

Machine Learning

15강

강화학습

컴퓨터과학과 이관용 교수

학습목차

01 강화학습의 개요

02 Q-학습과 심층 Q-신경망

1

강화학습의 개요

머신러닝의 유형

○ 지도학습 교사학습 supervised learning

→ 분류, 회귀

□ 학습할 때 시스템이 출력해야 할 목표 출력값('교사')을 함께 제공

○ 비지도학습 비교사학습 unsupervised learning

→ 군집화

□ 학습할 때 목표 출력값에 대한 정보가 없음

○ 준지도학습 반지도학습 semi-supervised learning

○ 약지도학습 weakly supervised learning

○ 강화학습 reinforcement learning

□ 출력값에 대한 교사 신호가 '보상' reward 형태로 제공

□ 교사 신호는 정확한 값이 아니고, 즉시 주어지지 않음

강화학습의 적용 사례

○ Google DeepMind, AlphaGo

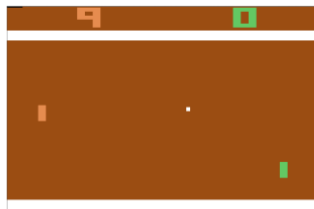
- 프로 기사를 맞바둑으로 이긴 최초의 프로그램 (2016.3)
- 몬테카를로 트리 탐색, 딥러닝, 강화학습 기술의 결합
- 이세돌과의 대국 이후 개량된 버전 출시
 - ✓ 2017.10. 'AlphaGo Zero'
→ 바둑 규칙만 입력, 스스로의 대국을 통해 이치 터득
 - ✓ 2017.12. 'AlphaZero'
→ 모든 보드게임을 위한 알파고 제로의 범용 버전



