TME5 RLD

Policy Gradient

Edwyn Brient, Léo Hein - Décembre 2021

1 Algorithme Actor Critic

L'algorithme Actor Critic fait partie des algorithmes de type policy gradient. Ces algorithmes reposent sur l'optimisation de fonctions paramétrées pour obtenir des politiques maximisant l'espérance des récompenses :

$$\max_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$$

Les réseaux de neurones sont un type de fonctions paramétrées pratique pour ce genre de problème d'optimisation. Contrairement à certains algorithmes de policy gradient, l'actor critic utilises deux réseaux de neurones qu'il apprend simultanément : le réseau critic a pour but d'apprendre à estimer la valeur d'un état et le réseau actor a pour objectif de rendre la distribution d'actions maximisant les récompenses à long terme.

Pour bien comprendre l'intérêt d'apprendre un réseau critic, il faut introduire la fonction avantage. La fonction avantage a pour but de modéliser l'avantage de prendre une action sur une autre.

$$A^{\pi_\theta}(s,a) = Q_\lambda(s,a) + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

Cette fonction permet d'estimer l'intérêt de choisir une action plutôt qu'une autre et nous servira donc pour le calcul de la loss pour le réseau actor. Si le V peut être estimer par le réseau critic que nous allons apprendre en simultané, Q sera ici approximé de différentes manières en utilisant le return, la récompense à l'instant t ou le λ-return.

$$Q_\lambda^{\pi_\theta}(s_t,a_t) = \sum_{t'=0}^{T-t} \lambda^{T-t'} R_{T-t'} \text{ avec T le temps de fin d'épisode}$$

Dans nos expériences, le réseau critic aura pour objectif d'estimer au mieux la valeur des états $V^{\pi_\theta}(s)$ afin d'orienter au mieux le gradient de politique. Pour entraîner notre réseau critic, on utilise la huber loss entre $V^w(s)$ et $Q^{\pi_\theta}(s,a)$. Pour le réseau actor, la loss est la suivante : $-\log(\pi_\theta(s,a)) * A^{\pi_\theta}(s,a)$

2 Expériences

2.1 Paramètres et implémentation

Dans le cadres des expériences que nous avons menés dans ce TP, nous avons implémenté les différentes versions de l'algorithme actor critic. Ces versions ont été testées sur les environnements Cartpole, LunarLander et gridworld. Dans toutes les expériences qui vont suivre, un second critic (appelé réseau target) est utilisé pour estimer $V^{\pi_\theta}(s')$ afin d'éviter une surrestimation de V. A chaque étape d'apprentissage, 10 optimisations sont effectuées par épisode. Une fois toutes les optimisations effectuées, le réseau target est mis à jour après avoir en copiant les poids du critic.

Les résultats qui suivent ont été produits avec l'algorithme actor critic sur l'environnement Cartpole. Les paramètres ont été les mêmes entre les différentes expériences afin de pouvoir comparer les versions sans qu'un autre paramètre influe sur le résultat. Les étapes d'apprentissages sont effectuées toutes les 1000 itérations en attendant la fin de l'épisode en cours. Le learning rate pour tous les réseaux a été fixé à 10^{-4}, la fonction utilisée pour la loss du critic est la huber loss. Enfin, tous les réseaux ont deux couches de 64 neurones à activation ReLU.

2.2 Résultats

Nos premières expériences ont été réalisées sur l'environnement cartpole d'openAI.

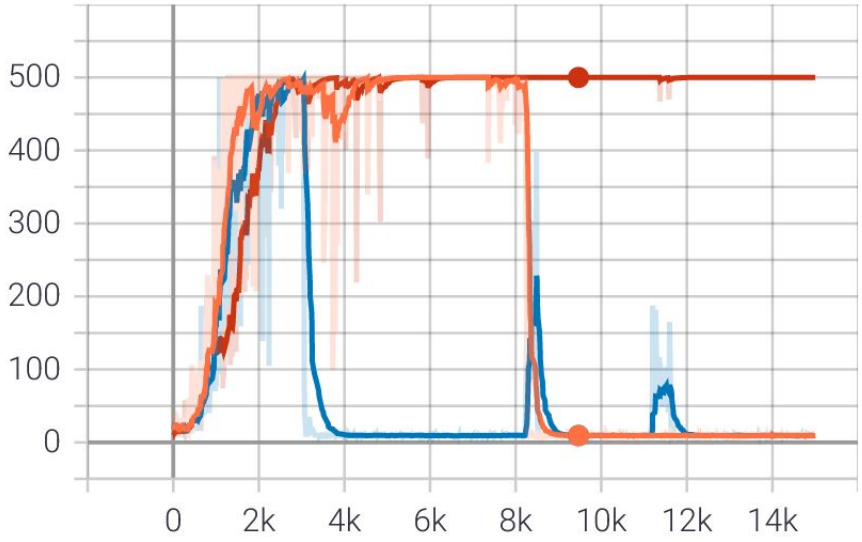

FIGURE 1 – Récompenses des version d'actor critic sur carptole en fonction du nombre d'épisodes : en *bleu* $\lambda = 0, \gamma = 0.99$, en *orange* $\lambda = 1, \gamma = 0$ et en *rouge* $\lambda = 0.99, \gamma = 0.99$

