6. PPO

경희대학교

기계공학과 RCI 연구실
박보형

2025-2 이동로봇

목차

Before PPO

REINFORCE

- Return이 좋았던 Action은 강화하고 나빴던 Action은 약화시키는 가장 기본적인 PG 알고리즘

$$\nabla_\theta J(\theta) = \mathbb{E}[\, G_t \nabla_\theta \log \pi_\theta(a_t|s_t)\,]$$

한계 : 환경의 Noise가 많으면 Return의 Variance가 커져서 학습이 불안정해짐

Advantage

- State-Value Function을 baseline으로 도입

$$\nabla_\theta J(\theta) = \mathbb{E}[(G_t - b(s_t))\, \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

- Advantage

$$A_t = G_t - V(s_t)$$

$$\nabla_\theta J(\theta) = \mathbb{E}[\, A_t \nabla_\theta \log \pi_\theta(a_t|s_t)\,]$$

- 이제 평균보다 잘한 Action만 강화할 수 있다.

Before PPO

Actor-Critic(A3C)

- Advantage를 계산하려면 Policy $\pi(a|s)$와 State-Value Function $V(s)$ 두 개의 네트워크 필요

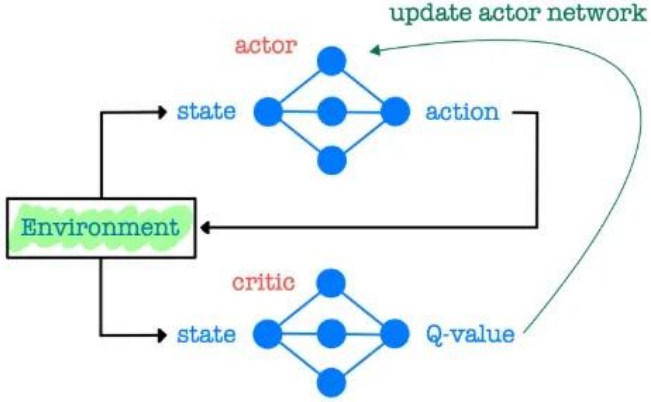
  REINFORCE에서는 $A_t = G_t - V(s_t)$ 에서 Return을 사용하기 때문에 Online Learning이 불가능

  - TD error로 변형해서 Online Learning을 가능하도록 함

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} (G_t - V_\phi(s_t)) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \quad \text{REINFORCE with baseline}$$

$$= \mathbb{E}_{\pi_\theta}\left[(r + \gamma V_\phi(s') - V_\phi(s)) \nabla_\theta \log \pi_\theta(a \mid s)\right] \quad \text{TD Actor-Critic}$$

$$\theta \leftarrow \theta + \alpha A_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$



$$L(\phi) = (G_t - V_\phi(s_t))^2$$

$$\phi - (-(target - V_\phi(s_t))\nabla_\phi V_\phi(s_t)) = \phi + \beta(target - V_\phi(s_t))\nabla_\phi V_\phi(s_t)$$

$$\phi \leftarrow \phi + \beta(G_t - V_\phi(s_t))\,\nabla_\phi V_\phi(s_t)$$

Actor-Critic

Initialize critic network $V(s;\phi)$ and actor network $\pi(a\,|\,s;\theta)$ randomly
Hyperparameters: stepsizes $\alpha > 0$, $\beta > 0$
for episode = $1, M$ **do**
  Initialize s, the first state of the episode
  $I \leftarrow 1$
  for s *is not terminal* **do**
    Select action a according to policy $\pi(\cdot\,|\,s;\theta)$
    Execute a and observe r, s'
    $\delta \leftarrow r + \gamma V(s';\phi) - V(s;\phi)$ — TD method
    $\phi \leftarrow \phi + \beta\,\delta\,\nabla_\phi V(s;\phi)$ — Critic
    $\theta \leftarrow \theta + \alpha I \delta\,\nabla_\theta \log \pi(a\,|\,s;\theta)$ — Actor
    $I \leftarrow \gamma I$
    $s \leftarrow s'$
  end
end

Before PPO

GAE(Generalized Advantage Estimation)

- Advantage
  - Return을 이용해서 사용 : Variance 크고, bias가 작음
  - TD error를 이용해서 사용 : Variance 작고, bias가 큼
- 중간인 n-step Advantage를 사용하자!

