Proximal Policy Optimization Algorithms

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

고경빈

2025.07.09

Background

- 최근 신경망 함수 근사기를 사용하는 강화 학습 방식들이 제안됨
  - 신경망 함수 근사기: 신경망을 사용하여 복잡한 함수를 근사
- 확장성, 데이터 효율성, 강건성을 갖춘 방법론에 대한 개선의 여지가 존재함

- 정책: 특정 상태에서 어떤 행동을 취해야 가장 좋은 결과를 얻을 수 있는가
- 가치 함수(V)/Q-함수(Q): 특정 상태의 가치/상태에서 특정 행동을 취했을 때의 가치

Deep Q-learning

- Q-learning
  - 목표: 각 상태-행동 쌍에 대한 최적의 Q-값을 학습하는 것
  - 한계점: 상태/행동 공간이 크면, 모든 상태-행동 쌍의 Q-값을 테이블 형태로 저장/업데이트 불가
- Deep Q-learning
  - Q-함수 근사기를 사용해 신경망이 주어진 상태를 입력 받아 각 행동에 대한 Q-값을 출력하도록 학습
  - 모델은 본 적 없는 새로운 상태에 대해서도 Q-값 예측할 수 있게 되어 일반화 성능 향상
  - 한계점
    - 간단한 문제에서 실패함
    - 작동 방식이 완전히 이해되지 못함
    - 이산 행동 공간은 가진 게임 환경에서는 잘 작동하지만, 연속적인 행동 공간에서는 성능이 좋지 않음

Vanilla Policy Gradient

- 작동 방식
  - 정책 정의: $\pi_\theta(a|s)$를 신경망과 같은 매개변수화된 함수로 정의
  - 데이터 수집: 정책을 사용하여 환경과 상호작용하며 상태(s_t), 행동(a_t), 보상(r_t) 등의 데이터 수집
  - 정책 경사 추정: 수집된 데이터를 바탕으로 정책 성능을 나타내는 목적 함수의 경사 계산

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

  - 매개변수 업데이트: SGA를 통해 정책을 업데이트하여 보상을 증가시키는 방향으로 정책 개선
- 한계점
  - 데이터 비효율성
  - 불안정성 및 낮은 강건성
  - 파괴적인 업데이트

TRPO

- 대리 목적 함수를 최대화하되, 이전 정책과 새로운 정책 간의 KL 발산에 강한 제약을 둠
- 목적 함수: 정책 업데이트 크기를 제한하는 조건 하에서 최대화

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right]$$

- 제약 조건: 정책 업데이트가 너무 크게 일어나지 않도록 제한

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta$$

- 장점: 정책이 급격하게 변하는 것을 방지하고, 단조로운 성능 향상을 기대할 수 있음
- 한계점
  - 계산 복잡성
  - 호환성 문제

TRPO

- 제약 조건 대신 KL 발산에 페널티를 주는 방식

$$\underset{\theta}{\text{maximize}}\, \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t - \beta\, \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right]$$

