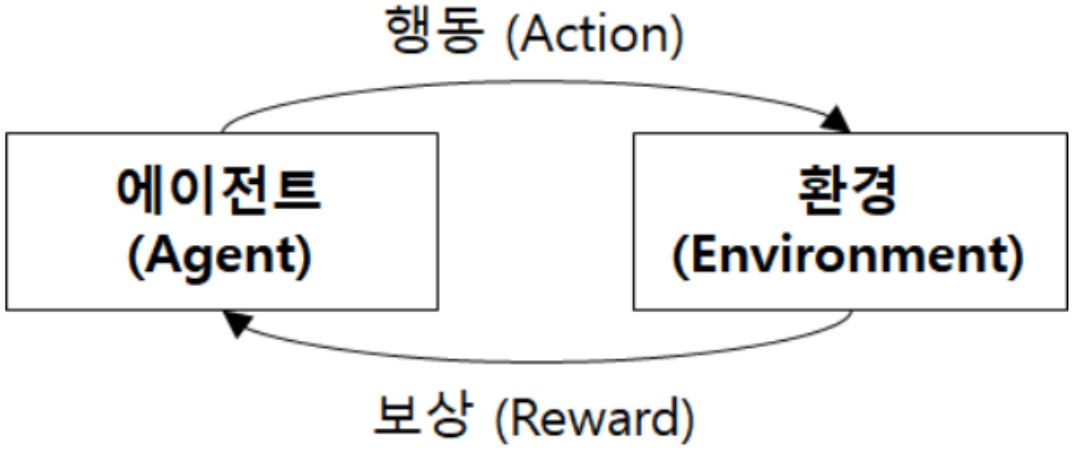
RL

REINFORCEMENT LEARNING



RL이란?

- 기계 학습의 한 분야로, 에이전트가 환경과 상호작용을 하며 최적의 행동을 학습하는 방법을 연구하는 기술

- 에이전트(agent)가 하는 일

RL이란?

- 에이전트의 목적

: 보상의 장기간 기대치를 최대로 만드는 행동을 학습

정책(policy)

- 에이전트가 행동을 결정하기 위해 사용하는 알고리즘

- 확정적 정책

: 상태 s가 주어졌을 때, 특정 행동 a를 항상 선택하는 정책

$$\pi(s) = a$$

- 확률적 정책

: 상태 s가 주어졌을 때, 각 행동 a를 선택할 확률을 정의하는 정책

$\pi(a|s)$ = 행동 a를 상태 s에서 선택할 확률

정책 탐색

❖ 에이전트가 주어진 환경에서 최적의 정책을 찾기 위해 정책의 파라미터를 조정하거나 정책 구조를 탐색하는 과정

❖ 접근 방법

: 정책 기반 방법 & 가치 기반 방법

정책 탐색

❖ 정책 기반 방법

: 직접적으로 정책의 파라미터를 조정하여 최적의 정책을 찾는 방법

: REINFORCE Algorithm, Proximal Policy Optimization(PPO)

❖ 가치 기반 방법

: 상태-행동 쌍의 가치를 평가하여 최적의 행동을 선택하는 방법

: Q-Learning, Deep Q-Network(DQN)

정책 기반 방법 vs 가치 기반 방법

정책 기반 방법	가치 기반 방법
정책 자체가 학습 목표	상태-행동 쌍의 가치를 평가하고, 이를 토대로 차후 행동 선택
정책의 파라미터를 직접 업데이트	가치 함수를 학습하여 이를 기반으로 정책 유도
정책의 성과를 기반으로 파라미터를 조정	Q-learning의 경우 Q-value를 업데이트하며 행동 결정
연속적 행동 공간도 자유롭게 다룰 수 있다.	이산적 행동 공간에 더 적합하다.
일반적으로 더 안정적, 복잡한 정책 구조 다룰 수 있음	구현 간단, 안정성 문제가 있을 수 있음
요리사의 레시피를 직접 만들어서 맛있는 요리를 만들어내는 과정	재료의 품질과 조리 방법을 각각 평가하여, 최적의 레시피를 찾아내는 과정

