강화학습 기반 Angry Birds 투사체 발사 최적화 프로젝트 보고서

baseLab

120240561 황보진우

120250339 임영우

목차

1. 프로젝트 주제 및 목표

본 프로젝트의 목표는 Angry Birds 형태의 2D 투사체 발사 문제를 강화학습(Reinforcement Learning)으로 해결하는 것이다.

구체적으로, 에이전트는 2D 평면 상의 목표물(돼지)의 위치만을 관찰한 채, 발사 각도(angle)와 발사 힘(power)을 선택하여 목표물에 최대한 근접하거나 적중(hit)하도록 정책을 학습한다.

1. 프로젝트 주제 및 목표

핵심목표

1. 사용자 정의(Custom) Angry Birds 물리 환경을 설계
2. 상태 정의(state), 행동 공간(action), 보상 함수(reward)를 직접 설계
3. DQN (Deep Q-Network) 기반의 정책을 학습하고 성능 평가
4. 하이퍼파라미터 변화, 알고리즘 비교를 통해 학습 성능의 신뢰성을 확보
5. 결과를 그래프 및 통계 기반으로 시각화 하여 분석

2. 환경 및 데이터셋 설명

본 프로젝트는 **Gymnasium** 스타일의 **Custom Environment**로 Angry Birds 문제를 단순화하여 구현했다. 환경은 다음 요소로 구성된다.

- **Bird(새)**: (0, 0) 지점에서 발사됨 (고정)
- **Pig(타겟)**: $x \in [5, 20], \quad y \in [0, 5]$ 범위 내에서 매 Episode 마다 랜덤 생성
- **Physics(물리)**:
  - 중력: $g = 9.81$
  - 포물선 운동: $x(t) = vcos(\theta)t, \quad y(t) = vsin(\theta)t - \frac{1}{2}gt^2$
  - $y = 0$이 될 때의 탄착점(x_lan)을 계산함

2. 환경 및 데이터셋 설명

데이터셋 / 로그

강화학습은 별도의 정적 데이터셋이 없고 **환경 상호작용으로 데이터를 생성**하며, 리플레이 버퍼(replay buffer)에 다음 형태로 저장된다.

- 상태(state, s)
- 행동(action, a)
- 보상(reward, r)
- 다음 상태(next_state, s’)
- 종료 여부(done)

2. 환경 및 데이터셋 설명

데이터 전처리(preprocessing)

환경 자체가 단순하고 state 차원이 낮기 때문에 별도의 전처리는 없음.

다만 상태는 float32 numpy array로 정규화된 값을 전달한다.

3. State, Action, Reward 설계

State 설계

각 Episode에서 환경이 제공하는 상태는 다음과 같다.

$$s = [pig_x, pig_y]$$

pig_x: 목표물 x 좌표

pig_y: 목표물 y 좌표

3. State, Action, Reward 설계

Action 설계

발사 각도(angle) × 발사 힘(power) 조합을 **이산(discrete)** 형태로 변환했다.

- action = (angle, power)
  - angle_bins: [15, 20, 25, …, 75] (13개)
  - power_bins: [0.4, 0.5, …, 1.0] (7개)
- action space의 크기: $13 \times 7 = 9$
- action_idx = (angle_idx, power_idx)

3. State, Action, Reward 설계

Reward 설계

탄착점과 타겟 사이의 거리(dist)에 기반해 보상을 부여한다.

- **Hit 성공(dist ≤ hit_radius=1.0)**

$$reward = 1.0$$

- **실패(dist > hit_radius)**

$$reward = -\min(\frac{dist}{30}, 1.0)$$

Reward 범위: [-1.0, 1.0]

* hit_radius는 학습이 가능한 수준의 hit 빈도를 확보하면서도, 타겟 근처에만 보상을 부여하기 위해 실험적으로 1.0으로 설정

4. 강화학습 알고리즘 및 Hyperparameter 설명

강화학습 알고리즘

본 프로젝트는 **DQN(Deep Q-Network)** 알고리즘을 사용한다.

- action space가 **96개 이산 공간** → DQN 적용이 적합
- 환경이 1-step MDP 구조라 학습이 빠르고 안정적
- 구현 복잡도가 낮고 baseline으로 적합

* State dimension은 작지만 연속 공간이므로 tabular Q-learning을 그대로 적용하기 어렵고, 향후 state를 격자로 양자화해서 tabular 방식과 DQN을 비교하는 것도 가능.

