TME3 RLD

Q-Learning

Edwyn Brient, Léo Hein - Décembre 2021

1 Algorithme Q-Learning

L'algorithme Q-Learning tabulaire est un des modèles de base de l'apprentissage par renforcement.

L'unique différence (mais pas des moindres) entre Q-learning et SARSA se situe dans la mise à jour de Q, la table des valeurs. Si pour l'algorithme Q-learning, méthode off policy, la mise à jour se fait en prenant le max sur Q à l'instant t+1, pour l'algorithme SARSA, la mise à jour se fait en choisissant l'action suivante en fonction de la politique de l'algorithme. Celui ci fait donc partie des méthodes on policy.

Pour bien comprendre la différence entre les deux, on peut comparer la phase de mise à jour entre les algorithmes :

— Pour Q-learning la mise à jour est la suivante :

$$Q(s_t, a_t)Q(s_t, a_t) + \alpha(r_t + \gamma max_{a' \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, a') - Q(s_t, a_t))$$

— Pour SARSA, on commence par choisir une action a_{t+1} par la politique de l'algorithme puis on met Q à jour :

$$Q^\pi(s_t, a_t)Q^\pi(s_t, a_t) + \alpha(r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t))$$

Dans ces deux algorithmes, α permet de jauger la vitesse d'apprentissage et γ, le discount, est une variable décrivant comment la valeur d'un état dépend des états suivants.

Enfin l'algorithme DynaQ permet d'apprendre le modèle tabulairement en parallèle de Q et d'apprendre Q en simulant le modèle à chaque apprentissage. Cette méthode est très sample efficient puisque l'algorithme peut apprendre parallèlement au actions jouées dans l'environnement. Pour cet algorithme, on introduit une nouvelle variable d'apprentissage du modèle α_r.

Si la mise à jour de Q est la même lors des simulations ou dans l'environnement, P et R, les probabilités de transition et les récompenses sont apprises de la manière suivante :

$$\hat{R}(s_t, a_t, s_{t+1})\hat{R}(s_t, a_t, s_{t+1}) + \alpha_r(r_t - \hat{R}(s_t, a_t, s_{t+1}))$$
$$\hat{P}(s_{t+1}|a_t, s_t)\hat{P}(s_{t+1}, a_t, s_t) + \alpha_r(1 - \hat{P}(s_{t+1}, a_t, s_t))$$
$$\forall s' \neq s_{t+1} : \hat{P}(s'|a_t, s_t)\hat{P}(s', a_t, s_t) + \alpha_r(0 - \hat{P}(s', a_t, s_t))$$

On peut remarquer dans la mise à jour de P que si l'environnement arrive dans un état s_{t+1}, une mise à jour des autres états est nécessaires pour que la probabilité somme toujours à 1 et pour réduire la probabilité que le modèle se trouve dans ces états après cette suite (état, action).

Une particularité des algorithmes mis en place est que lorsqu'ils trouvent un nouvel état, ils posent $Q(s, a) = 1 \ \ \forall a \in \mathcal{A}$. Cette initialisation permet de considérer ces Q valeurs comme bonnes par postulat avant de les mettre à jour.

2 Expériences

2.1 Paramètres

Les expériences que nous avons mené sur ces algorithmes se basent sur l'environnement gridworld et plus particulièrement sur les plans 0, 1 et 5. Les paramètres des expériences sont les suivants :

— learning rate : 0.1
— mode d'exploration : ϵ-greedy
— ϵ : 0.1
— decay : 0.9999
— discount γ : 0.999
— durée maximum des épisodes pendant l'entrainement : 100 pour le plan 1 et le plan 0 et 250 pour le plan 5.

2.2 Résultats

Le plan 0 a été utilisé pour tester la différence entre SARSA et Q-learning. En effet un exemple classique pour différencier experimentalement les deux algorithme est le voyage d'un point A à un point B au bord d'une falaise avec une probabilité non nulle d'aller vers la falaise en tentant de la longer. Ainsi l'algorithme SARSA ne mettant pas à jour Q de manière greedy, il prend souvent moins de risque, ne longeant pas la falaise mais prenant une case de marge. Le plan 0 est très similaire avec une case au centre où on ne peut pas passer, est ce que les algorithmes auront le même comportement qu'en théorie ?

