Sorbonne Université - Paris

IA pour la Robotique
Rapport

Mini Projet - Lunar Landing

Élève :
Yannis Montreuil

Enseignant :
Olivier Sigaud

22 octobre 2021

Table des matières

1 Introduction

Ce mini projet concerne l'atterrissage d'une navette sur la lune. Le but est de faire atterrir la navette via des méthodes de renforcement par apprentissage. Nous devons alors de déterminer la politique la plus performante possible dans cet environnement. Le renforcement par apprentissage étant basé sur une notion de récompense et d'épisode, la navette gagne une récompense quand elle s'approche de la zone d'atterrissage, au contraire, quand elle perd sa récompense quand elle s'écarte de la zone. L'épisode se termine quand la navette se crash. Un épisode est réussit quand la navette parvient à atterrir sans dégât.

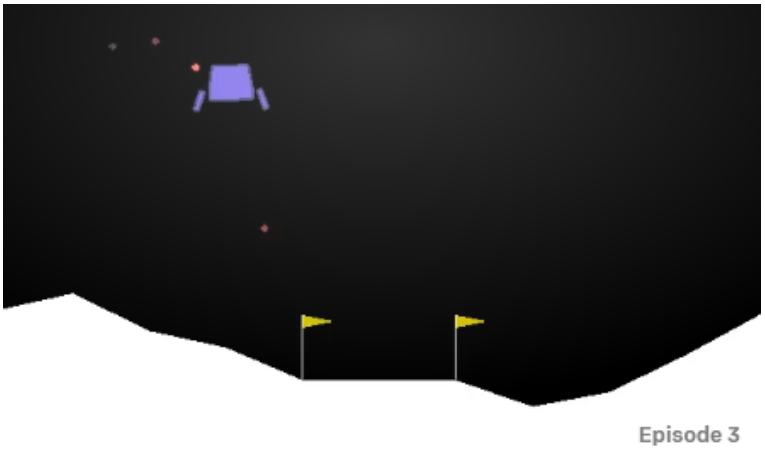

FIGURE 1 : Atterrissage de la navette

2 Obtention d'une politique

Dans un premier temps, nous décidons de tester les différentes politiques de l'environnement sans les optimiser. Nous implémentons donc les politiques nécessitant un environnement dit itératif : A2C, PPO, DQN et QR-DQN. Nous lançons alors les politiques avec les mêmes paramètres par défaut pour pouvoir les comparer sur un pied d'égalité. Nous utilisons l'environnement Tensorboard et pandas afin de visualiser les résultats de l'apprentissage plus rapidement et plus efficacement. Nous stockons alors tous les résultats dans un dossier, et nous obtenons le graphique d'apprentissage sur le reward des politiques en donnant un budget $N = 1 \times 10^6$. Nous obtenons les temps de calculs et résultats suivant :

Politiques	A2C	PPO	DQN	QR-DQN
Temps (mn)	45	42	40	210

FIGURE 2 : Apprentissage des politiques

Nous observons de grandes disparités dans les résultats malgré le budget accordé à chaque politique. Certaine semblent ne pas converger, probablement parce qu'elle requiert des hyper-paramètres optimisés. Nous retenons alors la politique *PPO* qui est la plus performante dans ce problème pour un budget donné. Il est important de bien comprendre que nous nous basons seulement sur une expérience pour jauger la performance de la politique. Ces derniers peuvent être sujet à de mauvaise performance à cause d'un manque de chance dans l'apprentissage. Il sera alors important de réaliser plusieurs simulations pour vérifier la reproductibilité d'une politique.

3 Reproductibilité

Comme expliqué dans la dernière section lors de l'obtention d'une politique, il est très important de vérifier si les résultats que nous avons obtenu sont cohérents ou non. Pour cela, nous allons réaliser plusieurs fois la simulation sur la politique *PPO* pour observer les variations de résultats que nous pouvons obtenir sur la version non optimisée.

