

10기 & 11기 정규

ToBig's 10기 이민주

강화학습

Reinforcement Learning

Content

Unit 01 | Introduction of RL

Unit 02 | MDP & Bellman Equation

Unit 03 | Dynamic Programming

Unit 04 | Model Free

Unit 05 | Deep RL

Unit 00 | Reference

[1부] 강화학습 소개

- 1장: 강화학습 개요
 - 강화학습의 개념
 - 스키너의 강화 연구
 - 우리 주변에서의 강화
 - 머신러닝과 강화학습
 - 스스로 학습하는 컴퓨터, 에이전트
- 강화학습 문제
 - 순차적 행동 결정 문제
 - 순차적 행동 결정 문제의 구성 요소
 - 방대한 상태를 가진 문제에서의 강화학습
- 강화학습의 예시: 브레이크아웃
 - 딥마인드에 의해 다시 빛을 본 아타리 게임
 - 브레이크아웃의 MDP와 학습 방법
- 정리
 - 강화학습의 개념
 - 강화학습 문제
 - 강화학습의 예시: 브레이크아웃

[2부] 강화학습 기초

- 2장: 강화학습 기초 1 - MDP와 벨만 방정식
 - MDP
 - 상태
 - 행동
 - 보상함수
 - 상태 변화 확률
 - 감가율
 - 정책
 - 가치함수
 - 가치함수
 - 큐함수
 - 벨만 방정식
 - 벨만 기대 방정식

— 벨만 최적 방정식

- 정리
- MDP
- 가치함수
- 벨만 방정식

3장: 강화학습 기초 2 - 그리드월드와 다이내믹 프로그래밍

- 다이내믹 프로그래밍과 그리드월드
 - 순차적 행동 결정 문제
 - 다이내믹 프로그래밍
 - 격자로 이뤄진 간단한 예제: 그리드월드
- 다이내믹 프로그래밍 1: 정책 이터레이션
 - 강화학습 알고리즘의 흐름
 - 정책 이터레이션
 - 정책 평가
 - 정책 발전
 - RLCode 깃허브 저장소
 - 정책 이터레이션 코드 설명
 - 정책 이터레이션 코드 실행

4장: 강화학습 기초 3 - 그리드월드와 큐러닝

- 강화학습과 정책 평가 1: 몬테카를로 예측
 - 사람의 학습 방법과 강화학습의 학습 방법
 - 강화학습의 예측과 제어
 - 몬테카를로 근사의 예시
 - 샘플링과 몬테카를로 예측
- 강화학습과 정책 평가 2: 시간차 예측
 - 시간차 예측
 - 강화학습 알고리즘 1: 살사
 - 살사
 - 살사 코드 설명
 - 살사 코드의 실행 및 결과
- 강화학습 알고리즘 2: 큐러닝
 - 살사의 한계
 - 큐러닝 이론
 - 큐러닝 코드 설명
 - 큐러닝 코드의 실행 결과
- 정리
 - 강화학습과 정책 평가 1: 몬테카를로 예측
 - 강화학습과 정책 평가 2: 시간차 예측
 - 강화학습 알고리즘 1: 살사
 - 강화학습 알고리즘 2: 큐러닝

[3부] 강화학습 심화

- 5장: 강화학습 심화 1 - 그리드월드와 근사함수
 - 근사함수
 - 몬테카를로, 살사, 큐러닝의 한계
 - 근사함수를 통한 가치함수의 매개변수화
 - 인공신경망
 - 인공신경망 1: 인공신경망의 개념
 - 인공신경망 2: 노드와 활성함수
 - 인공신경망 3: 딥러닝
 - 인공신경망 4: 신경망의 학습
 - 인공신경망 라이브러리: 캐라스

— 캐리스 소개

- 간단한 캐리스 예제

— 딥살사

- 딥살사 이론
- 딥살사 코드 설명
- 딥살사의 실행 및 결과

— 폴리시 그레이디언트

- 정책 기반 강화학습
- 폴리시 그레이디언트
- REINFORCE 코드 설명
- REINFORCE의 실행 및 결과

— 정리

- 근사함수
- 인공신경망
- 인공신경망 라이브러리: 캐리스
- 딥살사
- 폴리시 그레이디언트

6장: 강화학습 심화 2 - 카트풀 알고리즘 1: DQN

- 카트풀 예제의 정의
- DQN 이론
- DQN 코드 설명
- DQN 실행 및 결과

— 알고리즘 2: 액터-크리틱

- 액터-크리틱 이론 소개
- 액터-크리틱 코드 설명
- 액터-크리틱 실행 및 결과

— 정리

- 알고리즘 1: DQN
- 알고리즘 2: 액터-크리틱

7장: 강화학습 심화 3 - 아타리

- 브레이크아웃 DQN
- 아타리: 브레이크아웃

— 컨볼루션 신경망(CNN)이란?

