Projet d'Apprentissage par Renforcement pour Space Invaders

WAN Zihao

28/4/2023

1 Introduction

Dans ce rapport, je présente mon projet d'apprentissage par renforcement pour résoudre le problème du jeu Space Invaders. Mon objectif est de développer et implémenter un algorithme qui apprend à jouer et maximiser le score en détruisant le plus d'aliens envahisseurs possibles. Je détaillerai mon démarche, les compétences acquises, les sources utilisées et les résultats obtenus, ainsi que la conclusion.

2 Méthodologie

2.1 Choix de l'algorithme

Pour ce projet, j'ai choisi d'utiliser l'algorithme DQN (Deep Q-Network) pour apprendre à jouer au jeu Space Invaders. Le DQN combine les avantages des méthodes de table Q-learning et des réseaux de neurones profonds, ce qui permet d'apprendre des politiques efficaces même dans des environnements à grande échelle.

2.2 Environnement et problème

Le problème que nous avons choisi est le jeu Space Invaders, où l'objectif est de détruire le plus d'aliens envahisseurs possibles. L'agent contrôle un vaisseau spatial qui peut effectuer quatre actions différentes : aller à gauche, aller à droite, tirer et ne rien faire. L'agent reçoit une récompense de 1 point pour chaque alien détruit.

2.3 Implémentation

J'ai implémenté l'algorithme DQN en utilisant la bibliothèque PyTorch. J'ai créé une classe `DQNAgent` qui contient les méthodes pour sélectionner des actions, mémoriser des expériences, mettre à jour le réseau et sauvegarder/charger des

modèles. La classe `DQN` est un modèle de réseau de neurones profonds qui prend en entrée l'état du jeu et retourne les valeurs Q pour chaque action possible.

Dans mon implémentation, j'utilise l'image complète du jeu comme état. Pour entraîner mon agent, j'utilise une boucle d'apprentissage sur 501 épisodes 10000 étapes par épisode. À chaque étape, l'agent sélectionne une action, applique l'action à l'environnement et mémorise l'expérience. Ensuite, l'agent est entraîné en utilisant un échantillon d'expériences mémorisées.

3 Résultats

3.1 Performance

Pendant l'entraînement de mon agent sur 501 épisodes, j'ai pu observer une amélioration significative de la performance au fil du temps.

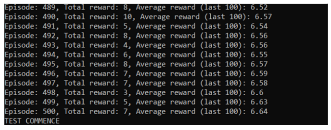

Figure 1: Processus d'entraînement

Après avoir terminé 501 épisodes, j'ai sauvegardé le modèle pour la dernière session d'entraînement de numéro 500. Et puis, j'ai chargé ce modèle et testé sa performance.(Nous pouvons commenter la boucle d'appretissage et exécuter la section de test directement)

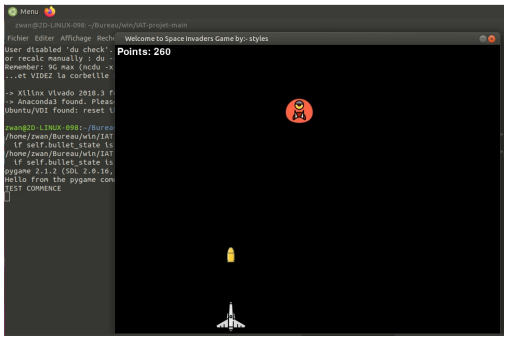

Figure 2: Processus de test

Les résultats des tests en page suivante montrent que notre modèle entraîné final peut atteindre de bonnes performances de jeu.

Figure 3: Résultat de test

3.2 Conclusion

Le politique d'action générée par notre algorithme montre que l'agent a appris une stratégie efficace pour jouer au jeu Space Invaders. Au fur et à mesure que les épisodes d'entraînement progressent, on observe que les récompenses totales augmentent, ce qui indique que l'agent devient de plus en plus compétent pour détruire les envahisseurs.

L'analyse des résultats a montré que mon agent était capable d'obtenir un score élevé dans le jeu Space Invaders. Cela confirme que l'algorithme DQN que j'ai choisi est bien adapté à ce problème et que les hyperparamètres que j'ai sélectionnés permettent d'atteindre une performance satisfaisante.

En conclusion, mon projet démontre que l'utilisation d'un algorithme d'apprentissage par renforcement, en l'occurrence DQN, est une approche efficace pour résoudre le problème du jeu Space Invaders. L'agent apprend à maximiser la somme des récompenses en exécutant les actions appropriées dans l'environnement du jeu. Les résultats obtenus sont prometteurs et montrent que cette méthode peut être appliquée avec succès à d'autres problèmes d'intelligence artificielle similaires.