Intelligence Artificielle pour les Telecommunications

Projet IAT: Space Invaders

Yassine LAHMER
Salma AZIZ-ALAOUI
Victor SULLY

1 Introduction

Le but du projet est d'utiliser un algorithme d'apprentissage par renforcement pour résoudre le problème du jeu Space Invaders. Le jeu consiste à maximiser le score obtenu par le vaisseau en détruisant les aliens envahisseurs. L'agent peut effectuer 4 actions différentes pour atteindre cet objectif. Ainsi, notre objectif est de choisir un algorithme approprié et l'intégrer au projet, en sélectionnant un échantillon d'hyperparamètres pertinents et en définissant une stratégie d'optimisation. Nous comptons aussi visualiser l'évolution de la qualité de l'apprentissage moyenne.

2 Définition de l'état

La phase de définition de l'état a été la plus importante du processus. Initialement, il était évident que tous les paramètres n'étaient pas nécessaires, donc seuls les paramètres jugés importants ont été conservés: (player_X, player_Y, invader_X, invader_Y, bullet_State)
Ensuite, il a été nécessaire de réduire la taille de l'état afin de limiter la consommation de ressources. Ainsi, la définition de l'état suivante a été adoptée: (player_X - invader_X, player_Y - invader_Y, bullet_State), avec:

- $(player\_X, invader\_X) \in A^2 \subset [0, screen\_width]^2$
- $(player\_Y, invader\_Y) \in A^2 \subset [0, screen\_height]^2$
- $bullet\_State \in \{0, 1\}$

3 Algorithme choisi: Q-learning

3.1 Explication du choix d'algorithme

Nous avons opté pour l'algorithme Q-learning pour plusieurs raisons: Tout d'abord, le jeu Space Invaders est relativement simple en termes de dynamique, avec un nombre limité d'actions et d'états discrets. Q-learning est un algorithme efficace pour ce type d'environnement où les politiques optimales peuvent être apprises de manière efficace à partir des expériences passées. De plus, Q-learning est un algorithme d'apprentissage par renforcement qui peut apprendre à maximiser les récompenses, qui dans ce cas seraient les points marqués dans le jeu. Ainsi, l'algorithme pourrait apprendre à prendre des décisions optimales pour chaque état du jeu, comme par exemple quelle cible attaquer ou quelle direction prendre pour éviter les tirs ennemis, en s'appuyant sur les récompenses immédiates et les états du jeu.

3.2 Explication de l'algorithme Q-learning

L'algorithme du Q-learning est une méthode de renforcement qui permet à un agent d'apprendre à prendre des décisions en maximisant une fonction de récompense. L'agent explore l'environnement en prenant des actions et observe les récompenses associées à ces actions. Il utilise ensuite ces informations pour mettre à jour ses valeurs Q, qui représentent la qualité d'une action dans un état donné. Les valeurs Q sont mises à jour en utilisant une règle de mise à jour de la forme:

$Q(s,a) = Q(s,a) + \alpha * (r + \gamma * max(Q(s',a')) - Q(s,a))$,

où **s** est l'état actuel, a est l'action choisie, **r** est la récompense reçue, **s'** est l'état suivant, **a'** est l'action choisie à l'état suivant, α est le taux d'apprentissage et γ est le facteur de réduction de la récompense future. L'agent choisit l'action à prendre en fonction de la valeur Q maximale pour l'état actuel. L'algorithme est itératif et l'agent continue à prendre des actions et à mettre à jour les valeurs Q jusqu'à ce que la convergence soit atteinte.

4 Hyperparamétres

Les hyperparamètres α, γ, $\epsilon_{initial}$, ϵ_{final}, max_step et n_episodes ont été évalués. Les résultats suivants ont été obtenus pour chaque paramètre :

- $\alpha = 1$: Le taux d'apprentissage a été fixé à 1 car les Q-values doivent s'adapter rapidement pendant l'apprentissage de l'agent. Cela a été confirmé en entraînant l'agent avec différentes valeurs de α .

- $\gamma = 0{,}95$: Différents facteurs de réduction γ entre 0 et 1 n'ont pas eu un impact significatif sur l'agent entraîné car les récompenses actuelles et futures étaient fortement corrélées. Par exemple, si l'agent meurt en raison de mauvaises actions ($\rightarrow$ petite récompense actuelle), les récompenses futures seront également petites car l'agent mort ne peut pas marquer de points. Cependant, les agents entraînés avec des valeurs de γ proches de 1 ont légèrement mieux performé que les agents entraînés avec un γ plus petit.

- $\epsilon_{initial} = 0{,}7$ et $\epsilon_{final} = 0{,}05$: L'évaluation de différentes paires de valeurs pour $\epsilon_{initial}$ et ϵ_{final} a montré que des valeurs proches de 1 pour $\epsilon_{initial}$ permettaient à l'agent d'explorer l'espace d'états. Des valeurs de ϵ_{final} très proches de 0 ont assuré que l'agent final ne sélectionnait pas des actions au hasard.

- max_step = 500 : Des valeurs entre 50 et 5000 ont été testées pour stepsmax. Des valeurs autour de 500 ont donné à l'agent suffisamment de temps pour progresser dans le jeu sans rester bloqué dans un coin trop longtemps.

