2021-2 한양대학교 HAI

강화학습 부트캠프

Chris Ohk
utilForever@gmail.com

오늘 배울 내용

- Trust Regions #2
  - Proximal Policy Optimization (PPO)

PPO - Introduction

- 강화학습의 대표적인 알고리즘
  - DQN
    - 행동 공간이 이산적인 문제들에는 효과적으로 적용 가능하지만,
      연속적인 문제들에도 잘 동작하는지는 검증되지 않았다.
  - A3C - “Vanilla” Policy Gradient Methods
    - 데이터 효율성과 강건성 측면에서 좋지 않다.
      (A3C는 온폴리시라서 한 번 쓴 데이터를 바로 버리기 때문에 데이터 효율성이 좋지 않다.)
  - TRPO
    - 간단히 말해 복잡하다.
    - 노이즈(예 : Dropout)나 매개 변수 공유(정책이나 가치 함수, 보조 작업 등)를 포함하는 아키텍처와 호환성이 없다.

PPO – Introduction

- 강화학습의 대표적인 알고리즘 대비 개선 사항
  - Scalability
    - 더 큰 모델에서 사용 가능하며 병렬로 구현할 수 있다.
  - Data Efficiency
  - Robustness
    - 하이퍼파라미터 튜닝 없이 다양한 문제들에 적용되어 해결

PPO – Introduction

- 제안하는 알고리즘
  - 논문에서 Clipped Probability Ratio를 포함하는 Objective Function을 제안한다.
    - TRPO의 데이터 효율성과 강건성을 유지하면서도 1차 근사만 사용한다.
    - 정책 성능에 대한 Lower Bound를 제공한다.

→ 따라서 정책으로부터 데이터 샘플링과 샘플링된 데이터를 이용한 최적화를 번갈아가면서 수행한다.

Background: Policy Optimization

- Policy Gradient Methods
  - 일반적인 PG 방법들은 실험적으로 Destructive Large Policy Update가 발생한다.
    - 더 자세히 말하자면, PG 방법들이 수행하는 매개 변수 공간에서의 점진적인 갱신이 정책 공간에서는 큰 변화를 유발할 수 있다는 말이다.

Background: Policy Optimization

- Trust Region Methods
  - Policy Update 크기에 대한 제약 조건 하에 Objective Function을 최대화하는 게 목표

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

$$\text{subject to}\ \hat{\mathbb{E}}_t\left[KL\left[\pi_{\theta_{\text{old}}}(\cdot\,|s_t), \pi_\theta(\cdot\,|s_t)\right]\right] \le \delta$$

  - 위의 수식은 제약 조건으로 인해 과도하게 정책이 갱신되는 걸 방지한다.
  - TRPO에서는 제약 조건이 있는 최적화 문제를 풀기 위해 다음과 같은 방법들이 필요하다.

1. Fisher Information Matrix인 KL Divergence의 2차 근사를 사용한다.
   → 2차 행렬을 구하기 위해 많은 계산량이 필요하다.
2. Conjugate Gradient를 사용한다.
   → 구현하기가 어렵다.

Background: Policy Optimization

- Trust Region Methods

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

  - 원래 이론적으로는 제약 조건이 아니라 목표에 패널티를 부여하는 형태다.
    하지만 다양한 문제들(혹은 학습 중에 특성이 변하는 문제)에서 모두 잘 동작하는 β를 찾는 게 어렵기 때문에, TRPO에서는 패널티 대신 제약 조건을 취하는 방식을 택했다.

Clipped Surrogate Objective

- TRPO의 Surrogate Objective Function을 강제적으로 클리핑하는 방법을 알아보자.
- 먼저 기존 TRPO의 Surrogate Function을 다음과 같이 표현한다.

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

$$r_t(\theta_{\text{old}}) = 1$$

- 위의 수식을 이용해 TRPO의 Surrogate Object를 최대화한다.

