Cahier des charges
Etude de Soft Actor-Critic à actions discrètes

Simon Groc et Yassin Lahbib

Mars 2025

1 Contexte du problème

Nous sommes dans le cadre d'un projet de développement et d'évaluation de modèles d'apprentissage par renforcement au sein de Sorbonne Université en master AI2D. Soft Actor-Critic (SAC) est reconnu comme un algorithme de référence en apprentissage par renforcement, principalement destiné aux environnements à actions continues. Une version adaptée aux actions discrètes a également été développée, bien que son utilisation et son étude soient moins répandues. Par ailleurs, certains travaux soulèvent des questions intéressantes sur cet algorithme et proposent diverses variantes notamment:

- **Christodoulou** propose une première adaptation directe de SAC aux actions discrètes, montrant que l'algorithme reste performant sans modifications majeures des hyperparamètres sur des environnements comme Atari . Christodoulou (2019)
- **Neo et Chen** introduisent DSAC-C, une variante de SAC discrète intégrant des contraintes statistiques pour une meilleure robustesse en généralisation. Neo et Chen (2023)

2 Objectif du projet

Dans ce contexte, l'objectif consiste à explorer et à comparer ces différentes variantes de SAC adaptées aux actions discrètes. Le projet vise à intégrer ces variantes dans la librairie pédagogique BBRL (BlackBoard Reinforcement Learning), développée pour faciliter l'enseignement de l'apprentissage par renforcement. Plus précisément, il s'agira de :

- **Implémenter :** Coder les différentes versions de SAC à actions discrètes au sein de BBRL.
- **Comparer :** Évaluer les performances de ces variantes entre elles et par rapport à DQN, qui est un algorithme de référence.

- **Analyser :** Étudier en profondeur les propriétés et comportements spécifiques à chacune des variantes afin d'identifier leurs forces et limites.

3 Contraintes

Ce projet intégrant de nombreuses notions nouvelles, une formation initiale a été dispensée portant sur l'apprentissage par renforcement. Nous avons également pris en main la librairie BBRL grâce à des notebooks dédiés, ce qui nous a permis d'en comprendre le fonctionnement et d'explorer divers algorithmes de RL, notamment DQN, DDPG et TD3.

4 Cadre technique

Pour ce faire, nous allons utiliser Python, un langage simple d'utilisation qui permet un développement rapide grâce à certaines de ses bibliothèques optimisées pour l'apprentissage par renforcement.

La bibliothèque principale utilisée s'appelle BBRL. L'idée générale de cette bibliothèque est de tout représenter sous forme d'agents interagissant dans un espace de travail, une approche similaire à celle du tableau noir dans les systèmes multi-agents (d'où le nom de la bibliothèque BlackBoard Reinforcement Learning)).

La bibliothèque BBRL a été dérivée de SaLinA, un modèle qui permet de coder efficacement des algorithmes séquentiels. Quant à BBRL, elle a été conçue pour permettre l'enseignement de plusieurs concepts d'apprentissage par renforcement. Cependant, grâce à de nombreuses mises à jour, elle peut maintenant être considérée comme une bibliothèque indépendante, bien que son code hérite toujours de SaLinA.

De plus, nous utilisons la bibliothèque PyTorch, qui offre de nombreuses fonctionnalités très utiles pour BBRL. En effet, PyTorch est une bibliothèque avancée qui permet d'utiliser efficacement des tenseurs et propose un calcul différentiel automatique (autograd). Cela nous permet de faciliter la rétropropagation des gradients dans nos réseaux multicouches.

Avec les outils évoqués précédemment, nous devons donc commencer par implémenter les algorithmes suivants pour prendre en main la bibliothèque BBRL :

- **DQN (Deep Q-Network)** : est un algorithme d'apprentissage par renforcement qui permet à un agent de prendre des décisions dans un environnement pour maximiser une récompense. Il utilise un réseau de neurones pour estimer une fonction appelée "Q-value", qui prédit la qualité (ou l'utilité) d'une action dans un état donné. L'agent sélectionne ensuite les actions les plus prometteuses en se basant sur ces prédictions, tout en explorant de nouvelles options de temps en temps. Grâce à une mémoire appelée "replay buffer", il réutilise des expériences passées pour améliorer

l'efficacité de l'entraînement. En bref, DQN combine des réseaux de neurones et des techniques d'apprentissage par renforcement pour permettre à des agents de résoudre des problèmes complexes.

