서강대학교

RL Term Project
TD3의 모든 것

GitHub : https://github.com/cjw94103/RL_Term_Project_A71054

과목명 : 강화학습의 기초
지도교수 : 소정민 교수님
제출일 : 2025-12-07
학번 : A71054
이름 : 최재원

목차

1. 프로젝트 주제 및 목표

프로젝트 목표 및 범위

주제	• TD3 알고리즘의 하이퍼파라미터 민감도 분석 : MuJuCo Continuous Control 환경에서의 실증 연구
범위	• TD3(Twin Delayed Deep Deterministic Policy Gradient) 알고리즘을 중심으로 연구를 진행 • HalfCheetah-v4 및 Ant-v4 환경을 기반으로 Continuous Control 알고리즘의 특성을 분석 • 27개의 하이퍼파라미터 조합 중 유망한 후보군을 식별 후 정량적, 정성적 분석을 통한 영향도 파악

목표

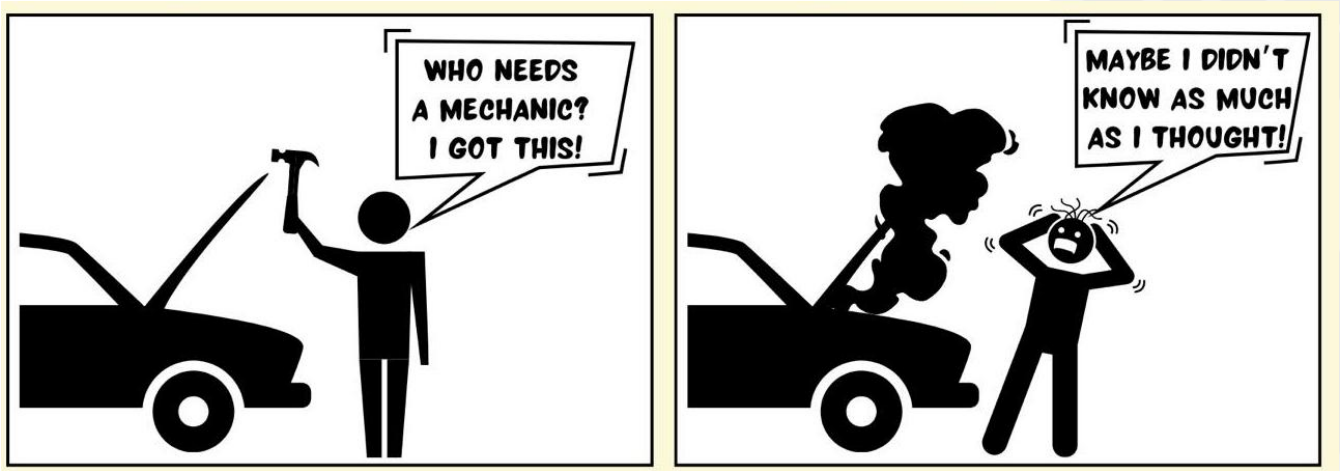
알고리즘 재현 및 검증	하이퍼파라미터 영향력 분석	실무적 가이드라인 제시
• TD3 논문을 심층 분석하고 핵심 알고리즘을 구현하여 재현성을 검증 • 논문에서 명시되지 않은 하이퍼파라미터를 식별하고 적절한 값을 설정하여 완전한 구현을 완성	• 핵심 하이퍼파라미터를 체계적으로 변화시키며 실험을 수행함 • 각 파라미터가 학습 성능, 안정성, 수렴 속도에 미치는 영향을 정량적 및 시각적으로 분석함	• 환경 특성에 따른 최적의 하이퍼파라미터 조합을 도출 • 실제 강화학습 프로젝트 적용 시 활용 가능한 하이퍼파라미터 선택 전략을 제시 • 실험 결과를 통한 정량적 근거도 함께 제공

