강화학습 이론 기초

동양미래대학교

학습목표

- 강화학습의 원리에 대해 이해한다.
- 강화학습의 알고리즘의 주요 성질에 대해 이해한다.

01
강화학습

01. 강화학습

1) 강화학습 (Reinforcement Learning, RL)이란

“강화학습 (RL) 이란”

보상을 최대한 많이 받는 방법을 학습 +100 -50 -50 -50

보상을 최대한 많이 받는 행동을 학습

보상의 합이 최대가 되는 행동을 학습

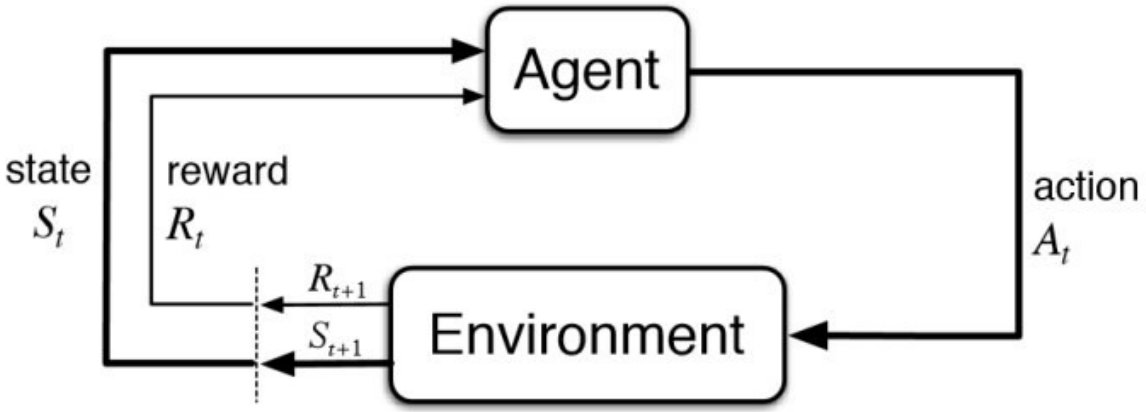
할인된 미래 보상의 합이 최대가 되는 행동을 학습

ex) 3일 뒤 100만원, 1년 뒤 100만원(60만원)

할인된 미래 보상의 합의 평균이 최대가 되는 행동을 학습

01. 강화학습

2) RL 기본 구조



Experience Buffer

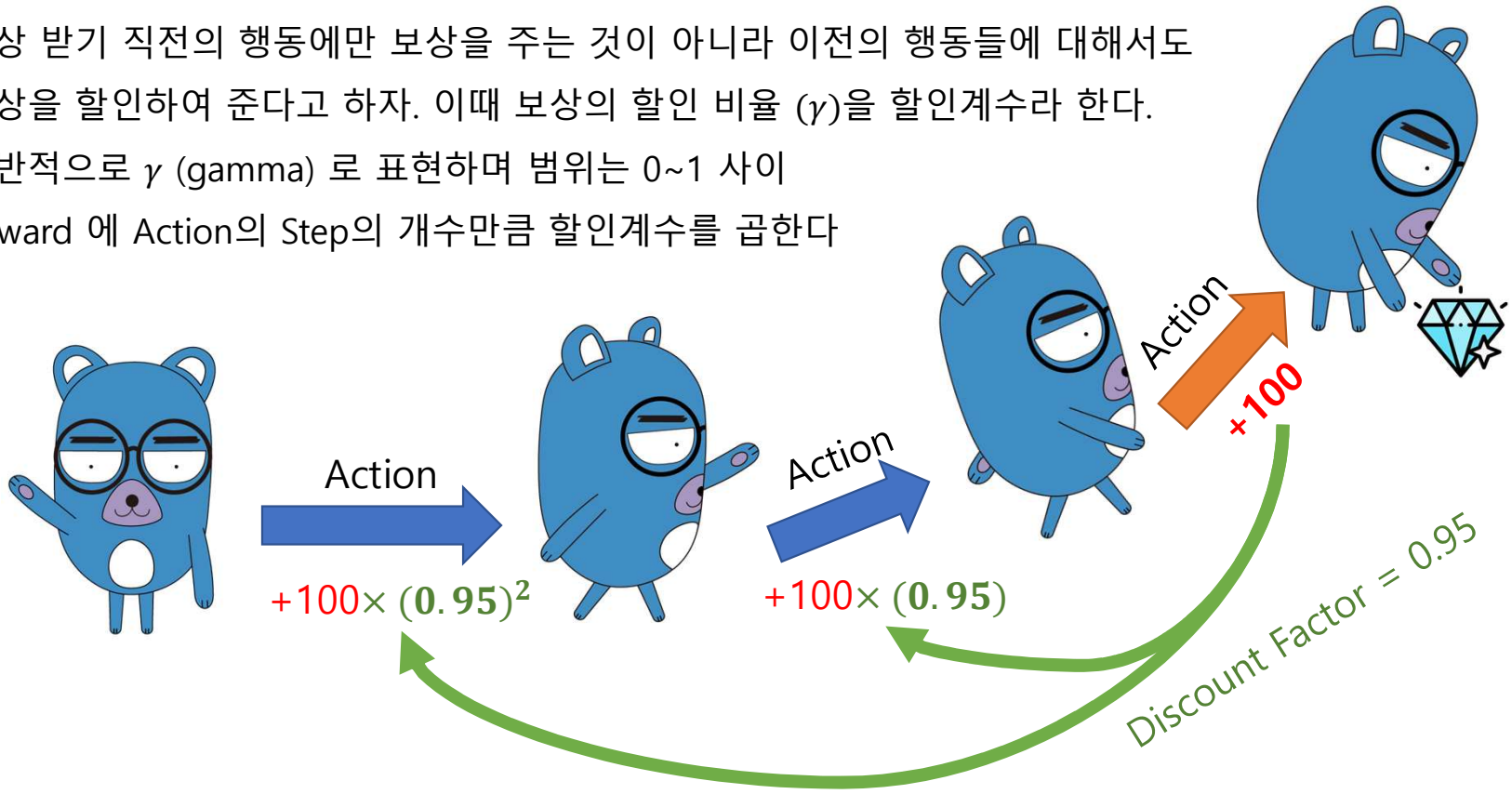
State	Action	Reward	Next State
State	Action	Reward	Next State
State	Action	Reward	Next State
⋮	⋮	⋮	⋮

