Reinforcement Learning

Compte rendu config 3

Github : https://github.com/BouaitaRayane/RL-highway/tree/main

Introduction

Dans cette configuration, nous tentons d'entraîner un agent pour résoudre l'environnement « roundabout-v0 ». Sa configuration est la suivante :

- Observation Kinematics : observation vectorielle des véhicules autour (contrairement à OccupancyGrid, GrayScaleObservation, ...)
- Actions discrètes

L'objectif est de rejoindre l'autre côté du rond-point en évitant toute collisions avec les autres véhicules.

Algorithme utilisé

Dans cette tâche, nous avons utilisé la librairie Stable-Baselines3, qui permet de s'abstraire de l'implémentation détaillée des modèles, des politiques et des processus d'entraînement, tout en facilitant les expérimentations.
Notre premier choix s'est porté sur l'algorithme A2C, avec une configuration d'hyperparamètres relativement standard. Ce choix est motivé par la nature de l'environnement « roundabout-v0 », jugé complexe en raison du nombre de véhicules et de leurs comportements aléatoires. A2C repose sur une politique stochastique, ce qui le rend plus souple face à l'incertitude et à la variabilité de l'environnement, contrairement à DQN qui applique une politique déterministe via la Q-fonction et une exploration ε-greedy, souvent moins adaptée à des contextes dynamiques.
Par ailleurs, A2C présente un bon compromis entre performance et temps d'entraînement, en comparaison avec PPO, ce qui en fait une solution pertinente pour établir un benchmark de référence dans cette première phase d'évaluation.

Algorithme A2C

Premier run

Configuration du modèle :

- **Learning rate :** 3e-4
- **Gamma :** 0.99
- **N_steps :** 8
- **vf_coef :** 0.5 *(Poids de la loss du critic (valeur) dans la fonction de coût globale pour régler l'équilibre entre : optimisation de la politique (actor) et qualité de la prédiction de valeur (critic))*.

- **ent_coef** : 0.01 *(Coefficient pour l'entropie de la politique influant sur le degré de diversité dans les actions. Cela encourage l'exploration.*
- **policy_kwargs** : 2 couches de 128 neurones *(personnalisation de l'architecture des réseaux de l'actor et critic).*

Modifications de l'environnement :

- **simulation frequency** : 20
- **duration** : 12
- **policy frequency** : 10
- **total_timesteps** : 200 000 ~ 833.33 épisodes
    - épisodes = total_timesteps / (simulation_freq * duration)

Analyse des résultats

Quantitative :

Le mean reward et men_ep_lenght sont stables et relativement haut laissant penser que le modèle a bien appris.

En revanche :

- l'entropy_loss : diminue progressivement vers -1.6 (l'agent explore moins les actions) mais reste très instable.

- policy_loss : l'allure en pics révèle des mises à jour brutes de politique, ce qui pourrait suggérer que le learning rate est trop élevé.
- value_loss : on observe également des pics suggérant que le critic est trop lent à s'adapter.

Qualitative :
A la suite de l'entraînement, nous avons visualisé 10 simulations. Le modèle a simplement appris à rouler continuellement jusqu'à son point d'arrivé, n'adaptant en aucun cas son choix d'action par rapport à l'environnement. Aucun comportement de freinage ou d'accélération n'a été observé, l'agent suit le même pattern à chaque épisode.

Second run

Etant donné que la durée d'entraînement est très longue, nous n'avons pas envisagé d'augmenter le nombre d'épisodes ou de diminuer le learning rate. L'approche choisie a donc été de tenter de générer plus de collisions durant l'entraînement en modifiant l'environnement de la manière suivante :

- Tentative **d'augmentation du nombre de véhicule** : nous ne sommes pas parvenu à le modifier correctement dans le fichier config
- Adoption d'une **conduite agressive** pour les autres véhicules : avec *"other_vehicles_type" : "highway_env.vehicle.behavior.AggressiveVehicle",* afin de générer plus de collisions lors de l'entraînement

Bien que l'on observait une légère différence dans le comportement des autres véhicules, l'agent ne montrait rien de nouveau d'après les simulations, et les métriques d'évaluation étaient relativement similaires.

Troisième run

Cette fois-ci, nous avons essayé de paramétrer les rewards de sorte à forcer certains comportements.

```
"collision_reward": -20,
"high_speed_reward": 2,
"right_lane_reward": 0,
"lane_change_reward": 2
```

Une nouvelle fois, les différents graphiques (evaluations et loss) n'avaient pas vraiment changés comparé au premier run en termes d'allure, et de comportement.
Cependant la visualisation nous a permis de voir de nouveaux comportements :

- prendre la voie intérieure du rond point, réduisant ainsi sa distance totale
- rouler constamment plus vite

En revanche aucune adaptation de l'agent n'a encore été observé à l'approche d'autres véhicules, le même pattern est réalisé à toutes les tentatives.