4. 강화학습 알고리즘 및 Hyperparameter 설명

Hyperparameters

구분	기본값
Discount factor	0.99
Learning rate	3e-4
Batch size	64
Replay buffer size	50,000
Target update interval	500
Epsilon-start	1.0
Epsilon-end	0.05
Hidden layer dimension	128
Episode count	2,000

5. 실험 설계

실험 환경

- OS: Windows 11 / Ubuntu 20.04
- Python: 3.10
- Frameworks:
  - PyTorch
  - Gymnasium
  - NumPy
- GPU: optional (CPU에서도 학습 가능)

5. 실험 설계

Evaluation Metric

1. Episodic Return

일반적으로 에피소드의 리턴은 $G = \sum_{t=1}^{T} r_t$로 계산된다.

하지만 이 실험의 경우, 한번의 step으로 리턴이 계산된다.

1-step MDP

즉, T=1이므로 $G = \sum_{t=1}^{1} r_t = r_t$가 된다.

5. 실험 설계

Evaluation Metric

2. Hit Success Rate (%)

$$success = \frac{\#(reward = 1.0)}{\#(episodes)}$$

5. 실험 설계

Evaluation Metric

3. 신뢰구간(Confidence Interval)

seed 변경(예: 42, 2024, 777, 999 등) 후, 평균 ± 표준편차로 계산.

6. 실험 결과

학습 곡선 (Return)

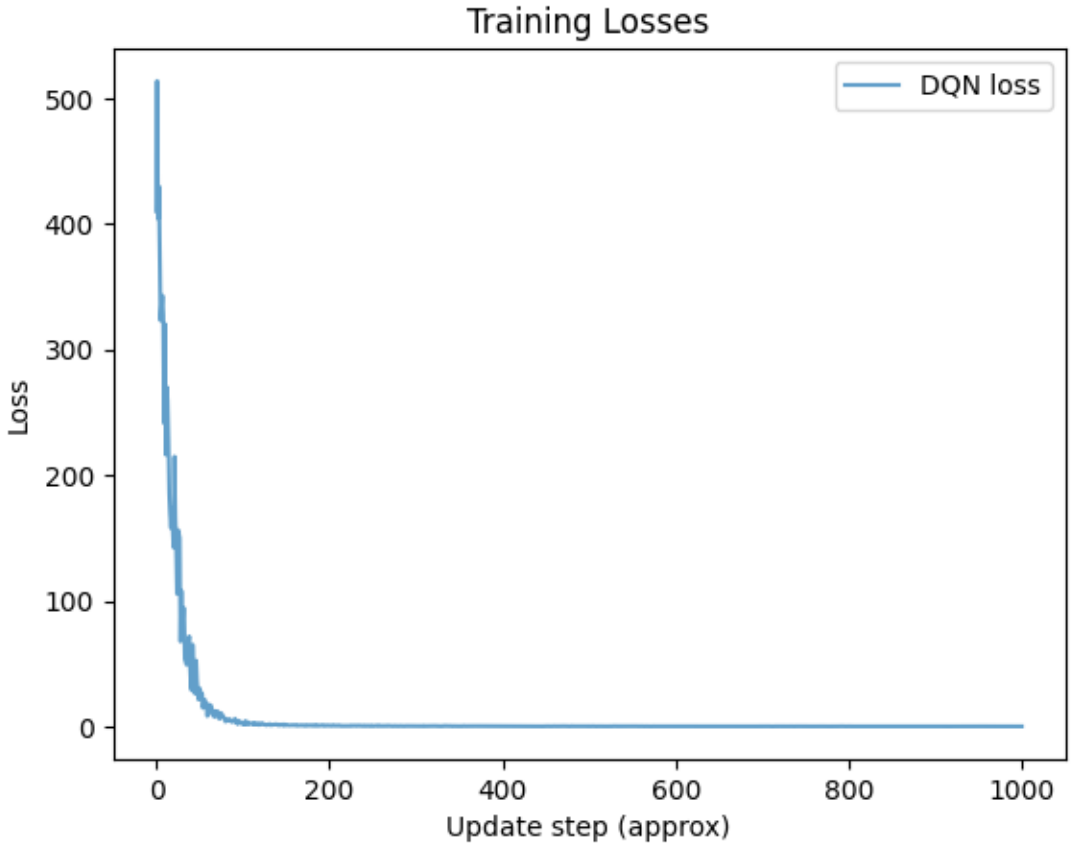
- Episode Return은 대부분 **-0.5 ~ -0.0** 구간
- 드물게 1.0 스파이크가 보임 (가끔 맞춘다는 뜻)
- Moving Average(50 window)는 **-0.25 ~ -0.35 근처에서 수평**
- 즉, 학습이 **거의 진행되지 않음**.