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

Clipped Surrogate Objective

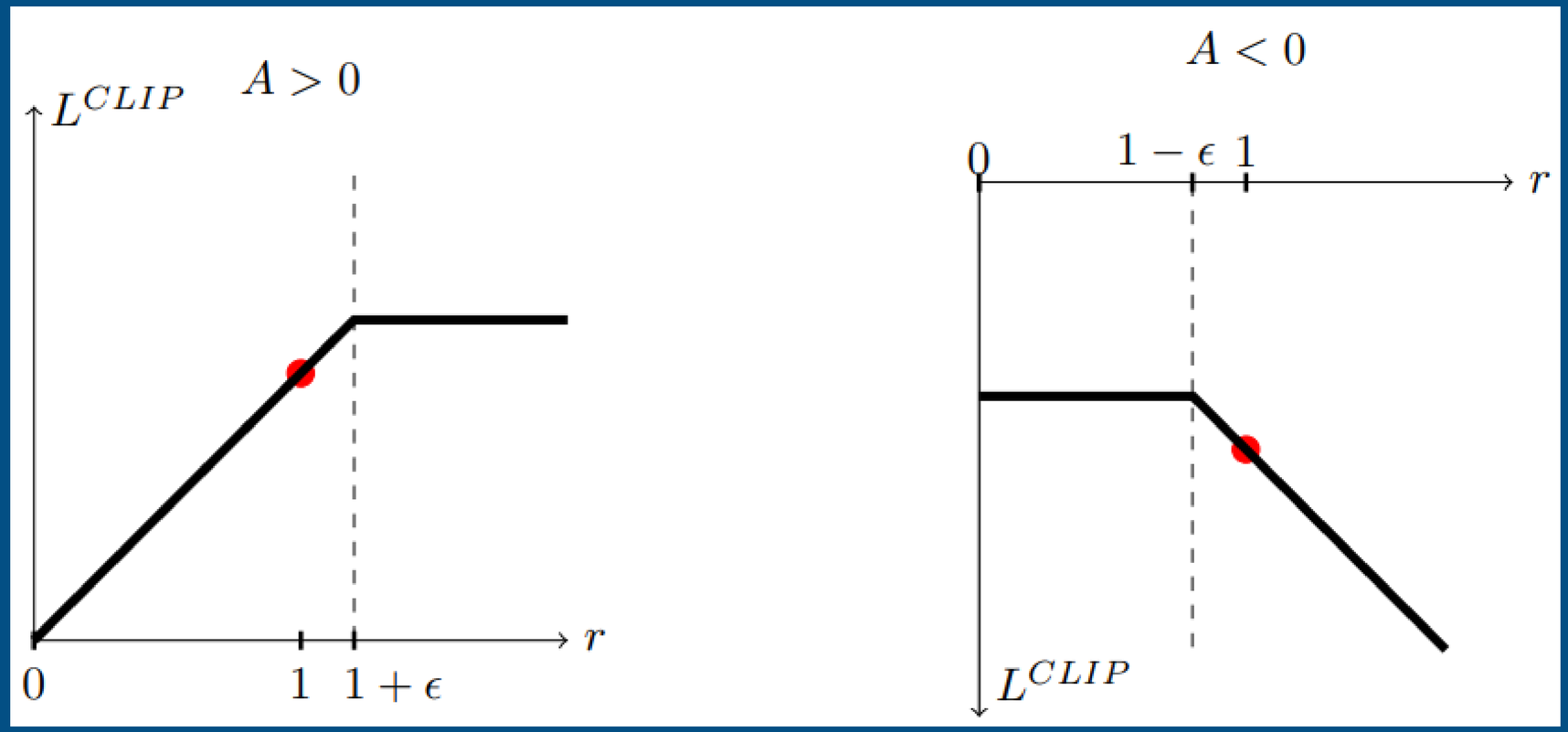
- 이 수식을 그대로 사용한다면 정책이 과도하게 갱신된다.
  따라서 패널티를 이용해 필요 이상의 정책 갱신을 방지한다.
  - TRPO에서는 KL Divergence를 이용해 패널티를 적용한다.
  - PPO에서는 계산적으로 효율적인 패널티를 적용하고
    정책이 과도하게 갱신되는 걸 방지하기 위해 아래와 같은 클리핑 기법을 사용한다.

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

  - ϵ는 하이퍼파라미터로 연속적인 제어 환경에서는 0.2일 때 성능이 가장 좋았으며,
    아타리 게임에서는 $0.1 \times \alpha$ 값을 사용한다. (α는 학습률로 1에서 시작해 진행하며 0으로 감소)
  - $L^{CLIP}(\theta)$ 은 Clipped와 Unclipped Object 중 최솟값을 택해 Unclipped Objective에 대한 Lower Bound가 된다.

Clipped Surrogate Objective

- 두 그래프로(Advantage Function $\hat{A}_t$의 부호에 따라) Clip에 대해 설명하고 있다.