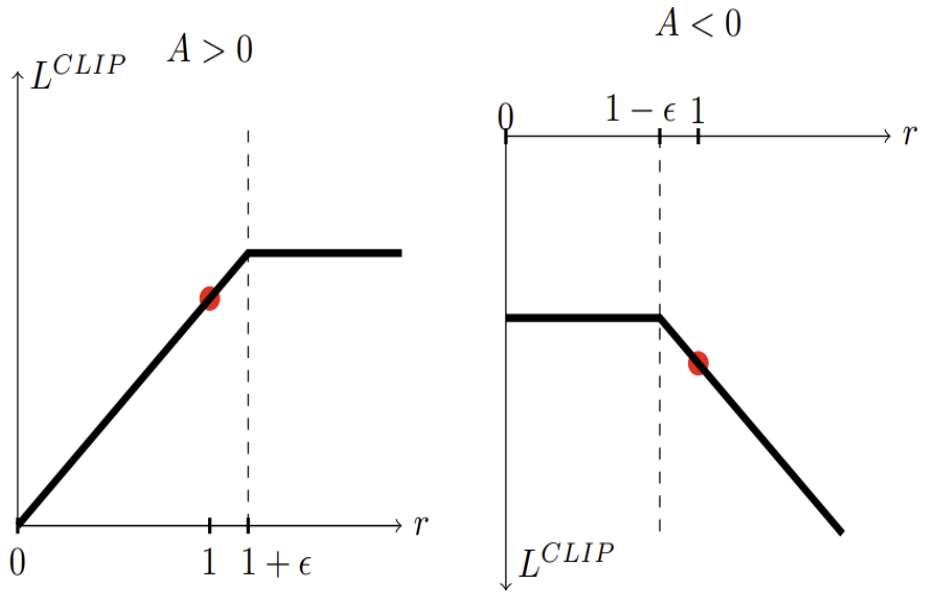
  - 서로 다른 문제에서 잘 작동하는 β를 적절하게 선택하기 어려움

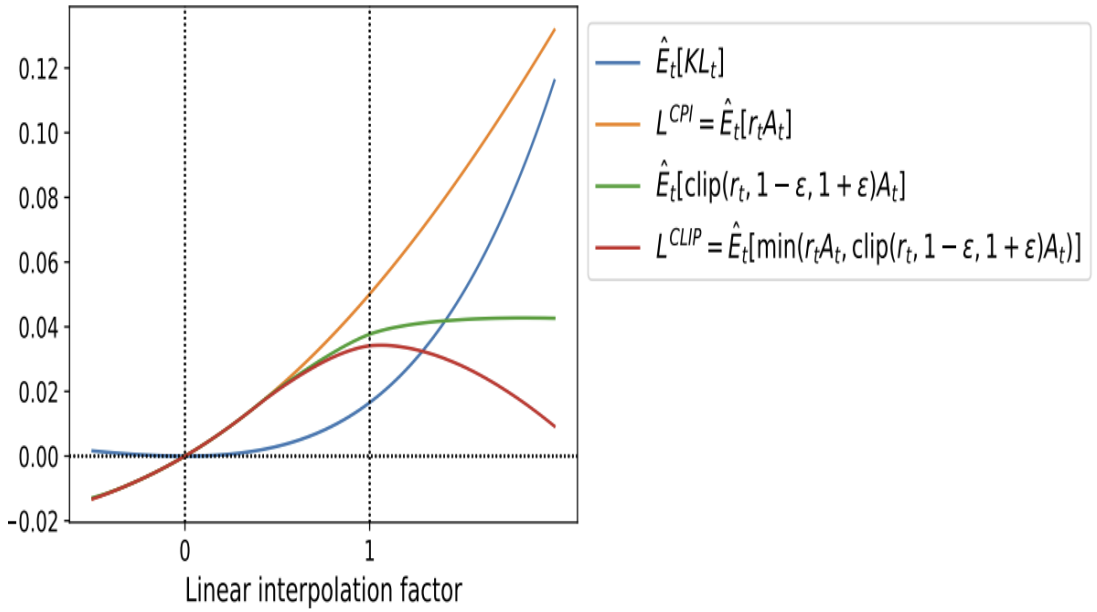
- 고정 β로 페널티 목적 함수를 SGD로 최적화하는 것만으로는 TRPO의 단조로운 개선을 모방하기 어렵고 추가 수정이 필요함

Clipped Surrogate Objective

- TRPO: $L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$

- 새로운 목적함수: $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$





Adaptive KL Penatly Coefficient

- KL 발산에 페널티를 적용하고, 각 정책 업데이트마다 KL 발산이 미리 정해둔 목표 값에 도달하도록 β를 조정하는 방식
- 각 정책 업데이트에서의 수행 단계
  1. KL 페널티 목적 함수 최적화

