

REINFORCEMENT LEARNING & ADVANCED DEEP


M2 DAC

TME 5. Policy Gradients

Ce TME a pour objectif d'expérimenter les approches de renforcement Policy Gradients vues en cours.

Implémenter l'algorithme actor-critic donné dans la figure ci-dessous et l'appliquer aux 3 problèmes du TP précédent (CartPole, LunarLander et GridWorld)

batch actor-critic algorithm:

- 
1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
 2. fit $\hat{V}_\phi^\pi(\mathbf{s})$ to sampled reward sums
 3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
 4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

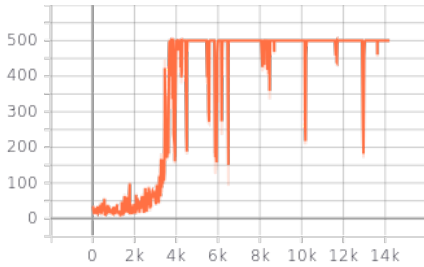
Plutôt que de mettre à jour après chaque action, on attend la fin d'un certain nombre de trajectoires avant toute optimisation.

À l'étape 2, la fonction V est mise à jour par un coût de Huber pour faire tendre la différence temporelle TD(0) vers 0 comme au TP précédent. Pour la cible $V(s')$ et pour la baseline dans la fonction d'avantage, il est conseillé d'utiliser un réseau annexe copiant les paramètres du réseau principal toutes les k itérations (par exemple 10000). À noter que cet algorithme est on-policy, le buffer de transitions est réinitialisé après chaque mise à jour de la politique (étape 5).

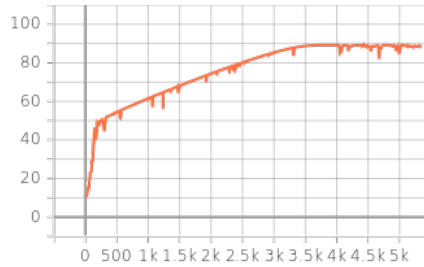
On pourra considérer une version Rollout Monte-Carlo (où V_t est comparé à R_t), une version TD(0) (où V_t est comparé à $r_t + \gamma V_{t+1}$ comme dans l'algorithme précédent) et une version TD(λ). Idem pour le calcul de l'avantage utilisé par l'acteur. Pour cela, il est conseillé de préparer les choses dans la fonction store, dans laquelle il est possible, lorsqu'une trajectoire est terminée (i.e., done=True), d'enregistrer les différences temporelles cumulées en plus des récompenses ponctuelles. Cela se fait en itérant à partir

de la fin de la trajectoire pour calculer les valeurs successives de GAE (Generalized Advantage Estimation) pour chaque transition jusqu'à l'état initial.

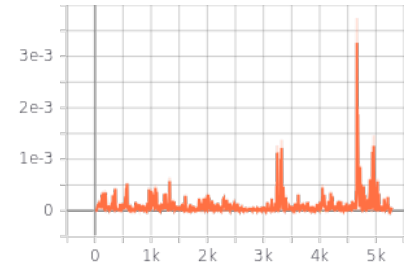
À titre indicatif, voici ce que l'on peut obtenir sur Cartpole, avec A2C utilisant un target network V (mis à jour après chaque optimisation, ayant lieu tous les 1000 événements en fin d'épisode), un discount de 0.99, un paramètre $\lambda = 0.99$ un pas d'apprentissage de 0.001, un batch de la taille du buffer (toutes les transitions observées depuis la dernière optimisation), une taille d'épisode maximale de 500 événements en apprentissage (également en test) et selon un réseau de neurones à deux couches cachées de 30 neurones chacune (avec activation tanh sur les couches cachées):



(a) Reward en apprentissage
(abscisse: nombre d'episodes)



(b) Valeur cible moyenne
(abscisse: nombre d'optimisations)



(c) Kulback-Leibler divergence
(abscisse: nombre d'optimisations)

On note quelques instabilités dans le reward moyen cumulé de la politique au cours de l'apprentissage, qui coïncident avec des changements brutaux de la politique (observables sur la courbe de divergence de Kulback-Leibler estimée). Nous réglerons ces instabilités au prochain TP avec des algorithmes moins sujets aux fortes oscillations.