- Environment (환경) = 게임
- Agent (에이전트) = 게이머
- State (상태) = 게임 화면
- Action (행동) = 키보드
- Reward (보상) = 점수

3) 할인계수, 할인된 미래 보상의 합

할인계수 (Discount Factor)

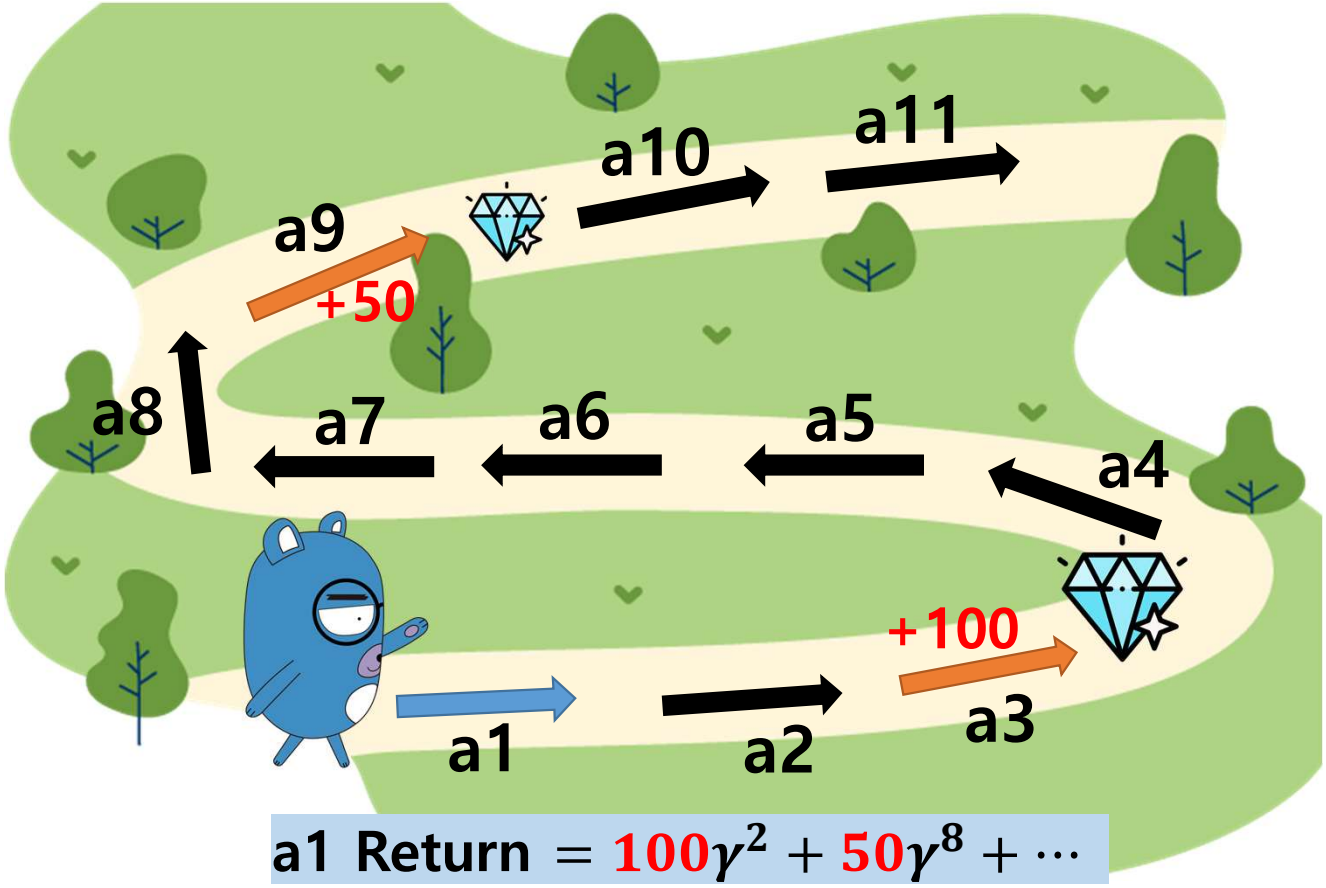
- 보상 받기 직전의 행동에만 보상을 주는 것이 아니라 이전의 행동들에 대해서도 보상을 할인하여 준다고 하자. 이때 보상의 할인 비율 (γ)을 할인계수라 한다.
- 일반적으로 γ (gamma) 로 표현하며 범위는 0~1 사이
- Reward 에 Action의 Step의 개수만큼 할인계수를 곱한다



