Laurie Azoulay, Jeanne Brom, Luc David, Alice Gangneux

Projet d'IAT:
SPACE INVADERS

4TC3

30 avril 2022

Table des matières

1 Approche du sujet

Le problème que nous voulions résoudre ici était le jeu Space Invaders. Notre but était de développer une IA capable de détruire le plus d'aliens possible.

Pour ce faire, il nous a fallu déterminer les états que pouvait avoir notre agent dans le jeu. Dans Space Invaders, l'agent a 4 actions possibles : aller à gauche, aller à droite, tirer et ne rien faire. Nous avons fait le choix de définir les états de notre agent par 4 champs :

— sa distance avec l'alien le plus proche de lui en x,

— sa distance avec l'alien le plus proche de lui en y,

— l'état de sa balle (tirée ou non),

— le sens de déplacement de l'alien le plus proche en x

Ces états représentent, à notre avis, les informations nécessaires et suffisantes que doit prendre en compte l'alien pour survivre. Pour limiter le nombre d'états au total, nous avons discrétisé l'espace. Ainsi, nous avons coupé l'axe des x en 16 intervalles de 50 pixels et l'axe des y en 10 intervalles de 60 pixels.

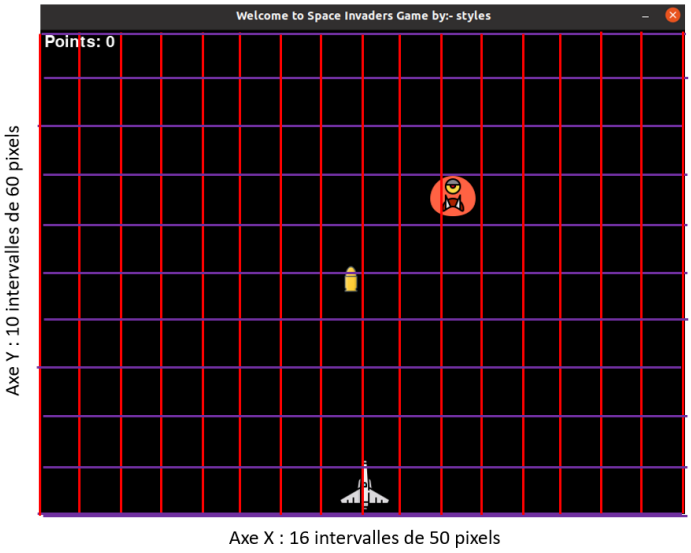

FIGURE 1 – Quadrillage de l'espace

Nous avons donc au total : 16 (intervalles x) *10 (intervalles y) *2 (tirée ou non) *2 (droite ou gauche alien) = 640 états. Et donc une matrice de Q d'une taille de 16*10*2*2*4 (actions) = 2560 qui sont nos états multipliés par le nombre d'action. Notre but premier ici était de limiter au maximum le nombre d'états à prendre en compte afin de diminuer la complexité de notre algorithme.

2 Algorithme choisi

Parmi les trois algorithmes qu'il nous était possible de choisir, nous avons décidé d'implémenter un apprentissage par renforcement grâce à la méthode **Q-Learning**. Ce choix a été principalement motivé par le fait que nous estimions que la méthode de Q-Learning était celle sur laquelle nous nous étions le plus attardé·e·s lors des séances de TP.

Comme tous les algorithmes d'apprentissage par renforcement, Q-Learning repose sur l'enchaînement des trois étapes principales : **échantillonnage, apprentissage et amélioration**.

2.1 Échantillonnage

Dans notre cas, l'échantillonnage des données d'apprentissage se faisait au cours d'une partie jouée par notre agent. A chaque action effectuée par l'agent (droite, gauche, tir ou non), l'agent va obtenir une récompense s'il a touché un alien (+1 sur le score) ou non (+0 sur le score). A chaque partie jouée, c'est le score attribué à notre agent qui constitue la valeur de retour d'un échantillon.

2.2 Apprentissage

L'apprentissage de notre algorithme est réalisée grâce à une implémentation de la fonction d'optimalité de Bellman :

$$V^{\pi} = [\sum_{t=0}^{\infty} \gamma^t.R(s_t, a_t)|s_0 = s] = [R(s, \pi(s)) + \gamma \sum_{t=0}^{\infty} \gamma^t.R(s_{t+1}, a_{t+1})|s_1 = s'] = R(s, \pi(s)) + \gamma V(s', \pi)$$

Avec γ^t le facteur de décompte, π la politique courante, V^{π} la performance de la politique, s_t un état, a_t une action, $R(s, a)$ ensemble des récompenses.

Notre algorithme cherche à estimer la fonction de valeur optimale Q*(s,a) avec une représentation tabulaire pour trouver la politique optimale. Cette fonction mesure la qualité d'une action faite par l'agent dans l'état actuel du système.

