강화학습 과제

61조
120250391 이승헌
github link

프로젝트 주제 및 목표

Proximal Policy Optimization Algorithms, Schulman et al, 2017

위 논문을 최대한 비슷한 방법으로 재현실험 및 논문에서 커버되지 않은 파라미터로 재현실험

환경

Gymnasium Mujoco

- Half Cheetah_v5

Half Cheetah - State

Joint Angles - 머리부터 발까지 8개 관절의 상대적인 각도 위치 정보 (8차원)

Joint Velocities - 8개 관절의 각속도 정보 (8차원)

Z-axis Coordinate - 로봇의 몸통이 지면에서 떨어진 높이 , Z좌표 (1차원)

총 17 차원으로 Agent 가 전방향 운동을 제어하기 위해 필요한 모든 정보

각 값들의 범위는 [-Inf , Inf]

Half Cheetah - action

Control Torque - 6개 관절 (앞다리 3개, 뒷다리 3개) 에 가해지는 제어 토크

각 값들의 범위는 [-1, 1]

Half Cheetah - rewards

reward = forward_reward - crtl_cost

1. forward_reward
   - Agent 의 X축 속도가 보상값이 됨. 후진하거나 제자리에 머무르면 0 or 음수값을 가짐
2. Control Cost
   - 최소한의 힘(토크)를 사용하여 전진하는 것을 목표로함
   - Control Cost = 0.1 * sum(a_t^2)

알고리즘 - PPO

Proximal Policy Optimization (PPO)

- 정책이 너무 급격하게 변하지 않도록 제한(clipping)하여 안정이면서 효율적으로 학습하는 알고리즘
- Trust Region Policy Optimization(TRPO)의 안정성과 Policy Gradient 의 단순함을 결합함
- TRPO에 비해 구현이 간단하면서 성능이 좋음

알고리즘 - PPO

기존 Policy Gradient 의 문제

- 한 번의 업데이트가 policy를 너무 크게 바꿈
- 불안정한 학습, 성능 급락 발생

TRPO의 문제

- 알고리즘이 상대적으로 복잡함
- noise 를 포함하거나, parameter를 sharing 하는 구조와 잘 맞지 않음

PPO 의 핵심 아이디어

- KL 제약 대신 확률 비율을 Clipping 하는 간단한 방식을 사용해 안정적 학습 유도

알고리즘 - PPO

PPO-Clipping

$$L^{CLIP} = \min\left(r_t A_t,\ \text{clip}(r_t, 1-\varepsilon, 1+\varepsilon)\, A_t\right) \qquad r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

r_t(θ) : 업데이트 전 후 정책의 행동 확률 비율

Clipping 의 역할 : r_t(θ) 가 [1-Ɛ , 1+Ɛ] 범위를 벗어나지 않도록 강제로 제한함

- **정책이 좋아지는 방향 (advantage > 0) 에서는 너무 크게 바뀌지 않도록 제한**
- **정책이 나빠지는 방향에서는 과한 update 방지**

알고리즘 - PPO

PPO-Adaptive KL Penalty

- KL divergence를 기준으로 penalty β 를 자동 조정하는 방법
- TRPO 처럼 KL divergence 를 활용해 정책 변화량을 제한
- TRPO 와 다르게 KL 이 목표값을 넘어가면 β를 증가, 너무 작으면 β를 감소시키는 구조
  - KL < d_target /1.5 -> $\beta = \beta / 2$ (penalty 완화)
  - KL > d_target *1.5 -> $\beta = 2\beta$ (penalty 강화)
  - 변경된 β 로 다음 update 진행

$$L_{\mathrm{KLPEN}}(\theta) = \hat{E}_t \left[r_t(\theta)\hat{A}_t - \beta\, KL(\pi_{\theta_{\mathrm{old}}}, \pi_\theta) \right]$$

