Reinforcement Learning & Advanced Deep

M2 DAC

TME 6. Advanced Policy Gradients

Ce TME a pour objectif d'expérimenter l'approche PPO.

1 PPO Adaptative KL

Implémenter l'algorithme PPO avec coût KL adaptatif donné dans la figure ci-dessous et l'appliquer aux 3 problèmes des TP précédents (CartPole, LunarLander et GridWorld)

Algorithme 1 : Algorithme PPO - version KL adaptatif

Input : Paramètres initiaux θ_0 et ϕ, KL cible δ, pas d'apprentissage α, nombre d'étapes d'optimisation K

$\beta_0 \leftarrow 1$
for $k = 0, 1, 2, ...$ **do**
  Collecte d'un ensemble de trajectoires $\mathcal{D}_k$ selon politique π_{θ_k}
  Calcul des avantages $\hat{A}^{\pi_{\theta_k}}$ pour toutes les transitions de $\mathcal{D}_k$ selon $TD(\lambda)$
  $\theta \leftarrow \theta_k$
  for s *de* 1 *à* K **do**
    $\theta \leftarrow \theta + \alpha\big(\nabla_\theta \mathcal{L}_{\theta_k}(\theta) - \beta_k \nabla_\theta \bar{D}_{KL}(\theta_k|\theta)\big)$
      avec $\mathcal{L}_{\theta_k}(\theta) = \frac{1}{|\mathcal{D}_k|} \sum_{(s,a)\in\mathcal{D}_k} \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} \hat{A}^{\pi_{\theta_k}}(s,a)$
      et $\bar{D}_{KL}(\theta_k|\theta) = \frac{1}{|\mathcal{D}_k|} \sum_{s\in\mathcal{D}_k} D_{KL}(\pi_{\theta_k}(.|s)|\pi_\theta(.|s))$.
  end
  $\theta_{k+1} \leftarrow \theta$
  if $\bar{D}_{KL}(\theta_k|\theta_{k+1}) \geq 1.5\delta$ **then**
    $\beta_{k+1} \leftarrow 2\beta_k$
  end
  if $\bar{D}_{KL}(\theta_k|\theta_{k+1}) \leq \delta/1.5$ **then**
    $\beta_{k+1} \leftarrow 0.5\beta_k$
  end
  Mise à jour de V_ϕ selon $TD(\lambda)$ sur $\mathcal{D}_k$
end

Notons les K pas de gradients à chaque optimisation sur les trajectoires récoltées (plutôt que 1 avec les PG classiques).

Au fait d'après vous, que vaut le gradient de la KL au premier passage ? À quoi correspond l'algo PPO si on prend $K = 1$?

Note: on pourra comparer deux versions de la KL, $\bar{D}_{KL}(\theta_k|\theta)$ comme dans le papier original et en mode reverse $\bar{D}_{KL}(\theta|\theta_k)$

2 PPO with Clipped Objective

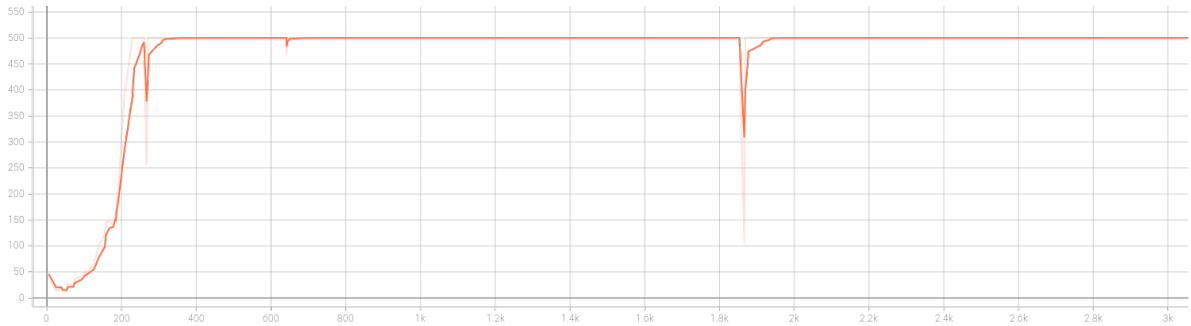
Implémenter l'algorithme PPO avec objectif "clippé" donné ci-dessous.

Algorithme 2 : Algorithme PPO - version clippée

Input : Paramètres initiaux θ_0 et ϕ, KL cible δ, pas d'apprentissage α, nombre d'étapes d'optimisation K

$\beta_0 \leftarrow 1$
for $k = 0, 1, 2, ...$ **do**
  Collecte d'un ensemble de trajectoires $\mathcal{D}_k$ selon politique π_{θ_k}
  Calcul des avantages $\hat{A}^{\pi_{\theta_k}}$ pour toutes les transitions de $\mathcal{D}_k$ selon $TD(\lambda)$
  $\theta \leftarrow \theta_k$
  for s *de* 1 *à* K **do**
    $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}^{CLIP}_{\theta_k}(\theta)$
    avec $\mathcal{L}^{CLIP}_{\theta_k}(\theta) =$
    $\frac{1}{|\mathcal{D}_k|} \sum_{(s,a)\in\mathcal{D}_k} min\Big(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} \hat{A}^{\pi_{\theta_k}}(s,a); clip(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon)\hat{A}^{\pi_{\theta_k}}(s,a)\Big)$
  end
  $\theta_{k+1} \leftarrow \theta$
  Mise à jour de V_ϕ selon $TD(\lambda)$ sur $\mathcal{D}_k$
end

À titre indicatif, voici ce qu'on peut obtenir sur cartpole avec PPO-clippé ($\epsilon = 0.2$), effectuant une optimisation de 100 étapes (i.e., $K = 100$) tous les 100 évènements (on attend quand même la fin de l'épisode en cours), utilisant un target network V (mis à jour tous les 1000 évènements), un discount de 0.99, un paramètre $\lambda = 0.99$, un pas d'apprentissage de 0.0001 pour l'acteur et de 0.0003 pour la critique, un batch de la taille du buffer (toutes les transitions observées depuis la dernière optimisation), une taille d'épisode maximale de 500 évènements 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):



Reward en apprentissage (abscisse: nombre d'episodes)

3 Comparaison

Comparer les performances des deux versions de PPO avec A2C et une version de PPO sans KL ni clipped objective (au moins sur Cartpole et LunarLander). Pour cela, une fois des "bons" hyper-paramètres trouvés, faire tourner les algos un certain nombre de fois (e.g., 10) et tracer les points moyens de performance (e.g., toutes les 100 trajectoires collectées).

4 Bonus: Entropie

Vous pourrez considérer une version avec coût d'entropie évitant aux politiques de converger trop rapidement vers des solutions sous-optimales.