노이즈 환경에서 거대 언어 모델 기반 보상 함수를 활용한 강화학습 에이전트 성능 평가

이재원, 지창훈*, 김주봉*, 한연희*†

한국기술교육대학교 컴퓨터공학부, *한국기술교육대학교 미래융합공학전공

{asd5742, koir5660, rlawnqhd, yhhan}@koreatech.ac.kr

Evaluation of reinforcement learning agent performance using a large language model based reward function in a noisy environment

Jae-Won Lee, Chang-Hun Ji*, Ju-Boing Kim*, Youn-Hee Han*†

Department of Computer Engineering, Korea University of Technology and Education

*Future Convergence Engineering, Korea University of Technology and Education

요 약

현실 세계의 물리적 외력과 같은 예측 불가능한 요인들은 시뮬레이터 환경에서의 보상 함수 설계에 어려움을 초래한다. 특히 로봇의 팔이나 손과 같은 복잡한 조작이 요구되는 환경에서 이러한 노이즈를 고려한 보상 함수를 설계하는 것은 어렵다. 이에 Evolution-driven Universal Reward Kit for Agent (EUREKA)는 환경 코드와 작업 설명을 기반으로 다양한 강화학습 시뮬레이션 환경에서 보상 함수를 생성한다. 본 논문에서는 두 가지 실험을 통해 Eureka의 노이즈 환경에 대한 대응 능력을 평가한다. 첫 번째 실험은 Cartpole 환경에서 다양한 무작위 노이즈를 적용하여 Eureka의 보상 함수 설계 성능을 평가한다. 두 번째 실험은 특정 노이즈 환경에서 Eureka의 대응 방법을 분석한다. 실험 결과, Eureka는 복잡한 환경에서도 높은 수준의 노이즈에 대응할 수 있음을 보인다.

Ⅰ. 서 론

현실 세계에 존재하는 기계 장치등을 강화학습으로 제어할 때에는 노이즈 등 물리적 외력과 같은 예상할 수 없는 요인들이 변수로 작용한다. 대부분의 시뮬레이터 환경에서는 이러한 변수를 반영하는 것이 어렵기 때문에, 노이즈가 반영되지 못한 환경이 사용된다. 특히, 로봇의 팔 혹은 손과 같이 복잡하고 섬세한 조작이 요구되는 작업에서는 물리적인 외력에 민감하다. 이러한 노이즈들은 보상 함수를 다시 설계함으로써 극복할 수 있다. 그러나 인간의 경험만으로 노이즈를 고려한 보상 함수를 설계하는 것은 어려운 작업이다.

보상 함수 설계의 어려움을 해소하기 위해 최근에는 GPT와 같은 대규모 언어 모델(Large Language Models, LLM)을 활용하여 보상 함수를 생성하는 연구가 진행되고 있다 [1]. 이러한 연구 중 하나인 Evolution-driven Universal Reward Kit for Agent (EUREKA) [2] 는 LLM을 사용하여 환경 코드와 작업 설명을 기반으로 다양한 강화학습 시뮬레이션 환경에서 보상 함수를 생성한다. 이렇게 생성된 보상 함수를 사용하면 기본 보상 함수를 사용할 때보다 더 좋은 성능을 산출할 수 있다.

따라서, 본 논문에서는 무작위의 물리적 외력이 적용되는 환경에서 Eureka의 보상 함수 설계 능력을 평가한다. 또한 Cartpole 환경에서 제어할 수 있는 관측 요소들에 고정 오차 혹은 무작위 노이즈를 추가하는 각각의 실험을 진행한다. 실험 결과에서 Eureka는 이와 같은 환경에서도 높은 수준의 노이즈에 대응할 수 있는 보상 함수를 잘 생성함을 보인다.

Ⅱ. 본 론

본 논문에서는 Eureka의 복잡한 노이즈 환경에 대한 대응 능력을 평가하기 위해 두 가지의 실험을 진행한다. 두 실험 모두 Proximal Policy Optimization (PPO) [3] 알고리즘을 사용하여 학습한다.

