Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

Abstract

We propose a new family of policy gradient methods for reinforcement learning, which alternate between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent. Whereas standard policy gradient methods perform one gradient update per data sample, we propose a novel objective function that enables multiple epochs of minibatch updates. The new methods, which we call proximal policy optimization (PPO), have some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement, more general, and have better sample complexity (empirically). Our experiments test PPO on a collection of benchmark tasks, including simulated robotic locomotion and Atari game playing, and we show that PPO outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time.

1 Introduction

Proximal Policy Optimization (PPO) overview

PPO is:

Policy Gradient algorithm w/ advantage estimate

+ mechanism to improve sample efficiency: importance sampling and an approximation

+ modified objective function to ensure approximation is valid

Let’s dig into it !

How do we pick step size α in Policy Gradient?

If α is too large, no convergence/learning.

If α is too small, lot's of environment interactions for very little gain

Possible fix: Use small α, but update policy many times with same data. This is the main idea behind the algorithms TRPO and PPO.

Issue with fix: Updating current policy with data from old policy. PG theorem doesn't apply. Invoke importance sampling theorem.

Aside: Importance Sampling

- Suppose p(x) and q(x) are two distributions over {1,2,..,n}
- Suppose we want to estimate : $E_{X\sim p}[\ f(X)\ ]$ using samples from q(.)

- Importance sampling theorem: $E_{X\sim p}[\ f(X)\ ] = E_{X\sim q}[\ f(X)\ \frac{p(X)}{q(X)}\ ]$

- Thus $f(x)\frac{p(x)}{q(x)}$ with x sampled from q(.) is an unbiased estimator for $E_{X\sim p}[f(X)]$

- Quiz: Prove importance sampling theorem. Hint: easy, 1-2 lines

Towards PPO: Applying importance sampling to policy gradient

- Recall $\nabla_\theta v_\theta = E_{\pi_\theta}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t) A_{\pi_\theta}(S_t, A_t)]$
- To improve sample efficiency, we are interested in estimating this with data generated by an earlier policy $\pi_{\theta'}$
- From importance sampling theorem:

$$E_\theta[\nabla_\theta \log \pi_\theta(A_t|S_t) A_{\pi_\theta}(S_t, A_t)] = E_\theta[f(S_t,A_t)] = E_{\theta'}[f(S_t,A_t) \frac{p_\theta(S_t,A_t)}{p_{\theta'}(S_t,A_t)}]$$

$$= E_{\theta'}[\nabla_\theta \log \pi_\theta(A_t|S_t) A_{\pi_\theta}(S_t, A_t) \frac{p_\theta(S_t,A_t)}{p_{\theta'}(S_t,A_t)}]$$

$$= E_{\theta'}[\nabla_\theta \log \pi_\theta(A_t|S_t) A_{\pi_\theta}(S_t, A_t) \frac{\pi_\theta(A_t|S_t)}{\pi_{\theta'}(A_t|S_t)} \frac{p_\theta(S_t)}{p_{\theta'}(S_t)}]$$

$$= E_{\theta'}[A_{\pi_\theta}(S_t, A_t) \frac{\nabla_\theta \pi_\theta(A_t|S_t)}{\pi_{\theta'}(A_t|S_t)} \frac{p_\theta(S_t)}{p_{\theta'}(S_t)}]$$

where last inequality follows from identity $\nabla_x f(x) = f(x) \nabla_x \log f(x)$

- Just showed:

$$\nabla_\theta v_\theta = E_\theta[\nabla_\theta \log \pi_\theta(A_t|S_t) A_{\pi_\theta}(S_t, A_t)] = E_{\theta'}[A_{\pi_\theta}(S_t, A_t) \frac{\nabla_\theta \pi_\theta(A_t|S_t)}{\pi_{\theta'}(A_t|S_t)} \frac{p_\theta(S_t)}{p_{\theta'}(S_t)}]$$

