


(a) $\gamma = 0.01$ (b) $\gamma = 0.1$ (c) $\gamma = 0.9$

FIGURE 3 – Valeurs et politiques associées à chaque cases après avoir exécuté l'algorithme Q-learning sur un labyrinthe de petite taille avec différentes valeurs de γ

Comparaison des algorithmes Q-learning et SARSA

Pour $\tau = 0.1$, $\alpha = 0.1$ et $\gamma = 0.9$, les résultats obtenus après exécution de Q-learning et SARSA sur le labyrinthe sont les suivants :



(a) Q-learning (b) SARSA

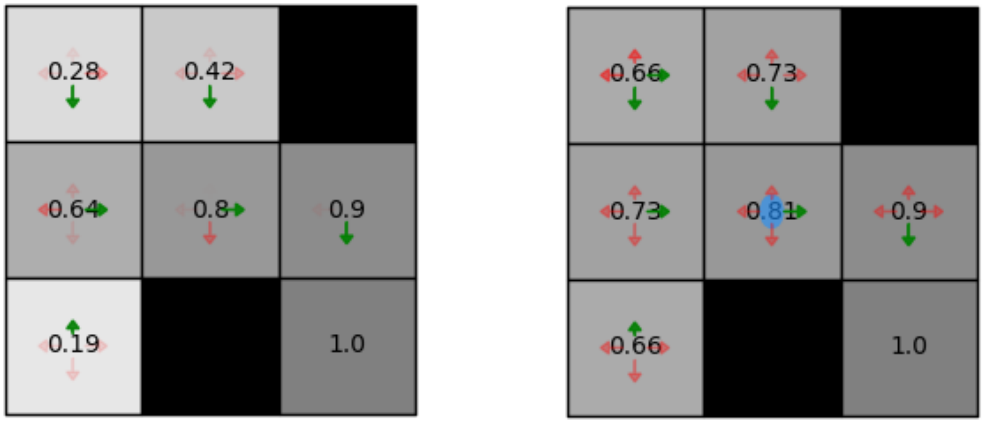
FIGURE 4 – Valeurs et politiques associées à chaque cases après avoir exécuté les algorithmes Q-learning et SARSA (100 épisodes) sur un labyrinthe de petite taille

En moyenne, 786 steps sont nécessaires pour que SARSA converge sur cette instance tandis qu'il en faut seulement 514 aec l'algorithme Q-learning. D'autre part, on remarque que la Q-table de l'agent a été davantage mise à jour avec SARSA qu'avec Q-learning. Il semblerait que SARSA calcule les valeurs de la Q-table en évaluant le maximum d'actions possibles pour éviter à l'agent d'effectuer de mauvais déplacement a posteriori tandis que Q-learning se contente de trouver la politique optimale à chaque étape en supposant que l'agent est effectivement en train de suivre cette politique. Ainsi, bien que ces algorithmes permettent tous deux de calculer une politique menant vers la case finale à partir de toutes positions du labyrinthe, on peut penser que SARSA permet d'avoir une politique plus "sûre" que Q-learning.

3 Model-Based Reinforcement Learning

Comparaison des algorithmes RTDP et Q-learning

Pour $\tau = 0.1$, $\alpha = 0.1$ et $\gamma = 0.9$, les résultats obtenus après exécution de Q-learning et SARSA sur le labyrinthe sont les suivants :



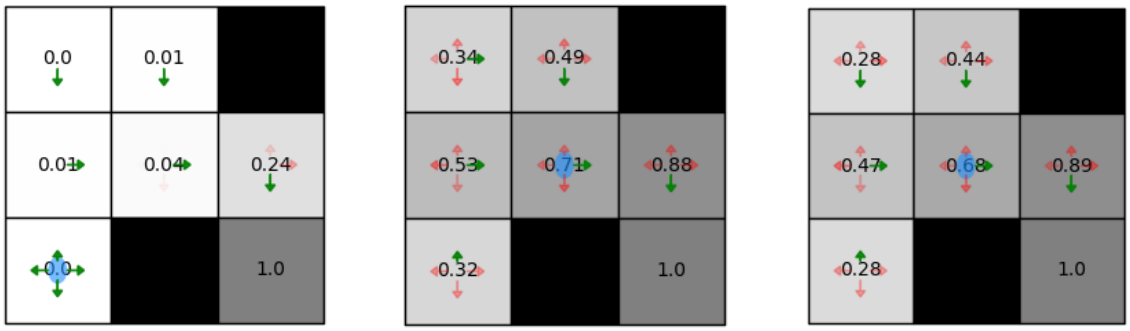
(a) Q-learning (b) RTDP

FIGURE 5 – Valeurs et politiques associées à chaque cases après avoir exécuté les algorithmes Q-learning et RTDP (100 épisodes) sur un labyrinthe de petite taille

On observe que RTDP converge beaucoup plus vite que Q-learning (0.21s contre 0.43s pour obtenir les valeurs attendues de la Q-table).
Lorsque la deviation est trop grande pour RTDP ($deviation = 1$), celui-ci converge moins vite vers la Q-table optimale.