3) 할인계수, 할인된 미래 보상의 합

할인된 미래 보상의 합 (Return)

$$G_t \overset{def}{=} r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$



3) 할인계수, 할인된 미래 보상의 합

“할인 계수 (Discount Factor)를 어떻게 설정해야 할까?”

- Discount Factor 는 강화학습에서 대표적인 하이퍼파라미터이다. 이 값을 어떻게 정하는지에 따라 학습 결과에 큰 영향을 준다.
- 가장 좋은 방법은 여러 Discount Factor 를 모두 시도해 보는 것인데... 시간과 비용이 매우 많이 든다.
- 전형적인 Discount Factor 의 값은 0.9에서 0.99 사이이다.
- 만약 환경이 연속적이고 초당 Frame 이 크다면 Discount Factor 의 값을 1에 가깝게 두는 것이 좋다.

(예) 만약 초당 24프레임인 연속적인 환경에서 Discount Factor 를 0.995 로 한다면 현재 보상 값에 $(0.995)^{24} \approx 0.89$ 를 곱하여 1초 전 행동에 보상이 지불된다.

한번 더

“강화학습 (RL) 이란”

보상을 최대한 많이 받는 방법을 학습

보상을 최대한 많이 받는 행동을 학습

보상의 합이 최대가 되는 행동을 학습

할인된 미래 보상의 합이 최대가 되는 행동을 학습

할인된 미래 보상의 합의 평균이 최대가 되는 행동을 학습

4) Policy

Policy (정책, π)

- 에이전트가 어떤 행동을 해야 할 지 계산하는 모델. 즉, Agent 의 뇌라고 볼 수 있다.

  우리가 궁극적으로 학습시키고자 하는 대상이다.

- 구체적으로, 현재 상태 S 와 행동 a를 입력 받으면 행동 a가 최대 Return일 가능성을 출력한다.

  수식으로는 $\pi(a|S) \coloneqq P(a\,|S)$ 로 정의한다.

- $\pi(a|S)$ 이 항상 1 또는 0이면 확정적 정책(Deterministic Policy)라 하고 그렇지 않으면

  확률적 정책(Stochastic Policy) 라고 한다.

4) Policy

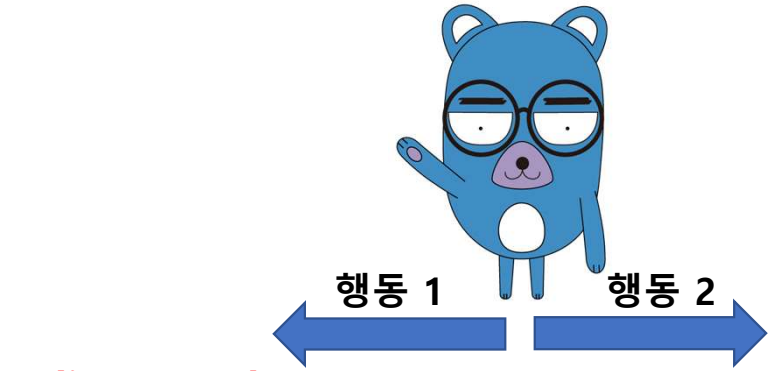
강화학습 모델의 예측 방식

A. 정책 기반 (Policy-Based) 직접적

- Return 이 최대가 되는 행동을 예측
- 구체적으로, 행동의 확률 (분포)를 output 으로 가진다.

