Introduction aux bibliothèques Gym et Stable-Baselines pour l'apprentissage par renforcement

Culture Sciences de l'Ingénieur

Guénolé CHÉROT – Augustin GODINOT

Édité le 21/07/2022

école normale supérieure paris-saclay

Cette ressource est issue d'une co-publication avec le numéro 109 de La Revue 3EI de juillet 2022 et fait partie du « Dossier Intelligence Artificielle » [5] sur Culture Sciences de l'Ingénieur. Guénolé Chérot est doctorant en deuxième année à l'ENS Rennes, rattaché au Laboratoire SATIE et Augustin Godinot est élève au DER Nikola Tesla de l'ENS Paris-Saclay.

Cette ressource présente les très populaires bibliothèques Gym et Stable-Baselines dédiées à l'apprentissage par renforcement. Il s'appuie sur une séance de travaux pratiques d'asservissement d'un pendule inversé.

Cette séance de travaux pratiques est aussi l'occasion de présenter le pédagogique outil jupyter notebook.

1 – Faire avancer la science reproductible

Malgré la simplicité apparente de la formulation d'un problème d'apprentissage par renforcement (Processus de Décision Markovien, stationnaire), l'analyse théorique de ses performances sur un système particulier reste compliquée, voire impossible. La validation des algorithmes repose donc majoritairement sur l'évaluation de leurs performances expérimentales. Or, une validation expérimentale rigoureuse nécessite une attention particulière. L'expérience doit-être :

- Contrôlée : Le problème à résoudre doit être clairement défini.
- Répétable : Les résultats obtenus doivent être réplicables par d'autre scientifique de manière indépendante.
- Statistiquement significative : Les algorithmes étant de nature stochastique, un grand nombre d'expérience identique doit être mené pour limiter les erreurs d'interprétation.

De nombreux efforts sont déployés dans la communauté Machine Learning pour faire face à la crise de la reproductibilité [1]. Par exemple, la société OpenAI a créé en 2016 **Gym**[1] [2]. Cette librairie open-source implémente une grande diversité d'environnements (pendule inversé, jeux Atari, robots...) avec une interface unifiée. Pour faciliter encore d'avantage la comparaison d'algorithme, **Stable-Baselines**[2] [4] fourni une implémentation open-source de l'ensemble des algorithmes à l'état de l'art.

Prenons maintenant un exemple pratique de l'utilisation de ces deux outils : si un chercheur propose un nouvel algorithme, il va commencer par le tester sur l'ensemble des environnements Gym. Ces expériences, lui donneront des scores (somme des récompenses obtenu sur un épisode) sur chacun des environnements.

[1] https://www.gymlibrary.ml/
[2] https://stable-baselines3.readthedocs.io/en/master/

Il pourra ensuite comparer ces scores avec ceux obtenus par les algorithmes de l'état de l'art grâce à la librairie **Stable-Baselines**. Cette procédure est illustrée **Erreur ! Source du renvoi introuvable.**.