Clipped Surrogate Objective

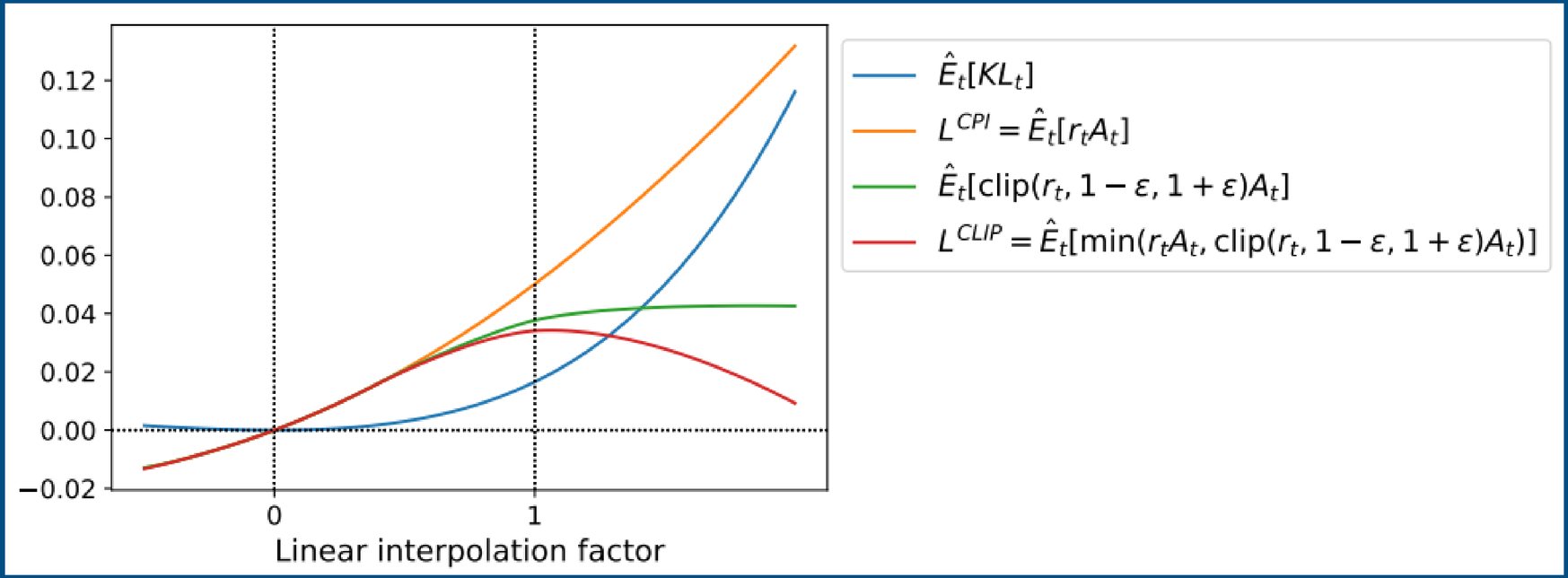
- Advantage Function $\hat{A}_t$가 양수일 때
  - Advantage가 현재보다 좋다라는 뜻이며 매개 변수를 + 방향으로 갱신해야 한다. 다시 말해 어떤 상태 s에서 행동 a가 평균보다 좋다는 의미다. 따라서 이를 취할 확률이 증가하게 되고, $r_t(\theta)$를 클리핑해 ϵ보다 커지지 않도록 유도하는 거다.
  - 추가적으로 TRPO에서 다뤘던 제약 조건이 아니라 단순히 클리핑하는 것이기 때문에 실제로 $\pi_\theta(a_t|s_t)$의 증가량이 ϵ보다 더 커질 수도 있다. 하지만 증가량이 더 커졌다고 하더라도 Objective Function을 갱신할 때 효과적이지 않을 수 있기 때문에 대부분 클리핑을 통해 ϵ 이하로 유지한다.
  - 또한 만약 $r_t(\theta)$가 Objective Function의 값을 감소시키는 방향으로 움직이는 경우더라도 $1-\epsilon$ 보다 작아져도 된다. 왜냐하면 최대한 Lower Bound를 구하는 게 목적이기 때문이다. 쉽게 말해서 $1+\epsilon$을 구하는 데 초점을 맞추고 있고, $1-\epsilon$는 신경쓰지 않고 있는 거다.

