Proximal Policy Optimization (PPO)

Shahzeb Aamir

Content:

1. Motivation for PPO
2. PPO Goals
3. PPO Algorithm in detail
4. Some cool implementations

Motivation

Training Data is not “static”

Training data is dependent upon current policy

High Sensitivity to Hyper-Parameters

If we get one “bad” policy, one outlier, it can end up in a fire pit. Or a very slow progress

PPO Goals

Developed by folks at OpenAI, based on Policy Gradient Methods

John Schulman et al., Proximal Policy Optimization Algorithms (2017 Paper)

1. Ease of implementation (Easy Code)

2. *Sample Efficiency ("Online Policy Method")*

3. *Robust (Less sensitive to Hyper Parameters)*

Policy Gradient Loss

1. Vanilla Policy Gradient

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t \mid s_t) \hat{A}_t \right].$$

1. Discounted Rewards (Return)
2. Baseline Estimate (Value Function)

Advantage: Relative value of the action

Policy: A neural network that takes in observed state and gives probability of actions

Trusted Region

New policy and old policy is not too different

This is the basis of PPO

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{aligned}$$

Value Function: Guess the final return starting from the current state

Usually a neural network: A noisy estimate

PPO

1. Crux of PPO

The surrogate policy loss function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t, \operatorname{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big]$$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

This probability ratio > 1: If the action is more likely now than it was before gradient update

0 < This probability ratio < 1: If the action is less likely now than it was before gradient update

PPO

1. If advantage is positive then the probability of taking that action increases but not too much (that it clips)
2. If advantage is negative then the probability of taking that action decreases but not too much

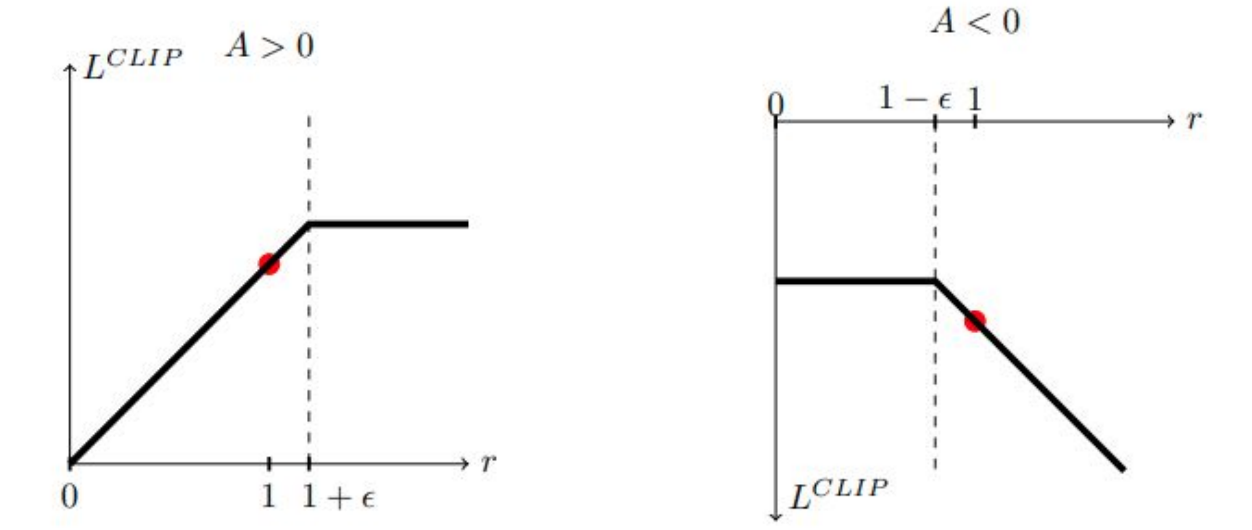

Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP}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

PPO

1. Final Objective function:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right]$$

Algorithm

Algorithm 1 PPO, Actor-Critic Style

for iteration$=1, 2, \ldots$ **do**
    for actor$=1, 2, \ldots, N$ **do**
        Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
        Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
    end for
    Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
    $\theta_{\text{old}} \leftarrow \theta$
end for

PPO

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1},$$
$$\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

PPO implementation

https://openai.com/blog/openai-baselines-ppo/

https://github.com/higgsfield/RL-Adventure-2/blob/master/3.ppo.ipynb

https://www.youtube.com/watch?v=WxQfQW48A4A&list=PLB79uOaPEEU6uU1-Pfaqr08RTTzhyB8hu&index=3

Thank You!