- **DDPG (Deep Deterministic Policy Gradient)** : est un algorithme d'apprentissage par renforcement conçu pour des environnements continus, où les actions ne sont pas discrètes mais peuvent prendre des valeurs dans un intervalle. Il combine deux réseaux de neurones : un pour la "politique" (qui décide des actions) et un autre pour estimer la qualité des actions (le critique). L'agent apprend à améliorer sa politique grâce à des gradients obtenus à partir du critique, en s'assurant de toujours maximiser les récompenses. Comme DQN, il utilise un replay buffer pour réutiliser les expériences passées et une technique appelée target network pour stabiliser l'entraînement. En résumé, DDPG est parfait pour des tâches complexes avec des actions continues, comme contrôler des robots ou piloter des véhicules.
- **TD3 (Twin Delayed Deep Deterministic Policy Gradient)** : Une amélioration de DDPG qui, au lieu de converger avec un seul critique, en utilise deux, ce qui permet de ne pas surestimer les Q-valeurs. On prend donc le minimum des deux critiques pour stabiliser l'apprentissage et permettre une convergence plus rapide dans des problèmes complexes.
- **SAC (Soft Actor-Critic)** : Un algorithme avancé de renforcement profond qui combine une exploration efficace et une stabilité d'apprentissage, avec une politique stochastique, tout en maximisant une version "adoucie" de la récompense cumulative future. Il a été conçu pour résoudre les problèmes liés aux politiques stochastiques dans des environnements continus et complexes. Il a la particularité de maximiser à la fois la récompense cumulative future et l'entropie, de sorte que la politique choisie ait toujours un maximum de chances d'explorer suffisamment pour trouver une meilleure politique.

La réalisation de ces algorithmes se fait au sein de notebooks Python conçus par Monsieur Olivier Sigaud. De plus, nous disposons également de vidéos expliquant le fonctionnement théorique de ces algorithmes, ainsi que d'une bibliographie d'articles scientifiques dédiée à chacun d'entre eux.

5 Protocole Expérimental

Après avoir implémenté ces modèles d'apprentissage par renforcement, il nous est demandé de développer deux versions discrètes de SAC. Dans la première version, le modèle produit en sortie une Q-valeur pour chacune des actions possibles. Ainsi, le modèle aura autant de valeurs en sortie qu'il y a d'actions possibles (comme pour DQN). Dans la seconde version, le modèle renvoie une seule Q-valeur en sortie, correspondant à une action spécifique dans un état donné. Dans ce cas, le modèle prend à la fois l'état et l'action en entrée.

Il nous faut ensuite comparer ces deux algorithmes entre eux ainsi qu'à DQN. Pour cela, nous partirons de l'hypothèse initiale selon laquelle ces algorithmes ont une complexité identique. L'objectif de ce projet est de déterminer si cette hypothèse est vraie ou fausse. Si elle est fausse, nous chercherons à comprendre les raisons pour lesquelles cette hypothèse n'est pas valide. De cette façon, nous pourrons établir un classement expérimental formel concernant la complexité de ces algorithmes.

Pour cela, nous allons entraîner les trois algorithmes sur un ensemble d'environnements à actions discrètes. Pour chacun de ces environnements, nous utiliserons plusieurs seeds différentes, sur lesquelles nous entraînerons tous les algorithmes à travers tous les environnements.

Nous allons ensuite collecter et stocker toutes les informations relatives à l'apprentissage et aux performances des trois modèles. Ces données seront traitées afin de lisser les résultats, notamment par le calcul de moyennes sur plusieurs exécutions indépendantes. Nous réaliserons ensuite des visualisations en nous appuyant sur les méthodes abordées lors du cours de la semaine 3 (Stats and Tuning dans AROB), centré sur l'évaluation rigoureuse des algorithmes en apprentissage par renforcement. Ce cours mettait en avant l'importance de répéter les expériences (via différents seeds), d'analyser statistiquement les performances, et de produire des graphiques qui reflètent à la fois les performances moyennes et la variabilité des résultats, en atténuant par exemple l'influence des valeurs extrêmes. L'objectif est d'obtenir des représentations visuelles claires et pertinentes pour comparer efficacement les différents apprentissages.

Enfin, nous réaliserons des analyses statistiques pour comparer, deux à deux, la convergence de l'apprentissage des algorithmes, ainsi que leur stabilité et leurs performances finales. Notre hypothèse sera validée si aucun des résultats statistiques ne montre de différence significative entre deux algorithmes.