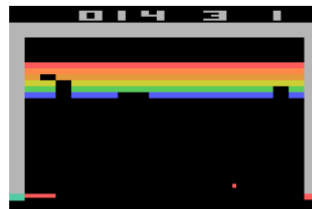
강화학습의 적용 사례

○ 게임

Atari 게임



Pong



Breakout



Space Invaders



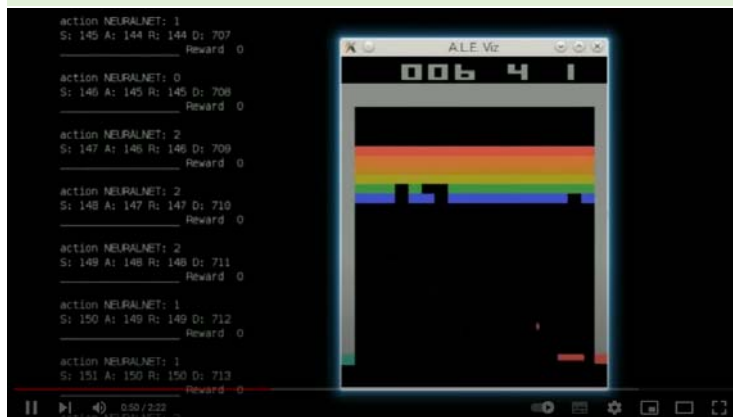
Seaquest



Beam Rider

<https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf>

<https://www.youtube.com/watch?v=MKtNv1UOaZA>



<https://www.youtube.com/watch?v=DMXvkbAtHNY>

AlphaStar

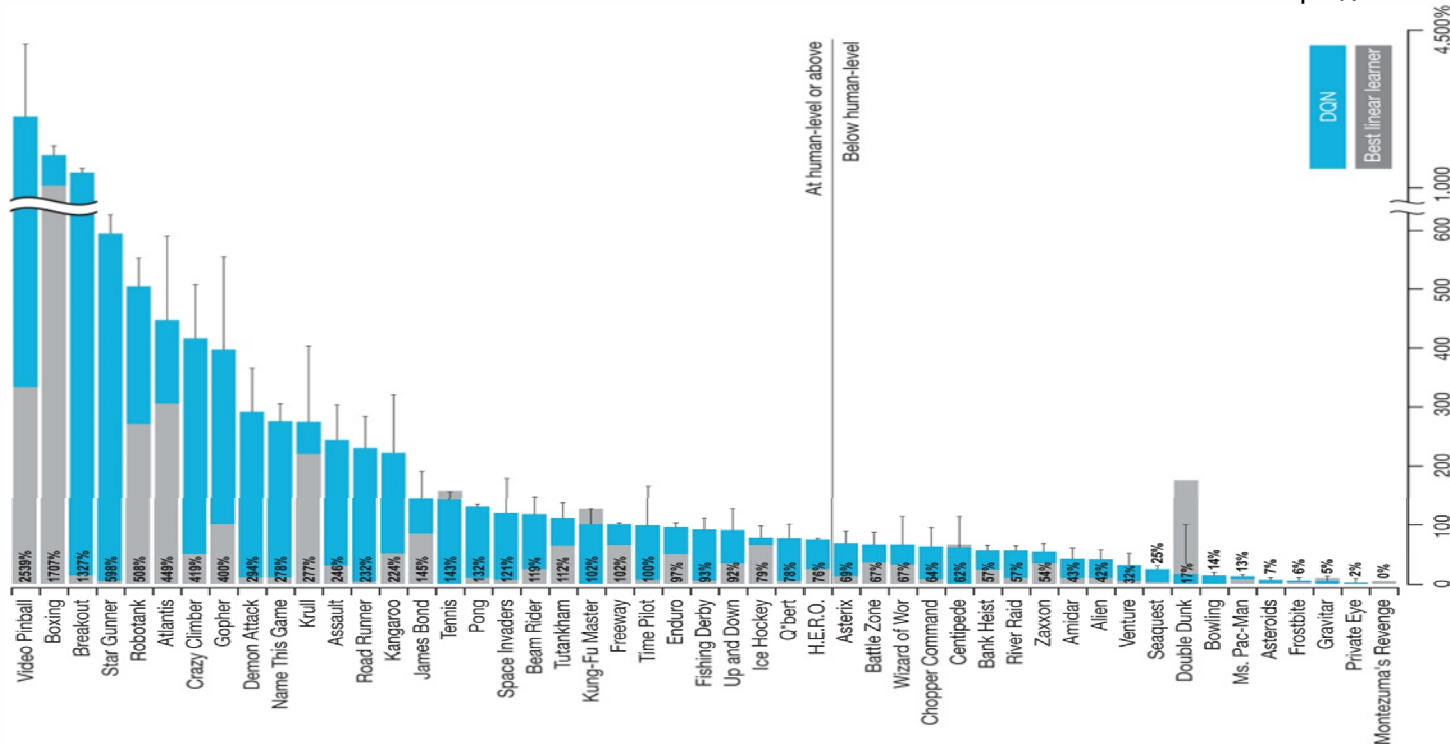
스타크래프트 게임



강화학습의 적용 사례

○ 다양한 게임에 대한 심층 강화학습의 성능 비교

<https://www.nature.com/articles/nature14236>



□ 모든 게임에 대해 하나의 신경망 사용 → DQN 모델(CNN과 Q-학습의 결합)

강화학습의 적용 사례

○ 로봇 제어

- 모방학습 → 비디오 이미지를 보고 로봇이 흉내 내서 동작을 배우도록 학습



<https://youtu.be/Q4bMcUk6pcw>

<https://arxiv.org/pdf/1504.00702.pdf>

강화학습의 적용 사례

○ 시스템 설정/제어



DeepMind AI Reduces Google Data Centre
Cooling Bill by 40%

<https://deepmind.com/applied/deepmind-for-google/>

강화학습의 응용 분야

- 보드게임, 비디오 게임, 로봇 제어, 시스템 제어 등
 - 주어진 상황에서 최적의 제어 신호에 대한 정확한 목표값을 모름
 - 제어 동작 후 결과에 대한 성공 여부의 평가는 가능
- 제어 문제를 표현하고 해결하는 방법으로 주로 사용
 - 주어진 조건이나 상황에서 어떤 동작을 취해야 할지를 결정하는 문제
 - ✓ 시간에 따른 순차적인 개념이 존재
 - 이전 상황이 현재 시점에서의 결정에 영향을 미침

강화학습 reinforcement learning

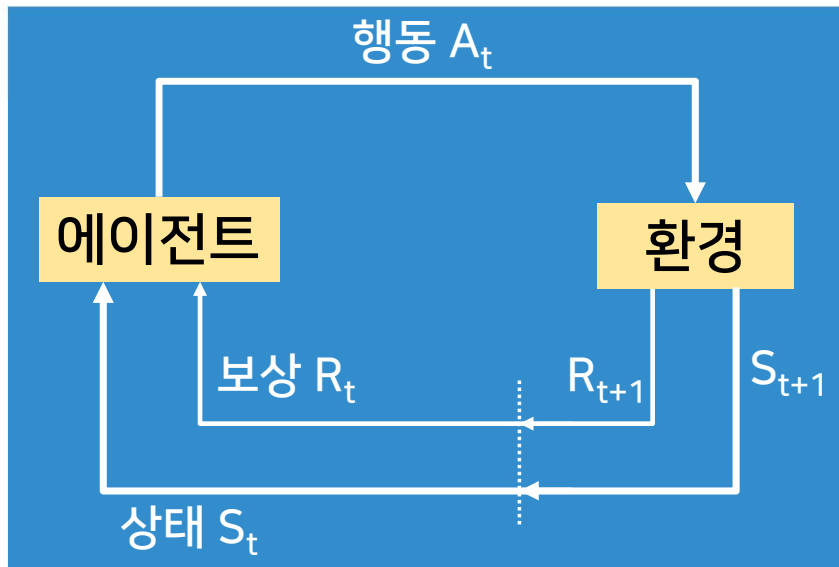
○ 출력값에 대한 교사 신호가 '보상' reward 형태로 제공

- 목표 출력값 없음 → 최종 목표를 지정하고
매 순간 학습 시스템이 취하는 행동의 결과에 따라 보상을 제공

○ 학습의 목적

- 최종적으로 얻게 되는 누적 보상의 최대화
 - ✓ 목표 달성에 적절한 행동 → 양의 보상
 - ✓ 목표 달성에 부적절한 행동 → 음의 보상(벌점)
- 보상을 통해 좋은/나쁜 행동을 배워
행동 방식을 긍정적 방향으로 "강화"한다.