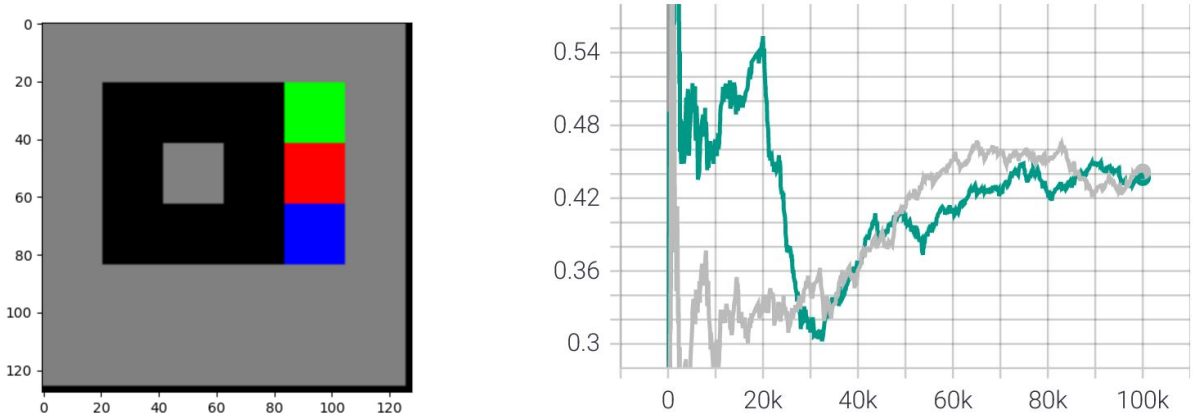

FIGURE 1 – A gauche : plan 0 avec en bleu la position de départ et en rouge et vert respectivement les cases finales de récompense -1 et 1. A droite : Récompenses des version SARSA et Q-learning avec et sans dynaQ en fonction du nombre d'épisodes. En *gris*, la version Q-learning, en *vert*, la version SARSA.

Les deux algorithmes ont finalement les mêmes résultats contrairement à ce qu'on aurait pu imaginer. Il y avait pourtant une longueur maximum d'épisode de 100 ce qui laissait largement le temps à l'agent de faire le tour. Il aurait pu être intéressant de tester un plan sans la case grise pour voir si le problème vient de l'implémentation ou si l'ajout de cette case fait en sorte que SARSA ne trouve plus le chemin le moins risqué.

Le plan 1 est identique au plan 0 mais avec une deuxième source de reward très proche de la falaise. Alors, la solution consistant à faire le tour du terrain pour arriver à l'état final de reward 1 n'est plus optimal. On peut s'attendre à des résultats similaires pour les algorithmes.

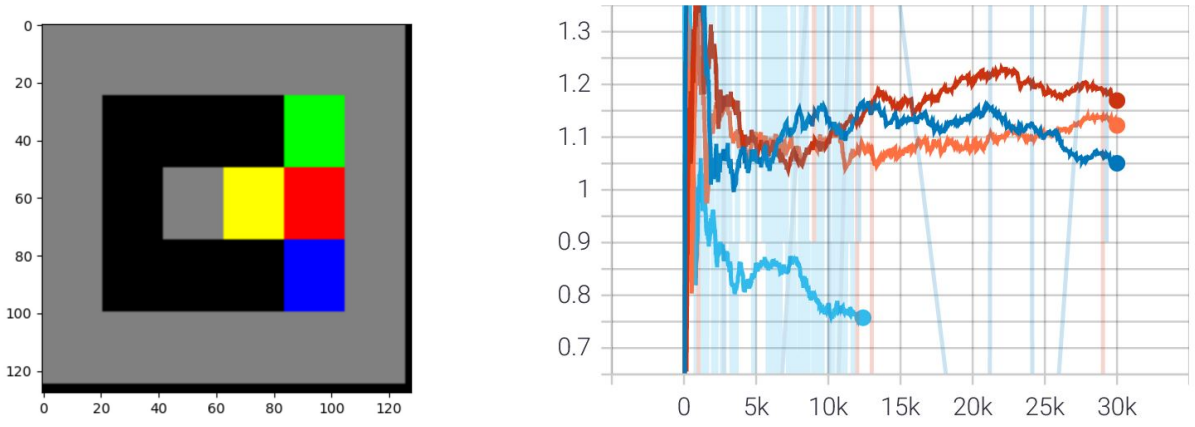

FIGURE 2 – A gauche : plan 1 avec en bleu la position de départ, en rouge et vert respectivement les cases finales de récompense -1 et 1 et en jaune une case de récompense 1. A droite : Récompenses des version SARSA et Q-learning avec et sans dynaQ en fonction du nombre d'épisodes. En *rouge*, la version Q-learning avec DynaQ, en *orange*, la version SARSA, en *bleu* la version Q-learning et en *cyan* la version SARSA avec DynaQ.