- If $\pi_\theta \approx \pi_{\theta'}$, then $\frac{p_\theta(S_t)}{p_{\theta'}(S_t)} \approx 1$. Thus, if $\pi_\theta \approx \pi_{\theta'}$ then

$$E_\theta[\nabla_\theta \log \pi_\theta(A_t|S_t) A_{\pi_\theta}(S_t, A_t)] \approx E_{\theta'}[\frac{\nabla_\theta \pi_\theta(A_t|S_t)}{\pi_{\theta'}(A_t|S_t)} A_{\pi_\theta}(S_t, A_t)]$$

$$\approx E_{\theta'}[\frac{\nabla_\theta \pi_\theta(A_t|S_t)}{\pi_{\theta'}(A_t|S_t)} \hat{A}_t]$$

where $\hat{A}_t := G_t - V_\phi(S_t)$

- THUS, if we use $\pi_{\theta'}$ to generate an episode $(s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T)$, we can approximate $\nabla_\theta v_\theta$ by $\sum_{t=0}^{T-1} \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} \hat{A}_t$ where $\hat{A}_t := G_t - V_\phi(s_t)$
- Approximation is okay as long as $\pi_\theta \approx \pi_{\theta'}$

Proximal Policy Optimization (PPO)

- Very popular on-policy algorithm: robust, even used in original ChatGPT
- Uses critic network ϕ and advantage estimates $\hat{A}_t$
- Use current policy π_{θ_k} to generate N episodes (π_{θ_k} is $\pi_{\theta'}$ on previous slide)
- Use data from these episodes to update actor θ multiple times.
- Multiple updates with same data: sample efficient!
- When updating actor, begin at $\theta = \theta_k$ and take several small gradient steps, always using the same N episodes from π_{θ_k} .
- After the several small updates on θ, set $\theta_{k+1} = \theta$, and run N new episodes using θ_{k+1} .
- In this way, we can use a small learning rate α and re-use the data from π_{θ_k} for many updates.

Almost PPO:

Initialize θ_0 and ϕ

For k = 0,1,2,… :

Use policy π_{θ_k} to generate N episodes:

$\tau^{(i)} = (s_0^{(i)}, a_0^{(i)}, r_1^{(i)}, s_1^{(i)}, a_1^{(i)}, r_2^{(i)}, \ldots, s_{T-1}^{(i)}, a_{T-1}^{(i)}, r_T^{(i)})$, i=1,…,N

Calculate the returns-to-go $G_t^{(i)}$ and the advantage estimates $\hat{A}_t^{(i)}$

Set $\theta_{k0}=\theta_k$

For j = 0,…,J-1:

$$\theta_{k(j+1)} = \theta_{kj} + \frac{\alpha}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{\nabla_\theta \pi_\theta(a_t|s_t)|_{\theta=\theta_{kj}}}{\pi_{\theta k}(a_t|s_t)} \hat{A}_t^{(i)}$$

Set $\pi_{\theta_{k+1}} = \pi_{\theta_{kJ}}$

Update critic:

$$\phi \leftarrow \phi - \frac{\alpha}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\phi (V_\phi(s_t^{(i)}) - G_t^{(i)})^2$$

Clipped Objective

- In Almost PPO, the original policy is π_{θ_k} , and the new policies are $\pi_{\theta_{kj}}$, j $= 1, \ldots,$ J, which is being updated in the inner loop
- If $\pi_{\theta_{kj}}$ drifts very far from π_{θ_k} the approximation becomes bad.
- PPO: when $\pi_{\theta_{kj}}$ drifts very far from π_{θ_k} use clipping to stop gradient updates.
- $\theta_{k(j+1)} = \theta_{kj} + \frac{\alpha}{N} \nabla_{\theta kj} \sum_{i=1}^{N} \sum_{t=1}^{T} clip\{\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta k}(a_t|s_t)}, 1+\epsilon, 1-\epsilon\} \hat{A}_t^{(i)}$
- (actual formula a little more complicated, but this is the main idea)
- Clipping roughly keeps $1 - \varepsilon \leq \frac{\pi_{\theta_{kj}}}{\pi_{\theta_k}} \leq 1 + \varepsilon$ (see Joshua Achiam lecture)

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
8: **end for**

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases}$$

importance sampling

Clipping term, small typo

Take several gradient steps here

GRPO (Group Relative Policy Optimization)