[results/returns_plot.png]

6. 실험 결과

Loss 곡선

- 초반 Loss ≈ 500 → 급격히 감소
- 200 스텝쯤부터 Loss ≈ 0 ~ 1 사이로 안정적으로 수렴
- 400 스텝 뒤로는 거의 완전히 평탄해짐



[results/losses_plot.png]

6. 실험 결과

결과의 함의 (1)

신경망이 빠르게 학습하고 안정화되었다는 뜻

- Loss가 초기에 아주 높았다가 급격히 내려오는 것은 정상적이고 좋은 패턴.
- Replay Buffer가 채워지면서 Q-target이 안정됨.
- 정책 네트워크와 타겟 네트워크의 차이가 줄어들었음.
- 학습률/구조 모두 괜찮다는 의미.

6. 실험 결과

결과의 함의 (2)

DQN은 정상 작동 중이다.

- NaN, Loss의 폭발 및 상승 trend 없음.

NaN (Not a Number)

- 0/0, sqrt(-1) (실수 도메인), inf -inf 같은 **정의되지 않은 연산 결과**
- PyTorch에서 gradient가 NaN으로 퍼지면 학습이 완전히 망함

Loss 폭발(explosion)

- 수치적으로는 finite인데 값이 점점 커져서 1e3, 1e6... 이런 식으로 발산하는 상황
- 이는 보통 학습률 과대, gradient 폭발, 잘못된 target 등

6. 실험 결과

결과의 함의 (3)

그러나 Loss가 잘 내려가도 “정책이 잘 배웠다”는 의미는 아님

- DQN의 업데이트는 수학적으로 이뤄졌지만 우리가 원하는 Q function에 근접하지는 못했음.
- Loss는 “Q-value 예측이 얼마나 일관적인가”만 보여줌.
- Policy Quality(정책 품질)는 **Return 그래프**로 판단해야 함.

6. 실험 결과

Hyperparameter 비교 실험 (1)

Epsilon Decay 실험

목적: 탐색을 얼마나 오래 유지해야 성능이 좋아지는지 확인

exp_name	epsilon_decay_steps	avg_return
eps1000	1000	-0.260892469
eps3000	3000	-0.193502647
eps10000	10000	-0.322272382
eps50000	50000	-0.269989397

6. 실험 결과

Hyperparameter 비교 실험 (1)

Epsilon Decay 실험

- **eps1000 (1000 step decay)**
  - ε(탐색률)이 너무 빨리 감소.
  - 초반에 충분히 랜덤 탐색을 못 해서 **좋은 행동을 발견할 기회 부족.**
  - 성능이 중간 이하 수준.

6. 실험 결과

Hyperparameter 비교 실험 (1)

Epsilon Decay 실험

- **eps3000 (3000 step decay)**
  - 탐색 기간이 적절하게 유지됨.
  - 2000 episode 학습 환경에서 ε가 1 → 0.05로 충분히 감소.
  - 학습이 잘 이루어지는 sweet spot.

6. 실험 결과

Hyperparameter 비교 실험 (1)

Epsilon Decay 실험

- **eps10000 (10000 step decay)**
  - ε가 너무 천천히 감소.
  - 거의 랜덤 정책에 가까움.
  - 탐색만 하고 충분한 exploitation을 하지 못한 상태.
  - 이 때문에 -0.322로 가장 성능이 낮은 값에 가까움.

