

# Proximal Policy Optimization

(2017, OpenAI)

[쉽게 읽는 강화학습 논문 6화]

판요랩 – 노승은, 전민영

2019.04.13

*"Probably the state-of-the-art current reinforcement learning algorithm"*

*"Might be your first choice if you just throw an algorithm at something"*

*- 2017, Pieter Abbeel*

# 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

In recent years, several different approaches have been proposed for reinforcement learning with neural network function approximators. The leading contenders are deep  $Q$ -learning [Mni+15], “vanilla” policy gradient methods [Mni+16], and trust region / natural policy gradient methods [Sch+15b]. However, there is room for improvement in developing a method that is scalable (to

Arxiv 2017, 647회 인용  
굉장히 실용적인 접근을 담고 있는 논문...

# Idea

- TRPO와 같은 질문으로부터 출발 :

지금 얻은 데이터로 가능한 큰 STEP 만큼 update 하고 싶은데..  
그렇다고 너무 멀리 가서 성능을 떨어뜨리고 싶지도 않은데..  
어떻게 하면 좋을까?

- TRPO는 이 문제를 복잡한 second-order method 로 풀려고 하였다.
- PPO는 first-order 방법론이다.
- 트릭을 사용하여 새로운 policy를 기존의 policy와 가깝도록 유지하게 해준다.
- 굉장히 구현이 간단하며, 실증적으로도 좋은 성능을 보여준다.

- Policy gradients

$$\hat{g} = \hat{\mathbb{E}}_t \left[ \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \hat{A}_t \right]$$

- Can differentiate the following loss

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

but don't want to optimize it too far

- Equivalently differentiate

$$L_{\theta_{\text{old}}}^{IS}(\theta) = \hat{\mathbb{E}}_t \left[ \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t \right].$$

at  $\theta = \theta_{\text{old}}$ , state-actions are sampled using  $\theta_{\text{old}}$ . (IS = importance sampling)


Just the chain rule:  $\nabla_{\theta} \log f(\theta) \big|_{\theta_{\text{old}}} = \frac{\nabla_{\theta} f(\theta) \big|_{\theta_{\text{old}}}}{f(\theta_{\text{old}})} = \nabla_{\theta} \left( \frac{f(\theta)}{f(\theta_{\text{old}})} \right) \big|_{\theta_{\text{old}}}$

# 잠깐 복습!

## Importance Sampling

- Estimate the expectation of a different distribution

$$\begin{aligned}\mathbb{E}_{X \sim P}[f(X)] &= \sum P(X)f(X) \\ &= \sum Q(X) \frac{P(X)}{Q(X)} f(X) \\ &= \mathbb{E}_{X \sim Q} \left[ \frac{P(X)}{Q(X)} f(X) \right]\end{aligned}$$


$$\sum_a \pi_{\theta}(a|s_n) A_{\theta_{\text{old}}}(s_n, a) = \mathbb{E}_{a \sim q} \left[ \frac{\pi_{\theta}(a|s_n)}{q(a|s_n)} A_{\theta_{\text{old}}}(s_n, a) \right]$$

q는 sampling distribution. 여기서 old policy를 사용하겠죠?

# 좀더 쉽게...

$$U(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[ \frac{P(\tau|\theta)}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$\nabla_{\theta} U(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[ \frac{\nabla_{\theta} P(\tau|\theta)}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$\nabla_{\theta} U(\theta)|_{\theta=\theta_{\text{old}}} = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[ \frac{\nabla_{\theta} P(\tau|\theta)|_{\theta_{\text{old}}}}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$= \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[ \nabla_{\theta} \log P(\tau|\theta)|_{\theta_{\text{old}}} R(\tau) \right]$$