On observe que seule la version SARSA avec dynaQ perds en efficacité au cours des épisodes et termine vers les 0.75 de reward moyenne. Les trois autres algorithmes sont de même efficacité à environ 1.15 de reward moyenne.

Finalement, le plan 5 est bien plus étendu que les plans précédents et la deuxième source de reward est beaucoup plus lointaine que pour les plans 0 et 1. Si l'efficacité de DynaQ ne s'est pas prouvé sur les derniers plans, sur ce plan ci, les samples permettant d'apprendre beaucoup plus vite, les états à $Q = 1$ peuvent propager leur bonne valeur et l'exploration se fait bien mieux que pour les algorithmes standards.

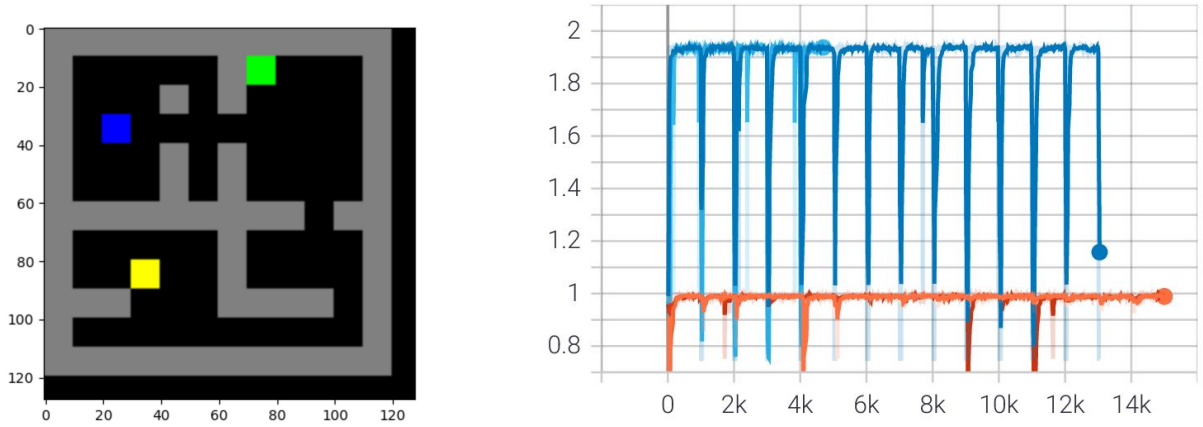

FIGURE 3 – A gauche : plan 5 avec en bleu la position de départ, en rouge et vert respectivement les cases finales de récompense -1 et 1 et en jaune une case de récompense 1. A droite : Récompenses des version SARSA et Q-learning avec et sans dynaQ en fonction du nombre d'épisodes. En *bleu*, la version Q-learning avec DynaQ, en *orange*, la version SARSA, en *rouge* la version Q-learning et en *cyan* la version SARSA avec DynaQ.

Sur ce dernier plan, l'exploration offerte par l'ajout de dynaQ avec l'optimisme (nouveaux états configurés à 1 de valeur) permet de découvrir le reward jaune contrairement aux algorithmes standards qui ne permettent pas de faire ruisseler la valeur des états peu explorés sur les états standards (les plus vus).

On peut cependant noter une certaine instabilité sur les algorithmes avec dynaQ, qui tous les 1000 épisodes n'arrivent pas à trouver la case verte et restent bloqués vers les 0.75 de reward à 250 step.

3 Conclusion

Ce TP nous a permis de découvrir les algorithmes Q-learning, SARSA et DynaQ très intéressants dans les problèmes à nombre de couples (état, action) finis. Lorsque les états passent en continu comme dans le problème de cartpole, une solution peut consister en discrétisant les états artificiellement mais des algorithmes de renforcement profonds prennent facilement le relai dans ce genre de problèmes.