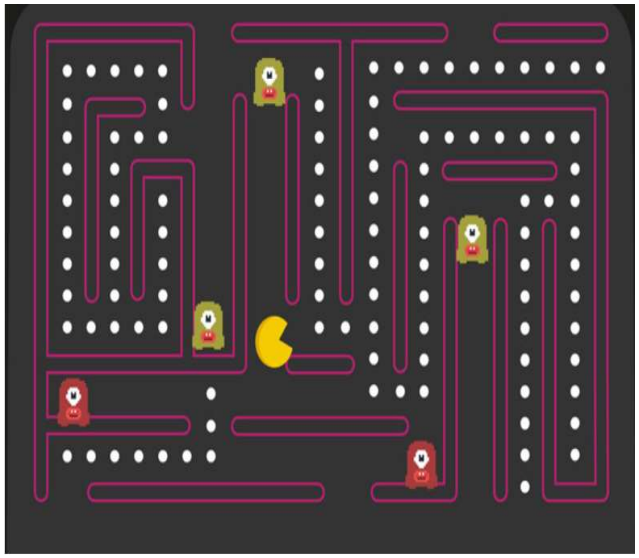
B. 가치 기반 (Value-Based) 간접적

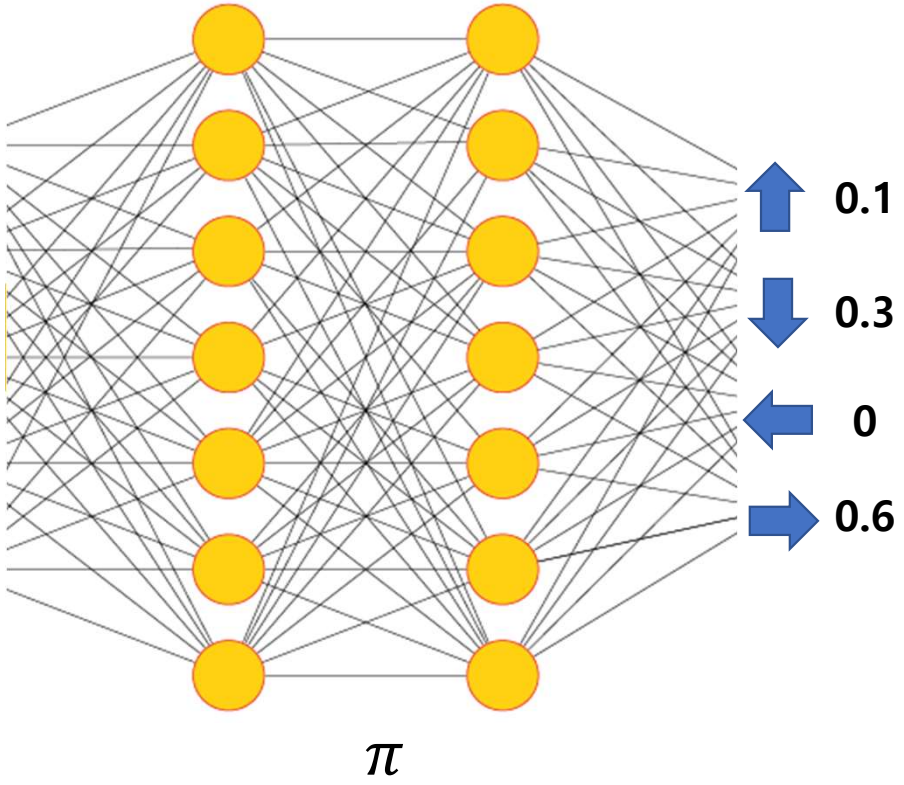
- Return 을 예측
- 예측된 Return을 기반으로 정책을 결정한다.
  (예를 들면, Return 이 가장 큰 값을 행동)



