

# Proximal Policy Optimization (PPO)

Fabrício Barth

Inspere Instituto de Ensino e Pesquisa

Maio de 2023

- Na última aula vimos o funcionamento do algoritmo **Reinforce**.
- O algoritmo **Reinforce** tem um problema relacionado com a atualização dos parâmetros do modelo.
- Dependendo do espaço de busca do problema, a atualização dos parâmetros pode ser grande demais e levar para situações indesejadas. Mesmo em situações onde a atualização dos parâmetros estava levando para uma situação de otimização.

# Proximal Policy Optimization (PPO)

- O PPO é um método de *policy gradient* que evita a atualização grande dos parâmetros da *policy*.
- Basicamente, o PPO usa uma razão entre a política atual e a política antiga e mantém esta razão entre  $[1 - \epsilon, 1 + \epsilon]$ .
- Desta forma o algoritmo PPO garante que a atualização do parâmetros não seja tão grande e o treinamento se torna mais estável.

# Objetivo desta aula

Ao final desta aula, você será capaz de compreender como funciona e como implementar o algoritmo **PPO**.

**function** *REINFORCE*( $\alpha, \gamma, \text{episódios}$ ):

inicializar os valores de  $\theta$  para a policy  $\pi(A|S, \theta)$  arbitrariamente

**for** todos os episódios **do**

gerar um episódio  $s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T$  seguindo  $\pi(.|., \theta)$

**for** cada passo dentro do episódio  $t = 0, 1, 2, \dots, T - 1$  **do**

$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} \times r_k$$

$$\theta \leftarrow \theta + \alpha \times \nabla \log \pi(a_t | s_t, \theta) \times G$$

**end for**

**end for**

# Equação central

A equação central do algoritmo REINFORCE é:

$$\theta \leftarrow \theta + \alpha \times \nabla \log \pi(a_t | s_t, \theta) \times G \quad (1)$$

que também pode ser escrita como:

$$L^{PG}(\theta) = \hat{E} \left[ \log \pi_{\theta}(a_t | s_t) \cdot \hat{G}(s_t, a_t) \right] \quad (2)$$

# Estimador de vantagem

Schulman, 2015, propõe o uso do conceito de *advantage estimation* ou estimador de vantagem no lugar do *discounted rewards*  $\hat{G}(s_t, a_t)$ .

$$L^{PG}(\theta) = \hat{E} \left[ \log \pi_{\theta}(a_t | s_t) \cdot \hat{A}(s_t, a_t) \right] \quad (3)$$

onde:

$$\hat{A}(s_t, a_t) = \hat{Q}(s_t, a_t) - \hat{V}(s_t) \quad (4)$$

- $\hat{Q}(s_t, a_t)$  é o *discounted rewards*, e;
- $\hat{V}(s_t)$  é a função baseline.

# Estimador de vantagem

$$\begin{aligned}\hat{A}(s_t, a_t) &= \hat{Q}(s_t, a_t) - \hat{V}(s_t) \\ \hat{Q}(s_t, a_t) &= \sum_{k=t+1}^T \gamma^{k-t-1} \times r_k \\ \hat{V}(s_t) &= \sum_{a_t \in A} \pi(a_t | s_t) \cdot \hat{Q}(s_t, a_t) \\ \hat{A}(s_t, a_t) &= \hat{Q}(s_t, a_t) - \sum_{a_t \in A} \pi(a_t | s_t) \cdot \hat{Q}(s_t, a_t)\end{aligned}\tag{5}$$



# Razão entre políticas

Schulman, 2017, propõe o uso da razão entre a política atual e a política antiga:

$$L^{TRPO}(\theta) = \hat{E} \left[ \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \cdot \hat{A}(s_t, a_t) \right] = \hat{E} \left[ r(\theta) \cdot \hat{A}(s_t, a_t) \right] \quad (6)$$

$$r(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (7)$$

- Lembrando que  $\pi_{\theta}(a_t|s_t)$  representa a probabilidade de  $a_t$  acontecer em  $s_t$ .
- Sendo assim, se  $r(\theta) < 1$  então  $a_t$  tem uma probabilidade maior de ocorrer em  $\theta_{old}$  do que em  $\theta$ .
- Se  $r(\theta) > 1$  então  $a_t$  tem uma probabilidade maior de ocorrer na política atual do que na antiga.

Mas e se  $r(\theta)$  for um número muito maior que 1? .. 100?

O algoritmo PPO define uma nova função objetivo que garante que a política nova não fique muito longe da política antiga:

$$L^{CLIP}(\theta) = \hat{E} \left[ \min \left( r(\theta) \cdot \hat{A}(s_t, a_t), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) \cdot \hat{A}(s_t, a_t) \right) \right] \quad (8)$$

- Com esta função CLIP, nós temos duas possibilidades de  $r(\theta)$ , a versão original (não limitada) e a versão limitada onde o limite é entre  $[1 - \epsilon, 1 + \epsilon]$ , onde  $\epsilon$  é um hiperparâmetro que nos ajuda a definir os limites.
- No artigo de Schulman, 2017 é utilizado o valor 0.2 para  $\epsilon$ .

# Função objetiva "clipped"

Esta função não permite que atualizações na política sejam feitas quando a razão ( $r(\theta)$ ) entre a política atual e a antiga fica fora da "zona de conforto".