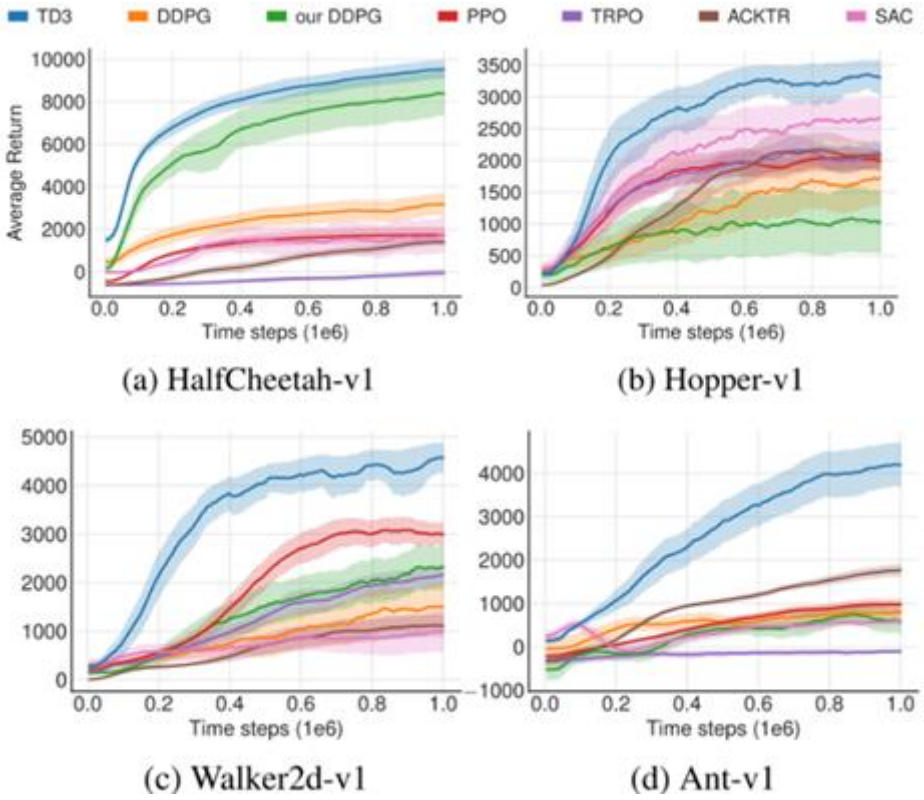

Figure 1 : Comparaison de sept algorithmes d'apprentissage par renforcement sur quatre environnements Gym [3]

Cette ressource présentera les librairies Gym et StableBaselines à travers une séance de travaux pratiques d'asservissement d'un pendule inversé. Aucune connaissance préalable de ces librairies n'est nécessaire, cependant, il est préférable d'avoir lu la ressource « *Introduction à l'apprentissage par renforcement* » [6], introduisant les principes généraux de l'apprentissage par renforcement. Une bonne maitrise du langage de programmation Python est également conseillée.

2 – Gym et Stable-Baselines pour l'enseignement de l'apprentissage par renforcement

La puissance de Gym et Stable-Baselines réside dans leur simplicité d'utilisation, plus précisément, la simplicité de leur interface. Dans une situation classique d'apprentissage par renforcement, à chaque instant t, un agent effectue une observation $o_t \in$ O de l'état $x_t \in X$ de son environnement. Suivant la valeur de son observation, il choisit d'effectuer une action $a_t \in A$. L'environnement récompense alors l'agent d'une valeur $r_t \in R$ dépendant de son action et de son état interne x_t.

2.1 - Introduction sur un exemple

Regardons, pas à pas, comment simuler cette situation avec Gym et comment y entraîner un agent avec Stable-Baselines. On s'intéresse au pendule inversé Cartpole, l'un des environnements les plus simples proposés par Gym.

Figure 2 : Environnement Cartpole de la bibliothèque Gym

On choisit l'agent PPO (proximal policy optimization) de Stable-Baselines, l'un des plus populaires et génériques. Le code complet est extrêmement court, les bibliothèques étant de très haut niveau.

Les variables (o_t, a_t, r_t) de la formulation mathématique du problème sont facilement identifiable dans le code ((o_t, a_t, r_t) ≡ (obs, action, reward)), tout comme leur dépendance. En pratique, un agent n'a jamais une connaissance parfaite de son environnement, l'état x_t de celui-ci n'apparaît donc pas parmi les variables manipulables dans le code.

Chaque algorithme de Stable-Baseline est une classe Python qui implémente une méthode `predict(observation)` prenant une observation en entrée et retournant une action, c'est donc un accès direct à la politique π de l'agent. En plus de proposer un ensemble d'agents (i.e. algorithmes retournant une action pour une observation) implémentés, Stable-Baseline se charge aussi de les entraîner via la fonction `model.learn()` de la ligne 17.

```
 9 from stable_baselines3 import PPO
10 import gym
11
12 # Création de l'environnement
13 env = gym.make("CartPole-v1")
14
15 #Lancement de l'apprentissage avec l'algorithme PPO
16 model = PPO(policy = "MlpPolicy",env =  env, verbose=1)
17 model.learn(total_timesteps=25000)
18
19 # (ré-)initialisation de l'environnement
20 obs = env.reset()
21
22 # Simule le rejeu 1000 pas de temps dans l'environnement
23 for i in range(1000):
24     # Selection de l'action à effectuer
25     action, _state = model.predict(obs)
26     # Appliquer l'action
27     obs, reward, done, info = env.step(action)
28     # Afficher
29     env.render()
30     # Si la simulation est terminé,
31     # on remet le pendule au centre
32     if done:
33       obs = env.reset()
```

Figure 3 : Code complet de l'apprentissage de l'asservissement d'un pendule inversé avec Gym et Stable-Baselines

2.2 - TP sur JupyterNotebook

L'exercice proposé utilise aussi le pendule inversé nommé Cartpole pour mettre en œuvre un apprentissage par renforcement avec la bibliothèque Stable-Baselines.