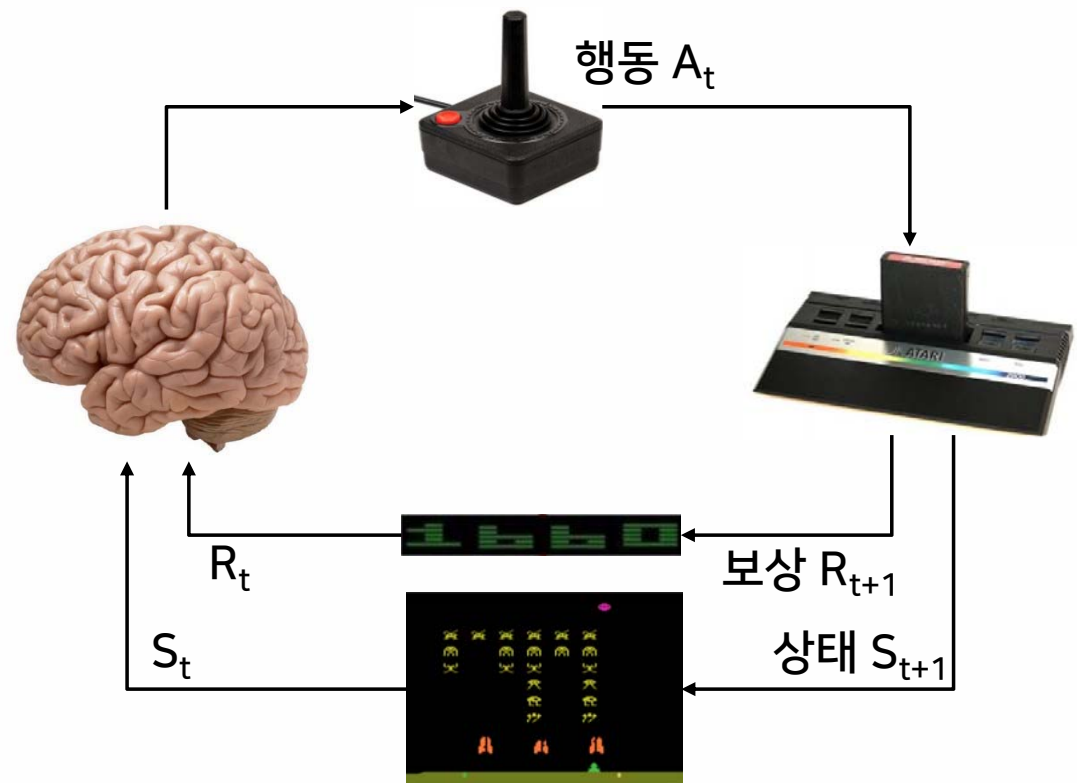
강화학습의 계산 모델



에이전트의 역할

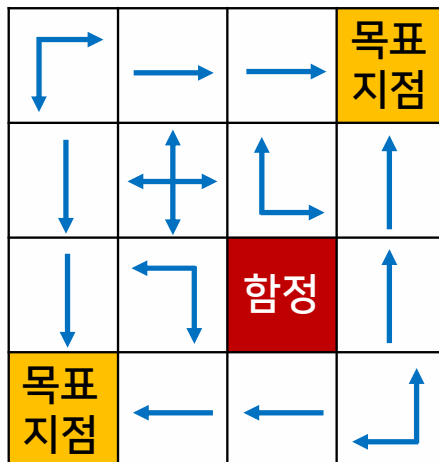
누적 보상을 최대화하는 최적의 행동 결정

집합 A, S, R 의 정의가 필요



상태와 보상의 예

○ 미로_{maze} 찾기



상태 집합 S → 움직일 수 있는 모든 위치로 구성

행동 집합 A → 상·하 좌·우로의 이동 방향

보상 집합 R → 목표지점에 도달하면 **1**
 함정에 빠지면 **-1**
 나머지의 경우 **0**

마르코프 결정 프로세스

○ MDP, Markov Decision Process

- ☐ 학습을 위해 에이전트-환경의 상호작용 상황을 수학적으로 표현한 것
- ☐ 정책 *policy* → 에이전트가 행동을 선택할 때 사용하는 규칙
 - ✓ 에이전트는 정책에 따라 행동을 결정하고,
환경은 주어진 MDP에 따라 다음 상태와 보상을 결정
- ☐ 학습의 목적
 - ✓ 주어진 MDP에서 누적 보상을 최대화하는 최적의 행동을 결정하는 정책을 찾는 것
- ☐ 마르코프 성질 *Markov property*
 - ✓ "다음 행동은 현재의 상태(상황)에 의해서만 결정된다."
 - 이전의 선택/결정들이 영향을 미치지 않음 → 현재 상태만 판단하면 됨

마르코프 결정 프로세스

○ 마르코프 결정 프로세스는 $\langle S, A, P, R, \gamma \rangle$ 의 튜플이다.

□ $S \rightarrow$ 가능한 상태의 유한집합

□ $A \rightarrow$ 가능한 행동의 유한집합

□ $P \rightarrow$ 상태 전이확률

✓ 상태 s 에서 행동 a 를 취했을 때 상태 s' 로 전이될 확률값 $P_{ss'}^a$


$$P_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

□ $R \rightarrow$ 보상 함수 \rightarrow 상태 s 에서 행동 a 를 취했을 때 얻어질 보상의 기대치 R_s^a

$$R_s^a = E[R_{t+1} \mid S_t = s, A_t = a]$$

□ $\gamma \rightarrow$ 보상을 계산할 때 사용되는 할인율 discount factor ($\gamma \in [0,1]$)

미로 찾기 문제에 대한 MDP의 예

1	2	3	4	5
6	7	8	9	10
11		13	14	15
16	17	18	19	20
21	22	23	24	25 Goal

상태 집합 S 로봇 위치의 집합

$$S = \{1, 2, 3, 4, 6, 9, 11, 12, 14, 15, 17, 18, 23, 24, 25\}$$

행동 집합 A 로봇의 위치에 따른 이동 방향의 집합

$$A = \{\text{상, 하, 좌, 우}\}$$

P 해당 위치에서 움직여서 다른 위치로 이동할 확률 \rightarrow 행렬로 표현

보상 집합 R 위치와 행동에 따라 정의

$$R = \{-1, 0, 5\}$$

이동할 수 없는 위치(회색 칸) 선택 \rightarrow 이동하지 않음. -1

이동 가능한 위치 선택 \rightarrow 해당 위치로 이동. 0

목표지점 도착 $\rightarrow 5$

학습해야 할 것은? \rightarrow "How to arrive at the goal"

정책과 가치함수

○ 정책 π

- 상태 S_t 와 행동 A_t 의 시퀀스를 결정하는 함수적 규칙

$$(S_0, A_0) \xrightarrow{R_1} (S_1, A_1) \xrightarrow{R_2} \dots \rightarrow (S_t, A_t) \xrightarrow{R_{t+1}} (S_{t+1}, A_{t+1}) \xrightarrow{R_{t+2}} \dots$$

$$\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s]$$


○ '좋은 정책'에 대한 평가 기준이 필요

- 수익 return, 이익 gain G_t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- ✓ 시점 t 에서부터 얻어지는 보상에 대해 할인율 γ 을 곱해서 더한 값 ("총할인 보상")

total discount reward

1	2	3	4	5
6	7	8	9	10
11		13	14	15
16	17	18	19	20
21	22	23	24	25 Goal

$$\pi(\text{우}|1) = \mathbb{P}[A_t = \text{우} \mid S_t = 1] = \frac{1}{2}$$

$$\pi(\text{하}|1) = \mathbb{P}[A_t = \text{하} \mid S_t = 1] = \frac{1}{2}$$

정책과 가치함수

○ 학습의 목적

- 최종 누적 보상을 최대화하는 최적 정책 optimal policy $\pi^*: S \mapsto A$ 을 찾는 것
- 어떻게 최적 정책을 찾을까? → 상태와 행동에 대한 가치 평가를 수행