Clipped Surrogate Objective

- Advantage Function $\hat{A}_t$가 음수일 때
  - Advantage가 현재보다 좋지 않다라는 뜻이며 매개 변수를 – 방향으로 갱신해야 한다. 다시 말해 어떤 상태 s에서 행동 a가 평균보다 좋지 않다는 의미다. 따라서 이를 취할 확률이 감소하게 되고, $r_t(\theta)$를 클리핑해 ϵ보다 작아지지 않도록 유도하는 거다.
  - 또한 $r_t(\theta)$가 확률을 뜻하는 두 함수를 분자/분모로 갖고 있으며, 분수로 구성되어 있기 때문에 무조건 양수로 이뤄져 있다. Advantage Function인 $\hat{A}_t$과 곱해져 Objective Function인 L^{CLIP}은 Advantage Function과 방향이 같아진다.

Clipped Surrogate Objective

- 다음 그림에서 빨간색 그래프를 보면,
  - L^{CLIP}은 최솟값들의 평균이기 때문에 평균들의 최솟값보다는 더 작아진다.
    즉, 주황색 그래프와 초록색 그래프 중 작은 값보다 더 작아지는 걸 볼 수 있다.
  - L^{CLIP}을 최대화하는 θ가 다음 π_θ가 된다. 다시 말해 PPO는 기존 PG 방법들처럼 매개 변수 공간에서
    매개 변수 θ를 점진적으로 갱신하는 게 아니라, 매번 정책을 최대화하는 방향으로 갱신하는 걸로 볼 수 있다.



Adaptive KL Penalty Coefficient

- 이전까지 설명했던 Clipped Surrogate Objective 방법과 달리 기존의 TRPO에서 Adaptive한 매개 변수를 적용한 또 다른 방법을 알아보자.
  (이 방법은 앞에서 살펴본 Clip을 사용한 방법보다는 성능이 좋지 않다.)
  - Clip에서 다뤘던 Probability Ratio $r_t(\theta)$ 대신 KL Divergence를 이용해 패널티를 준다.
    그 결과 각각의 정책 갱신에 대해 KL Divergence d_{targ}의 Target Value를 얻는다.
    - Clipped Surrogate Objective를 대신하거나 추가적으로 사용할 수 있다.
    - 자체 실험 결과에서는 Clipped Surrogate Objective보다는 성능이 좋지 않았다.

Adaptive KL Penalty Coefficient

- 이 알고리즘에서 각각의 정책 갱신에 적용하는 단계는 다음과 같다.
  - KL Penalized Objective 최적화를 한다. 기존의 TRPO의 Objective Function에 β를 적용해 다음과 같이 β를 Adaptive하게 조절한다.

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

  - 다음을 계산한다.

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

    - 만약 $d < d_{targ}/1.5$이면, $\beta \leftarrow \beta/2$
    - 만약 $d > d_{targ} \times 1.5$이면, $\beta \leftarrow \beta \times 2$
    - 즉, KL Divergence 값이 일정 이상 커지게 되면 Objective Function에 패널티를 더 크게 부과한다.
    - 갱신된 β 값은 다음 정책 갱신 때 사용한다.

Algorithm

- 지금까지는 어떻게 정책을 갱신하는지 설명했다.
  지금부터는 가치 함수, 엔트로피 탐험과 합쳐서 어떻게 통합적으로 갱신하는지 살펴보자.
  - 먼저 Variance-Reduced Advantage Function Estimator를 계산하기 위해 Learned State-Value Function $V(s)$를 사용한다. 여기에는 두 가지 방법이 있다.
    - Generalized Advantage Estimation
    - Finite Horizontal Estimators
  - 만약 정책과 가치 함수 간에 매개 변수를 공유하는 신경망 구조를 사용한다면, Policy Surrogate와 Value Function Error Term을 결합한 Loss Function을 사용해야 한다. 또한 이 Loss Function과 Entropy Bonus Term을 추가해 충분한 탐험이 될 수 있도록 한다.

Algorithm

- 그래서 앞서 다뤘던 Objective Function을 L^{CLIP}라고 표현한다면, PPO에서 제시하는 통합적으로 최대화해야 하는 Objective Function은 이렇게 표현할 수 있다.

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

  - c_1, c_2 : Coefficients
  - S : Entropy Bonus
  - L_t^{VF} : Squared-Error Loss $\left(V_\theta(s_t) - V_t^{\text{targ}}\right)^2$

- 위의 Objective Function을 최대화하면 강화학습을 통한 정책 학습, 상태-가치 함수 학습 및 탐험을 할 수 있다.

