Sorbonne Université
Master 2 - Androide

Emmanuelle AMANN
Bruce ROSE

Coordination et Consensus Multiagents :

Modèles, Algorithmes, Protocoles

Multi-agent Reinforcement Learning using PettingZoo : Tic-tac-toe

14 décembre 2021

Introduction

Dans ce projet, nous nous intéressons à l'apprentissage multi-agents par renforcement. L'approche la plus simple pour l'apprentissage multi-agents est d'entraîner les agents indépendamment les uns des autres, sans connaissance des autres. Du point de vue d'un agent, les autres agents font simplement partie de l'environnement. Nous allons implémenter l'apprentissage multi-agents par Qtable pour des agents jouant au morpion (*Tic Tac Toe*).

1 Implémentation

Nous avons implémenté la fonction `epsilon_greedy_policy` ainsi que la fonction `update_Q_value`. Ces fonctions sont utilisées pour l'apprentissage dans la fonction `marl_q_learning`. Cette dernière donne la possibilité d'afficher une courbe de progression de l'apprentissage au fil de celui-ci. Cette courbe est lissée par un paramètre qui est de 2 par défaut (moyenne mobile). Elle permet également de choisir de quels types sont les joueurs entre `"Q"` et `"random"`.

La fonction `learn` est un *wrapper* pour la fonction d'apprentissage.

La fonction `test_policy` permet de faire s'affronter deux joueurs de types paramétrables, avec la possibilité d'afficher le jeu en cours de route. Cette fonction renvoie le nombre de victoires pour les deux joueurs.

2 Résultats

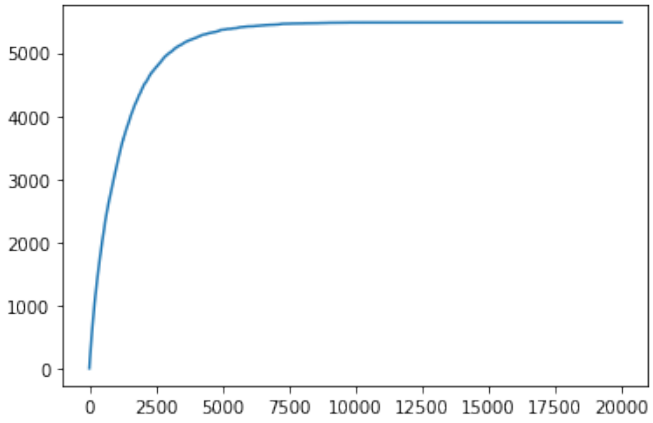

FIGURE 1 – Nombre d'états du jeu uniques trouvés au fil de 20 000 parties en, jouant aléatoirement.

Tous les états du jeu sont rencontrés au bout d'un nombre d'itérations égal à peu près à 7500 (cf. Figure 1). Il y a en tout 5478 états du jeu.

2.1 Apprentissage en self-play

Dans cette section nous entraînons deux joueurs en les faisant jouer l'un contre l'autre.

2.1.1 Courbe de progression

Nous avons affiché la progression du joueur 1 au cours de l'apprentissage. On constate que le joueur 1 obtient un niveau significativement bon contre le joueur aléatoire au bout d'un faible nombre d'épisodes.

2.1.2 Joueur 1 entraîné contre joueur 2 aléatoire

Pour tester l'apprentissage de notre joueur 1, nous avons exécuté 5000 parties contre un joueur aléatoire et avons compté les victoires et les défaites. Le joueur 1 gagne ainsi 98.62% du temps, le joueur 2 aléatoire 0% du temps. Un match nul arrive 1.38% du temps. Le fait que le joueur 2 ne gagne jamais montre que le joueur 1 est effectivement bien entraîné.