○ 가치함수

- 상태 가치함수 $v_\pi(s)$ state-value function $v_\pi(s) = E_\pi[G_t | S_t = s]$
 - ✓ 상태 s 에서 시작해서 정책 π 에 따라 행동을 취하였을 때 얻을 수 있는 기대 보상
total reward, return
- 행동 가치함수 $q_\pi(s, a)$ action-value function $q_\pi(s, a) = E[G_t | S_t = s, A_t = a]$
 - ✓ 정책 π 에 따라 상태 s 에서 행동 a 를 선택하였을 때 얻을 수 있는 기대 보상

최적의 정책과 가치함수

○ 최적 가치 함수

□ 최적 상태 가치함수 $v^*(s)$ $v^*(s) = \max_{\pi} v_{\pi}(s)$

✓ 가능한 모든 정책에 대해 최대값을 갖는 상태 가치함수

□ 최적 행동 가치함수 $q^*(s, a)$ $q^*(s, a) = \max_{\pi} q_{\pi}(s, a)$

✓ 가능한 모든 정책에 대해 최대값을 갖는 행동 가치함수

○ 최적 정책

□ 최적 행동 가치함수 $q^*(s, a)$ 를 최대화하는 행동을 찾으면 됨

$$\pi^*(a|s) = \begin{cases} 1 & \text{if } a = \operatorname{argmax}_{a \in A} q^*(s, a) \\ 0 & \text{otherwise} \end{cases}$$

MDP에서 학습의 어려움

- 에이전트는 상태와 보상에 관해 오직 지역적/부분적 정보만 이용 가능
- 보상이 즉시 주어지기보다는 긴 시간 동안의 지연 발생
- 상태 전이와 보상이 비결정론적인 경우, 함수식으로 정해지지 않거나 알려지지 않은 경우도 존재
- 상태공간과 정책공간이 너무 방대
- 비효율적 학습

→ 심층 Q-학습(Deep Q-learning)이 좋은 해결책이 되고 있음

2

Q-학습과 심층 Q-신경망

Q-학습

○ Q-함수 Q-function

□ 행동 가치함수

$$Q_{\pi}(s, a) = E_{\pi}[G_t | s, a]$$

□ Q-값 Q-value → 주어진 상태-행동 쌍의 기대 보상 → Q-함수의 값

○ Q-학습 Q-learning

□ 최적 정책 π^* 를 얻기 위해서 최적 Q-함수 $Q^*(s, a)$ 를 추정하는 방법

Q-학습 알고리즘

- ① 각 상태 s 와 행동 a 에 대해 $\hat{Q}(s, a)$ 를 0으로 초기화
- ② 임의의 상태 s 를 현재 상태로 지정
- ③ 다음 과정을 무한히 반복
 - ③-1. 행동 a 를 선택하고 실행
 - ③-2. 즉각적인 보상 r 을 얻음
 - ③-3. 새로운 상태 s' 를 얻음
 - ③-4. 보상 r 과 새로운 상태 s' 정보를 사용해서 $\hat{Q}(s, a)$ 를 갱신
$$\hat{Q}(s, a) = r + \max_{a'} \hat{Q}(s', a')$$
 - ③-5. $s \leftarrow s'$

간단한 Q-학습의 예

1 시작	2	3	4
5	6 함정 (-1)	7	8
9 함정 (-1)	10	11	12 함정 (-1)
13	14	15	16 목표 (1)

- 결정론적 → 하나의 상태로만 전이가 발생
- 상태/보상 정보는 부분적으로만 관찰
- $S = \{ 1, 2, 3, \dots, 16 \}$
- $A = \{ R, L, U, D \}$
- $R = \{ -1, 1, 0 \}$

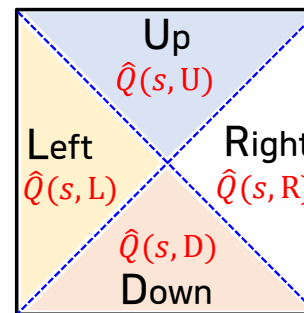

 함정 목표 지점 나머지

Q-테이블

○ Q-함수의 값이 이산적이므로 테이블 형태로 표현 가능

□ 16가지 상태 × 4가지 행동

1 0 시작 0	2 0 0	3 0 0	4 0 0
5 0 0	6 0 함정 (-1) 0	7 0 0	8 0 0
9 0 함정 (-1) 0	10 0 0	11 0 0	12 0 함정 (-1) 0
13 0 0	14 0 0	15 0 0	16 0 목표 (1) 0



Q-학습의 예

현재 상태 선택 $s \rightarrow$ 행동 선택 $a \rightarrow$ 보상 r

\rightarrow 새로운 상태 $s' \rightarrow$ Q 갱신 $\hat{Q}(s, a) \leftarrow r + \max_{a'} \hat{Q}(s', a')$

$s \leftarrow s'$

○ 갱신 과정

$$\begin{aligned}\hat{Q}(1, R) &\leftarrow r + \max_{a'} \hat{Q}(2, a') \\ &= 0 + \max\{\hat{Q}(2, R), \hat{Q}(2, L), \hat{Q}(2, U), \hat{Q}(2, D)\} \\ &= 0 + \max\{0, 0, 0, 0\} = 0\end{aligned}$$

$$\begin{aligned}\hat{Q}(15, R) &\leftarrow r + \max_{a'} \hat{Q}(16, a') \\ &= 1 + \max\{\hat{Q}(16, R), \hat{Q}(16, L), \hat{Q}(16, U), \hat{Q}(16, D)\} \\ &= 1 + \max\{0, 0, 0, 0\} = 1\end{aligned}$$

$$\begin{aligned}\hat{Q}(14, R) &\leftarrow r + \max_{a'} \hat{Q}(15, a') \\ &= 0 + \max\{1, 0, 0, 0\} = 1\end{aligned}$$

1 0 시작 1 0	2 0 0 1	3 0 0 1	4 0 0 0
5 0 0 0	6 0 함정 (-1) 0 0	7 0 0 1	8 0 0 0
9 0 함정 (-1) 0 0	10 0 0 1	11 0 0 1	12 0 함정 (-1) 0 0
13 0 0 0	14 0 0 1	15 0 0 1	16 0 목표 (1) 0 0