Environment

❖ Agent가 상호작용하는 외부시스템으로, 에이전트의 행동에 반응하여 상태와 보상을 제공하는 시스템을 의미

: 에이전트는 이와 상호작용하여 상태를 관찰하고, 행동을 취하며, 보상을 받아 정책을 학습

❖ 구성 요소

: 상태집합, 행동집합, 상태전이확률, 보상함수, 할인율

Environment

❖ 상태집합

: 환경에서 가능한 모든 상태들의 집합

: ex) 체스판의 모든 가능한 배치

❖ 행동집합

: 에이전트가 임의의 상태에서 선택할 수 있는 모든 행동들의 집합

: ex) 체스에서 가능한 모든 수

Environment

❖ 상태 전이 확률

: 상태 s에서 행동 a를 취했을 때, 다음 상태 s'로의 확률

$$P(s'|s,a)$$

❖ 보상 함수

: 상태 s와 행동 a의 조합에서 받는 보상

$$R(s,a)$$

Environment

❖ Return

: reward의 합을 의미, Agent의 궁극적인 목표

1. Discounted Return

: 미래의 보상을 현재의 가치로 반환하기 위해 할인율을 사용하는 방법

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots$$

2. Undiscounted Return

: 미래의 모든 보상을 할인율을 사용하지 않고 합산하는 방법

$$G_t = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \cdots$$

Environment

- 할인율

: 미래 보상의 중요성을 결정하는 파라미터

: 1에 가까울수록 먼 미래의 보상이 현재의 보상만큼 중요

: 0에 가까울수록 미래의 보상은 중요하게 취급되지 않음

$$0 \leq \gamma \leq 1$$

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots$$

예제

```
# 환경 초기화
env = gym.make('CartPole-v1')

# Q 테이블 초기화
state_space_bins = [20, 20, 20, 20]  # 각 상태 변수를 이산화하기 위한 구간 수
state_space_bounds = list(zip(env.observation_space.low, env.observation_space.high))
state_space_bounds[1] = [-0.5, 0.5]
state_space_bounds[3] = [-math.radians(50), math.radians(50)]

q_table = np.zeros(state_space_bins + [env.action_space.n])

# 하이퍼파라미터 설정
learning_rate = 0.1
discount_factor = 0.99
num_episodes = 10000
epsilon = 1.0  # 탐험(exploration) 비율 초기값
epsilon_decay = 0.999  # 탐험 비율 감소율
epsilon_min = 0.01  # 탐험 비율의 최소값
```

예제

```
def discretize_state(state):
    discrete_state = []
    for i in range(len(state)):
        low, high = state_space_bounds[i]
        if state[i] <= low:
            discrete_state.append(0)
        elif state[i] >= high:
            discrete_state.append(state_space_bins[i] - 1)
        else:
            scale = (state[i] - low) / (high - low)
            discrete_state.append(int(scale * state_space_bins[i]))
    return tuple(discrete_state)
```

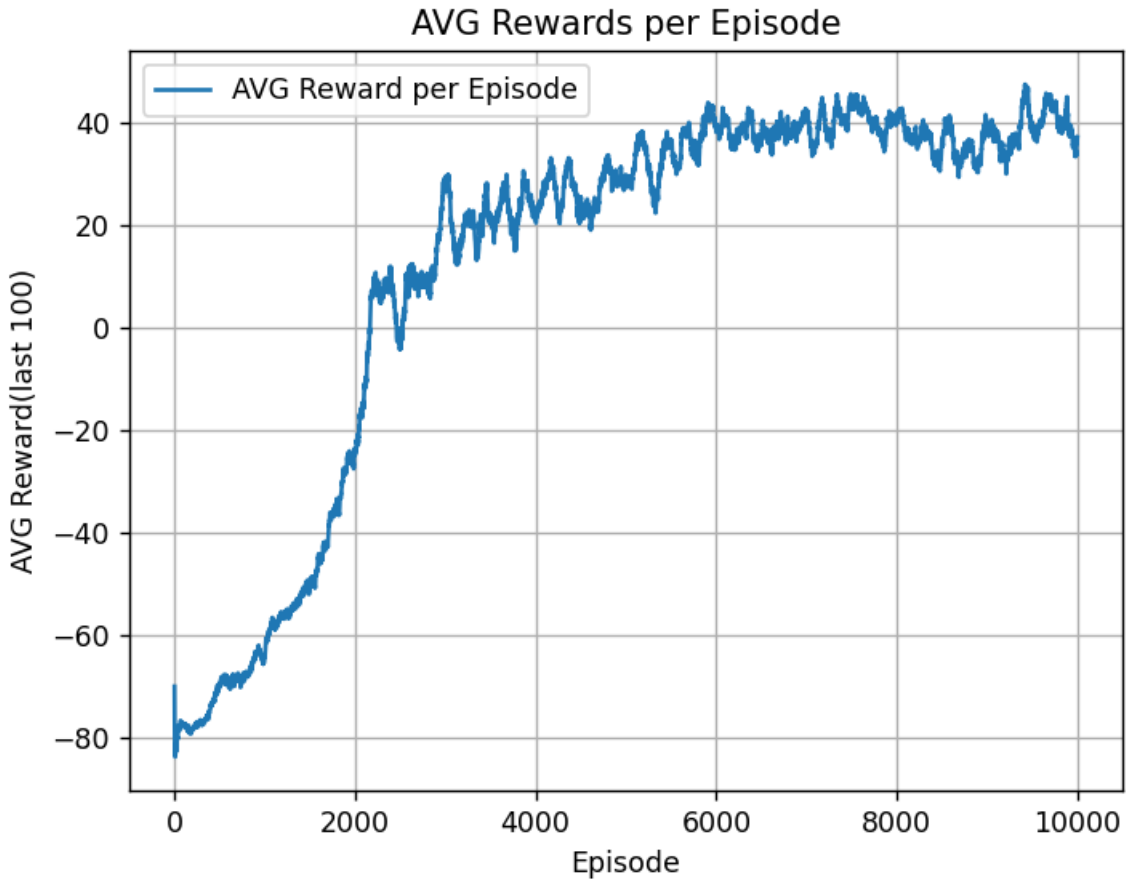
```python
# 에피소드 별 총 리워드를 저장하는 리스트
total_rewards = []
avg_rewards = []

for episode in range(num_episodes):
    state = discretize_state(env.reset()[0])
    done = False
    total_reward = 0

    while not done:
        if np.random.rand() < epsilon:
            action = env.action_space.sample()  # 탐험
        else:
            action = np.argmax(q_table[state])  # 활용

        next_state, reward, done, truncated, _ = env.step(action)
        next_state = discretize_state(next_state)

        if done:
            reward = -100  # 막대가 넘어지면 큰 벌점

        # Q-테이블 업데이트
        q_table[state][action] = q_table[state][action] + learning_rate * (
            reward + discount_factor * np.max(q_table[next_state]) - q_table[state][action]
        )

        total_reward += reward
        state = next_state

    total_rewards.append(total_reward)
    epsilon = max(epsilon_min, epsilon * epsilon_decay)  # 탐험 비율 감소

    if (episode + 1) % 100 == 0:
        print(f"Episode: {episode + 1}, Average Reward (last 100): {np.mean(total_rewards[-100:])}, Epsilon: {epsilon}")
    avg_rewards.append(np.mean(total_rewards[-100:]))
```