- 브레이크아웃의 컨볼루션 신경망

— DON 학습 전 준비 사항

- DON 코드 설명

— 텐서보드 사용법

- 브레이크아웃 DON 실행 및 결과

— 브레이크아웃 A3C

- DQN의 한계

— A3C란?

- 멀티스레딩 소개

— 브레이크아웃 A3C 코드 설명

— 브레이크아웃 A3C 실행 결과

- 브레이크아웃 DQN, A3C의 저장된 모델 플레이

— 정리

- 브레이크아웃 DQN

— 브레이크아웃 A3C

부록A: 학습결과 업로드

- 오픈에이아이 짐의 다양한 예제
- 오픈에이아이 짐에 학습 결과를 업로드

파이썬
캐리스로 배우는
강화학습



내 속으로
학습
구현하는
개념
실행
기록
모든 내용
온라인으로
제공되는
강화학습

DQN

3

Unit 00 | Reference

Lecture 1: [Introduction to Reinforcement Learning](#)

Lecture 2: [Markov Decision Processes](#)

Lecture 3: [Planning by Dynamic Programming](#)

Lecture 4: [Model-Free Prediction](#)

Lecture 5: [Model-Free Control](#)

Lecture 6: [Value Function Approximation](#)

Lecture 7: [Policy Gradient Methods](#)

Lecture 8: [Integrating Learning and Planning](#)

Lecture 9: [Exploration and Exploitation](#)

Lecture 10: [Case Study: RL in Classic Games](#)

<http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>
David Silver RL

Unit 01 | Introduction of RL

• Reinforcement Learning

행동심리학의 “강화”

- 시행착오를 통해 학습하는 방법
- 이전에 배우지 않았지만 직접 시도하면서 행동과 결과로 나타나는 좋은 보상 사이의 상관관계를 학습하고, 좋은 보상을 얻게 해주는 행동을 점점 더 많이 하는 것

Unit 01 | Introduction of RL

• Reinforcement Learning

스키너 상자 실험

먹이(보상)에 의해 강화

1. 배고픈 상태의 흰 쥐를 스키너 상자에 넣는다.
2. 흰 쥐는 스키너 상자 안에서 돌아다니다가 우연히 지렛대를 누르게 된다.
3. **지렛대를 누르자 먹이가 나온다.**
4. 지렛대와 먹이 간의 상관관계를 알지 못하는 쥐는 다시 상자 안을 돌아다닌다.
5. 다시 우연히 지렛대를 누른 흰 쥐는 또 먹이가 나오는 것을 보고
지렛대를 누르는 행동을 자주 하게 된다.
6. 이러한 과정이 반복되면서 흰 쥐는 지렛대를 누르면 먹이가 나온다는 사실을 학습하게 된다.

Unit 01 | Introduction of RL

- Example of Reinforcement Learning - Atari Breakout

보상 :

