Proximal Policy Optimization (PPO)

Fabrício Barth

Insper Instituto de Ensino e Pesquisa

Abril de 2025

Proximal Policy Optimization (PPO)

- Em algoritmos da família *policy optimization*, 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.
- 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 mantêm 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.

Algoritmo Reinforce

function *REINFORCE(α, γ, episódios):*
inicializar os valores de θ para a policy $\pi(A|S,\theta)$ arbitrariamente
for todos os episódios **do**
  gerar um episódio $s_0, a_0, r_1, \cdots, s_{T-1}, a_{T-1}, r_T$ seguindo $\pi(.|.,\theta)$
  for cada passo dentro do episódio $t = 0, 1, 2, \cdots, T-1$ **do**
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} \times r_k$
    $\theta \leftarrow \theta + \alpha \times \bigtriangledown \log \pi(a_t|s_t,\theta) \times G$
  end for
end for

Equação central

A equação central do algoritmo REINFORCE é:

$$\theta \leftarrow \theta + \alpha \times \bigtriangledown \log \pi(a_t|s_t, \theta) \times G \tag{1}$$

que também pode ser escrita como:

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

1

[1] Schulman, 2015 usa a segunda forma de anotação

Estimador de vantagem

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

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

onde:

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

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

[2]

[2] Mesmo conceito que é proposto em Mnih et al. (2016)

Razão entre políticas

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

$$L^{TRPO}(\theta) = \hat{E}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}.\hat{A}(s_t, a_t)\right] = \hat{E}\left[r(\theta).\hat{A}(s_t, a_t)\right] \quad (5)$$

$$r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)} \quad (6)$$

- 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 θold do que em θ.
- 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:

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

- 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 ϵ é um hiperparâmetro que nos ajuda a definir os limites.
- No artigo de Schulman, 2017 é utilizado o valor 0.2 para ϵ.

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".

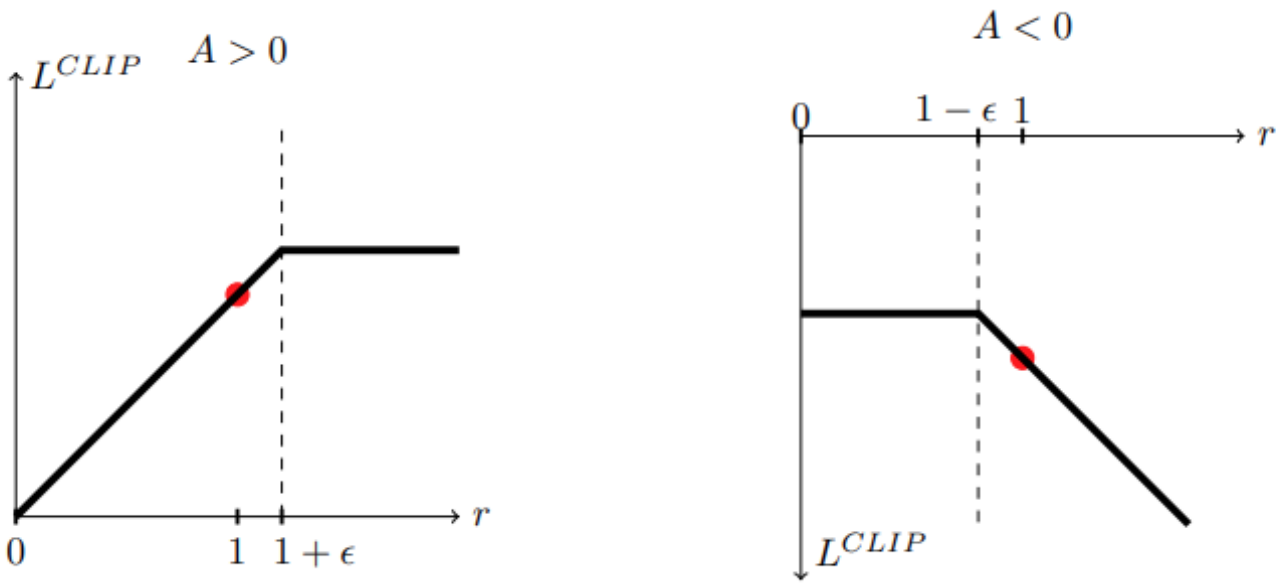

Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP}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

Algoritmo PPO, estilo "Actor-Critic"

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

Algumas observações:

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

Hiperparâmetros do algoritmo PPO

- K: quantidade de épocas usadas no treinamento de θ.
- M: tamanho do minibath - que não pode ser maior que N (número de atores) $\times$ T (tamanho máximo das trajetórias).
- ϵ: define as margens da função *CLIP*.
- α: define o *learning rate* do aprendizado.
- γ: define o valor que deve ser dado aos *rewards* futuros.
- *episodios*: define a quantidade de episódios.

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 make_vec_env

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

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()
```

Referências

- 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. 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.