# TRPO

- **Constrained Form**

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

- **Penalized Form**

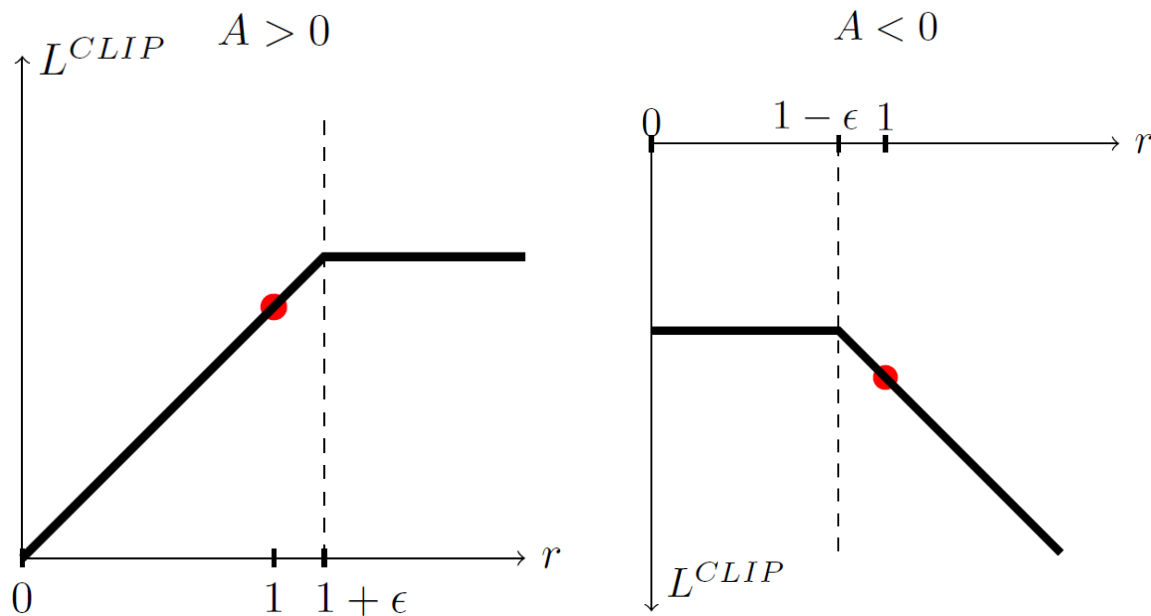
$$\underset{\theta}{\text{maximize}} \hat{\mathbb{E}}_t \left[ \underbrace{\frac{\pi_{\theta}(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}}_{\downarrow} \hat{A}_t - \beta \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_{\theta}(\cdot \mid s_t)] \right]$$
$$r_t(\theta) = \frac{\pi_{\theta}(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \quad \text{so } r(\theta_{\text{old}}) = 1.$$



# 방법 1 - Clipped Surrogate Objective

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[ \min(\underbrace{r_t(\theta)}_{\text{Original Loss}}, \underbrace{\text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)}_{\text{Clipped Loss}} \hat{A}_t) \right]$$

Lower bound of unclipped objective =  $L^{CPI}$



- $\theta_{old}$  근처에서는 (즉,  $r=1$  근처)  $L^{CPI}$ 와  $L^{CLIP}$ 가 1차 근사.

# Algorithm – PPO Clipped

**for** iteration=1, 2, ... **do**

Run policy for  $T$  timesteps or  $N$  trajectories

Estimate advantage function at all timesteps

Do SGD on  $L^{CLIP}(\theta)$  objective for some number of epochs

**end for**

## 방법 2 - Adaptive KL Penalty

- Using several epochs of minibatch SGD, optimize the KL-penalized objective

$$L^{KL PEN}(\theta) = \hat{\mathbb{E}}_t \left[ \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t - \beta \text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)] \right]$$

- Compute  $d = \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)]]$

- If  $d < d_{\text{targ}}/1.5$ ,  $\beta \leftarrow \beta/2$
- If  $d > d_{\text{targ}} \times 1.5$ ,  $\beta \leftarrow \beta \times 2$

- ▶ Pseudocode:

**for** iteration=1, 2, ... **do**

Run policy for  $T$  timesteps or  $N$  trajectories

Estimate advantage function at all timesteps

Do SGD on above objective for some number of epochs

If KL too high, increase  $\beta$ . If KL too low, decrease  $\beta$ .

**end for**

# 1. Surrogate Objectives 비교 실험

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
<b>Clipping, <math>\epsilon = 0.2</math></b>	<b>0.82</b>
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

- MuJuCo 환경에서 7개의 문제에 각각 3번씩 학습 시켜서 총 21개의 점수의 평균을 구함.
- policy, value 함수는 같은 구조.
- 1m step 학습.
- 마지막 100 episode의 평균 성능 측정.
- 각 환경별로 점수의 scale은 다 같도록 normalize 함.