2. TD3 이론적 배경

연구 배경

Continuous Control의 Overestimation 문제

- Discrete action space에서는 approximation error로 인한 value overestimation 문제가 잘 연구되어 있지만 continuous control task에서의 actor-critic method에서는 이 문제가 거의 다뤄지지 않음
- Temporal Difference Learning에서 부정확한 estimation이 반복적으로 사용되면서 error가 누적됨
- Overestimation bias로 인해 실제로는 나쁜 state를 높은 value로 잘못 추정하게 되어 policy update에 악영향을 끼침



논문의 Main Contribution

서강대학교

TD3 알고리즘의 Core Components

Clipped Double Q-Learning

- 두 개의 독립적인 critic을 사용하여 overestimation bias를 감소
- 낮은 value estimate를 선택함으로써 underestimation을 선호 (학습 과정에서 전파되지 않음)

Delayed Policy Updates

- Value estimate가 충분히 수렴할 때까지 policy update를 지연
- Value function과 policy의 결합 문제 해결

Target Policy Smoothing

- SARSA 방식의 update를 통해 유사한 action의 estimate를 bootstrapping
- Variance를 추가로 감소시키는 regularization 효과

알고리즘의 우수성

- DDPG 알고리즘을 기반으로 function approximation error의 상호작용을 고려하여 개선함
- OpenAI Gym의 7개 continuous control 환경에서 기존 최신 기법들을 큰 폭으로 능가함

Overestimation bias in actor-critic

Overestimation이 왜 생기나?

- Actor는 근사된 Critic Q_θ의 gradient를 따라 policy를 업데이트를 수행하는데, 이 과정에서 estimated value가 true value보다 체계적으로 높아지는 bias가 발생함

발생 과정

- π_{approx} : 근사 Critic Q_θ의 gradient로 업데이트한 policy
- π_{true} : 실제 가치 Q^π의 gradient로 업데이트한 policy일 때, 아래가 성립함

$$\mathbb{E}[Q_\theta(s, \pi_{\text{approx}}(s))] \geq \mathbb{E}[Q_\theta(s, \pi_{\text{true}}(s))].$$

- 즉, Q_θ를 최대화하는 방향으로 업데이트했으므로, Q_θ 기준으로는 π_{approx}가 π_{true} 보다 "더 좋아 보이도록" 값이 증가함
- 반대로, 실제 가치 Q^π로 보면, π_{true} 가 π_{approx}보다 더 낮고, 즉, 아래의 수식이 성립함

$$\mathbb{E}[Q^\pi(s, \pi_{\text{true}}(s))] \geq \mathbb{E}[Q^\pi(s, \pi_{\text{approx}}(s))].$$

- 두 부등식을 합치면, 아래의 수식이 유도됨

$$\mathbb{E}[Q_\theta(s, \pi_{\text{approx}}(s))] \geq \mathbb{E}[Q^\pi(s, \pi_{\text{approx}}(s))]$$

- 즉, π_{approx} 의 "추정 가치가 실제 가치보다 항상 크거나 같음" → Overestimation bias

Solution 1 : Clipped Double Q-Learning

Underestimation 선호 전략

- Overestimation bias가 있는 value estimate를 실제 value의 approximate upper-bound로 활용함
- Value가 낮게 추정된 action이 policy에 의해 회피되기 때문에, underestimate은 학습 과정에서 전파되지 않으며, 따라서 overestimation 보다 underestimation을 선호하는 것이 안전함

Critic Network

- 두 개의 독립적인 critic 네트워크를 사용하며, 각각 독립적으로 학습됨
- Target value 계산시 아래와 같은 전략을 사용함

$$y_1 = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \pi_{\phi_1}(s')).$$

- 즉, 두 critic 중 더 낮은 값을 선택하여 target value로 취함 → Clipping

Actor Network

- Actor는 첫 번째 critic network만 사용하여 업데이트를 수행함

$$\nabla_\phi J(\phi) = \mathbb{E}_s \left[\nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s) \right]$$