Il utilise un jupyter notebook hébergé par googlecolab[3] pour alterner code et explications :

[3] https://colab.research.google.com/drive/1AZh_FSSImncpOuy_vdGbewK6OyqksAme

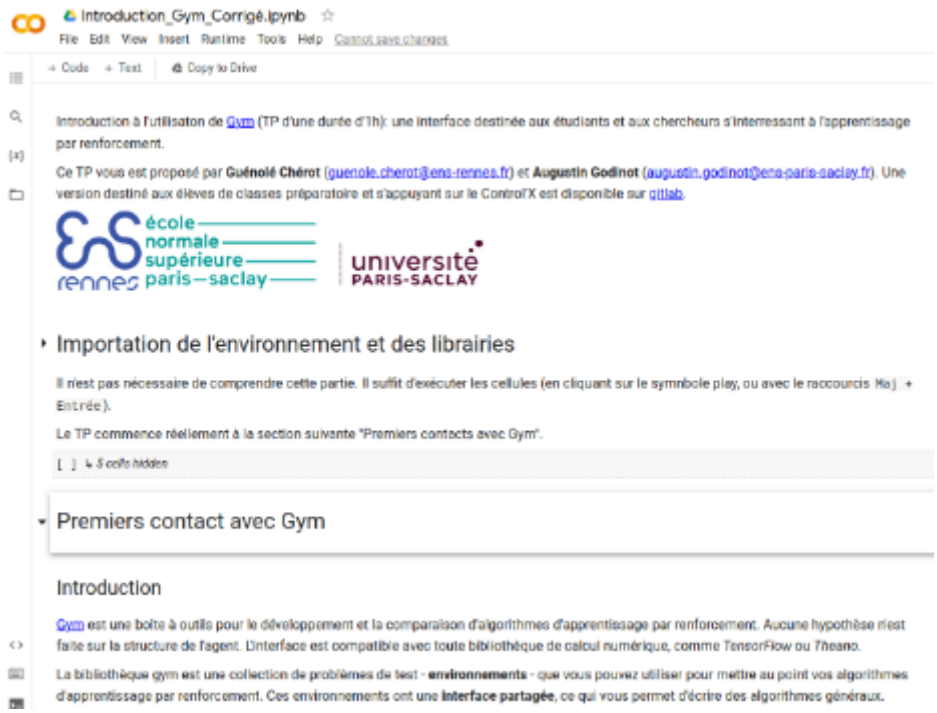

Figure 4 : Page d'accueil de la séance de travaux pratiques pendule inversé avec Gym et Stable-Baselines

La consultation du site montrera l'intérêt de l'outil jupyter Notebook pour la pédagogie. Les lignes de codes sont exécutées bloc par bloc, les explications accompagnent chaque bloc. Il est possible de demander aux étudiants de compléter certains blocs.

3 – Conclusion

Grâce aux bibliothèques Gym et Stable-Baselines, il est possible de simuler un environnement et un agent en une dizaine de lignes de Python. Les variables manipulées par les implémentations reflètent la modélisation mathématique du problème, ce qui facilite la prise en main et l'apprentissage par les étudiants. En bref, Gym fournit un ensemble de problèmes à résoudre pour comparer les agents sur une même base. Stable-Baselines fournit un ensemble d'agents fonctionnels à utiliser en enseignement et auxquels se comparer lors de la conception de nouveaux agents.

Il est également possible d'utiliser avec Gym et Stable-Baselines son propre environnement pour travailler sur un système original. La ressource « *Apprentissage par renforcement de la conduite d'un véhicule sur AirSim* » [7] présente une utilisation de Gym et Stable-Baselines pour de l'apprentissage par renforcement avec le simulateur de voitures autonomes réaliste AirSim.

Références :

[1] Baker, M. (2016). Reproducibility crisis. *Nature*, *533*(26), 353-66.

[2] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

[3] Fujimoto, S., Hoof, H., & Meger, D. (2018, July). Addressing function approximation error in actor-critic methods. In *International conference on machine learning* (pp. 1587-1596). PMLR.

[4] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*

[5] Dossier Intelligence Artificielle, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/dossier-intelligence-artificielle

[6] Introduction à l'apprentissage par renforcement, A. Juton, V. Noël, Rida Lali, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/introduction-a-lapprentissage-par-renforcement

[7] Apprentissage par renforcement de la conduite d'un véhicule sur AirSim, L. De Matteïs, S. Radosavljevic, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/apprentissage-par-renforcement-dela-conduite-dun-vehicule-sur-airsim