No clipping or penalty:

$$L_t(\theta) = r_t(\theta) \hat{A}_t$$

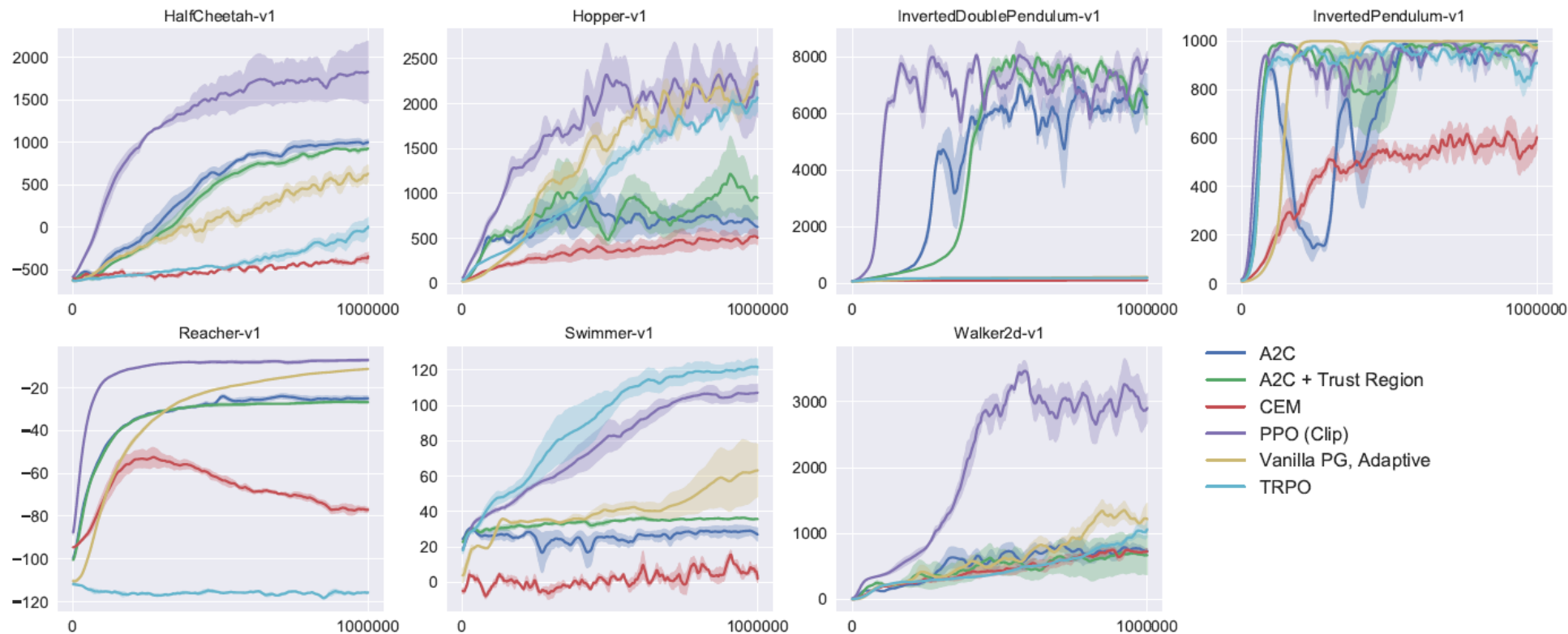
Clipping:

$$L_t(\theta) = \min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta)), 1 - \epsilon, 1 + \epsilon) \hat{A}_t$$

KL penalty (fixed or adaptive)

$$L_t(\theta) = r_t(\theta) \hat{A}_t - \beta \text{KL}[\pi_{\theta_{\text{old}}}, \pi_{\theta}]$$

## 2. Continuous Domain에서의 타 알고리즘과 비교 실험



- MuJuCo 환경에서 7개의 문제에 대해 학습시킴.
- 학습은 알고리즘 별로 모두 똑같이 1m step 진행.
- A2C가 A3C보다 더 잘 돼서 A2C로 실험 진행.
- PPO의  $\varepsilon = 0.2$  로 함.

### 3. Atari Domain에서의 타 알고리즘과 비교 실험

	A2C	ACER	PPO	Tie
(1) avg. episode reward over all of training	1	18	<b>30</b>	0
(2) avg. episode reward over last 100 episodes	1	<b>28</b>	19	1

- (1), (2)두 가지 기준을 적용하여 평가함.
  - (1) 은 학습이 얼마나 빠르지 평가하는 기준
  - (2)는 최종 성능이 얼마나 좋은지 평가하는 기준.
- 총 49개의 각각의 게임에서 어느 알고리즘이 젤 좋았나 적어놓은 후, 각 알고리즘 별로 그 개수를 셈.
  - 예컨대 PPO는 49개 중 30개의 게임에서 (1) 기준 제일 좋았음.
  - 각 점수는 3번 학습하여 그 평균을 측정.
- 모두 총 40m step 학습 시켜서 비교함.