Solution 2 : Delayed Policy Updates

Value estimate의 수렴을 기다리는 전략

- Critic이 여러 번 업데이트되는 동안, Actor 업데이트를 delay 시키는 방법
- Value estimate가 충분히 수렴할 때까지 policy update를 미루어 더 정확한 gradient를 사용함

효과 1 : Variance 감소

- Policy gradient의 variance가 줄어들어 학습이 더 안정적임
- 일관된 gradient 방향으로 학습이 진행됨

효과 2 : Error 누적 억제

- Value function의 부정확성이 policy에 즉각적으로 반영되지 않아 error의 누적을 방지함
- Critic이 더 정확해질 시간을 확보할 수 있음

효과 3 : 계산 효율성

- Actor 업데이트는 critic 업데이트보다 계산 비용이 큼
- 업데이트 빈도를 줄임으로써 전체 학습 속도가 향상될 수 있음

효과 4 : 학습 안정성

- Policy와 value function이 서로 안정적으로 발전할 수 있는 환경을 조성함
- 발산 위험을 크게 줄일 수 있음

Solution 3 : Target Policy Smoothing

Similar Actions의 Similar Values 가정

- 비슷한 action들은 비슷한 value를 가져야 한다는 직관을 활용함
- Action space에서 가까운 action들의 value를 함께 고려하여 smoothing 효과를 만듦

Regularization을 통한 Variance 감소

- Target action에 noise를 추가하여 주변 action들의 value도 함께 고려함
- 이는 SARSA 방식의 update와 유사한 효과를 내며, bootstrapping similar actions를 통해 variance를 추가로 감소시킴

구현 방법

- Target policy smoothing을 적용한 수식은 아래와 같음

$$\tilde{a} = \pi_\phi(s') + \epsilon, \qquad \epsilon \sim \mathrm{clip}\big(\mathcal{N}(0, \sigma), -c, c\big)$$

- 즉, action에 [-c, c] 범위의 가우시안 노이즈를 추가한 형태이며, 이에 따른 target value는 아래와 같음

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i}(s', \tilde{a})$$

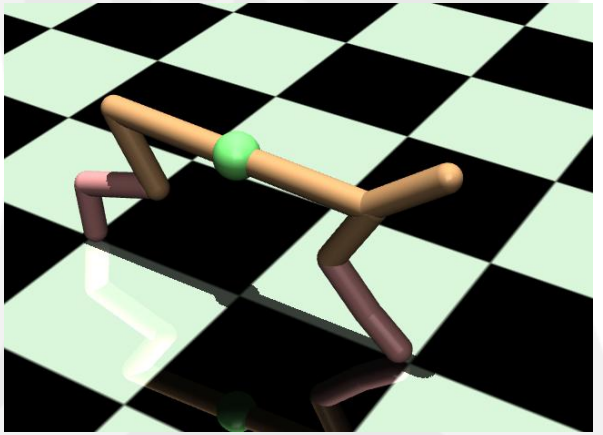
- c의 범위는 TD3를 학습할 때, 중요한 하이퍼파라미터 중 하나이며, noise를 [-c, c] 범위로 clipping하여 target action이 너무 멀리 벗어나지 않도록 함
- 이는 smoothing과 action의 유효성 사이의 균형을 유지함

