Comment

Frrishtog, Zidrrog, Jjbri

January 2021

1 Introduction

Dans ce projet notre but est de mettre en place un algorithme de reinforcement learning efficace pour le problème car-racingV0 du package gym. Le problème car-racingV0 correspond à un jeu dans lequel une voiture doit compléter un circuit. Ce circuit est composé de tiles et à chaque fois que la voiture passe sur une tile sur laquelle elle n'est jamais passée, son score augmente. Le but est donc de parcourir un maximum de tiles différentes en un nombre fini d'actions. Les actions de la voitures sont représentées par un vecteur de taille 3: la première valeur correspond à l'angle des roue (-1 tourne à gauche, +1 tourne à droite), la deuxième valeur correspond à l'accélération souhaitée et la troisième valeur correspond au freinage appliqué. Nous allons mettre en place un algorithme DQN pour résoudre ce problème.

2 Architecture

Premièrement, on observe, dans la définitions des actions de la voitures, qu'il s'agit d'un problème continu. Or, DQN s'applique uniquement aux problèmes à actions discrètes. Notre approche a donc été de définir un action space d'une dizaine d'actions parmi lesquelles l'agent choisira sa prochaine action. Pour arriver à faire cela, nous avons redéfini une classe correspondant à un environnement car-racingV0 qui utilise ce nouvel action space.
Une fois ce problème résolu, il faut mettre en place le reste de l'algorithme, en l'occurrence le CNN permettant de faire de l'extraction de feature qui seront utilisées par l'agent DQN. Notre CNN est composé de 2 conv2D suivi de 2 couches denses. De plus la dernière couche dense possède une output size de la taille de notre action space.
Nous avons ensuite utilisé le package rl.agent.dqn pour notre algorithme de DQN. Ce package a l'avantage de laisser une grande diversité de choix pour les méthodes d'entraînement, les politiques utilisées et est très efficace. En tout, sur le meilleur modèle nous avons réalisé 150 entraînements de 9000 steps.

3 Amélioration

Notre premier modèle comportait 10 actions. Nous avions fait le choix d'en prendre peu pour augmenter la vitesse d'apprentissage et pouvoir observer de premiers résultats. Il s'est avéré que très rapidement, la voiture enchaînait des choix contradictoires : accélérer puis tout de suite freiner, tourner à gauche puis à droite. Ce problème vient en partie du petit nombre d'actions de notre action space mais aussi de l'instabilité des choix de l'agent. Pour augmenter cette stabilité, nous avons utilisé le système de répétition des actions qui permet de choisir une action qui sera utilisée plusieurs frame. Avec ce système, l'agent a commencé à apprendre un bon comportement mais il restait de nombreux problèmes dû à cette technique. En effet, la répétition d'action nuit à la précision des choix et notre petit action space n'arrange pas ce problème.
Après avoir obtenu un agent au comportement très variable et ne dépassant que rarement les 300 points de score, nous avons décidé de changer d'action space pour rajouter de la précision. Ce changement a entraîné des ajustements dans notre classe d'environnement carRacing ainsi que dans le CNN utilisé pour l'extraction de features. Ce nouvel agent montrait déjà de meilleurs résultats que le précédent pour un entraînement similaire mais il existait encore un problème. En effet, la voiture avait tendance soit à beaucoup passer dans l'herbe pour couper la route, soit à avancer très lentement et rester très clairement dans les limites du circuit. Pour palier ce problème, nous avons utilisé notre classe CarRacing pour ajuster les rewards. Nous avons mis en place un système délivrant une reward de -50 si la voiture n'a pas découvert de nouvelle tile dans les 30 dernières frame et passant l'agent dans un état terminal. Cette méthode permet d'éviter les voyages dans l'herbe, durant lesquels la voiture ne découvre aucune tiles et cette méthode a permis de faire accélérer la voiture dans le cas général.

4 Statistiques

Nous avons alors réalisé des statistiques avec ce modèle entraîné 150 fois sur 9000 steps. On observe alors la courbe de la figure 1 montrant le score obtenu pour 100 runs. On observe un comportement très variable avec de très bons scores (935 pour le maximum) et de plutôt mauvais scores (322 pour le minimum). Cependant la moyenne des runs reste de 768 ce qui est très élevé pour ce problème avec les limitations que nous impose le DQN.
On peut tout de même expliquer les mauvais résultats par un manque certain d'entraînements de la voiture dans certains cas. En effet, si la voiture se perd dans l'herbe, elle n'arrive jamais à revenir sur la piste. Cela est dû principalement à notre système d'arrêt en cas de trop longue période sans découverte de tiles. Même si pour la phase de test cette méthode n'est pas utilisée, le DQN n'a pas pu apprendre le comportement à adopter dans le cas où la voiture se trouve dans l'herbe. De ce fait, il est presque impossible que la voiture corrige sa trajectoire dans ce cas. Une amélioration pourrait donc être de faire des entraînements spécifique pour apprendre à rejoindre la route le plus rapidement,

une fois la voiture dans l'herbe.

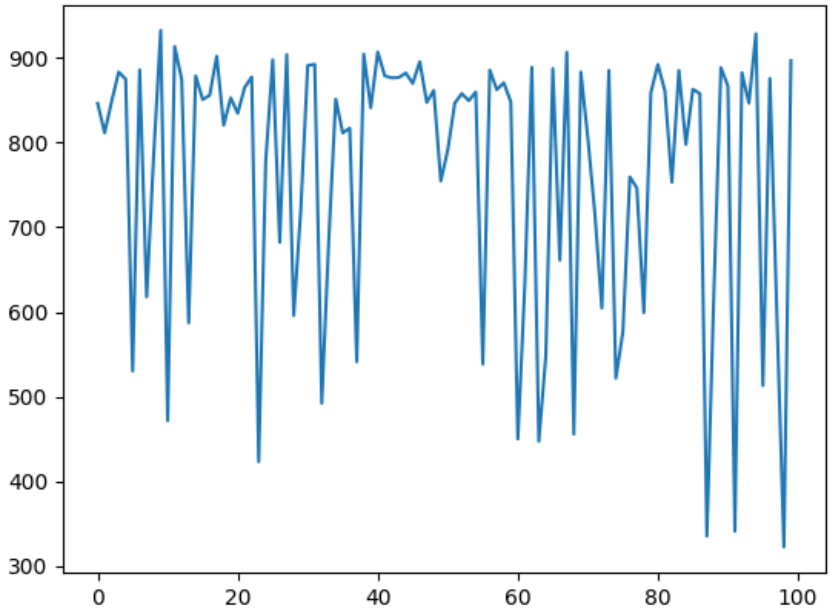

Figure 1: Scores obtenus pour 100 runs

5 Conclusion

Pour conclure, nous avons obtenu un agent plutôt efficace pour le problème carRacing. Cependant il existe encore des voies d'améliorations notamment des nouvelles phases d'entraînement spécifiques permettant d'apprendre certains comportements. De plus le DQN n'est pas la meilleure approche pour ce type de problème et nous n'avons pas pu non plus entraîner suffisamment l'agent pour obtenir des résultats optimaux.