실험 1. 고정 오차 존재 환경에 대한 Eureka의 대응

우선 실험 1 에서는 고정 오차에 대한 Eureka의 대응 방법을 분석하기 위해 4가지 관측 요소인 카트의 위치, 카트의 속도, 폴의 각도, 폴의 각속도 중 폴의 각도에 0.17라디안(약 10도)의 고정된 오차를 적용하여 실험을 진행한다.

그림 1의 (a) 코드에서 카트와 폴의 위치 및 속도에 대한 보상은 음수 부호가 붙은 지수함수를 사용하여 계산된다. 이 함수는 각 요소의 절댓값이 커질수록 보상이 감소하므로, 에이전트가 카트를 목표 위치와 상태에 가깝게 유지하도록 유도한다. 반면 (b) 코드에서는 폴의 각도에만 이 지수함수를 적용하고, ‘angle_reward_temperature’ 값을 5.0으로 증가시켜, 절댓값이 증가할수록 보상이 더 빨리 0에 가까워진다. 이는 폴 각도의 변화에 대해 에이전트의 반응을 덜 민감하게 만들어, 고정 오차에 의한 각도 변화에도 과도한 페널티를 받지 않도록 한다. 또한, (b) 코드에서 카트 위치와 속도, 폴 각속도에 대한

† 한연희(Youn-Hee Han, yhhan@koreatech.ac.kr): 교신저자

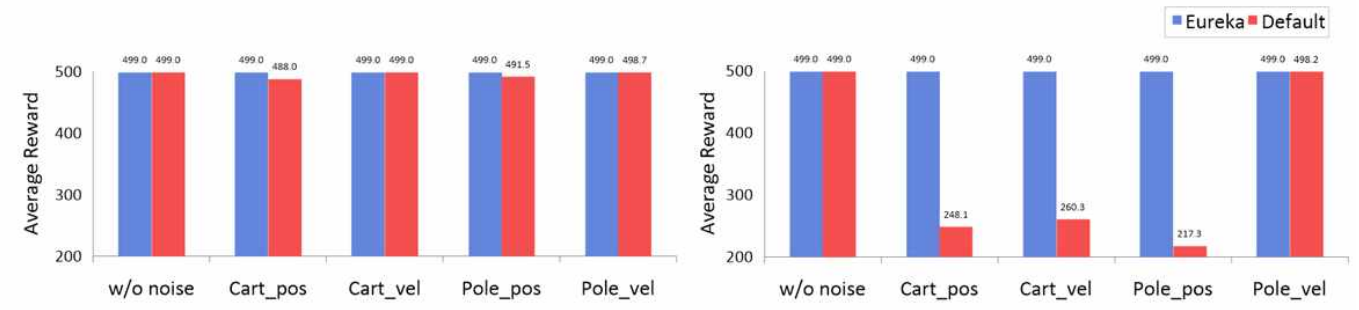
```python
# Define reward components
angle_reward_temperature = 2.0
angle_reward = torch.exp(-pole_angle.abs() * angle_reward_temperature)

velocity_reward_temperature = 0.5
velocity_reward = torch.exp(-cart_velocity.abs().pow(2) * velocity_reward_temperature)

velocity_pole_temperature = 0.05
velocity_pole_reward = torch.exp(-pole_velocity.abs() * velocity_pole_temperature)

position_reward_temperature = 3.0
position_reward = torch.exp(-cart_position.abs().pow(2) * position_reward_temperature)

# Combine and normalize individual reward components
reward_normalized = torch.stack([position_reward, velocity_reward, angle_reward,
                                 velocity_pole_reward]).mean(dim=0)
```