3. 실험 셋업

실험 환경 1 : HalfCheetah

기본 정보	• 유형 : MuJoCo 기반 continuous control environment • 목표 : 2D 평면에서 cheetah 로봇을 최대한 빠르게 전진시키기 • 특징 : 비교적 단순한 구조로 continuous control 알고리즘의 성능을 평가하는 표준 벤치마크
State 정보	• 17차원 연속 벡터 • 위치 : 로봇 몸통의 각도, 6개 관절의 각도 • 속도 : 로봇 몸통의 속도, 6개 관절의 각속도 • 로봇의 현재 상태를 완전히 관측 가능함
Action 정보	• 6차원 연속 벡터 • 6개 관절 (앞/뒷 다리의 hip, knee, ankle)에 적용할 torque(힘) • [-1, 1] 범위로 정규화 • continuous control로 부드러운 움직임 생성
Reward 정보	• Forward Reward : 전진 속도에 비례 (빠를수록 높은 보상) • Control Cost : Action 크기에 대한 패널티 (에너지 효율성 유도) • 목표 : 최소한의 힘으로 최대한 빠르게 전진 → Forward Reward - Control Cost

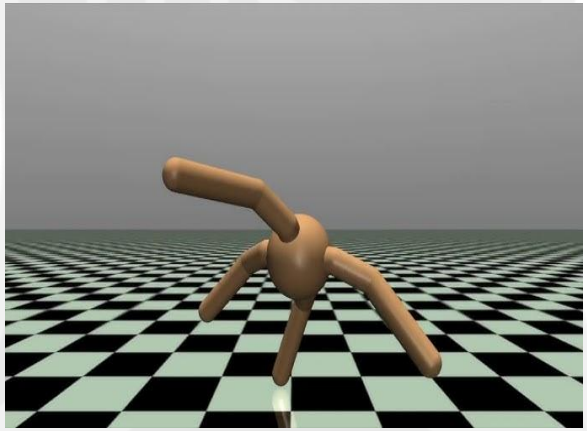
예시 이미지



실험 환경 2 : Ant

기본 정보	• 유형 : MuJoCo 기반 continuous control environment • 목표 : 3D 공간에서 4족 로봇(개미)를 제어하여 최대한 빠르게 전진시키기 • 특징 : HalfCheetah 보다 복잡한 구조로 더 어려운 control task
State 정보	• 27차원 연속 벡터 • 위치 : 로봇 몸통의 3D 위치 및 방향, 8개 관절의 각도 • 속도 : 로봇 몸통의 3D 선속도 및 각속도, 8개 관절의 각속도 • 3D 공간에서 로봇의 완전한 상태를 관측
Action 정보	• 8차원 연속 벡터 • 4개 다리의 8개 관절(각 다리당, hip, ankle)에 적용할 torque • 범위 : [-1, 1]로 정규화 • 4족 보행을 위한 다리 간 협응 제어 필요
Reward 정보	• Forward Reward : 전진 속도에 비례 (빠를수록 높은 보상) • Control Cost : Action 크기에 대한 패널티 (에너지 효율성) • Contact Cost : 비정상적인 접촉에 대한 패널티 • Survive Reward : 매 step마다 생존 보상 (+1.0) • 목표 : 균형을 유지하며 에너지 효율적으로 빠르게 전진

예시 이미지



TD3의 주요 하이퍼파라미터

유형	파라미터	논문 기본 값	설명
Network	Actor learning rate	3e-4	Actor network의 learning rate
	Critic learning rate	3e-4	Critic network의 learning rate
	Batch size	256	Mini-batch size
Exploration	Exploration noise	0.1	학습 중 action에 추가되는 noise
Target update	Target update rate	0.005	Target network의 soft update의 비율
	Discount factor	0.99	미래 보상 할인율
Policy smoothing	Target policy noise	0.2	Target action에 추가되는 noise
	Noise clip range	0.5	Target noise의 clipping 범위
Delayed Update	Policy update frequency	2	Actor update delay interval
Replay Buffer	Buffer size	1e6	Replay Buffer의 최대 크기
	Initial random steps	10000	초기 random exploration steps
Training	Total timesteps	1e6	전체 학습 timesteps
	Evaluation frequency	5000	Policy(Actor) evaluation 주기