행동 선택을 위한 전략

○ 탐험_{exploration} vs 탐사_{exploitation}

- 탐험 → 탐색공간 전체를 골고루 찾음 → 랜덤하게 다른 행동을 선택
- 탐사 → 특정한 곳을 중심으로 주변을 집중적으로 찾음
 - 현재의 Q-함수 정보를 바탕으로 최적의 행동을 선택

○ 혼합된 선택 전략 → **엡실론 탐욕** _{ϵ -greedy}

For time t during learning

$$\epsilon = 0.1/(t + 1)$$

if $\text{rand}(1) < \epsilon$

randomly select a **#탐험**

else

$a = \text{argmax } Q(s, a)$ **#탐사**

시간이 지남에 따라

탐험 정도는 감소, 탐사 정도는 증가

어떤 경로가 더 좋은 것인가?

○ 할인된 미래 보상 discounted future reward

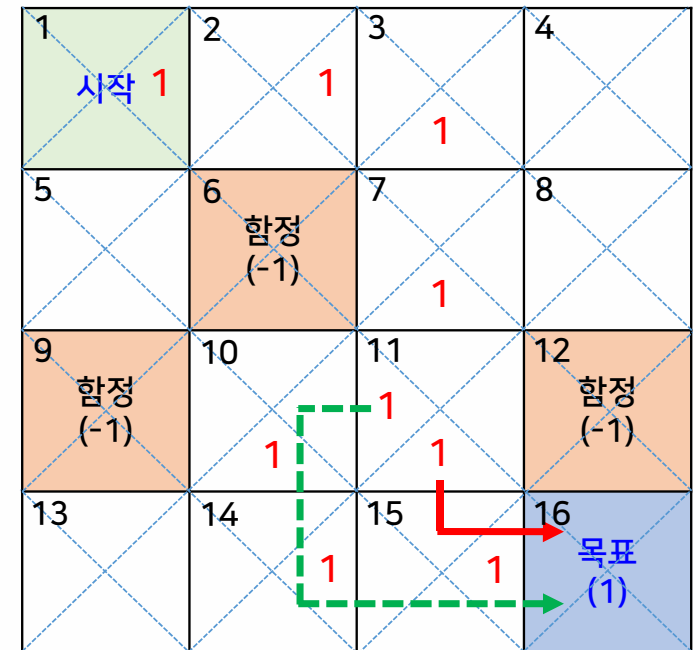
- 미래 보상에 할인율 γ 을 적용

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots + \gamma^{n-t} R_n$$

$$= R_t + \gamma G_{t+1}$$

○ 할인율을 적용한 Q-함수의 갱신 규칙

$$\hat{Q}(s, a) = r + \gamma \max_{a'} \hat{Q}(s', a')$$



어떤 경로가 더 좋은 것인가?

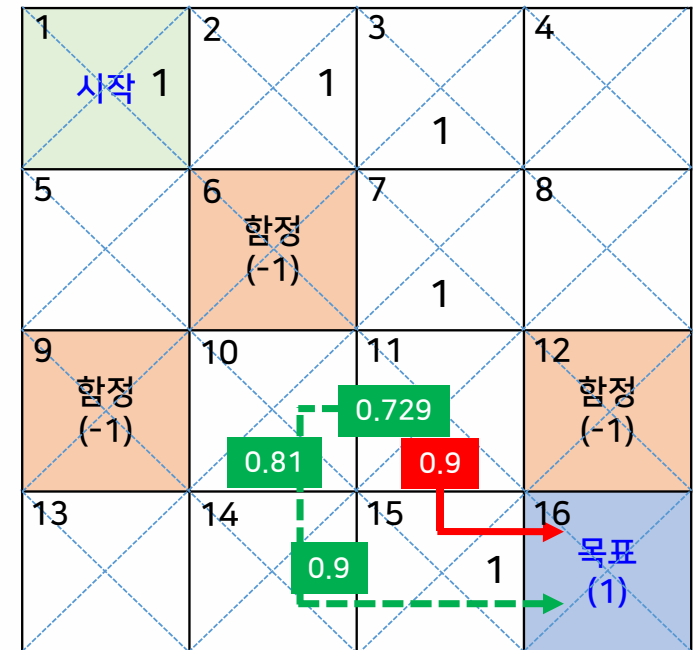
○ 상태 11에서의 Q-값 ($\gamma = 0.9$)

$$\begin{aligned}
 \hat{Q}(11, D) &\leftarrow r + \gamma \max_{a'} \hat{Q}(15, a') \\
 &= 0 + 0.9 \times \max\{\hat{Q}(15, R), \hat{Q}(15, L), \hat{Q}(15, U), \hat{Q}(15, D)\} \\
 &= 0 + 0.9 \times \max\{1, 0, 0, 0\} = 0.9
 \end{aligned}$$

$$\hat{Q}(14, R) = 0.9$$

$$\begin{aligned}
 \hat{Q}(10, D) &\leftarrow r + \gamma \max_{a'} \hat{Q}(14, a') \\
 &= 0 + 0.9 \times \max\{\hat{Q}(14, R), \hat{Q}(14, L), \hat{Q}(14, U), \hat{Q}(14, D)\} \\
 &= 0 + 0.9 \times \max\{0.9, 0, 0, 0\} = 0.81
 \end{aligned}$$

$$\begin{aligned}
 \hat{Q}(11, L) &\leftarrow r + \gamma \max_{a'} \hat{Q}(10, a') \\
 &= 0 + 0.9 \times \max\{\hat{Q}(10, R), \hat{Q}(10, L), \hat{Q}(10, U), \hat{Q}(10, D)\} \\
 &= 0 + 0.9 \times \max\{0, 0, 0, 0.81\} = 0.729
 \end{aligned}$$



비결정론적 환경

○ 결정론적_{deterministic} 모델

- ☐ 모델의 출력과 행동이 어떤 임의성도 없이 초기 조건과 파라미터에 의해서만 전적으로 결정
- ☐ 확률적 불확실성이 존재하지 않음

○ 비결정론적_{non-deterministic} 모델

- ☐ 모델에 임의성이 내재되어 있음
- ☐ 동일한 초기 조건과 파라미터 집합일지라도 서로 다른 여러 출력을 생성
- ☐ 예: 장기, 바둑, 체스 등