L'interprétation la plus intéressante se situe avec l'environnement Cartpole. Cette inter-

prétation se base sur une expérience, cependant, il aurait certainement été utile de répéter l'expérience de multiples fois pour comparer les apprentissages sur différents cas de figures.

Sur cette expérience, on peut remarquer que les trois méthodes commencent par apprendre assez rapidement puisqu'elles atteignent toutes les 450+ reward après environ 2K épisodes.

Rapidement la versions TD(0) retombe au reward minimum. Ceci peut s'expliquer par le manque de vision que donne la comparaison avec un reward à l'instant t comparé à TD(λ) même avec l'ajout de la baseline pour le TD(0). A la suite de cette chute, cette version semble avoir des problèmes à converger à nouveau.

La version Monte Carlo fonctionne correctement mais manque de stabilité donné par la baseline pour les deux autres versions. Cette version tombe donc à 0 comme la précédente après un certain nombre d'itérations et y reste.

Finalement, la version combinant la vision à long terme donnée par le TD(λ) et l'introduction de la baseline converge et reste stable sur les 15K épisodes, elle semble donc être la meilleure.

L'expérience sur gridworld plan 5 suit. Elle a été réalisée avec les mêmes paramètres que sur l'environnement Cartpole, avec $\lambda = 0.999$ et $\gamma = 0.999$. Ce choix de paramètres vient de la durée des épisodes, là où pour Cartpole, quand la durée des épisodes augmente, l'algorithme est déjà entrain de converger.

FIGURE 2 – Récompenses de la version d'actor critic $\lambda = 0.999$ et $\gamma = 0.999$ sur gridworld plan5 en fonction du nombre d'épisodes

Les résultats décevants pour l'algorithme sur ce problème proviennent certainement du feature extractor utilisé qui donnait comme taille d'input aux réseaux 108. Malgré la forte taille du problème, une meilleure extraction (par exemple à l'aide d'un CNN même cela semble légèrement puissant pour ce problème simple) pourrait certainement aider à obtenir de meilleures performances.

Enfin, l'expérience sur l'environnement lunarlander clos la phase d'expérimentations.

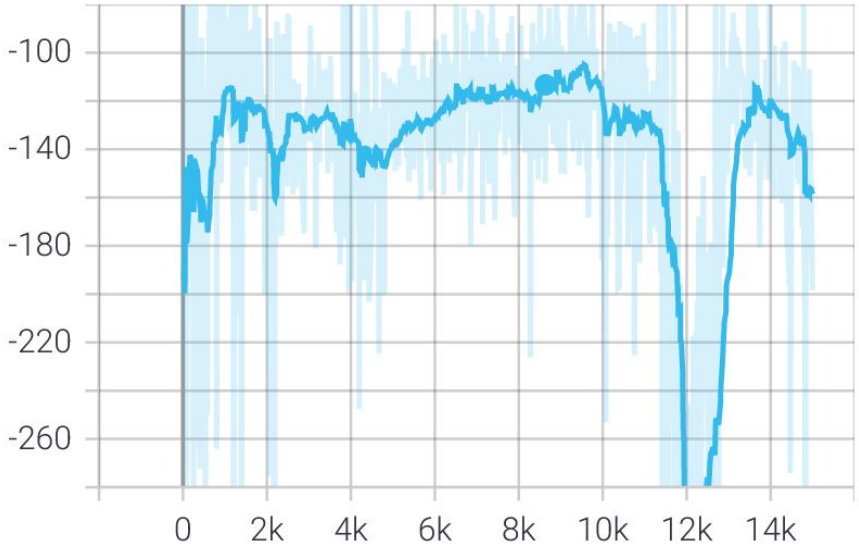

FIGURE 3 – Récompenses de la version d'actor critic $\lambda = 0.999$ et $\gamma = 0.999$ sur lunarlander en fonction du nombre d'épisodes

Au vu des nombres d'actions par épisodes effectués par l'actor critic sur ce problème (1000), l'algorithme semble avoir convergé vers un minimum local ne faisant pas atterir la fusée. L'introduction dans la loss de l'entropie aurait peut-être pu pallier à ce problème.

3 Conclusion

Ce TP nous a permis de découvrir l'algorithme d'actor critic et une partie de ses sous algorithmes sur différents problèmes.