Algorithm

- 추가적으로 A3C처럼 인기있는 PG의 스타일에서는 T time-step 동안 정책에 따라서 샘플들을 얻고, 이 샘플들을 갱신할 때 사용한다. 이 방식은 T time-step까지만 고려하는 Advantage Estimator가 필요하며, A3C에서 다음과 같이 사용한다.

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T)$$

  - 여기서 t는 주어진 T 길이의 Trajectory Segment 내에서 $[0, T]$에 있는 Time Index다.
- 위의 수식에 더해 Generalized Version인 GAE는 Truncated Version을 사용한다.

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}$$

$$\text{where } \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

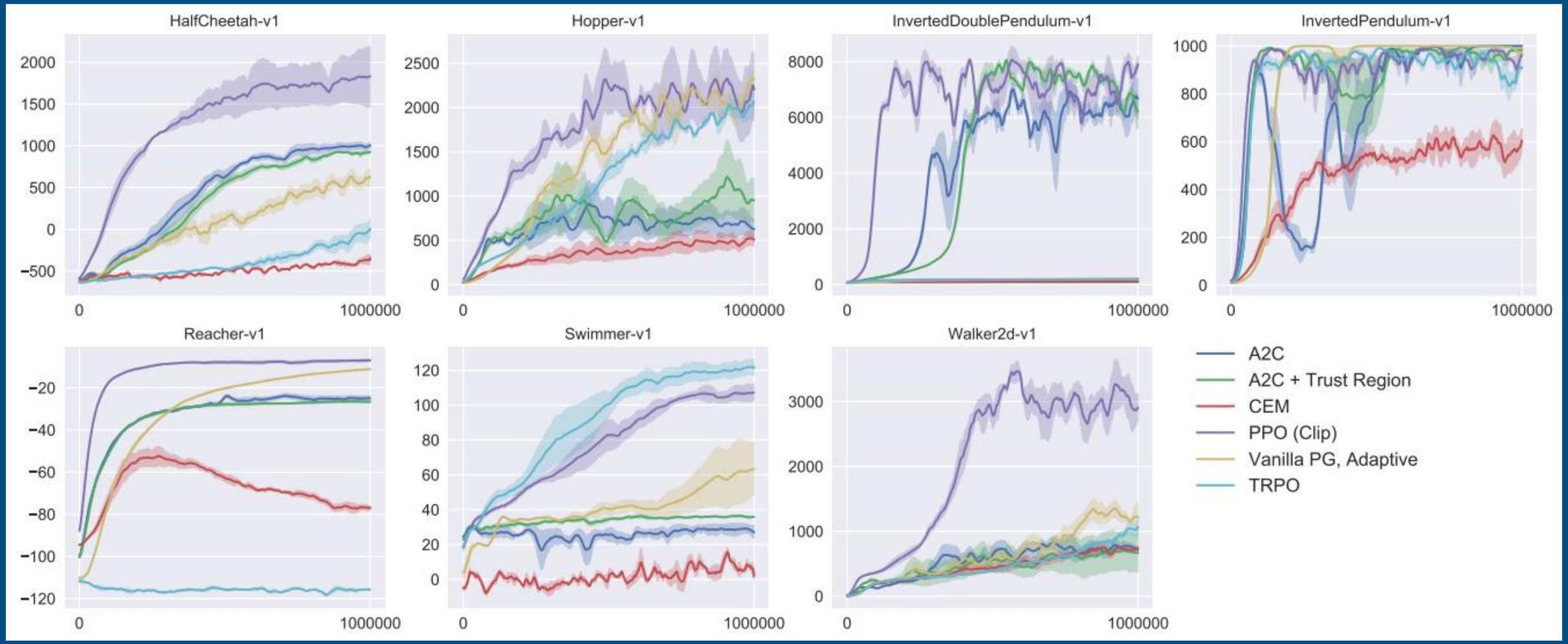
Algorithm

- 그래서 고정된 T 길이의 Trajectory Segment를 사용한 PPO 알고리즘은 다음과 같다.
  - 여기서 액터는 A3C처럼 병렬로 둬도 상관없다.
  - Advantage Estimate 계산을 통해 Surrogate Loss를 계산한다.

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

Experiments



감사합니다!

스터디 듣느라 고생 많았습니다.