비결정론적 환경

○ 상태 전이와 $\hat{Q}(s', a') \rightarrow$ 비결정론적

- 결정론적 버전의 Q-학습이 제대로 동작하지 않음
- 해결책 \rightarrow 신뢰 요소 belief factor α 도입

○ α 를 가진 Q-학습의 갱신 규칙

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

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

① 각 상태 s 와 행동 a 에 대해 $\hat{Q}(s, a)$ 를 0으로 초기화

② 임의의 상태 s 를 현재 상태로 지정

③ 다음 과정을 무한히 반복

③-1. 행동 a 를 선택하고 실행

③-2. 즉각적인 보상 r 을 얻음

③-3. 새로운 상태 s' 를 얻음

③-4. r 과 s' 정보를 사용해서 $\hat{Q}(s, a)$ 를 갱신

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

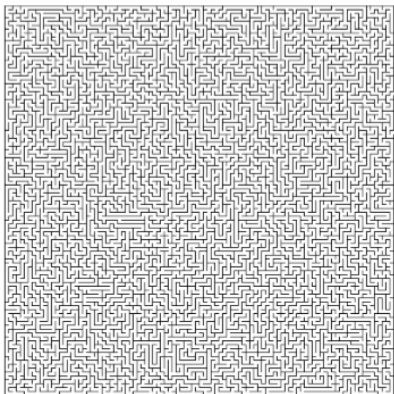
③-5. $s \leftarrow s'$

수렴 특성이 증명됨(Watkins and Dayan, 1992)

Q-테이블에서 Q-신경망으로

○ Q-테이블의 한계

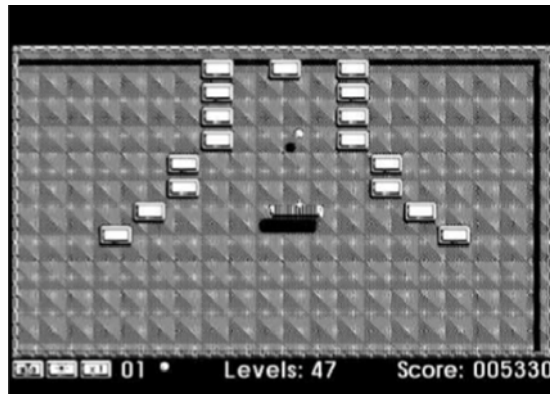
□ 실제 응용에서의 비효율성



미로 (100×100)