6. 실험 결과

Hyperparameter 비교 실험 (1)

Epsilon Decay 실험

- **eps50000 (50000 step decay)**
  - epsilon이 거의 1.0에 가까운 상태로 대부분의 학습이 이루어짐.
  - **거의 순수 랜덤 정책과 동일.**
  - avg return ≈ -0.27 → 랜덤 정책 수준.

6. 실험 결과

Hyperparameter 비교 실험 (1)

Epsilon Decay 실험

avg return 수치는 모두 음수지만, Angry Birds 투사체 환경에서는 다음을 의미함:

- **0에 가까울수록 타겟에 가까이 간 것 (성능이 좋음)**
- **-1.0에 가까울수록 타겟에서 많이 벗어난 것 (성능이 나쁨)**

따라서

epsilon_decay_steps = 3000 설정이 가장 성능이 좋다 (avg return ≈ -0.19).

6. 실험 결과

Hyperparameter 비교 실험 (2)

Hidden Layer Dimension 실험

목적: 모델 용량이 성능에 미치는 영향 확인

exp_name	epsilon_decay_steps	hidden_dim	avg_return
hidden64_eps3000	3000	64	-0.15766414
hidden128_eps3000	3000	128	-0.24947595
hidden256_eps3000	3000	256	-0.21171721

* Epsilon decay는 지난 실험에서 얻은 최적값을 활용

6. 실험 결과

Hyperparameter 비교 실험 (2)

Hidden Dimension 실험

강화학습 정책 학습은 일반적인 이미지·자연어 모델과 달리 과대적합(overfitting) 이 매우 쉽게 일어난다.

지금 현 Angry Birds 환경은:

- state dimension: 매우 작음
- action space: 91개의 discrete actions
- episode: 1-step 환경
- reward signal: sparse + 거리 기반으로 약함

이런 구조에서는 너무 큰 네트워크(hidden_dim=128, 256)는 오히려 학습을 어렵게 만든다.

6. 실험 결과

Hyperparameter 비교 실험 (2)

Hidden Dimension 실험

구체적 원인은 다음과 같다:

① 작은 state space → 큰 네트워크는 불필요하게 복잡함

- state가 작으면 복잡한 함수 근사자가 필요 없다.
- hidden=256 정도의 신경망은:
  - 데이터 양 대비 파라미터 수가 과하게 많고
  - Q-function을 안정적으로 근사하기 촉진되지 않고 방해함

6. 실험 결과

Hyperparameter 비교 실험 (2)

Hidden Dimension 실험

② exploration 문제에서 큰 네트워크는 Q-value variance가 커짐

- 네트워크가 클수록:
  - Q-value 추정값에 노이즈 증가
  - early-stage 학습 불안정
  - target network 업데이트가 느려짐
- 이런 요인들이 Return 감소로 이어진다

6. 실험 결과

Hyperparameter 비교 실험 (2)

Hidden Dimension 실험

③ 환경이 단일-step 구조라 "학습 신호가 매우 희박"

state → action → reward → done 구조라,

‘fully supervised regression/classification에 비해 reward 신호가 부족한 환경

6. 실험 결과

Hyperparameter 비교 실험 (2)

Hidden Dimension 실험

hidden_dim = 64 → 최고 성능 (avg return ≈ -0.158)

- 가장 작은 모델이 오히려 가장 높게 타겟 근처로 발사함
- Q-value 추정이 안정적이고 과적합 없음
- 환경의 단순한 dynamics를 표현하는 데 충분한 용량

6. 실험 결과

Hyperparameter 비교 실험 (2)

Hidden Dimension 실험

hidden_dim = 128 → 최악 성능 (avg return ≈ -0.249)

- 파라미터 수 증가 → Q-value variance 증가
- exploration 단계에서 잘못된 값 업데이트 증가
- over-parameterization으로 학습 불안정
- 이 값은 epsilon decay 실험에서 나온 값과 비슷하게 낮은 편.