5) 심층 강화학습

심층 강화학습 = 딥러닝 + 강화학습

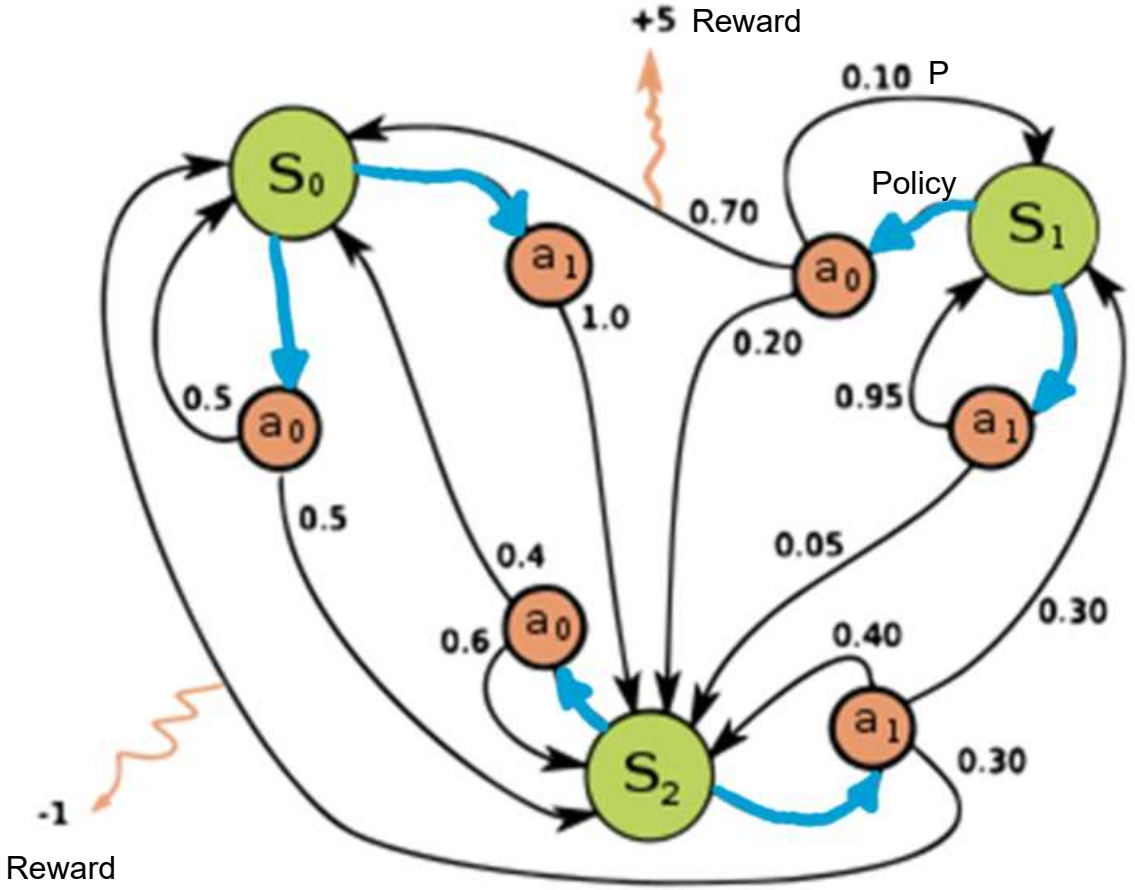




01. 강화학습

6) MDP (Markov Decision Process, 마르코프 결정 과정)

MDP (Markov Decision Process)



- MDP는 강화학습 문제를 풀기 위한 환경의 기본 요소를 정의는 공간으로 $< S, A, P, R, \gamma >$와 같이 표현한다.
- S 는 State(상태) , A 는 Action(행동) , P 는 state transition Probability (상태전이확률), R 은 Reward (보상) , γ는 discount factor (할인계수) 이다.

검은 화살표 = state transition Probability

분홍 화살표 = Reward

하늘색 화살표 = Policy

" 하늘색 화살표(Policy)는 우리가 찾아야 한다"

6) MDP (Markov Decision Process, 마르코프 결정 과정)

상태 전이 확률 P (state transition Probability)

- 상태 S 에서 행동 a를 선택했을 때 S′으로 이동 할 확률. 즉, $P(S'|S,a)$ 로 수식적으로 정의 할 수 있다.
- 상태 전이 확률이 1 또는 0으로만 구성된 환경을 결정론적 환경이라 부른다.
- 상태전이 확률 $P(S'|S,a)$ 에 대한 정보를 알면 Model based, 모르면 Model free 라 한다

보상 R (Reward)

- 상태 S 에서 행동 a를 선택했을 때 S′으로 이동하면 받는 점수.
- $\mathrm{R(S,a,S')}$ 와 같이 함수의 형태로 정의할 수 있다.

7) 가치 함수

가치함수 (Value Function)

A. 상태 가치함수 $V(S)$ (State Value)

- 현재 상태 S에서의 Return의 기댓값. 즉, $V(S) := E[G \mid S]$ (G : Return)
- 정책 π에 의존적이다. 따라서 $V_\pi(S)$ (평균) 와 같이 표기하기도 한다.
- 최적의 상태 가치함수 $V^*(\mathrm{S}) := \max_\pi V_\pi(S)$

B. 행동 가치함수 $\mathrm{Q}(S, A)$ (Q-Value)

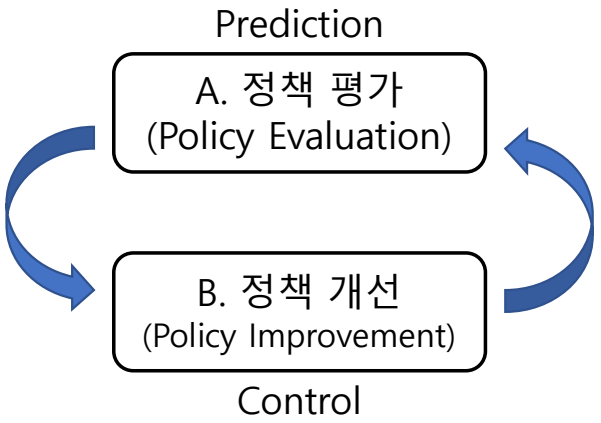
- 현재의 상태 S에서 어떤 행동 A을 했을 때의 Return의 기댓값. 즉, $\mathrm{Q}(S, A) := E[G \mid S, A]$
- 정책 π에 의존적이다. 따라서 $Q_\pi(S, a)$ 와 같이 표기하기도 한다
- 최적의 행동 가치함수 $Q^*(\mathrm{S}, \mathrm{a}) := \max_\pi Q_\pi(S, a)$

Q(S, 위) = 20
Q(S, 아래) = 50
Q(S, 왼) = 80
Q(S, 오) = 200

(참고) V 와 Q 는 아래와 같은 관계를 가진다.