- Clipping PPO 보다 성능이 낮은 결과를 보임

hyperparameter

env_id : 학습 진행할 Gymnasium 환경 이름

total_timesteps : 환경과 상호작용할 전체 timestep 크기 [default = 1000000]

horizon : 한 번의 정책 update에 사용하기 위해 환경에서 수집할 경험의 timestep (Rollout buffer size) [default = 2048]

adam_stepsize : 최적화 함수의 learning rate [default = 3e-4]

n_epochs : 수집된 data (horizon 크기)를 가지고 policy, value 네트워크를 몇 번 반복하여 최적화 할 것인지 결정 [default = 10]

minibatch_size : n_epochs 동안 데이터 셋을 나눌 minibatch 단위 [default = 64]

gamma : Discount factor, 미래 보상을 현재 가치로 얼마나 반영할지 정하는 계수 [default = 0.99]

gae_lamda : Generalized Advantage Estimate 계수 , Advantage 추정 시 variance 와 bias 사이의 균형을 조절 [default = 0.95]

epsilon : clipping 계수, r_t 를 [1-epsilon, 1+epsilon] 로 제한함 [가변값]

beta : KL_penalty 초기 계수 , KL Divergence 가 정책 손실에 미치는 영향력을 조절 [가변값]

kl_target : 목표 KL Divergence, 업데이트 후 이전 정책과 새 정책간의 KL Divergence 목표값, 실제 KL Divergence 값이 목표치를 벗어나면 beta 조정 [가변값]

실험환경

운영체제 : window11

하드웨어 (GPU) : RTX 4070ti

하드웨어 (CPU) : intel Core I9 14900k

Python version : 3.10.16

주요 라이브러리 : PyTorch, Numpy, Gymnasium

실행 방식 : Single Process

평가 기준

1. Episode Reward
   a. 학습 기록 : 학습 중 매 update 시점마다 최근 10개의 episode 평균 reward를 기록
   b. 최종 성능 : 전체 학습 종료 후, 기록된 모든 episode 중 마지막 100개의 episode 의 평균 reward로 평가

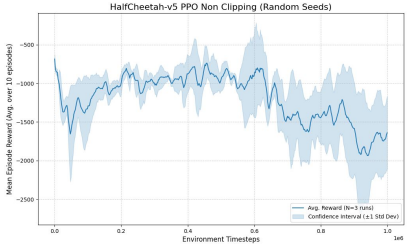
2. 신뢰구간
   a. 3개의 무작위 랜덤시드([1, 10000])를 사용하여 학습을 진행한 뒤 평균 reward 계산
   b. 평균 reward 에서 ±표준편차를 통해 신뢰구간을 표현함으로써 안정성 확보

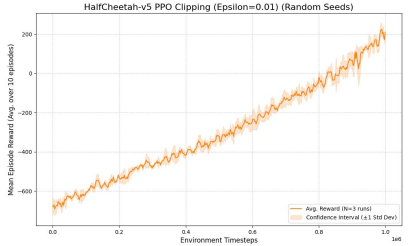
실험 결과 - reward table

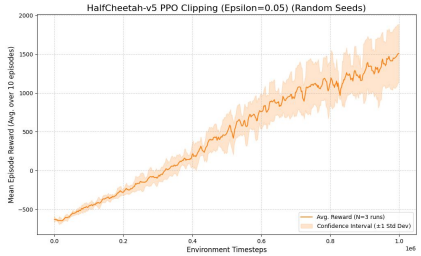
non clipping	-1771.19
clipping, ε = 0.01	139.04
clipping, ε = 0.05	1386.36
clipping, ε = 0.1	1751.20
clipping, ε = 0.2	1968.70
clipping, ε = 0.3	1134.35
clipping, ε = 0.5	-83.35
fixed β = 0.3	-466.62