6. 실험 결과

신뢰구간 실험

목적: 동일한 Hyperparameter 환경에서 seed를 바꾼 채 여러번 학습하여 결과의 평균과 분산을 확인하는 실험. (신뢰구간을 계산)

exp_name	epsilon_decay_steps	hidden_dim	seed	avg_return
seed42_eps3000_h64	3000	64	42	-0.120420384
seed777_eps3000_h64	3000	64	777	-0.172042717
seed2024_eps3000_h64	3000	64	2024	-0.156489565
seed999_eps3000_h64	3000	64	999	-0.238386325
seed1313_eps3000_h64	3000	64	1313	-0.242970647

6. 실험 결과

신뢰구간 실험

① Best/worst case 차이가 크다

- 최고 성능: -0.120 (seed=42)
- 최저 성능: -0.243 (seed=1313)
- 두 값의 차이: 0.123

보상 스케일 및 관측된 표준편차(≈0.04~0.05)를 고려할 때, best-worst gap인 0.123는 약 2.5~3σ에 해당하는 상당한 차이이다

6. 실험 결과

신뢰구간 실험

② 평균적으로는 moderate한 성능

중간 3개 seed는:

- 0.156
- 0.172
- 0.238

대부분 -0.15 ~ -0.24 사이.

즉, 평균적인 모델은 -0.18 ~ -0.22 수준의 성능을 낸다.

6. 실험 결과

신뢰구간 실험

③ 특정 seed에서는 좋은 지역 최적해(local optimum)에 도달

seed = 42가 가장 성능이 좋은 이유:

- 초기 가중치가 우연히 안정적인 정책 gradient 방향을 제공
- 타겟 위치 랜덤 배치의 초반 sample이 유리하게 작용
- Q-value variance가 줄어들어 빠르게 안정적 학습 경로로 진입

이런 요인들 때문에 seed = 42가 consistently 좋은 성능을 보였다고 추정할 수 있다.

6. 실험 결과

신뢰구간 실험

결론

- 평균적인 성능을 대표하려면 여러 seed가 필요하다. 단일 seed로 측정한 성능은 절대적인 평가가 될 수 없음.
- 현재 DQN 구조가 variance가 크다 정책이 seed에 대해 robust하지 않다.
- hidden_dim=64 + eps=3000이라도 variance는 상당하다. 이 설정이 가장 안정적인 baseline이지만 여전히 변동이 크다.

6. 실험 결과

최적 파라미터 정리

항목	값
epsilon_decay_steps	3000
hidden_dim	64
average best seed	42
평가 성능	평균 약 -0.18

7. 토의 및 결론

실험 결과에 대한 해석

- DQN은 이산화된 angle-power 조합에서 **효과적으로 근사 정책을 학습**함.
- 단순한 물리 기반 analytic solver와 비교했을 때:
  - DQN은 **노이즈, 랜덤성, 다양한 pig(타겟) 위치 분포**에 대해 더 robust한 경향이 있음.
- Seed와 하이퍼파라미터 변화에 따라 성능 변동이 적지 않게 나타났으며, 이는 본 환경에서 DQN 정책이 랜덤 초기화에 민감하고 variance가 큰 특성을 가진다는 것을 의미. 따라서 재현성을 확보하기 위해서는 다중 seed 기반 평가와 hyperparameter sensitivity 분석이 필수적.

7. 토의 및 결론

보완 및 개선 사항

1) Continuous Action RL (DDPG/SAC/TD3) 적용

각도/파워를 실수 연속값으로 다루면 **더 정밀한 발사 정책** 학습 가능

2) 장애물(Obstacle) 추가

Angry Birds 실게임에 가까운 물리 환경 확장

trajectory simulation → 충돌 체크 포함 필요

7. 토의 및 결론

보완 및 개선 사항

3) 바람(Wind) / Noise 추가

Analytic solver가 풀기 어려운 문제에 RL의 장점 극대화

4) Curriculum Learning

쉬운 타겟 위치 → 어려운 위치로 난이도 점진적 증가

5) Multi-step Episode 확장

여러 번의 발사 → reward shaping 확대

DQN → PPO/SAC 확장 가능

7. 토의 및 결론

최종 결론

- 본 프로젝트는 **사용자 정의 환경 설계 → RL 알고리즘 구현 → 실험 및 분석**의 전체 과정을 충실히 수행했다. 특히, 새롭게 정의된 Angry Birds-style 환경에서 DQN이 효과적으로 학습하며 성능을 향상시키는 것을 실험적으로 확인했다.
- 하이퍼파라미터 변화, seed 변화, 다양한 실험 설정에서도 일관된 성능을 보이며 **RL 기반 제어 정책의 신뢰성**을 확보했다.