2.3 Amélioration

Une fois l'apprentissage commencé, le but est d'améliorer la politique choisi par l'agent afin de faire un score le plus élevé possible. Pour cela, il faut choisir la meilleure solution obtenue grâce à l'algorithme d'apprentissage et donc grâce à la fonction Q.

3 Mesures et analyses

Par la suite, nous avons déployé une série de tests grâce à notre algorithme de Q-Learnig adapté au jeu Space-Invaders, afin de trouver les paramètres adaptés à notre algorithme afin d'avoir une courbe d'apprentissage satisfaisante.Nous avons fait le choix de fixer le nombre de pas à 2000 car nous considérions plus important que l'agent apprenne à tuer le premier alien du jeu. Le reste du jeu devient ensuite plus ou moins la même stratégie.

Nous pouvons voir sur l'Annexe les Figures 1 à 5 qui nous montrent les courbes d'apprentissage, c'est à dire le score de l'alien avec une certaine valeur de γ (0,5 ou 0,8) sur 300 et 1200 épisodes.

Le but de l'IA est de chercher à observer la convergence de Q vers une valeur fixe le plus rapidement possible, et pour cela, on fait évoluer le facteur γ et le nombre d'épisodes pour que l'agent ait le plus gros score possible à chaque partie.

Nous avons décidé d'observer les variations de scores de notre agent au fur et à mesure que le nombre d'épisodes augmente. Au début les scores sont très faibles et varient entre 1 et 4 points avec énormément de scores nuls à chaque épisode. Quand on augmente le nombre d'épisodes, l'agent fait de moins en moins de score nul mais toujours pas de scores dépassant 5. On peut donc conclure de ces courbes que notre agent n'apprend pas ou peu, et nous ne comprenons pas ce à quoi cela est dû.

4 Conclusion

En conclusion, nous n'avons pas réussi à obtenir des scores vraiment convaincants après avoir testé plusieurs nombres d'épisodes, il faudrait sûrement en faire bien plus pour avoir un agent qui puisse apprendre vraiment.

Nous aurions sûrement pu avoir des scores encore plus élevés si notre agent était plus précis donc si notre espace était discrétisé en plus de cases mais il aurait fallu plus d'épisodes pour avoir des résultats car il y aurait eu plus d'états.

5 Annexe

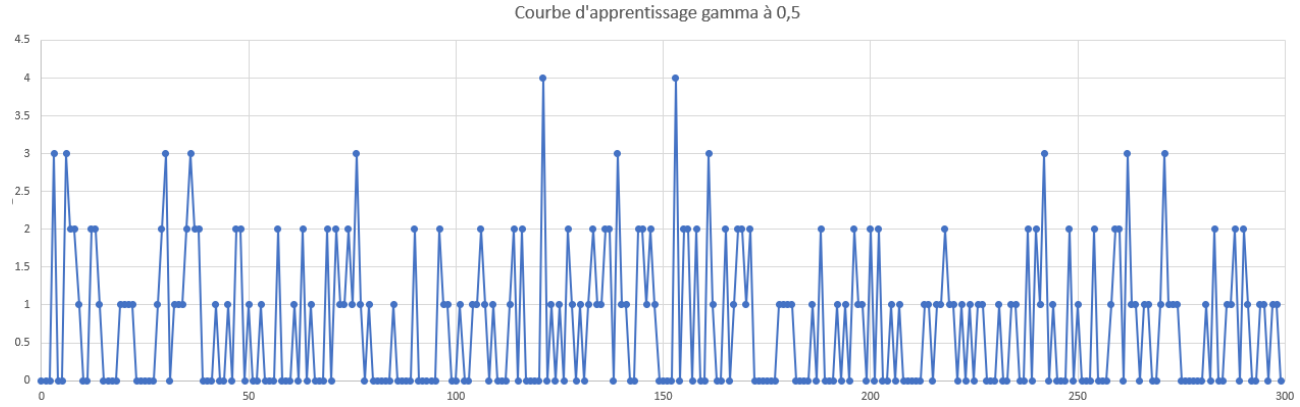

FIGURE 2 – Courbe d'apprentissage du score selon les épisodes avec un gamma de 0.5