하이퍼파라미터 튜닝 범위

서강대학교

실험 개요

- TD3의 세 가지 핵심 매커니즘과 직접적으로 연관된 하이퍼파라미터를 튜닝 대상으로 선정
- 각 하이퍼파라미터의 값을 범위에 따라 3개를 선정하고 grid search를 통해 튜닝 수행 (27 cases)
- 나머지 하이퍼파라미터는 논문에서 설정한 기본 값을 사용

파라미터 값 범위

- Target policy noise : [0.1, 0.2, 0.3]
- Noise clip range : [0.3, 0.5, 0.7]
- Policy update frequency : [1, 2, 3]

Evaluation Metric

- Training : 5,000 step 단위 10 episode의 평균 return 값
- Inference : 10개의 다른 seed를 사용하여 평균 return 및 표준편차 산출

선정 이유

Target Policy Noise	Noise Clip Range	실무적 가이드라인 제시
• TD3의 핵심 기법 중 하나인 Target policy smoothing을 직접 제어하는 파라미터 • Critic의 부정확한 value estimate에 대한 overfitting을 방지하는 regularization 역할 수행 • Variance 감소와 학습 안정성에 직접적인 영향을 끼침	• Noise를 clipping하여 과도한 smoothing 방지 • 너무 큰 noise는 target value의 confidence를 해칠 수 있음 • Target policy noise와 함께 조정되어야 하는 dependent 파라미터임	• TD3의 핵심 기법 중 하나인 Delayed policy updates를 직접 제어하는 파라미터 • Value function과 policy의 결합 문제를 해결함 • Critic의 수렴 시간을 확보하여 더 정확한 policy gradient를 사용할 수 있게 함

4. 실험 결과

환경별 최고 성능 비교

서강대학교

HalfCheetah-v4

환경	Target Policy Noise	Noise Clip	Policy Frequency	평균 return	표준편차
baseline	0.2	0.5	2	7597	62
ours	0.3	0.5	1	8813	81
차이	-	-	-	1324	-

Ant-v4

환경	Target Policy Noise	Noise Clip	Policy Frequency	평균 return	표준편차
baseline	0.2	0.5	2	2999	381
ours	0.3	0.5	2	5084	84
차이	-	-	-	1973	-

- 학습 완료 후, 10개의 random seed를 사용하여 각 episode에 대한 return의 평균 및 표준편차를 기록
- baseline : 논문에서 제시한 파라미터의 기본 값을 이용한 학습 결과를 기록
- ours : 나머지 하이퍼파라미터 중 평균 return이 가장 높은 학습 결과를 기록
- 두 환경 모두 Policy Noise가 0.3이 optimal이며, 논문의 기본 설정인 0.2는 conservative한 선택이었을 가능성이 있음

Baseline 기준 HalfCheetah-v4 결과 분석

서강대학교

Policy Noise : 0.2 → 0.3 증가

- 더 강한 exploration이 HalfCheetah의 복잡한 locomotion 학습에 도움
- HalfCheetah는 다리 coordination이 중요한데, 더 많은 exploration이 다양한 움직임에 대한 패턴 학습을 촉진함

Policy Update Frequency : 2 → 1 감소

- 더 빈번한 policy update가 이 환경에서 효과적
- HalfCheetah는 비교적 안정적인 환경이라 aggressive update가 가능하며 더 도움이 될 수 있음을 보여줌

표준편차 비교

- Baseline : 62 → 상대적으로 안정적임
- Ours : 81 → 약간 증가함
- 성능이 향상되었지만 variance가 소폭 증가함 → 전형적인 exploration-exploitation trade-off 현상이 발생함

Baseline 기준 Ant-v4 결과 분석

Policy Noise : 0.2 → 0.3 증가

- Baseline 보다 50% 더 강한 exploration을 수행함
- 이로 인해, high-dimensional space에서 더 넓은 영역을 탐색할 수 있고, local optima 탈출에 유리함