- $V_\pi(S) = \sum_a \pi(a|S) Q_\pi(S, a)$
- $Q_\pi(S, a) = \sum_{S'} P(S'|S, a)[R(S, a, S') + \gamma V_\pi(S')]$

8) 정책 반복 (Policy Iteration)



정책 반복 (Policy Iteration)이란 오른쪽 그림과 같이
A. 정책 평가 (Policy Evaluation) 와 B. 정책 개선
(Policy Improvement)를 반복하는 과정을 의미한다.

- A. 정책 평가 : 가치 함수 (V 또는 Q)를 계산하는 과정
- B. 정책 개선 : 가치 함수를 토대로 정책 (π)를 업데이트 하는 과정
  (즉, 최적의 가치함수 V* 또는 Q* 를 만들기 위해 정책을 수정하는 과정)

일반적으로, 강화학습이 이루어지는 과정은 정책 반복의 과정으로 이루어진다.

8) 정책 반복 (Policy Iteration)

A. 정책 평가

정책 평가에서 가치 함수 (V, Q)를 계산할 때, 주로 반복 알고리즘을 통해 계산한다.

A1) 동적 계획법 (Dynamic Programming, DP)

$$V_{k+1}(s) \leftarrow \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_k(s')]$$

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma Q_k(s',a')]$$

- 상태 전이 확률 P 에 대한 정보를 알 때 (즉, Model Based)에서 사용하는 방식

8) 정책 반복 (Policy Iteration)

A. 정책 평가

정책 평가에서 가치 함수 (V, Q)를 계산할 때, 주로 반복 알고리즘을 통해 계산한다.

A2) 몬테 카를로 방법 (Monte Carlo Method, MC)

$$V_{k+1}(s) \leftarrow (1-\alpha)V_k(s) + \alpha G$$
$$Q_{k+1}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha G$$

- 상태 전이 확률 P 에 대한 정보를 모를 때 (즉, Model Free)에서 사용하는 방식이다.
- G는 할인된 미래 보상의 합(Return) 이다. G를 계산하기 위해서는 에피소드에 끝이 있어야 한다.
- α 는 학습률이다. (예 0.01) 학습률이 작으면 안정적인 학습이 가능하나 수렴속도는 느려진다.

8) 정책 반복 (Policy Iteration)

A. 정책 평가

정책 평가에서 가치 함수 (V, Q)를 계산할 때, 주로 반복 알고리즘을 통해 계산한다.

A3) 시간차 학습 (Temporal Difference learning, TD)

$$V_{k+1}(s) \leftarrow (1-\alpha)V_k(s) + \alpha(r + \gamma V_k(s'))$$
$$Q_{k+1}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha(r + \gamma Q_k(s',a'))$$

- MC와 마찬가지로 상태 전이 확률 P 에 대한 정보를 모를 때 (Model Free) 사용하는 방식이다. MC와 차이점은 TD에서는 에피소드에 끝이 없어도 학습이 가능하다는 점이다.
- α 는 학습률(예 0.01)이다. 학습률이 작으면 안정적인 학습이 가능하나 수렴속도는 느려진다.
- 어느 환경에서도 일반적으로 사용 가능한 정책 평가 알고리즘이다.

8) 정책 반복 (Policy Iteration)

B. 정책 개선

B1) Greedy (탐욕)

$$\pi(S,a) \leftarrow \begin{cases} 1 & (a = \text{argmax}\, Q(S,a)) \\ 0 & \text{otherwise} \end{cases}$$

- 행동 가치 Q가 가장 큰 행동 a만 취하는 방법이다.
- 확정적 정책 (Deterministic Policy)에서의 정책 개선 방식이다.
- 예시

Action	Q_1	π_1	Q_2	π_2	Q_3	π_3	…
up	10	0	30	0	40	0	…
down	20	0	35	0	45	1	…
left	30	0	40	1	40	0	…
right	40	1	35	0	30	0	…

8) 정책 반복 (Policy Iteration)

B. 정책 개선

B2) REINFORCE (강화)

$$\pi_{k+1}(S,a) \leftarrow \pi_k(S,a) + \delta$$

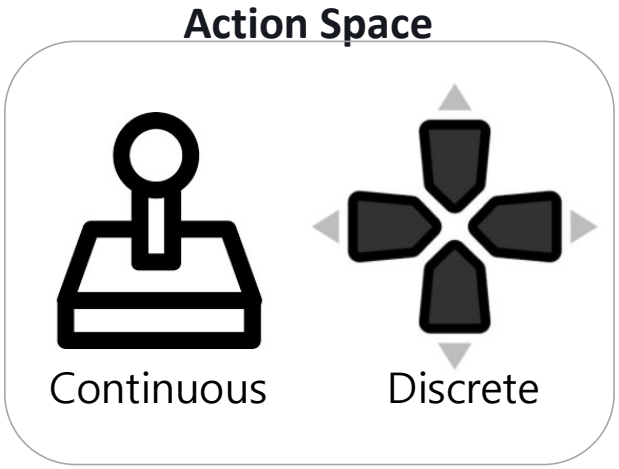
- 가치를 고려하여 정책 확률을 강화 또는 약화시키는 방법이다.
- 확률적 정책 (Stochastic Policy)에서의 정책 개선 방식이다.
- 예시