$$A_t^{(n)} = G_t^{(n)} - V(s_t)$$

  - 어떠한 n값도 모든 상황에 항상 잘 맞을 수 없다.
- 지수 가중 평균을 내서 사용하자 : GAE

$$A_t^{GAE(\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

γ: discount

λ: bias–variance trade-off (0~1 사이)

$\lambda = 0 \rightarrow$ pure TD(1-step)에 가까움

$\lambda \rightarrow 1 \rightarrow$ Monte Carlo에 가까움

  - 단순 평균을 사용하면 n=1과 n=H(매우 큰 값) 모두 강하게 섞여서 불안정
    - TD error의 기여를 시간에 따라 지수적으로 줄여서 섞자.

Before PPO

Policy Ratio $r_t(\theta)$

- Sample 효율을 높이기 위해서 Rollout을 여러 번 사용해야 함
  - Rollout : 현재 정책으로 환경에서 실행한 최신 trajectory $(s_0, a_0, r_1, s_1, a_1, r_2, \ldots)$
    - On-Policy에서 사용 : 현재 Policy에서 수집된 Sample의 Expectation 사용

$$\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}\left[A^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a \mid s)\right]$$

  - Replay Buffer : 지금까지 수행된 전체 transition tuple (s, a, r, s')
    - Off-Policy에서 사용 : Policy에 상관없음

$$Q^*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q^*(s', a') \mid s, a\right]$$

- Rollout을 여러 번 쓰려고 하니, Sample data를 수집했을 때 Policy의 θ_{old}와 현재 Policy의 θ가 다르다.
  - 과거 Policy로 모든 Sample data를 현재 Policy 기준으로 해석해야 함 : Policy Ratio

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

$r_t > 1$: 새 정책이 그 행동을 예전보다 더 선호
$r_t < 1$: 새 정책이 그 행동을 덜 선호

Before PPO

Importance Sampling

- 원래 분포 p(x)에 대한 Expectation을 구하고 싶다면

$$\mathbb{E}_{x\sim p}[f(x)] = \int f(x)p(x)\,dx$$

- 그런데 Sample을 뽑을 수 있는 분포는 q(x)일 때

$$\mathbb{E}_{x\sim p}[f(x)] = \int f(x)\frac{p(x)}{q(x)}q(x)\,dx = \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right]$$

- 원래 쓰고 싶던 분포 p 대신에 Sample을 뽑을 수 있는 분포 q를 사용하되, $\frac{\mathrm{p}}{\mathrm{q}}$ 의 가중치를 곱해주면 된다.

- 우리가 구하고 싶은 건 $\nabla_\theta J(\theta) = \mathbb{E}_{s_t,a_t\sim\pi_\theta}\left[A_t\,\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$

- 그런데 과거 Sample θ_{old}에 대해서 Importance Sampling을 적용하면

$$\mathbb{E}_{s_t,a_t\sim\pi_\theta}\left[A_t\,\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] = \mathbb{E}_{s_t,a_t\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}A_t\,\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

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

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_t,a_t\sim\pi_{\theta_{\text{old}}}}\left[r_t(\theta)\,A_t\,\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

$r_t(\theta) > 1$:
새 정책이 이 행동을 **더 자주 하고 싶어 한다**
→ 이 샘플의 영향력을 **증폭**시킴.
$r_t(\theta) < 1$:
새 정책이 이 행동을 **덜 자주 하고 싶어 한다**
→ 이 샘플의 영향력을 **줄임**.

Before PPO

Trust Region

- 지금까지의 DRL

$$\nabla_\theta J(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}}\left[r_t(\theta)\, A_t^{GAE(\lambda)}\, \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

- 이때 gradient가 커지는 경우
  - Policy가 한 번에 너무 많이 바뀜
  - Critic이 예측하던 Value가 새로운 Policy에는 맞지 않음
  - Sample 효율이 나빠지고 학습이 불안정해짐

- 핵심 아이디어
  - Policy를 업데이트할 때 너무 급격하게 변하지 않도록 안전 범위 안에서만 움직이자.
  - TRPO 알고리즘에서는 그 기준으로 KL Divergence를 사용
  - 구현이 어렵고 정확히 적용하기 어려움

PPO(Proximal Policy Optimization)

- Trust Region의 개념인 정책의 급변 방지를 KL Divergence 대신에 Ratio Clipping으로 해결

- Policy Ratio

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

$r_t(\theta) > 1$:
새 정책이 이 행동을 **더 자주 하고 싶어 한다**
→ 이 샘플의 영향력을 **증폭**시킴.