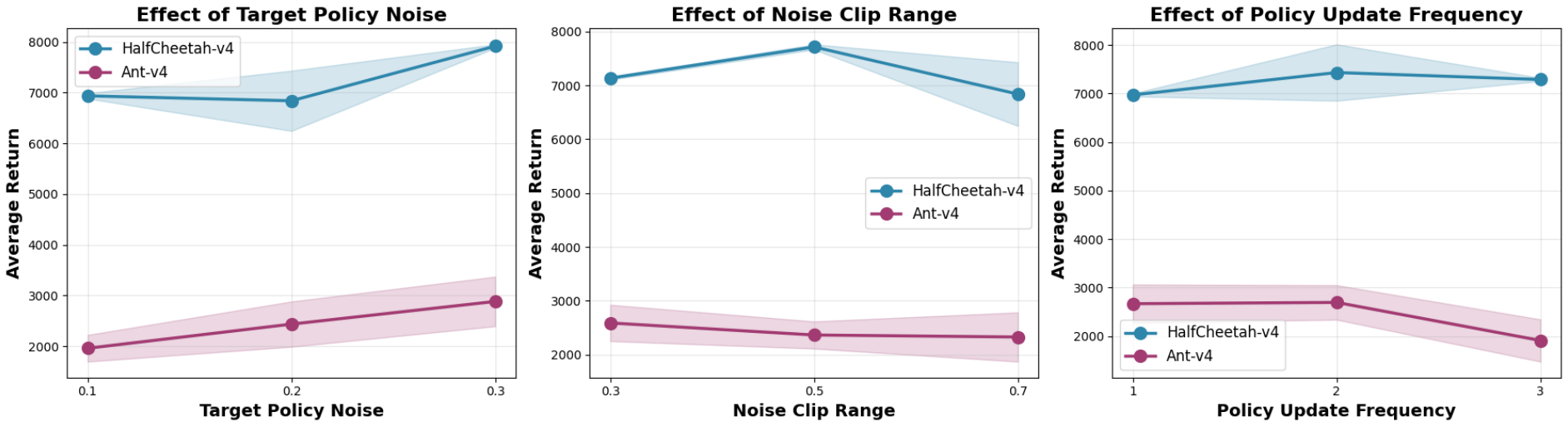
Policy Frequency 2 유지

- Ant처럼 복잡한 환경에서는 critic이 충분히 안정화되는 것이 중요함
- 너무 빈번한 policy update는 학습의 불안정성을 야기함
- 이는 TD3의 “delayed policy update” 철학과 일치함

표준편차 분석

- 단순히, 평균 return만 향상된 것이 아닌 표준편차가 큰 폭으로 감소함 (baseline : 381 → ours : 84)
- 이는 알고리즘의 robustness가 극적으로 개선된 것을 의미함
- Evaluation seed 간 일관성이 크게 향상

하이퍼파라미터의 민감도 분석



Target Noise의 효과 (왼쪽 그래프)

- 두 환경 모두 policy noise가 증가할수록 성능이 향상되며, 특히 0.3에서 최고 성능을 보임
- HalfCheetah-v4는 0.1(7,000)과 0.2(6,900)에서 비슷한 성능을 보이다가 0.3(8,000)에서 급격히 향상되는데, 이는 일정 임계값을 넘으면 강한 exploration이 더 효율적인 이동 패턴 발견에 도움이 됨을 시사함
- Ant-v4는 일관된 상승세를 보임 : 0.1(2,000) → 0.2(2,500) → 0.3(2,900), 총 45% 향상
- 8개의 관절을 가진 high-dimensional action space에서는 각 step마다 exploration 증가가 지속적으로 도움이 됨
- 두 환경 모두 TD3 논문의 기본값(0.2)보다 0.3에서 최적 성능을 보이며, 원논문은 다양한 태스크에서의 안정성을 우선시한 보수적 선택이었던 것으로 보임