Action	Q_1	π_1	Q_2	π_2	Q_3	π_3	…
up	10	0.02	30	0.10	40	0.20	…
down	20	0.08	35	0.20	45	0.50	…
left	30	0.20	40	0.50	40	0.20	…
right	40	0.70	35	0.20	30	0.10	…

9) 행동 공간 (Action Space)

이산 행동 공간, 연속 행동 공간

- 이산 행동 공간(Discrete Action Space)
  : 취할 수 있는 행동이 유한
- 연속 행동 공간 (Continuous Action Space)
  : 취할 수 있는 행동이 연속적이고 무한



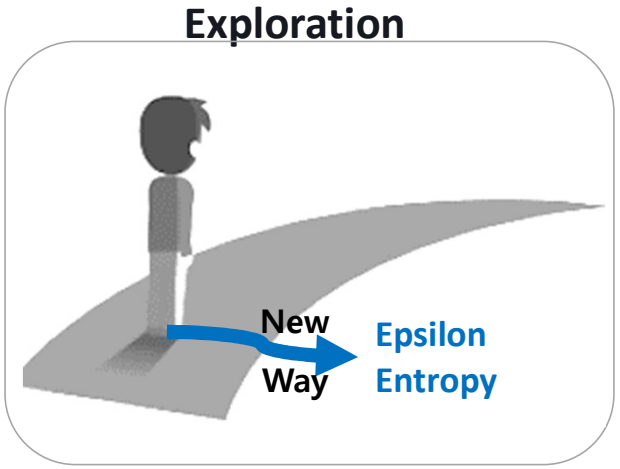
연속 행동 공간에서 강화학습

- 앞에서 주로 이산 행동 공간을 예시로 설명했다. 연속 행동 공간에서는 정책을 어떻게 구성해야 할까?
- 첫째, 연속 공간을 이산 공간으로 만들어서 하는 방법이다. 이를 양자화라고 한다.
- 둘째, 확률질량함수(pmf)가 아닌 확률밀도함수(pdf)를 활용하는 방식이다.
  예를 들어 최적의 행동 분포를 정규분포로 가정하고 평균 m 과 표준편차 σ를 업데이트하며 학습한다.

10) 탐험 (Exploration)

탐험

- 모든 상태와 행동을 경험하는 것은 극단적으로 오랜 시간이 걸릴 수 있다.
- 하지만 너무 Policy 에만 의존하여 경험하는 것은 더 좋은 방법을 찾는 데에 방해가 될 수 있다.
- 따라서 적절한 Exploration 를 섞어가며 Policy 에 기반한 경험이 중요하다.



탐험 하이퍼파라미터

1) Epsilon-Greedy (ϵ-Greedy)
   - 확정적 정책 (Deterministic Policy) 에서 사용하는 탐험 방식이다.
   - ϵ (탐험 비율) 값을 크게 설정하면, 탐험을 장려하여 학습하게 된다.
2) Entropy 최대화 목표
   - 확률적 정책 (Stochastic Policy) 에서 사용하는 탐험 방식이다.
   - Entropy 의 반영비율을 크게 설정하면, 탐험을 장려하여 학습하게 된다.

11) On-Policy , Off-Policy

* 행동 정책 (손) : 경험을 하기 위한 행동을 결정하는 정책.
* 타겟 정책 (뇌) : 학습의 대상이 되는 정책.
* 경험 버퍼 (Experience Buffer) : 경험 데이터가 저장된 장소

On-Policy

- 행동 정책과 타겟 정책이 같아야 하는 알고리즘.
- 자기 반성 스타일
- 정책이 업데이트 되면 기존 경험 버퍼는 모두 폐기해 함

Off-Policy

- 행동 정책과 타겟 정책이 같지 않아도 되는 알고리즘.
- 분석 스타일
- 정책이 업데이트 되기 이전의 경험 버퍼도 활용 가능

- 정확히 말하자면, Experience Buffer 의 분포가 타겟 정책의 분포와 같아야 학습이 가능한 알고리즘이 On-Policy 이고 그렇지 않으면 Off-Policy 이다.
- On-Policy 의 경우 학습이 안정적인 반면 데이터 효율성이 떨어진다는 단점이 있다.

12) 보상의 희소성

보상의 희소성 (간헐적 보상)

- 게임에서 이기면 보상 1점을 준다고 하자. 이때 바둑 또는 스타크래프트와 같은 게임들은 매우 긴 시간이 지나야 보상을 겨우 받을 수 있다.
- 이렇게 보상이 간헐적인 경우 중간 단계에서의 행동에 대한 가치 추정이 불안정해진다.
- 이를 해결하기 위한 방법 중 하나로 Reward Function (보상 함수)를 통해 보상을 단계별로 나누어 주는 것이다.