- Used in Deepseek R1
- No critic network
- Denote $G^{(1)}$, $G^{(2)}$,…,$G^{(N)}$ for the returns of the N episodes
- Let m and σ^2 be the mean and variance of $G^{(1)}$, $G^{(2)}$,…,$G^{(N)}$
- Normalize returns: $G^{(i)} \leftarrow (G^{(i)} - m)/\sigma$

- Replace $\hat{A}_t^{(i)}$ in PPO algorithm with normalized returns $G^{(i)}$:

- $\theta_{k(j+1)} = \theta_{kj} + \frac{\alpha}{N} \nabla_{\theta kj} \sum_{i=1}^{N} clip\{\frac{\pi_\theta(o_i)}{\pi_{\theta k}(oi)}, 1+\epsilon, 1-\epsilon\} G^{(i)}$

Group Relative Policy Optimization In order to save the training costs of RL, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which foregoes the critic model that is typically the same size as the policy model, and estimates the baseline from group scores instead. Specifically, for each question q, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model π_θ by maximizing the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \tag{1}$$

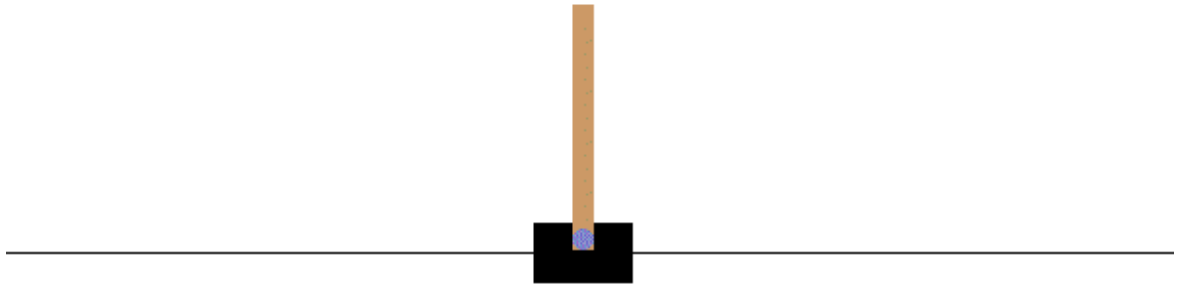
$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \tag{2}$$

where ε and β are hyper-parameters, and A_i is the advantage, computed using a group of rewards $\{r_1, r_2, \ldots, r_G\}$ corresponding to the outputs within each group:

$$A_i = \frac{r_i - mean(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}. \tag{3}$$

Homework 5

- You will apply Policy Gradient and PPO to the cartpole environment.
- Two actions: push car left, push cart right
- State space is 4D and continuous: cart position, cart velocity, pole angle, pole angular velocity
- The episode ends if any one of the following occurs:
  1. Pole Angle is greater than ±12°
  2. Cart Position is greater than ±2.4 (center of the cart reaches the edge of the display)
  3. Episode length is greater than 500
- Reward: +1 for every time step. So goal is maximize length of episode



Increasing Exploration

- Policy gradient intrinsically provides some exploration since the policy network provides a stochastic policy via the softmax.
- But often this exploration is not enough: policy network policy can become close to deterministic.
- Often add an entropy term to objective function to enhance exploration:

$$H(\pi_\theta|\mathrm{s}) := -\sum_a \pi_\theta(a|s)\log\pi_\theta(a|s)$$

- Along a single episode, we would use

$$-\frac{1}{T}\sum_{t=1}^{T}\sum_a \pi_\theta(a|s_t)\log\pi_\theta(a|s_t)$$

- Note that in the above formula, we use the states in the episode but not the actions.

Update with Advantage Estimate & Entropy

Update the policy:

$$\theta \leftarrow \theta + \frac{\alpha}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\hat{A}_t^{(i)}\nabla_\theta log\, \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

$$- \frac{\lambda}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_a \nabla_\theta\, \pi_\theta(a|s_t^{(i)}) log\, \pi_\theta(a|s_t^{(i)})$$