- n_episodes = 1900 : Laisser l'agent apprendre pendant environ 19000 épisodes a donné des agents bien entraînés. Cependant, notre agent entraîné pendant 1900 épisodes a produit les meilleurs résultats.

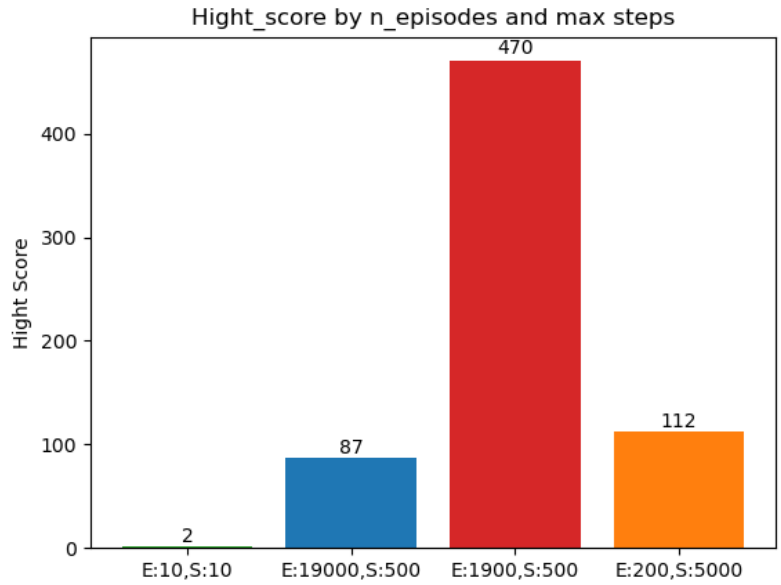

Figure 1: Hight score by the number of episodes and max steps.

5 Résultats

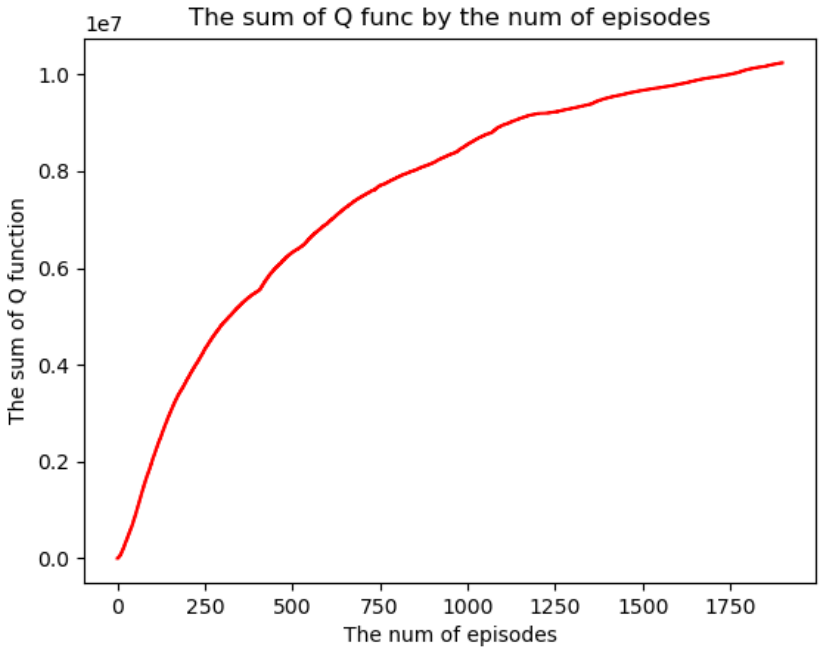

Figure 2: The sum of Q func by the num of episodes.

Le graphe ci-dessus illustre la somme des Q-valeurs sur l'axe des y en fonction du nombre d'épisodes sur l'axe des x. Au début, à l'épisode 0, la somme des Q-valeurs est observée à 0 car la table Q a été initialisée à zéro. Cependant, à mesure que le nombre d'épisodes augmente, il y a une augmentation notable de la somme des Q-valeurs. Aux alentours de 1900 épisodes, on peut observer que la foncton commance a tendre vers un plateau, indiquant que la table Q n'évolue plus ou évolue de manière très subtile.

6 Conclusion

Une amélioration possible serait d'augmenter la taille de notre espace d'état. Cependant, cela entraînerait une augmentation significative de l'espace d'état-action combiné, nécessitant plus de données pour apprendre une estimation précise de la fonction Q. Pour ce faire, nous devrions approximer la fonction Q à l'aide d'un réseau neuronal profond, appelé apprentissage profond par renforcement. Cependant, cette approche serait plus coûteuse en termes de puissance de calcul.

En revenant à notre méthode actuelle, qui est l'apprentissage Q tabulaire, nous pouvons conclure que nous avons obtenu des résultats satisfaisants car notre table Q a convergé et le jeu avec l'agent entraîné est bon.

7 Sources

- https://moodle.insa-lyon.fr/course/view.php?id=5875
- https://fr.wikipedia.org/wiki/Q-learning
- https://www.pygame.org/docs/ref/music.html#pygame.mixer.music.play
- https://github.com/blavad/IAT-correction

8 Code source

Cliquez ici pour voir le code source.