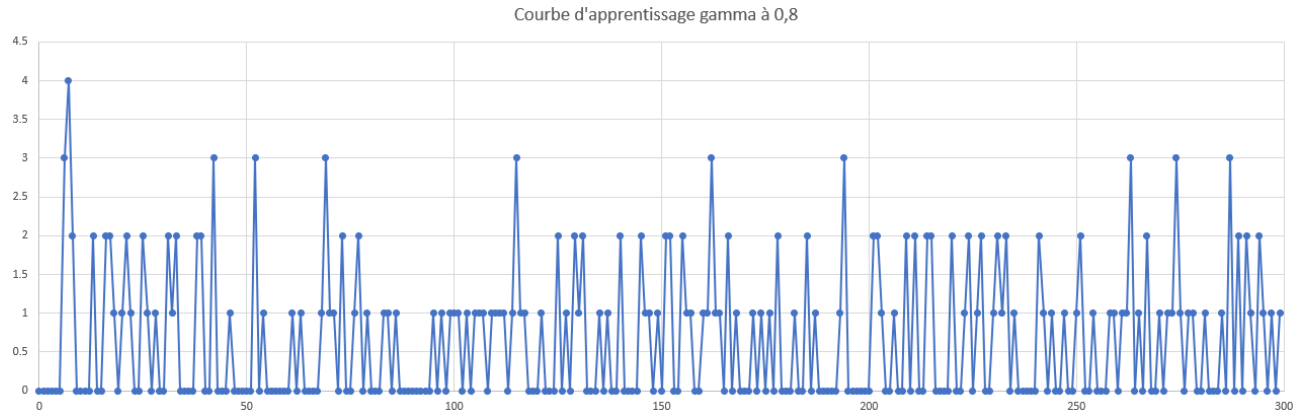

FIGURE 3 – Courbe d'apprentissage du score selon les épisodes avec un gamma de 0.8

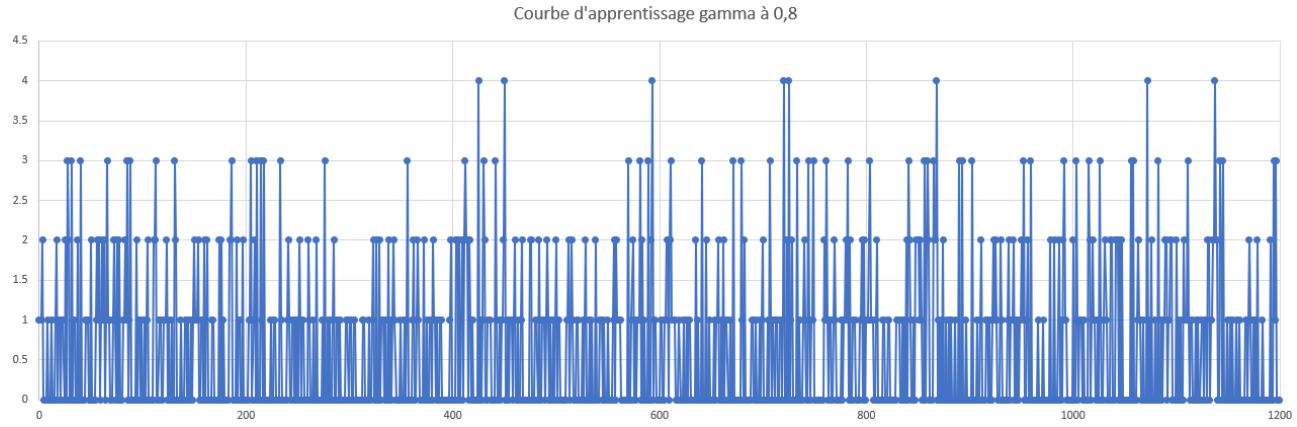

FIGURE 4 – Courbe d'apprentissage du score selon les épisodes avec un gamma de 0.5

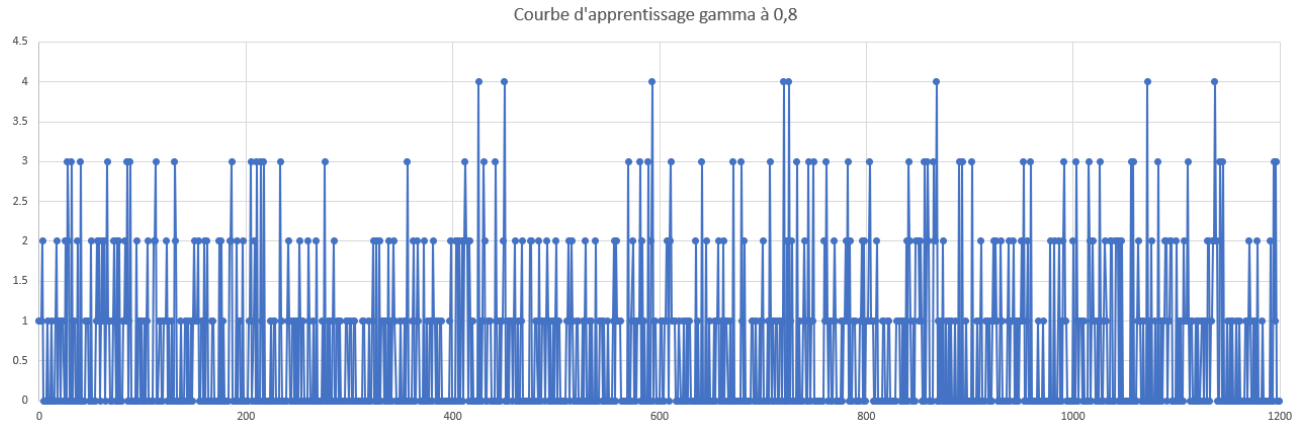

FIGURE 5 – Courbe d'apprentissage du score selon les épisodes avec un gamma de 0.8