$$Q(s,a) \leftarrow (1-\alpha) \cdot Q(s,a) + \alpha \cdot [r + \gamma \cdot \max_{a'} Q(s',a')]$$

예제

예제

```
Episode: 2200, Average Reward (last 100): 7.2, Epsilon: 0.11068126067226178
Episode: 2300, Average Reward (last 100): 6.36, Epsilon: 0.10014353548890782
Episode: 2400, Average Reward (last 100): 10.36, Epsilon: 0.09060908449456685
Episode: 2500, Average Reward (last 100): -3.57, Epsilon: 0.08198238810784661
Episode: 2600, Average Reward (last 100): 10.31, Epsilon: 0.07417702096160789
Episode: 2700, Average Reward (last 100): 6.19, Epsilon: 0.06711478606235186
Episode: 2800, Average Reward (last 100): 10.15, Epsilon: 0.06072493138443261
Episode: 2900, Average Reward (last 100): 13.47, Epsilon: 0.05494344105065345
Episode: 3000, Average Reward (last 100): 29.47, Epsilon: 0.04971239399803625
Episode: 3100, Average Reward (last 100): 15.05, Epsilon: 0.044979383703645896
Episode: 3200, Average Reward (last 100): 19.64, Epsilon: 0.04069699315707315
Episode: 3300, Average Reward (last 100): 20.57, Epsilon: 0.036822319819660124
Episode: 3400, Average Reward (last 100): 18.84, Epsilon: 0.03331654581133795
Episode: 3500, Average Reward (last 100): 21.48, Epsilon: 0.030144549019052724
Episode: 3600, Average Reward (last 100): 23.74, Epsilon: 0.027274551230723157
Episode: 3700, Average Reward (last 100): 22.91, Epsilon: 0.024677799769608873
Episode: 3800, Average Reward (last 100): 21.53, Epsilon: 0.022328279439586606
Episode: 3900, Average Reward (last 100): 26.64, Epsilon: 0.02020245189549843
Episode: 4000, Average Reward (last 100): 22.37, Epsilon: 0.018279019827489446
Episode: 4100, Average Reward (last 100): 26.26, Epsilon: 0.016538713596848224
Episode: 4200, Average Reward (last 100): 31.27, Epsilon: 0.014964098185791003
Episode: 4300, Average Reward (last 100): 24.16, Epsilon: 0.013539398527142203
Episode: 4400, Average Reward (last 100): 27.77, Epsilon: 0.012250341464001188
Episode: 4500, Average Reward (last 100): 25.05, Epsilon: 0.011084012756089733
Episode: 4600, Average Reward (last 100): 21.78, Epsilon: 0.010028727700218176
Episode: 4700, Average Reward (last 100): 24.51, Epsilon: 0.01
Episode: 4800, Average Reward (last 100): 33.56, Epsilon: 0.01
```



Thank you

2024.07.17