$$\downarrow$$

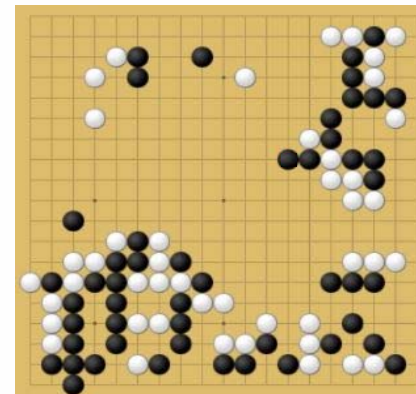
$$10^4 \times 4$$



아타리 게임

$$\downarrow$$

$$256^{84 \times 84}$$



바둑

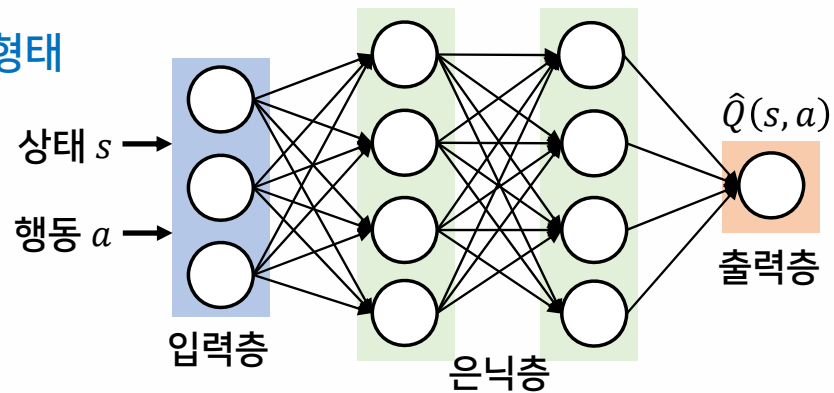
$$\downarrow$$

$$10^{170}$$

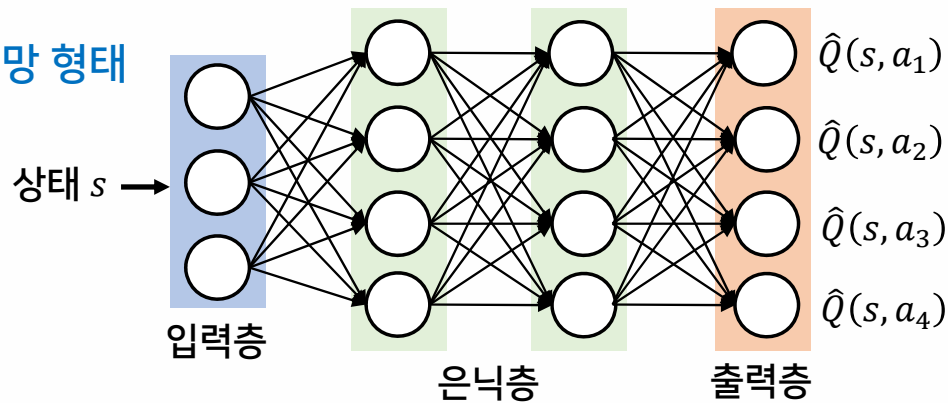
Q-신경망 Q-Network

○ 신경망을 이용하여 Q-함수를 표현하고 추정

Q-신경망의 기본 형태

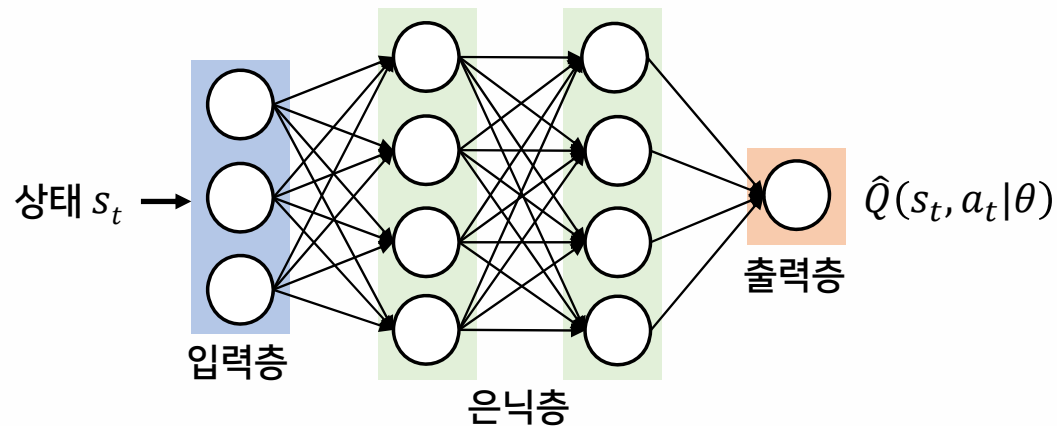


구글 딥마인드가 재정의한 Q-신경망 형태



Q-신경망의 학습

- 시점 t 에서 학습 파라미터 θ 에 대한 Q-신경망의 실제 출력



- 목표 출력값 $y_t = r_t + \gamma \max_{a'} Q^*(s_{t+1}, a') \approx r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta)$

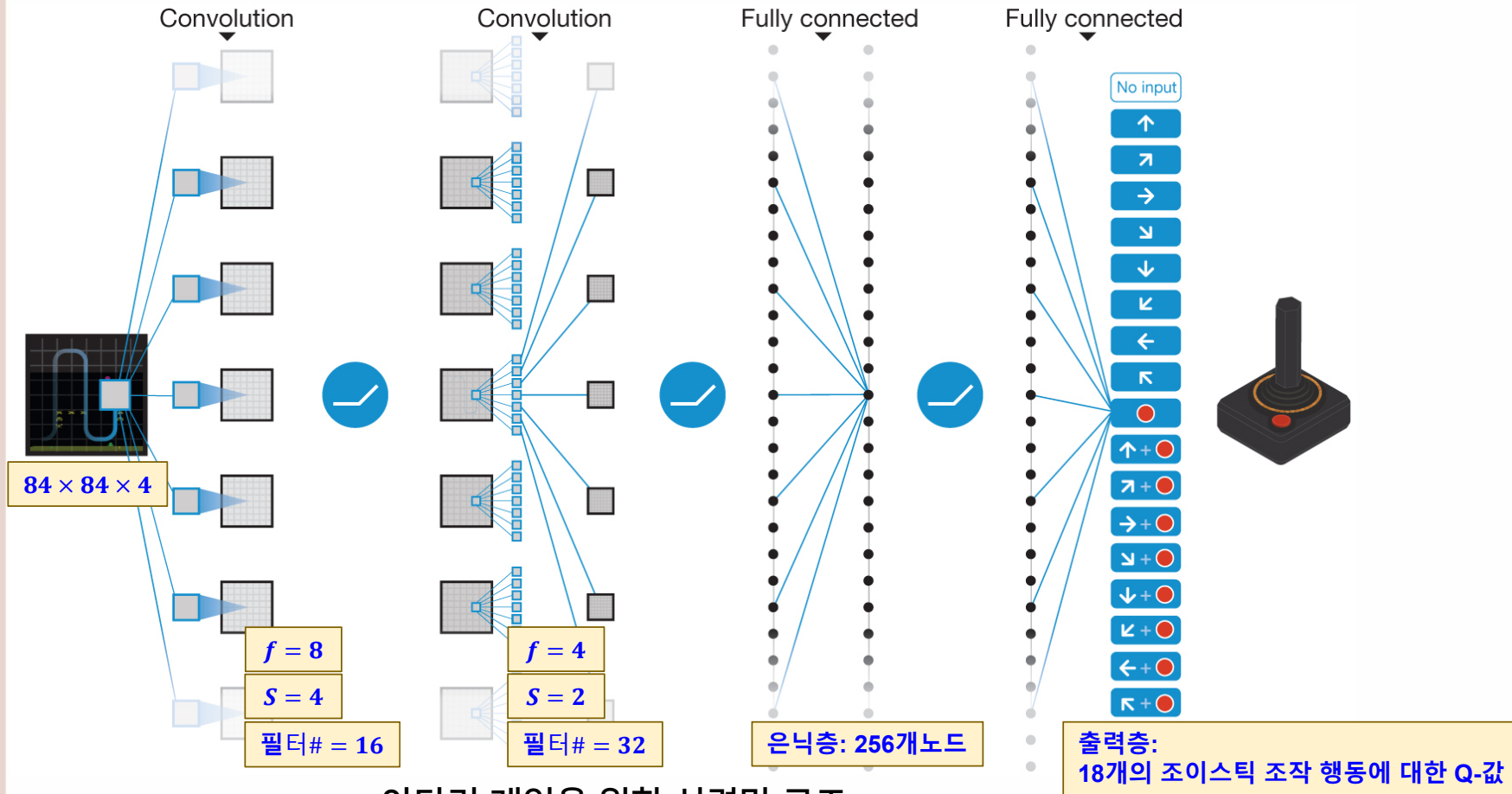
- 목적함수 $E(\theta) = \sum_{t=0}^T \left\{ \hat{Q}(s_t, a_t | \theta) - \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta) \right) \right\}^2$

Q-신경망 학습의 불안정성

- 목적함수 $E(\theta) = \sum_{t=0}^T \left\{ \hat{Q}(s_t, a_t | \theta) - \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta) \right) \right\}^2$
 - Q-테이블 표현에 대해서는 수렴, Q-신경망은 발산
- 불안정성의 원인
 - 데이터 간의 높은 상관관계
 - 목표 출력값이 시간에 따라 변함
- 해결책
 - DQN Deep Q-Network
 - ✓ 심층 신경망
 - ✓ 경험 재현 experience replay, 목표망 target network