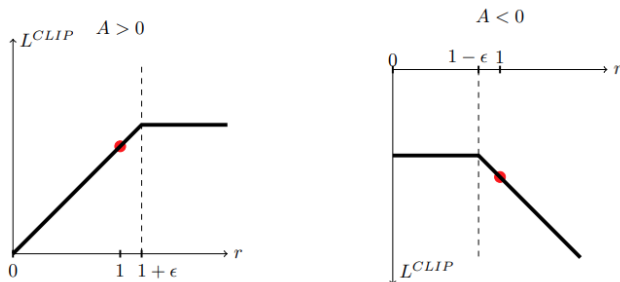


Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function  $L^{CLIP}$  as a function of the probability ratio  $r$ , for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e.,  $r = 1$ . Note that  $L^{CLIP}$  sums many of these terms.

# Algoritmo PPO, estilo "Actor-Critic"

```
for iterações=1,2,... do  
  for actor=1,2,..., N do  
    gerar um episódio  $s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T$  seguindo  $\pi(\cdot | \cdot, \theta_{old})$   
    calcular as estimativas de vantagem  $\hat{A}_1, \dots, \hat{A}_T$   
  end for  
  Otimiza  $L^{CLIP}(\theta)$  com  $K$  épocas e minibatch com tamanho  $M \leq NT$ .  
   $\theta_{old} \leftarrow \theta$   
end for
```

# Algoritmo PPO, estilo "Actor-Critic"

```
for iterações=1,2,... do  
  for actor=1,2,..., N do  
    gerar um episódio  $s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T$  seguindo  $\pi(.|., \theta_{old})$   
    calcular as estimativas de vantagem  $\hat{A}_1, \dots, \hat{A}_T$   
  end for  
  Otimiza  $L^{CLIP}(\theta)$  com  $K$  épocas e minibatch com tamanho  $M \leq NT$ .  
   $\theta_{old} \leftarrow \theta$   
end for
```

Este algoritmo é dividido em duas threads:

- atores: responsáveis por gerar os episódios e calcular toda a sequência de  $S, A, R, \hat{A}$  com base na policy vigente.
- críticos: responsáveis por otimizar os valores da policy.

# Algoritmo PPO, estilo "Actor-Critic"

```
for iterações=1,2,... do  
  for actor=1,2,..., N do  
    gerar um episódio  $s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T$  seguindo  $\pi(\cdot | \cdot, \theta_{old})$   
    calcular as estimativas de vantagem  $\hat{A}_1, \dots, \hat{A}_T$   
  end for  
  Otimiza  $L^{CLIP}(\theta)$  com  $K$  épocas e minibatch com tamanho  $M \leq NT$ .  
   $\theta_{old} \leftarrow \theta$   
end for
```

Algumas observações:

- $T$  não necessariamente é um estado terminal.  $T$  define o limite da trajetória.
- a geração dos episódios acontece por  $N$  atores independentes e paralelos.

# Algoritmo PPO, estilo "Actor-Critic"

```
for iterações=1,2,... do  
  for actor=1,2,..., N do  
    gerar um episódio  $s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T$  seguindo  $\pi(\cdot|., \theta_{old})$   
    calcular as estimativas de vantagem  $\hat{A}_1, \dots, \hat{A}_T$   
  end for  
  Otimiza  $L^{CLIP}(\theta)$  com  $K$  épocas e minibatch com tamanho  $M \leq NT$ .  
   $\theta_{old} \leftarrow \theta$   
end for
```

Algumas observações:

- Tempos em tempos, a thread para atualização da policy dispara um Stochastic Gradient Descent sobre a função  $L^{CLIP}(\theta)$ .
- executar treinamentos usando minibatch e com  $K$  épocas não é usual em Reinforcement Learning. No entanto, devido ao uso da função  $L^{CLIP}$  isso é possível.

# Usando a biblioteca stable\_baselines3

A biblioteca **stable\_baselines3** é um conjunto de implementações de algoritmos de Reinforcement Learning em pytorch:

<https://stable-baselines3.readthedocs.io/en/master/>

```
import gymnasium as gym

from stable_baselines3 import PPO
from stable_baselines3.common.env_util import import make_vec_env

# Parallel environments
env = make_vec_env("LunarLander-v2", n_envs=1)

model = PPO("MlpPolicy", env, verbose=1)
model.learn(total_timesteps=200_000)
model.save("ppo_lunarlander")

del model # remove to demonstrate saving and loading

model = PPO.load("ppo_lunarlander")

print('model trained')
obs = env.reset()
while True:
    action, _states = model.predict(obs)
    obs, rewards, dones, info = env.step(action)
    env.render()
```



- The 37 Implementation Details of Proximal Policy Optimization. Disponível <https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/>. Último acesso em maio de 2023.
- Schulman J, Levine S, Abbeel P, Jordan M, Moritz P. Trust region policy optimization. In International conference on machine learning 2015 Jun 1 (pp. 1889-1897). PMLR.v <https://doi.org/10.48550/arXiv.1502.05477>
- Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. 2017 Jul 20.
- Understanding Proximal Policy Optimization (Schulman et al., 2017). Disponível [blog.tylertaewook.com/post/proximal-policy-optimization](https://blog.tylertaewook.com/post/proximal-policy-optimization). Último acesso em maio de 2023.
- Simonini, T. Proximal Policy Optimization (PPO). Unit 8, of the Deep Reinforcement Learning Class with Hugging Face. Disponível em <https://huggingface.co/blog/deep-rl-ppo>. Último acesso em maio de 2023.