(a) 기본 환경

```python
# Reward components
angle_reward_temperature = 5.0
angle_reward = torch.exp(-angle_reward_temperature * torch.abs(pole_angle))

velocity_penalty_temperature = 0.05
velocity_penalty = -velocity_penalty_temperature * (cart_velocity ** 2 + pole_angular_velocity ** 2)

position_penalty_temperature = 0.25
position_penalty = -position_penalty_temperature * (cart_position ** 2)

# Normalize reward components
angle_reward = (angle_reward - angle_reward.min()) / (angle_reward.max() - angle_reward.min())
velocity_penalty = (velocity_penalty - velocity_penalty.min()) / (velocity_penalty.max() -
                   velocity_penalty.min())
position_penalty = (position_penalty - position_penalty.min()) / (position_penalty.max() -
                   position_penalty.min())

# Combine reward components
reward = angle_reward + velocity_penalty + position_penalty
```

(b) 폴의 각도에 0.17라디안의 고정 오차가 존재하는 환경

그림 1. Cartpole 환경에서 기본 환경과 폴의 각도에 고정 오차를 추가한 환경에서 Eureka가 생성한 보상 함수 비교



(a) 표준편차 0.1 (b) 표준편차 0.2

그림 2. Cartpole 환경에서 무작위 노이즈가 추가된 제어 요소별 실험 결과

페널티를 적용하여 과도한 조정을 방지한다. 이러한 전략은 고정 오차 환경에서의 폴 각도 변화를 넓게 허용하고, 빠른 움직임과 큰 편차에 대한 페널티를 통해 시스템의 안정성을 유지하는 Eureka의 적응성을 보여준다.

실험 2. 무작위 노이즈 환경에서 Eureka의 보상 함수 설계

실험 2에서는 무작위 노이즈 환경에서 Eureka의 보상 함수 설계 성능을 평가하기 위해 Cartpole 환경에서 제어할 수 있는 4가지 관측 요소 각각에게 평균 0, 표준편차 0.1 또는 0.2인 정규분포에서 값을 샘플링한 값을 더하였다.

그림 2에서 알 수 있듯이 노이즈를 적용하지 않은 환경(w/o noise)에서는 Eureka와 기본 보상 함수 모두 안정적으로 폴을 세우는 것을 볼 수 있다. 표준편차 0.1 에서는 Eureka와 기본 보상 함수 모두 높은 평균 점수를 받았다. Eureka는 어떤 노이즈에도 안정적인 실험 결과를 보이지만, 기본 보상 함수는 카트의 위치와 폴의 각도에 대한 노이즈에 영향을 받는 것으로 보인다. 표준편차 0.2 에서 기본 보상 함수는 폴의 각속도에 대한 노이즈 환경을 제외한 나머지 노이즈 환경들에서 평균 보상이 급격히 떨어졌지만, Eureka는 여전히 안정적인 실험 결과를 보인다는 점을 통해 Eureka가 노이즈를 고려한 보상 함수 설계 능력을 갖췄음을 알 수 있다.

Ⅲ. 결 론

본 논문에서는 노이즈 환경에서 Eureka의 보상 함수 설계 성능을 평가하기 위해 두 가지의 실험을 진행했다. 실험 1 에서는 Eureka가 고정 오차에 대해 합리적으로 대응할 수 있음을 보인다. 실험 2 에서는 Eureka가 무작위 노이즈가 추가된 복잡한 환경에서도 높은 수준의 보상 함수 설계 능력을 갖췄음을 보인다. 두 실험을 통해 Eureka가 복잡한 환경에서 높은 수준의 노이즈에 대응할 수 있음을 알 수 있다.

ACKNOWLEDGMENT

이 논문은 2023년 및 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2023R1A2C1003143 & NRF-2018R1A6A1A03025526).

참 고 문 헌

[1] Tianbao Xie et al., “TEXT 2 REWARD: AUTOMATED DENSE REWARD FUNCTION GENERATION FOR REINFORCEMENT LEARNING,” arXiv preprint arXiv:2309.11489, 2023.

[2] Yecheng Jason Ma et al., “EUREKA: HUMAN-LEVEL REWARD DESIGN VIA CODING LARGE LANGUAGE MODELS,” arXiv preprint arXiv:2310.12931, 2023.

[3] John Schulman et al., “Proximal Policy Optimization Algorithms,” arXiv preprint arXiv:1707.06347, 2017.