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

  2. KL 발산 (d) 계산 및 페널티 계수 (β) 조정

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

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

  3. 다음 업데이트에 β 사용

Algorithm

- L^{CLIP}과 L^{KLPEN}은 일반적인 정책 경사 구현에서 약간의 변형만으로 계산/미분 가능
- 정책 학습을 위한 요소
  - 정책 대리 손실과 가치 함수 오차 항을 함께 최적화해야 함
  - 충분한 탐색을 보장하기 위해 엔트로피 보너스를 추가
- PPO: $L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right]$
- 일반적인 정책 경사 구현 방법: $\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T)$
- 더 일반화된 방법(Truncated Generalized Advantage Estimation)

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

PPO, Actor-Critic Style

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

대리 목적 함수 비교

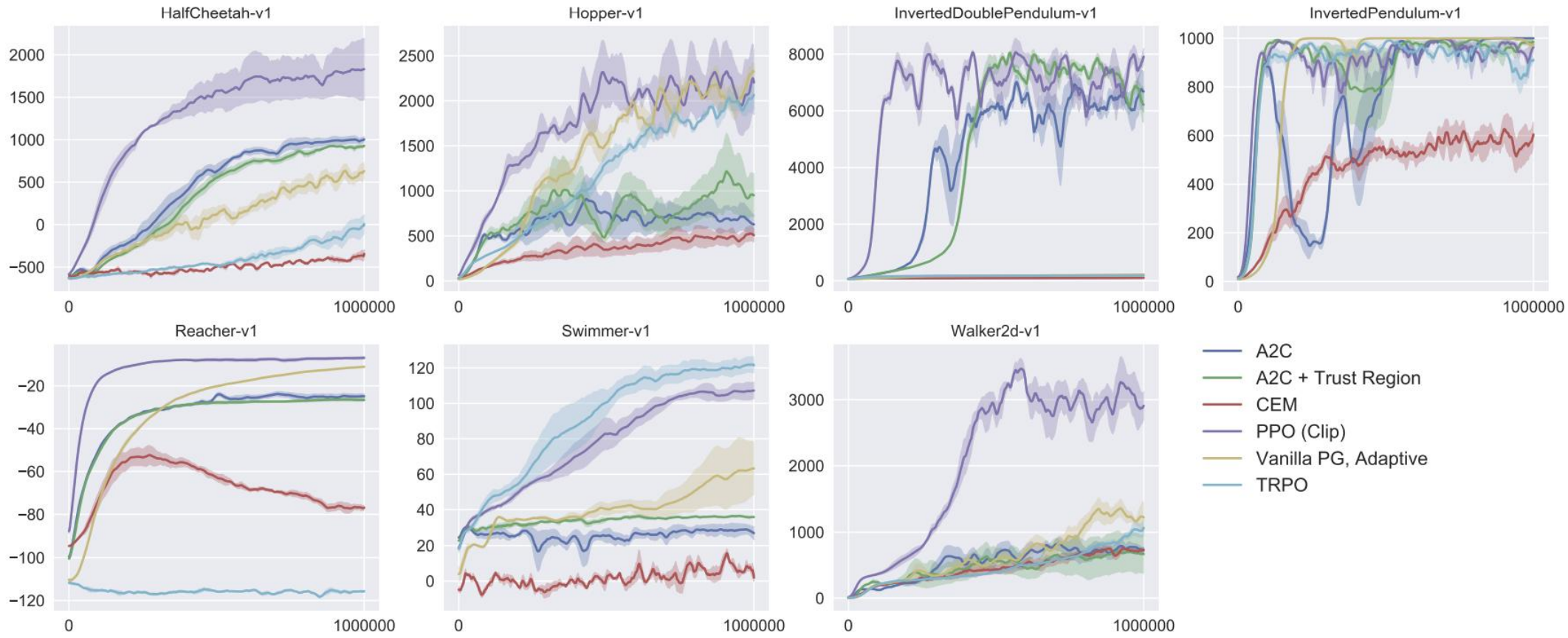
No clipping or penalty: $L_t(\theta) = r_t(\theta)\hat{A}_t$