Deep Q-Network

<https://www.nature.com/articles/nature14236.pdf>



아타리 게임을 위한 신경망 구조

경험 재현, 목표망

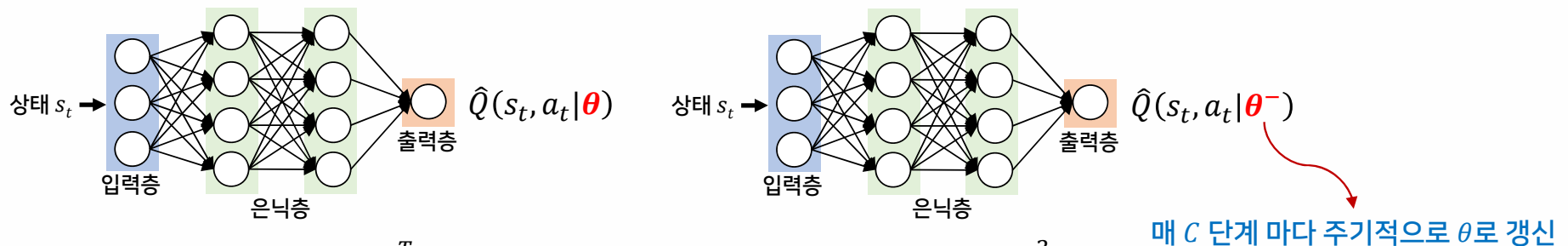
○ 경험 재현 → 데이터 간의 높은 상관관계에 따른 문제 해결

□ 재현을 통해 학습 데이터의 시퀀스를 재구성

- ✓ 에이전트의 경험 (s_t, a_t, r_t, s_{t+1}) 을 시간 간격 단위로 재현 메모리 D 에 저장한 후, D 로부터 균등 무작위 추출을 통해 미니 배치를 구성하여 학습 진행

○ 목표망 → 시변적인 목표 출력값 문제 해결 $E(\theta) = \sum_{t=0}^T \left\{ \hat{Q}(s_t, a_t | \theta) - \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta) \right) \right\}^2$

□ 원래 Q-신경망과 같은 구조, 다른 파라미터 θ^- 를 가진 별도의 신경망



□ 목적함수 $E(\theta) = \sum_{t=0}^T \left\{ \hat{Q}(s_t, a_t | \theta) - \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta^-) \right) \right\}^2$

DQN 학습 알고리즘

<https://www.nature.com/articles/nature14236.pdf>
Algorithm 1: deep Q-learning with experience replay.Initialize replay memory D to capacity N Initialize action-value function Q with random weights θ Initialize target action-value function \hat{Q} with weights $\theta^- = \theta$ **For** episode = 1, M **do**Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$ **For** $t = 1, T$ **do**

With probability ε select a random action a_t } ε -greedy
 otherwise select $a_t = \operatorname{argmax}_a Q(\phi(s_t), a; \theta)$

Execute action a_t in emulator and observe reward r_t and image x_{t+1} Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$ Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ Every C steps reset $\hat{Q} = Q$ **End For****End For**

경험 재현

목표망

DQN 학습 알고리즘

초기화 단계

- 재현 메모리 D 준비
- 학습을 통해 최적화할 행동 가치함수 $Q(\theta)$ 준비 (θ 를 랜덤 초기화)
- 학습 신호를 제공할 목표망 함수 $\hat{Q}(\theta^-)$ 준비 ($\theta^- = \theta$)

반복 수행

• 시작 상태 s_1 설정

- ε -탐욕 방법에 의한 행동 선택
 - ε 의 확률로 랜덤하게 행동 a_t 를 선택
 - $1 - \varepsilon$ 의 확률로 $Q(\theta)$ 를 이용한 행동 선택: $a_t = \operatorname{argmax}_a Q(s_t, a|\theta)$

- 행동 실행 → 행동 a_t 를 수행하여 보상 r_t 와 다음 상태 s_{t+1} 을 관찰

- 경험 재현 방법 적용: 메모리에 저장 및 미니 배치 구성
 - 경험 (s_t, a_t, r_t, s_{t+1}) 를 재현 메모리 D 에 저장
 - D 로부터 랜덤 샘플링을 하여 학습용 미니 배치 (s_j, a_j, r_j, s_{j+1}) 구성

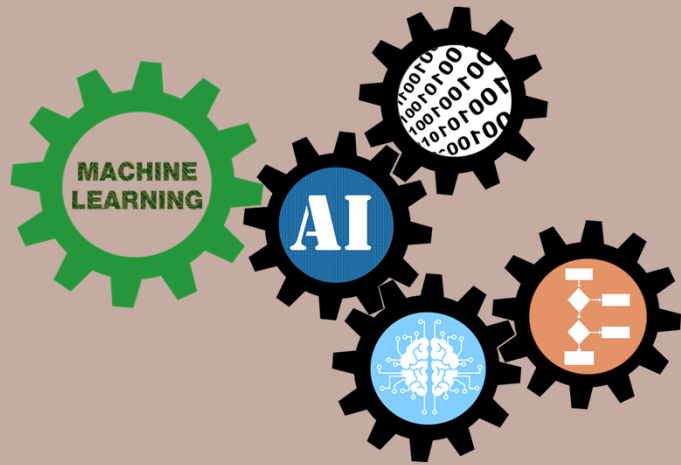
- 기울기 강하 기법으로 θ 학습
 - 목표망을 이용한 목표값 계산

$$y_j = \begin{cases} r_j & \text{if episode terminates at step } j + 1 \\ r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'|\theta^-) & \text{otherwise} \end{cases}$$

목표값을 이용하여 손실함수 $(y_j - Q(s_j, a_j|\theta))^2$ 의 기울기를 따라 θ 수정

- 목표망의 주기적 갱신 → 매 C 단계마다 목표망 수정: $\hat{Q} = Q$ (즉 $\theta^- = \theta$)

$t = 1, \dots, T$



강의 끝

지금까지 함께 해 준 여러분
감사합니다.!