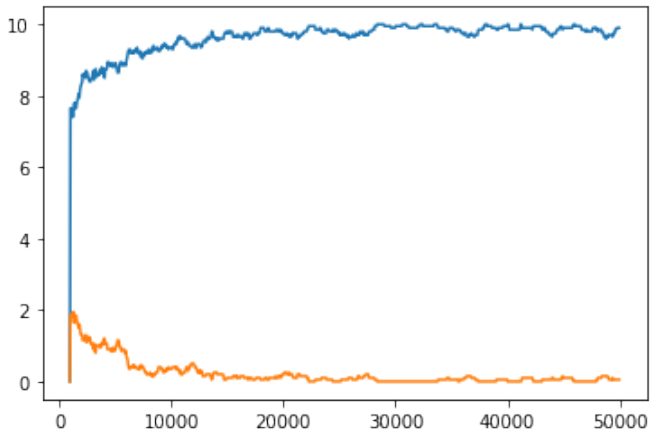

FIGURE 2 – Nombre de parties gagnées (moyenne mobile sur 20 valeurs) par le joueur 1 contre un joueur aléatoire parmi 10 au cours de l'apprentissage. Ordonnées : nombre de parties. Abscisses : nombre d'épisodes d'apprentissage.

2.1.3 Joueur 1 entraîné contre joueur 2 entraîné

Dans cette section, nous avons fait s'affronter les deux joueurs entraînés l'un contre l'autre. Le joueur 1 gagne ainsi 100% du temps. Ce résultat est étonnant, car normalement, le joueur 2 entraîné devrait toujours pouvoir causer un match nul. Si l'on observe certaines parties, on s'aperçoit que les joueurs répètent toujours les mêmes actions. Il est possible que le joueur 2 durant l'apprentissage ne différencie pas une défaite d'un match nul, ce qui expliquerait qu'il ne préférerait pas l'un à l'autre. Toutefois, le reward obtenu est 0 en cas de match nul et -1 en cas de défaite, donc cette théorie ne tient pas la route.

2.1.4 Joueur 1 aléatoire contre joueur 2 entraîné

Dans cette section, nous avons fait s'affronter le joueur 1 aléatoire contre le joueur 2 entraîné. Le joueur 1 gagne ainsi 0.6% du temps, le joueur 2, 85.44% du temps. Un match nul arrive 13.96% du temps. Le joueur 2 gagne moins souvent que le joueur 1 entraîné car il a le désavantage de jouer en deuxième.

2.2 Apprentissage du joueur 1 contre le joueur aléatoire

On peut penser que si l'apprentissage se fait contre un joueur aléatoire, alors la politique obtenue sera plus robuste. C'est ce que nous allons étudier dans cette section.

2.2.1 Courbe de progression

Nous avons affiché la progression du joueur 1 au cours de l'apprentissage. On constate une progression similaire à celle du joueur 1 entraîné contre un joueur 2 entraîné lui-aussi (cf. Figure 3).

2.2.2 Résultats contre aléatoire

Pour tester l'apprentissage de notre joueur 1 contre aléatoire, nous avons exécuté 5000 parties contre un joueur aléatoire et avons compté les victoires et les défaites. Le joueur 1 gagne ainsi 98.62% du temps, le joueur 2 aléatoire 0% du temps. Un match nul arrive 1.38% du temps, comme quand il était entraîné en self-play.

2.2.3 Résultats contre le joueur 2 entraîné en self-play

Dans cette section, nous avons confronté un joueur 1 entraîné contre un joueur aléatoire, et un joueur 2 entraîné en self-play. Sur les 5000 parties, chacune se finit en match nul, montrant l'optimalité des politiques employées par les deux joueurs.

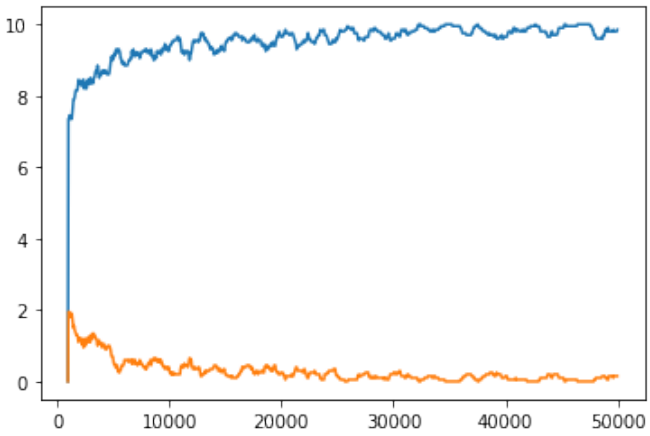

FIGURE 3 – Nombre de parties gagnées (moyenne mobile sur 20 valeurs) par le joueur 1 contre un joueur aléatoire parmi 10 au cours de l'apprentissage. Ordonnées : nombre de parties. Abscisses : nombre d'épisodes d'apprentissage.