Algorithme DQN

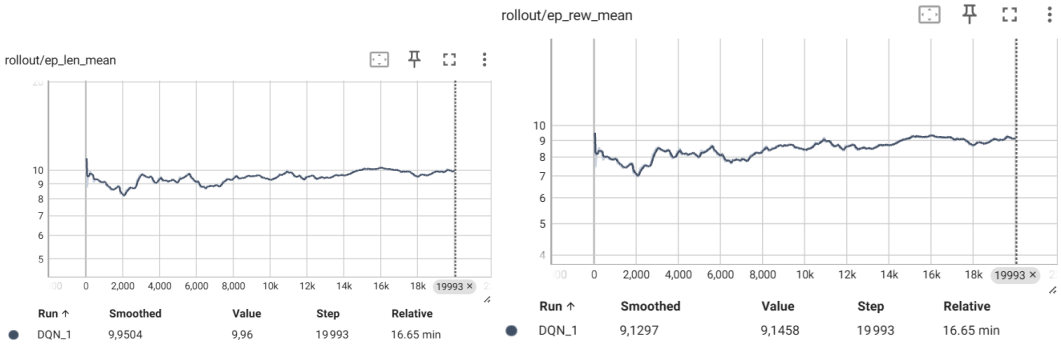
Au regard des résultats précédents, nous avons émis l'hypothèse que le nombre d'épisodes durant l'entraînement pouvait être insuffisant pour permettre une convergence efficace. Dans cette optique, nous avons choisi d'entraîner un agent DQN, en partant du principe qu'il serait potentiellement moins performant que d'autres approches, mais qu'il permettrait d'être entraîné sur un nombre d'épisodes nettement plus élevé à durée d'entraînement équivalente.

Ses paramètres étaient les suivants :

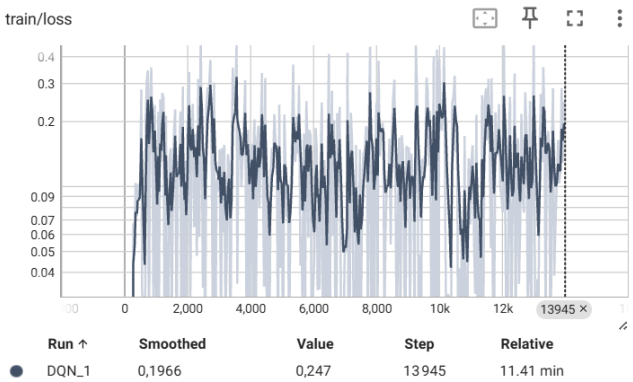
- **policy_kwargs** : *cette fois-ci deux couches cachées de 256 neurones chacune*
- **learning_rate** : 5e-4
- **buffer_size** : 15000 *mémoire dans laquelle les transitions sont stockées*
- **learning_starts** : 200
- **batch_size** : 32
- **gamma** : 0.8
- **train_freq** : 1
- **gradient_steps** : 1
- **target_update_interval** : 50
- **exploration_fraction** : 0.7 *fraction de l'apprentissage pendant laquelle l'exploration ε-greedy est active. L'agent passe donc plus de temps à explorer*

Analyse des résultats

Quantitative :



On observe que ep_len_mean et ep_rew_mean sont très stables et relativement élevé.



Il en est de même pour la loss qui est relativement basse et semble avoir convergé.

En ce qui concerne la visualisation, sur un total de 10 épisodes, l'agent n'est entré en collision avec un autre véhicule qu'à une seule reprise. Par ailleurs, les vidéos montrent qu'il est capable d'adapter son comportement de manière pertinente, que ce soit lors des insertions, de la circulation sur une même voie ou encore en sortie de rond-point.

Conclusion

Dans cette expérimentation, nous avons cherché à entraîner un agent à naviguer dans l'environnement « roundabout-v0 », avec pour objectif d'éviter les collisions tout en atteignant sa destination. L'algorithme A2C a d'abord été testé, offrant une bonne stabilité des métriques globales, mais un comportement limité et peu réactif aux autres véhicules, malgré plusieurs ajustements (récompenses, agressivité des autres véhicules, etc.).
Face à ces limites, un DQN a été entraîné avec l'hypothèse qu'un plus grand nombre d'épisodes améliorerait l'apprentissage. Cette approche s'est révélée pertinente : l'agent a montré une meilleure capacité d'adaptation, tout en conservant un faible taux de collision.
En résumé, A2C offre une base robuste mais nécessite un entraînement plus long pour exprimer son plein potentiel, tandis que DQN permet une convergence plus rapide avec des résultats satisfaisants dans cet environnement. Des pistes d'amélioration incluent un meilleur tuning des récompenses et des hyperparamètres du modèle pour tendre davantage vers un comportement idéal de l'agent.