하이퍼파라미터의 민감도 분석

서강대학교

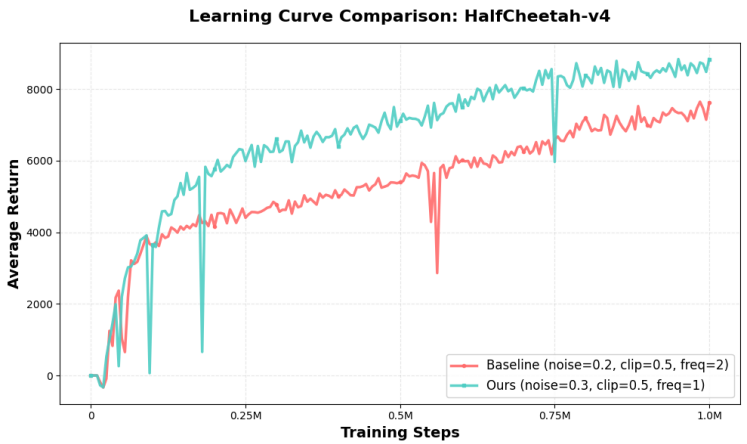
Noise Clip Range의 효과 (중간 그래프)

- Noise clip range는 전형적인 역-U자 패턴을 보이며, 0.5가 두 환경 모두에서 최적값임
- HalfCheetah-v4는 0.3(7,150) → 0.5(7,700, +7.7%) → 0.7(6,850, -11%) 즉, 0.7에서의 성능 저하는 과도하게 허용적인 노이즈 범위가 target policy에 불안정성을 초래함을 시사함
- Ant-v4는 0.3(2,250) → 0.5(2,350) → 0.7(2,300) 즉, clip range 변화에 덜 민감하며, poliy noise에 비해 상대적으로 작은 영향을 받음
- 적절한 clipping(0.5)는 충분한 action의 다양성을 허용하면서도 exploration을 효과적으로 정규화함

Policy Update Frequency의 효과 (중간 그래프)

- Update frequency는 환경 복잡도에 따라 정반대의 선호를 보임
- HalfCheetah-v4는 빈번한 업데이트를 선호하는데, 1(7,000), 2(7,400)가 3(7,250)보다 우수함
- 비교적 단순한 locomotion 태스크인 HalfCheetah는 안정성을 유지하면서도 빠른 policy optimization이 가능함
- Ant-v4는 반대 경향을 보이는데, 1(2,700) → 2(2,800, +37%) → 3(1,900, -32%)
- 2는 TD3의 delayed policy update 원칙과 정확히 일치하며, 특히 복잡한 8자 유도 coordination 문제에서는 critic이 안정화되기 전에 actor를 너무 자주 업데이트하면 value estimation error가 전파되고, 과도한 delaying은 학습 진행을 느리게 함
- TD3의 핵심 파라미터인 “delayed policy update”가 모든 환경에 보편적으로 최적은 아니며, 단순한 태스크는 빈번한 업데이트를, 복잡한 태스크는 보수적 접근을 요구함

하이퍼파라미터의 학습 안정성 분석



HalfCheetah-v4

- 더 강한 exploration(noise=0.3) + 빈번한 업데이트(freq=1) = 빠른 수렴 + 높은 최종 성능
- 두 설정 모두 HalfCheetah에서 안정적인 학습을 보임
- Ours가 약 500k step 부터 명확한 성능 우위 확립

Ant-v4

- 강한 exploration(noise=0.3)이 초기 학습 속도를 크게 개선하는 경향을 보임
- 그러나 두 설정 모두 Ant에서 높은 변동성 문제가 존재함
- Ours가 더 높은 평균 성능이지만, 안정성 측면에서는 여전히 불충분함
- 복잡한 8-DOF 환경에서 학습 안정성 확보의 어려움을 확인함