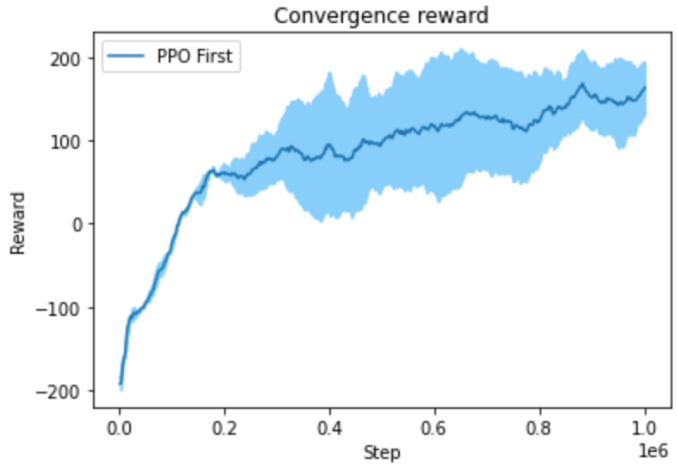

FIGURE 3 : Apprentissage PPO

Nous avons tout d'abord tracé plusieurs simulations du *PPO* en conservant les mêmes paramètres (défaut). La zone bleue clair permet de rendre en compte des allures des différentes simulations réalisées. La courbe bleue foncée correspond aux moyennes des 4 simulations. Nous pouvons donc observer que les résultats ne sont pas très convainquant étant donné qu'il existe une très grande différence entre les résultats. Ce phénomène peut être dû à la non optimisation des hyper-paramètres.

4 Optimisation d'une politique

Une fois la politique retenue, l'objectif est alors d'optimiser les hyper-paramètres afin d'améliorer les résultats de notre politique. Nous n'optimiserons pas les autres politiques dans un soucis de temps de calcul qui peut devenir très conséquent. Nous utilisons Optuna qui est une API d'optimisation d'hyper-paramètres. L'optimisation se déroule en deux étapes :

— L'étude : optimisation basée sur une fonction objectif
— L'essai : une exécution de la fonction objectif

Le but est de trouver un set de paramètre maximisant les valeurs du reward de la politique. Pour cela, nous implémentons une méthode *sample_ PPO* qui permet d'adapter notre futur optimisation en modifiant les paramètres et méthodes :

Lr	Architecture	Active function
[1e-5, 1]	["tiny", "small","medium","big"]	["tanh", "relu"]

Après convergence, nous obtenons le set d'hyper-paramètres suivant :

N step	N epoch	γ
1024	7	0.9959463232832901

La prochaine étape consiste alors à relancer notre politique et observer les résultats sur notre problème. Nous obtenons alors le graphique d'apprentissage suivant :

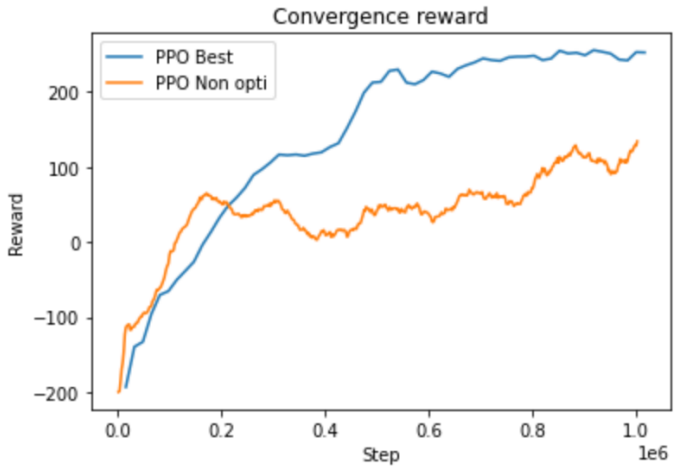

FIGURE 4 : Apprentissage PPO Best

Nous pouvons donc bien confirmer que l'optimisation que nous avions réalisé impact de manière bénéfique les résultats sur l'apprentissage de notre politique. Après convergence en moyenne, la politique converge vers une valeur de reward 4 fois supérieur. Nous pouvons, comme précédemment réaliser une étude de reproductibilité permettant de valider le modèle. Nous relançons donc la simulation et obtenons :