$r_t(\theta) < 1$:
새 정책이 이 행동을 **덜 자주 하고 싶어 한다**
→ 이 샘플의 영향력을 **줄임**.

  - Policy Ratio가 너무 크거나 작으면 Policy가 급변한다는 신호
  - 그래서 PPO에서는 이를 기준으로 clipping 적용

$$\text{clip}(r, 1-\epsilon, 1+\epsilon) = \begin{cases} 1+\epsilon, & r > 1+\epsilon \\ 1-\epsilon, & r < 1-\epsilon \\ r, & \text{otherwise} \end{cases}$$

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

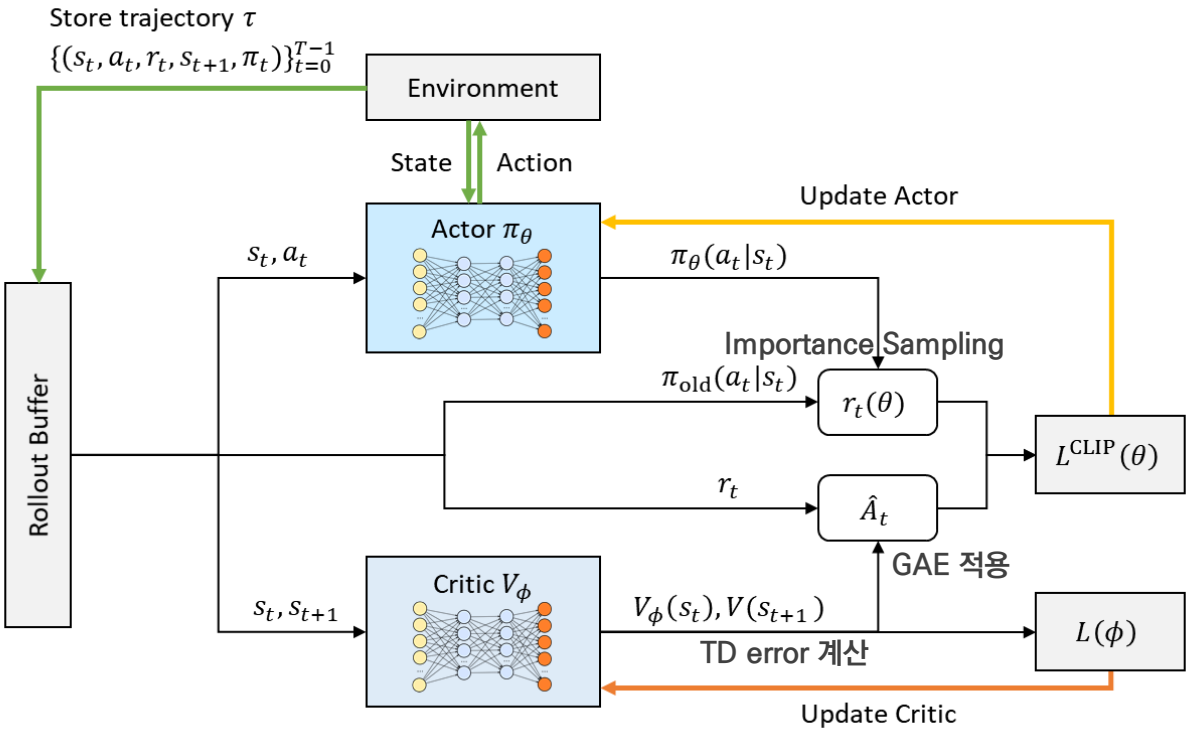
- PPO의 최종 Object Function

$$\nabla_\theta J(\theta) = \mathbb{E}\big[\nabla_\theta L^{CLIP}(\theta)\big]$$

PPO(Proximal Policy Optimization)

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 코드 실습

- Isaac Sim과 PPO를 이용한 코드 실습

감사합니다