5. 결론 및 개선 사항

실험 결론 및 향후 과제

Core Insights 1 : 환경 복잡도가 하이퍼파라미터 전략을 결정

- 저차원/단순 태스크 : 강한 exploration(0.3) + 빈번한 업데이트(1~2)
- 고차원/복잡 태스크 : 강한 exploration(0.3) + 보수적 업데이트(2)

Core Insights 1 : TD3 원논문 하이퍼파라미터의 재평가

- 논문의 기본값(noise=0.2)은 안정성 우선의 보수적 선택
- 개별 환경 특성에 맞춘 튜닝으로 추가 성능 향상 가능

현재 연구의 한계

- 단일 training seed 사용 (preliminary screening 단계만 수행)
- Multiple training seed를 통한 robustness 검증 필요

향후 연구 방향

- 상위 3개 하이퍼파라미터 설정에 대한 multi-seed validation
- Ant 환경의 학습 안정성 추가 개선 (replay buffer, gradient clipping 등)
- 다른 MuJuCo 환경으로 일반화 가능성 검증

실험 결론 및 향후 과제

결론

- 본 프로젝트는 체계적인 하이퍼파라미터 탐색을 통해 TD3 성능을 환경별로 향상시킬 수 있음을 입증했으며, 특히 환경 복잡도에 따른 맞춤형 하이퍼파라미터 선택의 중요성을 실증적으로 확인함
- 더 나아가 TD3의 핵심 메커니즘(특히, policy noise 및 delayed policy update)이 환경 특성에 따라 유연하게 조정되어야 함을 보임
- 이는 향후 강화학습 알고리즘 적용 시 환경 분석 기반의 하이퍼파라미터 전략 수립이 필수적임을 시사함

6. 부록

HalfCheetah-v4 파라미터별 성능

Target Policy Noise	Noise Clip	Policy Frequency	평균 return	표준편차
0.1	0.3	1	7487	102
0.1	0.3	2	7494	110
0.1	0.3	3	7015	103
0.1	0.5	1	6647	137
0.1	0.5	2	7919	219
0.1	0.5	3	7778	132
0.1	0.7	1	4803	30
0.1	0.7	2	7598	123
0.1	0.7	3	5674	34
0.2	0.3	1	5936	67
0.2	0.3	2	6521	85
0.2	0.3	3	6996	70
0.2	0.5	1	7358	55
0.2	0.5	2	7597	82
0.2	0.5	3	7721	60
0.2	0.7	1	6346	98
0.2	0.7	2	6916	1966
0.2	0.7	3	6152	112
0.3	0.3	1	7701	56
0.3	0.3	2	7260	136
0.3	0.3	3	7802	30
0.3	0.5	1	8813	81
0.3	0.5	2	7794	82
0.3	0.5	3	7806	103
0.3	0.7	1	7642	54
0.3	0.7	2	7778	140
0.3	0.7	3	8641	105

Ant-v4 파라미터별 성능

Target Policy Noise	Noise Clip	Policy Frequency	평균 return	표준편차
0.1	0.3	1	2263	39
0.1	0.3	2	2601	66
0.1	0.3	3	1853	19
0.1	0.5	1	1925	369
0.1	0.5	2	1985	22
0.1	0.5	3	1533	13
0.1	0.7	1	2277	626
0.1	0.7	2	1525	564
0.1	0.7	3	1676	610
0.2	0.3	1	2049	529
0.2	0.3	2	2807	46
0.2	0.3	3	2201	394
0.2	0.5	1	3183	68
0.2	0.5	2	2999	381
0.2	0.5	3	955	728
0.2	0.7	1	2770	1315
0.2	0.7	2	2553	77
0.2	0.7	3	2405	1196
0.3	0.3	1	2892	105
0.3	0.3	2	3426	68
0.3	0.3	3	3199	1087
0.3	0.5	1	2970	116
0.3	0.5	2	5084	84
0.3	0.5	3	640	606
0.3	0.7	1	3701	77
0.3	0.7	2	1288	1150
0.3	0.7	3	2745	1228