02
알고리즘

강화학습

02. 알고리즘

1) DQN

DQN (Deep Q Network) 알고리즘

- 가치 기반 (Value_Based) 알고리즘. Q-Value 를 DNN 모델로 만들어 학습.
- 정책 평가 방법으로 시간차 학습 (TD)을 이용.
- 정책 개선 방법으로 탐욕 정책(Greedy-Policy) 방식을 이용.
- 이산 행동 공간(Discrete Action Space)에서만 학습이 가능.
- 탐험 (Exploration) 정책은 Epsilon-Greedy 를 사용함.
- Off-Policy 알고리즘이다.
- 학습이 매우 불안정하다. 이를 개선하기 위한 많은 변종 들이 존재한다.
  (예) DDQN(Double DQN), 중요도 샘플링(Importance Sampling),
  듀얼링 DQN (Dueling DQN), ...등등
- DQN의 여러 변종들은 서로 혼합이 가능하며 이렇게 혼합하여 만든 기법을
  Rainbow 알고리즘이라 한다.

2) PG

PG (Policy Gradient) 알고리즘

- 정책 기반 (Policy_Based) 알고리즘.
- 가치 함수를 활용하지 않음.
- 몬테카를로 방법 (MC)을 이용하여 Return을 구하여 정책을 개선.
- 확률적 정책(Stochastic Policy) 이며 Policy의 Output(확률)의 Gradient를 계산하여 이를 강화 또는 약화시키는 방법으로 정책을 개선함.
- 이산 행동 공간과 연속 행동 공간에서 모두 학습이 가능.
- 탐험 (Exploration) 정책은 사용하지 않는다.
- On-Policy 알고리즘이다.

3) Actor-Critic

Actor-Critic 알고리즘

- DQN 과 PG를 혼합하여 만든 알고리즘.
  따라서 가치 기반 (Value Based)인 동시에 정책 기반 (Policy_Based)이다.
- Critic은 DQN 을 의미하며 DQN과 같은 방식으로 행동 가치 Q-Value 를 추정한다.
- Actor는 PG 를 의미하며 정책의 확률을 추정한다. PG에서는 MC 방법을 이용하여 Return을 직접 구한 후 정책을 개선하지만, Actor는 Critic이 예측한 Q-Value 를 기반으로 정책을 개선해 나간다는 점에서 차이가 있다.
- 이산 행동 공간과 연속 행동 공간에서 모두 학습이 가능.
- 탐험 (Exploration) 정책은 사용하지 않는다.
- On-Policy 알고리즘이다.

4) A2C / A3C

A2C (Advantage Actor-Critic) / A3C (Asynchronous Advantage Actor-Critic) 알고리즘

- Actor-Critic 알고리즘의 변종이다.
- Actor-Critic 에서 Critic 은 행동가치(Q-Value)를 예측하는 반면,
  A2C와 A3C에서 Critic 은 상태가치 V를 예측한다.
- Actor-Critic 에서 Actor는 Q-Value 를 통해 정책을 개선해 나가지만,
  A2C와 A3C에서는 Advantage = $\mathrm{R} + \gamma V(S') - V(S)$ 를 통해 정책을 개선한다.
  이를 통해 Actor-Critic 에서의 단점인 분산이 큰 문제(학습이 불안정)를 개선하였다.
- A3C는 여러 에이전트가 복사된 다른 환경을 탐색하면서 병렬로 학습을 진행한다.
- 이산 행동 공간과 연속 행동 공간에서 모두 학습이 가능.
- On-Policy 알고리즘이다.

5) SAC

SAC (Soft Actor-Critic) 알고리즘

- Actor-Critic 알고리즘의 변종이다.
- Q-Value 를 확률로 변환한 Soft Q-Value 를 타겟으로 하여 Actor의 정책을 개선한다.

  이로 인하여 SAC는 Off-Policy 이며 데이터 효율성을 개선하였다.
- Entropy 를 이용하여 탐험 (Exploration) 정책으로 활용한다.
- 연속 행동 공간에서만 학습이 가능.

  (이산 행동 공간에서 가능하도록 개량한 알고리즘도 존재)

6) PPO

PPO (Proximal Policy Optimization) 알고리즘

- A2C 기반의 TRPO 알고리즘의 간소화 버전이다.
- A2C 알고리즘은 On-Policy 이므로 데이터 효율성이 떨어진다. 하지만 PPO 에서는 과거 데이터 중 현재 정책과 비슷한 정책에서 발생한 데이터를 활용하는 Clipping 기법을 사용하여 데이터 효율성을 높였다.
- A2C에서 정책을 개선할 때 사용하는 Advantage 를 개량하여 GAE(Generalized Advantaged Estimation) 기법을 사용하여 알고리즘의 성능을 높였다.
- Entropy 를 이용하여 탐험 (Exploration) 정책으로 활용한다.
- 연속 행동 공간 및 이산 행동 공간에서 모두 학습이 가능.