Les environnements sur lesquels nous allons effectuer nos tests sont les suivants :

- **CartPole** : L'objectif est de maintenir un poteau (pole) en équilibre sur un chariot se déplaçant sur un axe horizontal.
- **MountainCar** : Le but est de contrôler une voiture afin qu'elle atteigne le sommet d'une colline. La voiture n'étant pas assez puissante pour y parvenir directement, elle doit prendre de l'élan en oscillant entre deux collines.
- **Acrobot** : Contrôler un robot à double pendule fixé à une base pour amener l'extrémité du pendule au-dessus d'une certaine hauteur le plus rapidement possible.
- **FrozenLake** : Déplacer un agent sur une grille pour atteindre un objectif sans tomber dans des trous (lac gelé). La grille est entièrement discrète.

- **Taxi** : Un taxi doit récupérer un passager et le déposer à un emplacement cible sur une grille.

6 Méthodologie d'évaluation

Il est maintenant nécessaire de définir les méthodes qui nous permettront d'évaluer les modèles.

Courbes d'apprentissage : Les courbes d'apprentissage permettent d'observer la vitesse d'amélioration du modèle à travers différents indicateurs, tels que l'évolution des récompenses obtenues, mais également via l'analyse des pertes du critique (critic loss) et de l'acteur (actor loss).

Stabilité : Nous devrons aussi évaluer la stabilité en analysant, à partir des 13 runs, la variance des performances tout au long de l'exécution des différents algorithmes, afin de déterminer leur robustesse face à l'aléa des initialisations et de l'exploration. Nous avons choisi 13 seeds car cela représente un bon compromis entre le temps de calcul et la fiabilité statistique.

Test statistique : Pour faire nos tests statistiques, nous allons partir de nos hypothèses initiales, qui sont que les trois algorithmes sont égaux, et pour les vérifier, nous allons utiliser le test de Welch, car il ne prend pas l'hypothèse que les variences sont égales. Nous considérerons un seuil de significativité de (p¡0,05).

7 Organisation du projet

L'organisation des tâches à réaliser dans le projet est la suivante :

Partie 1 - Janvier : Prendre en main la bibliothèque BBRL, comprendre son fonctionnement et la manière dont on l'utilise pour faire du renforcement de l'apprentissage (RL). Comprendre le fonctionnement des algorithmes à actions discrètes DQN et DDQN, puis les implémenter en BBRL.

Partie 2 - Février : Au cours de ce mois, nous allons comprendre comment fonctionne l'algorithme à actions continues DDPG, qui utilise un acteur et un critique, et l'implémenter en BBRL. De même pour l'algorithme TD3, qui est similaire à DDPG, mais avec l'utilisation d'acteurs et de critiques pour éviter la surestimation des valeurs Q.

Partie 3 - Mars : Au cours de ce mois, nous allons commencer à comprendre le fonctionnement de l'algorithme SAC et l'implémenter en actions continues. Cela nous permettra de comprendre l'idée derrière cet algorithme et comment l'appliquer dans le cas d'actions discrètes.

Partie 4 - Avril : Maintenant que nous avons vu les prérequis, nous allons comprendre et implémenter les deux formes de DSAC : celle où le critique a une sortie par action et prend un état en entrée, et celle où le critique prend un

état et une action en entrée et rend une valeur Q de cette paire d'état-action en sortie.

Partie 5 - Mai : Durant ce dernier mois, nous allons devoir faire une analyse sur l'apprentissage des différents algorithmes et faire une comparaison statistique avec le test de Welch.

8 Critères de réussite

La réussite complète de ce projet repose sur les trois critères suivants :

- **Algorithme DSAC** : Nous devons implémenter les deux algorithmes DSAC en BBRL de sorte que nous puissions les faire tourner sur tous les environnements de test évoqués ci-dessus, avec les mêmes 13 seeds.
- **Vérification de nos hypothèses** : Avec les résultats obtenus au travers de l'apprentissage des deux modèles DSAC et de DQN sur les différents environnements avec les mêmes 13 seeds, nous devons déterminer statistiquement si nos hypothèses sont justes.
- **Analyse de l'état de l'art** : Pour finir, nous devons vérifier en lisant des articles scientifiques si la comparaison entre les trois algorithmes a déjà été réalisée dans un précédent papier de recherche.