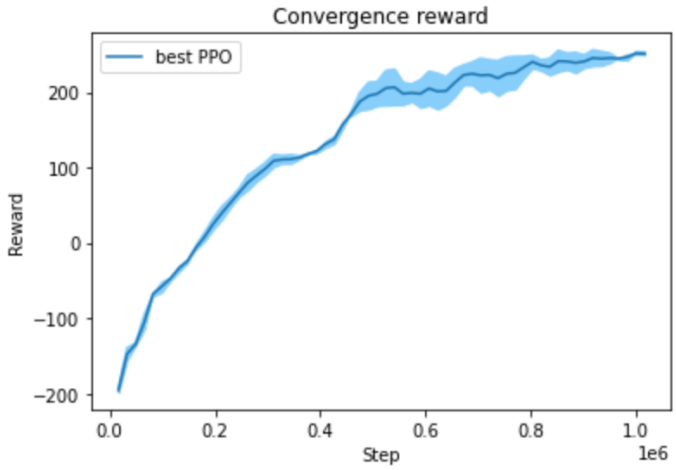

FIGURE 5 : Apprentissage PPO Best - fluctuation

Contrairement aux simulations réalisées pour la politique non optimisé, nous obtenons des résultats bien plus convainquant. En effet, la fluctuation de l'apprentissage entre plusieurs simulations semblent bien moins forte. Nous pouvons donc en déduire que l'optimisation des hyper-paramètres permet aussi de diminuer cette instabilité. Nous allons maintenant tenter d'améliorer le score obtenu.

5 Amélioration du modèle

Via les différents graphiques que nous avons obtenues, nous pouvons observer que l'apprentissage ne semble pas avoir totalement finis de converger. Pour en être certain, nous augmentons le budget jusqu'au nombre raisonnable de $N = 8 \times 10^6$. Nous observons alors les résultats et les comparons avec les autres en y ajoutant les temps d'apprentissage :

PPO	$N_0 = E6$	$N_1 = E7$	$N_2 = 4 \times E6$	$N_3 = 8 \times E6$
R_{max}	9	250	283	290
ΔT (mn)	3.35	19.25	58	107

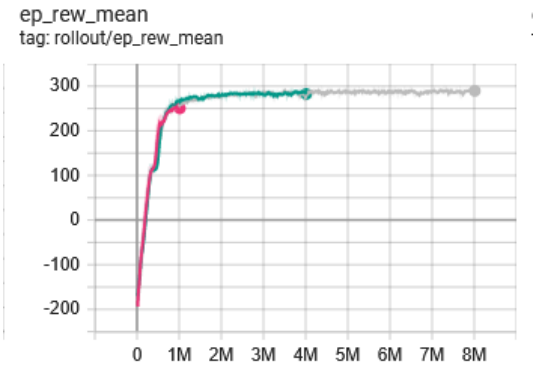

FIGURE 6 : Apprentissage PPO Opti

Nous obtenons donc de meilleurs résultats en augmentant le budget prévu pour l'apprentissage. Nous observons que la politique n'évolue plus considérablement malgré le budget apporter. Nous pouvons donc maintenir ce budget qui nous permet d'avoir une excellente valeur de reward. Nous sauvegardons donc le modèle afin de le faire tester via le script d'évaluation.

6 Conclusion

Ce projet nous a permis de prendre en main de nouvelle méthode de détermination de politique et de les appliquer sur un environnement graphique. Nous avons aussi mis en place et expliquer une démarche de recherche d'optimisation afin de clairement démontrer les résultats. Nous avons finalement obtenu des résultats extrêmement convainquant.

Références

[1] https ://github.com/osigaud/rl-baselines3-zoo/
[2] https ://sb3-contrib.readthedocs.io/en/master/
[3] https ://github.com/DLR-RM/stable-baselines3
[4] https ://www.tensorflow.org/tensorboard/dataframe
[5] https ://gist.github.com/ptschandl/ef67bbaa93ec67aba2cab0a7af47700b