Rapport d'Apprentissage par Renforcement

TP3 : Q-Learning et Sarsa

EPITA - [SCIA] - 2025

Octobre 2024

Alice CARIOU ***(alice.cariou@epita.fr)***

Eliana JUNKER ***(eliana.junker@epita.fr)***

1 Introduction

L'objectif de ce tp est d'implémenter un agent qui apprend à jouer au jeu Taxi-v3 de OpenAI Gym. Le but du jeu est de déposer un passager à une destination spécifique en un minimum de temps. Le jeu est composé d'une grille de 5x5 cases et le taxi peut se déplacer dans les 4 directions (haut, bas, gauche, droite). Le taxi peut prendre un passager sur une case spécifique et le déposer à une destination spécifique. Le jeu est terminé lorsque le passager est déposé à la destination. Le jeu est aussi terminé si le taxi prend plus de 200 actions.

Nous avons implémenté un agent qui apprend à jouer à ce jeu en utilisant les algorithmes Q-Learning et SARSA :

2 Implémentations

Le premier grand choix d'implémentation que nous avons été amené à effectuer concerne l'optimisation. En effet, l'algorithme Qlearning implémenté comme demandé ne passait pas le assert (la moyenne des 100 dernières récompenses était inférieure à 0). Ceci peut être expliqué par un epsilon bien trop élevé, ce qui introduit trop d'aléatoire dans nos choix en permanence.

Pour régler ce problème, nous testons donc tous les epsilons situés entre 0 et 0.25 (avec un pas de 0.05) et gardons celui qui obtient les meilleurs résultats. Cette optimisation n'a pas été effectuée sur SARSA (pas d'epsilon) et sur Qlearning eps scheduling (cela empirait ou conservait les performances en restant sur 1000 itérations d'entraînement).

Un autre point que nous avons été amené à choisir a été l'algorithme utilisé pour la mise à jour de epsilon pour Qlearning epsilon scheduling. En effet, après avoir fait des recherches, nous avons constaté que le decayed epsilon greedy était l'algorithme majoritairement utilisé pour l'epsilon-scheduling. Son fonctionnement est le suivant : L'epsilon initial est grand, puis diminue avec le temps, lorsque l'on se rapproche à priori de la solution.

3 Résultats

En ce qui concerne nos animations, nous avons décidé d'en créer toutes les 100 itérations, ce qui nous semblait raisonnable pour montrer les capacités de notre agent à différentes étapes d'apprentissage.

Ensuite, nous avons affiché des courbes pour observer ou justifier nos résultats : la première montre l'évolution des rewards au cours de l'apprentissage (Voir Figure 1, 2 et 3). Cela nous permet d'observer la progression des performances au cours du temps.

La deuxième représente les rewards moyens (sur les 100 derniers résultats) en fonction du choix de epsilon, en faisant varier epsilon entre 0 et 0.25, avec un pas de 0.05 (Voir Figure 4).

Voici notre tableau récapitulatif des résultats (le reward moyen est calculé sur les 100 dernières itérations). Le temps a été calculé pour les 1000 itérations d'entraînement et avec la génération des animations. Les temps sont donc comparables, mais plus élevés que ce qu'ils devraient vraiment être.

	reward moyen	temps (s)
Q-Learning	7.74	34.3
Q-Learning avec eps-scheduling	4.36	26.5
Sarsa	8.21	39.2

Au final, SARSA possède les meilleurs performances mais est plus lent à se stabiliser. Qlearning esp scheduling grimpe le plus rapidement mais possède des résultats moins bons (et plus changeant également, le relancer peut modifier totalement la courbe et les résultats). Qlearning avec le meilleur epsilon est proche des résultats de SARSA mais est un peu plus rapide à tourner et à se stabiliser.

4 Annexes

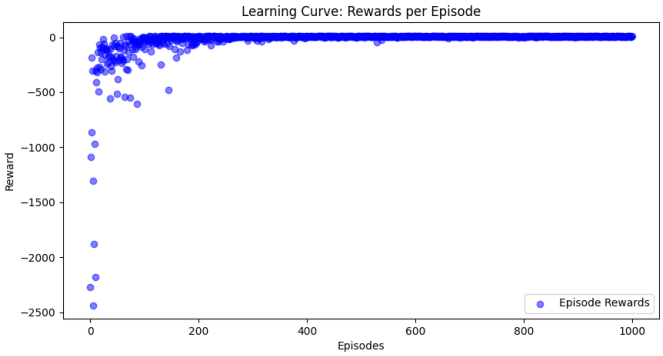

FIGURE 1 – Q-Learning

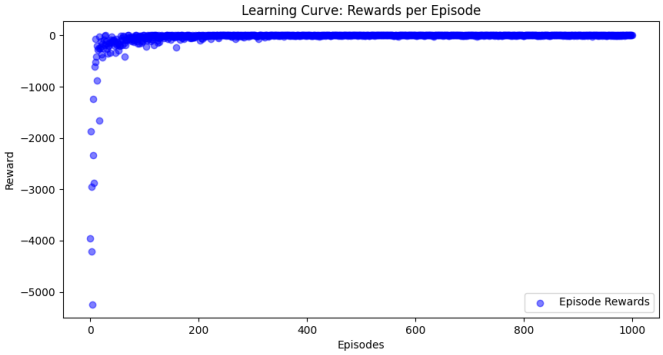

FIGURE 2 – e-scheduling

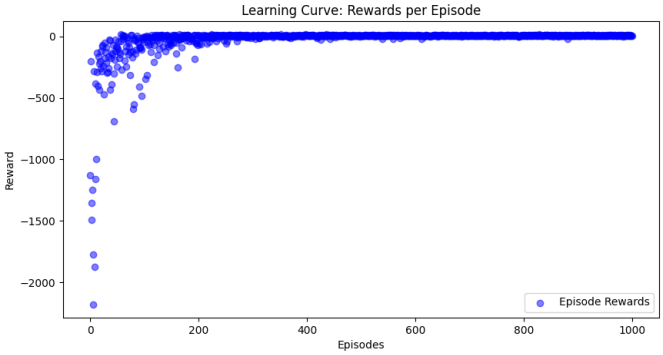

FIGURE 3 – Sarsa

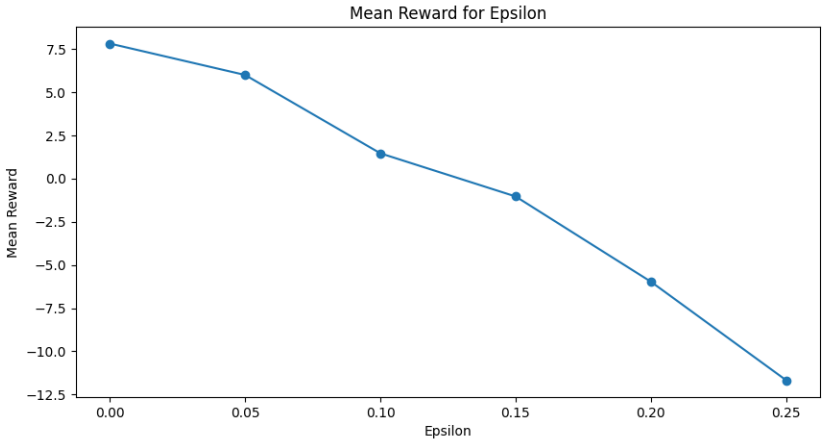

FIGURE 4 – Rewards en fonction de epsilon - Q-Learning