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

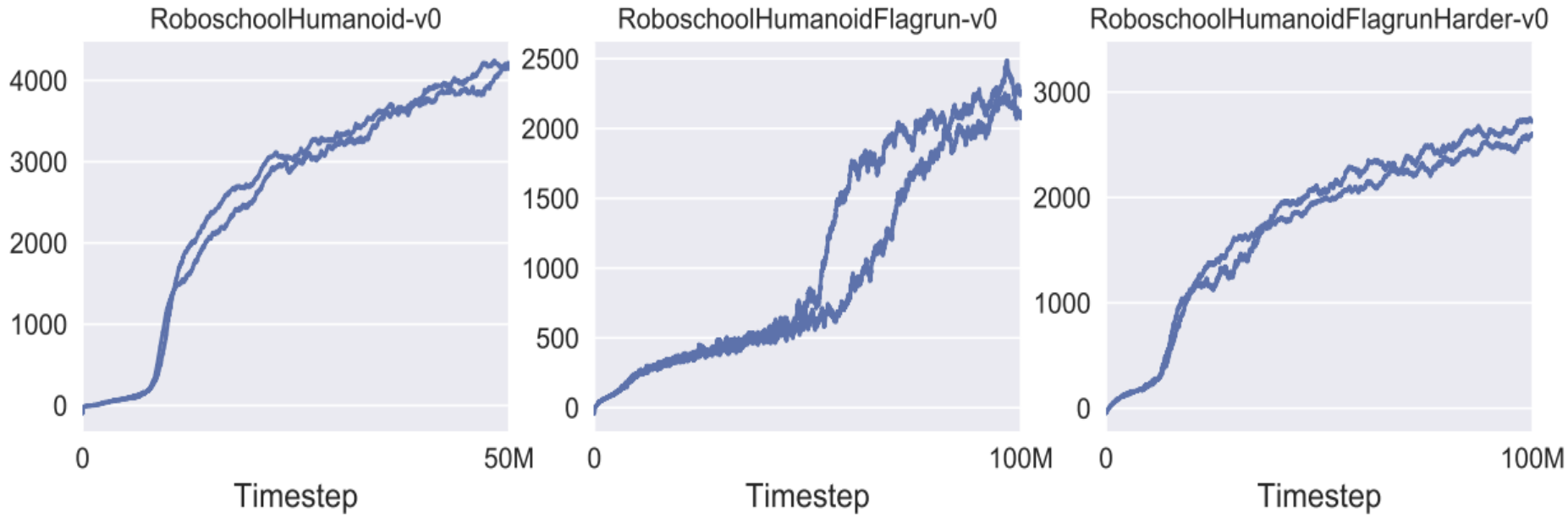
algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	**0.82**
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta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

연속 도메인에서의 비교


HalfCheetah-v1
Hopper-v1
InvertedDoublePendulum-v1
InvertedPendulum-v1
Reacher-v1
Swimmer-v1
Walker2d-v1
A2C
A2C + Trust Region
CEM
PPO (Clip)
Vanilla PG, Adaptive
TRPO

연속 도메인에서의 시연

- Tasks
  - RoboschoolHumanoid: Only 전방 이동
  - RoboschoolHumanoidFlagrun: 목표물의 위치가 200 타입 스텝 또는 목표에 도달할 때마다 무작위로 변함
  - RoboschoolHumanoidFlagrunHarder: 로봇이 큐브에 맞고, 땅에서 일어나야 함



Atari 도메인에서의 비교

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

- (1): 얼마나 빨리 학습하는가
- (2): 훈련 후반에 얼마나 높은 최종 성능에 도달하는가

Conclusion

- PPO는 각 정책 업데이트를 수행하기 위해 여러 에포크의 확률적 경사 상승을 사용하는 정책 최적화 방법들의 한 종류
- TRPO의 안정성과 신뢰성을 가지면서도 구현이 훨씬 간단하여 바닐라 정책 경사 구현에 몇 줄의 코드 변경만 필요함
- 더 일반적인 설정에 적용 가능하며, 전반적으로 더 나은 성능을 보임