4 On-policy and Off-policy, with SARSA and Q-learning

L'algorithme SARSA avec Replay Buffer parvient à calculer une Q-table quasi-optimale en 100 épisodes tandis qu'il en faut 1000 pour l'algorithme Q-learning avec Replay Buffer.



(a) Qlearning avec Replay Buffer (100 épisodes)

(b) Qlearning avec Replay Buffer (1000 épisodes)

(c) SARSA avec Replay Buffer

FIGURE 6 – Valeurs et politiques associées à chaque cases après avoir exécuté les algorithmes Q-learning et SARSA avec Replay Buffer sur un labyrinthe de petite taille

5 N-step Q-learning

L'efficacité de N-step Q-learning a été testée sur le labyrinthe suivant :

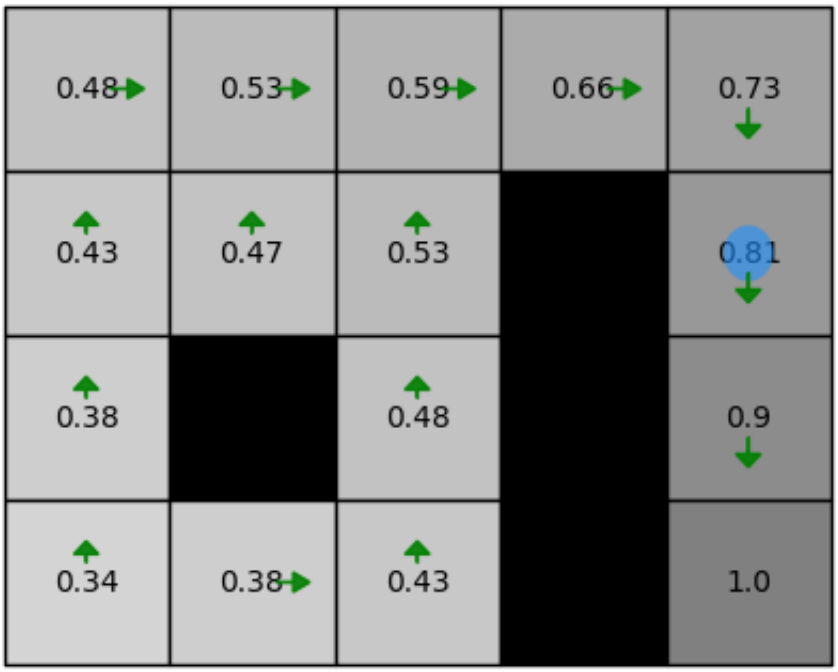
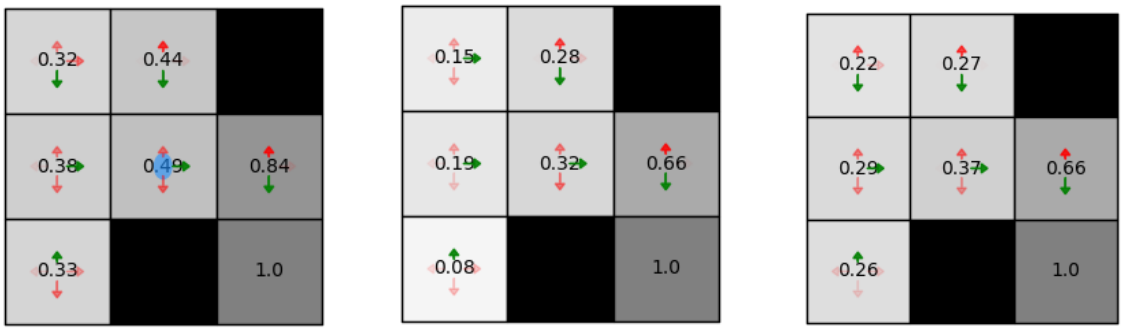


FIGURE 7 – Labyrinthe utilisé pour l'analyse de l'algorithme NStep Q-learning

6 Actor-Critic

Efficacité de l'algorithme Q-learning en fonction des valeurs de α_1, α_2

Intuitivement, on peut se dire que les valeurs de α_1 et α_2 doivent être les mêmes afin que l'Acteur et le Critique aient le même poids lorsqu'ils explorent le labyrinthe. Cette intuition semble se vérifier avec les résultats obtenus ci-dessous :



(a) $\alpha_1 = 0.1$ et $\alpha_2 = 0.1$ (b) $\alpha_1 = 0.2$ et $\alpha_2 = 0.8$ (c) $\alpha_1 = 0.6$ et $\alpha_2 = 0.4$

FIGURE 8 – Valeurs et politiques associées à chaque cases après avoir exécuté l'algorithme Actor-Critic sur un labyrinthe de petite taille