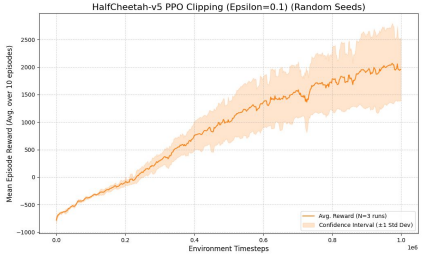
fixed β = 1	-82.10
fixed β = 3	-183.51
fixed β = 10	35.96
adaptive KL d_target=0.001	-44.06
adaptive KL d_target=0.003	226.00
adaptive KL d_target=0.01	-517.93
adaptive KL d_target=0.03	-54.10
adaptive KL d_target=0.05	-332.33

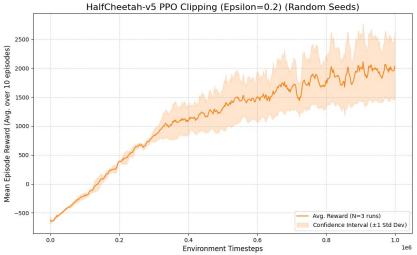
실험 결과 - Graph

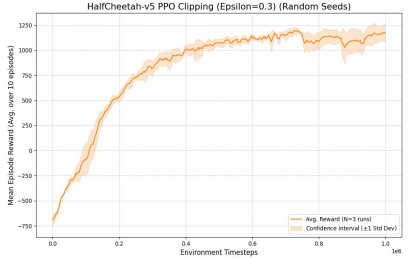

HalfCheetah-v5 PPO Non Clipping (Random Seeds)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)

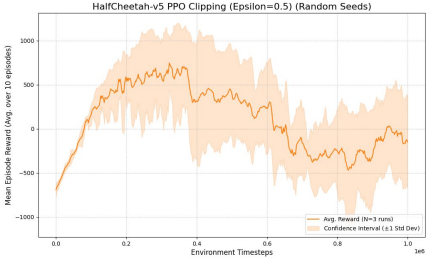

HalfCheetah-v5 PPO Clipping (Epsilon=0.01) (Random Seeds)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)

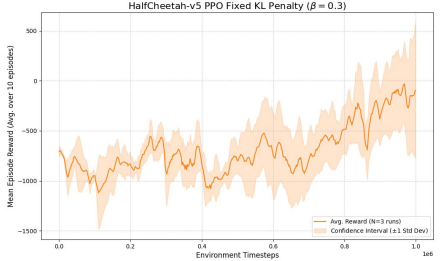

HalfCheetah-v5 PPO Clipping (Epsilon=0.05) (Random Seeds)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)


HalfCheetah-v5 PPO Clipping (Epsilon=0.1) (Random Seeds)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)

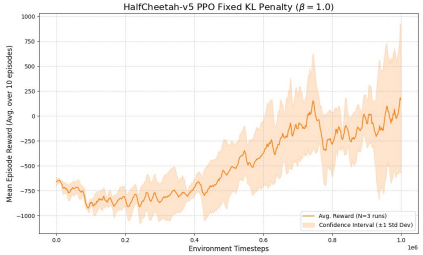

HalfCheetah-v5 PPO Clipping (Epsilon=0.2) (Random Seeds)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)

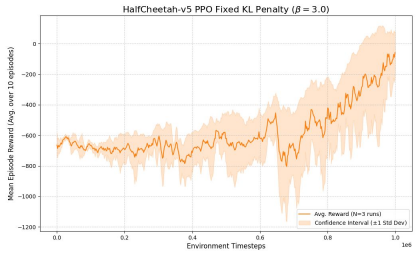

HalfCheetah-v5 PPO Clipping (Epsilon=0.3) (Random Seeds)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)

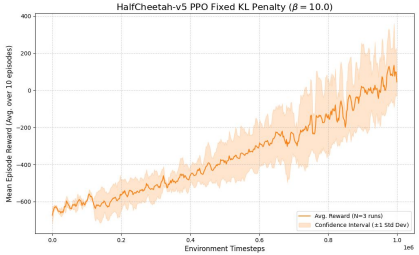

HalfCheetah-v5 PPO Clipping (Epsilon=0.5) (Random Seeds)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)

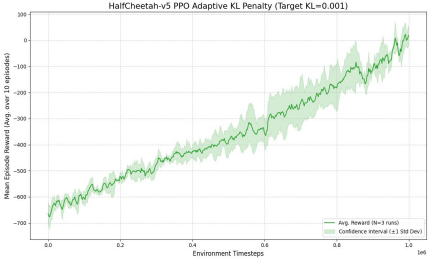

HalfCheetah-v5 PPO Fixed KL Penalty ($\beta = 0.3$)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)

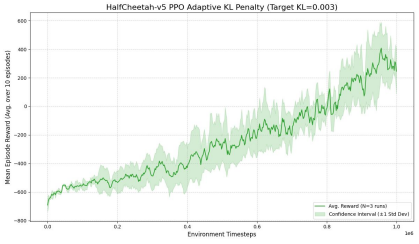
실험 결과 - Graph

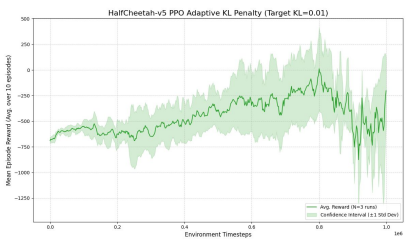

HalfCheetah-v5 PPO Fixed KL Penalty ($\beta = 1.0$)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)

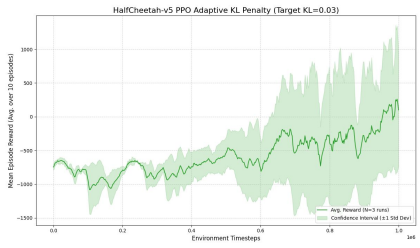

HalfCheetah-v5 PPO Fixed KL Penalty ($\beta = 3.0$)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)

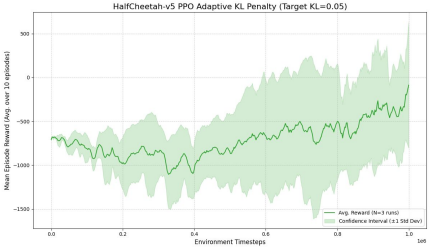

HalfCheetah-v5 PPO Fixed KL Penalty ($\beta = 10.0$)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)


HalfCheetah-v5 PPO Adaptive KL Penalty (Target KL=0.001)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)


HalfCheetah-v5 PPO Adaptive KL Penalty (Target KL=0.003)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)


HalfCheetah-v5 PPO Adaptive KL Penalty (Target KL=0.01)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)


HalfCheetah-v5 PPO Adaptive KL Penalty (Target KL=0.03)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)


HalfCheetah-v5 PPO Adaptive KL Penalty (Target KL=0.05)
Mean Episode Reward (Avg. over 10 episodes)
Environment Timesteps
1e6
Avg. Reward (N=3 runs)
Confidence Interval (±1 Std Dev)

실험 결과

1. PPO Algorithm 은 HyperParameter 에 매우 민감하게 반응한다는 사실을 확인함
2. 원 논문에 비해 Clipping 과 fixed, adaptive KL 의 성능 차이가 크게 나타남
3. 원 논문에서 권장된 HyperParameter 설정에서만 안정적이고 우수한 성능을 보였으며, 그 외의 설정에서는 학습이 불안정하거나 저조한 보상에 머무름

보완 및 개선사항

본 프로젝트에서는 HalfCheetah 1가지 환경에서만 실험을 진행하여 알고리즘 일반화 능력 검증에 한계가 존재함. 원 논문이 7가지 Gymnasium Mujoco 환경에서의 평균 성능을 통해 신뢰도를 확보했듯이, Hopper, Walker2d 등 7가지 환경 전체에 걸쳐 실험을 확장하고 성능을 비교 분석하여 구현된 알고리즘의 견고성과 일반화 능력을 검증 하는 방향으로 개선해야 함