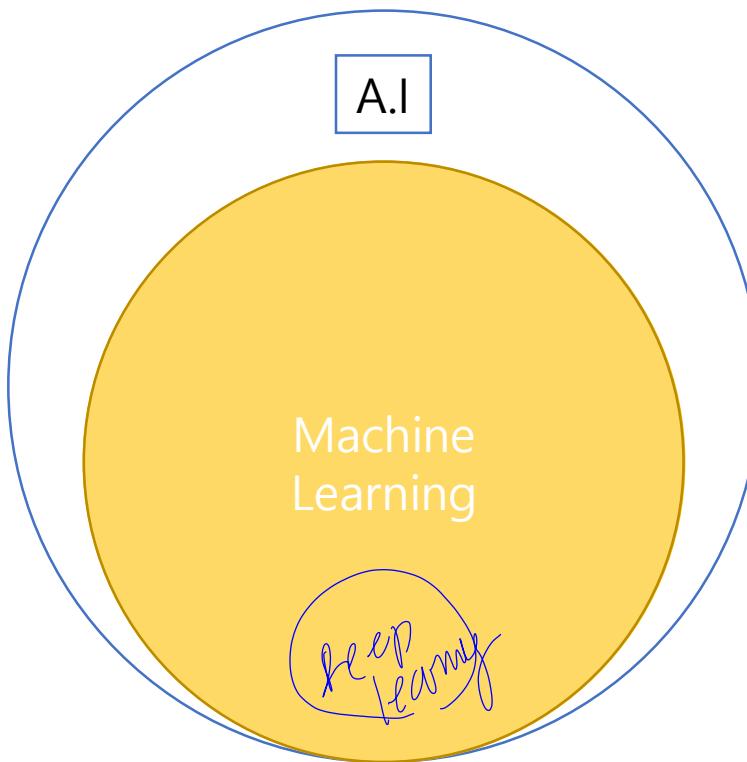
score



<https://www.youtube.com/watch?v=V1eYniJ0Rnk>

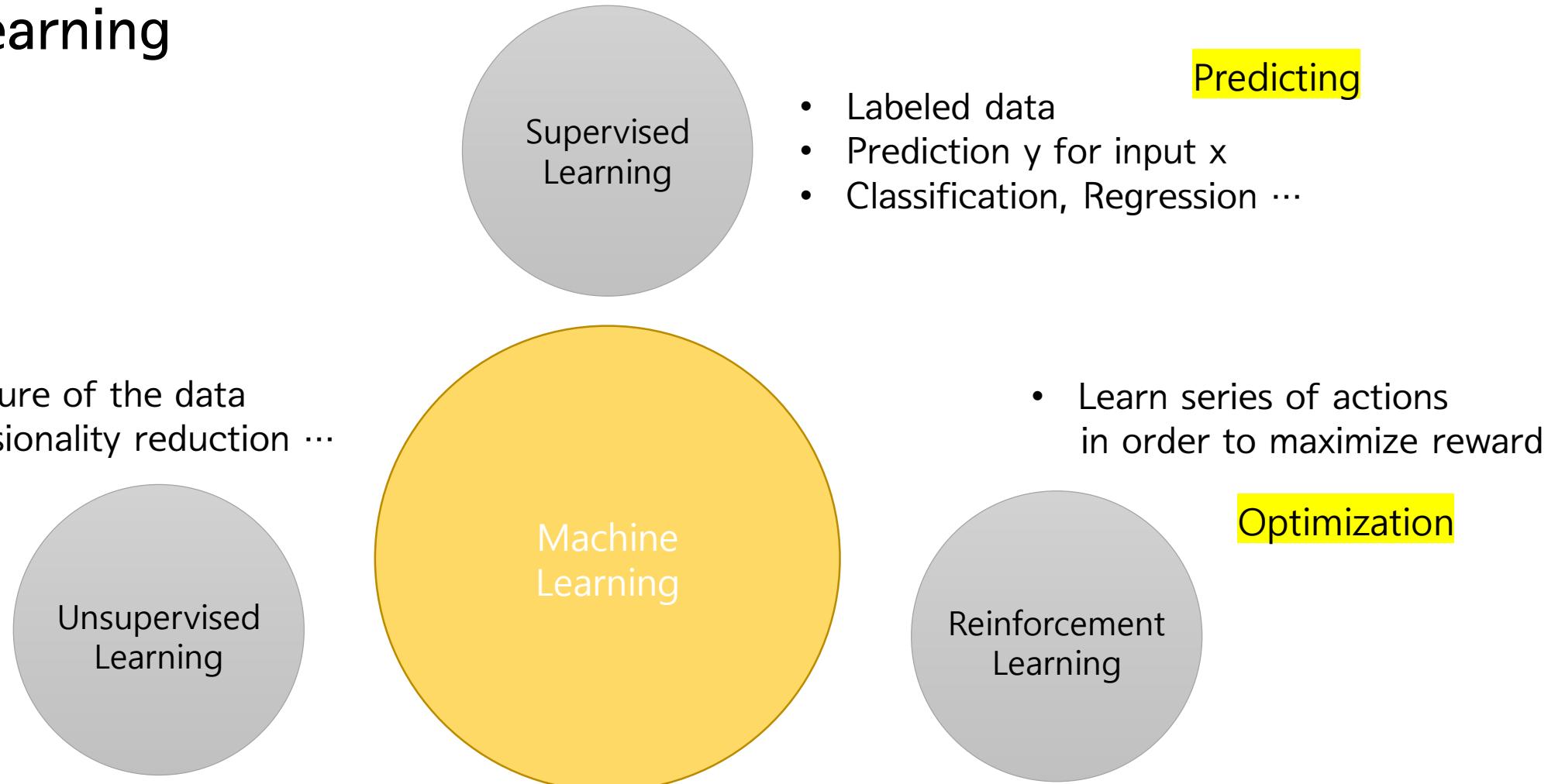
Unit 01 | Introduction of RL

- Machine Learning



Unit 01 | Introduction of RL

• Machine Learning



Unit 01 | Introduction of RL

• Reinforcement Learning

“어떤 **환경** 안에서 정의된 **에이전트**가 현재의 **상태**를 인식하여, 선택 가능한 행동들 중 **보상**을 최대화(최소화)하는 **행동** 혹은 행동 순서를 선택하는 방법”, 위키백과



Environment
환경



Agent
에이전트



State
상태



Reward
보상



Action
행동

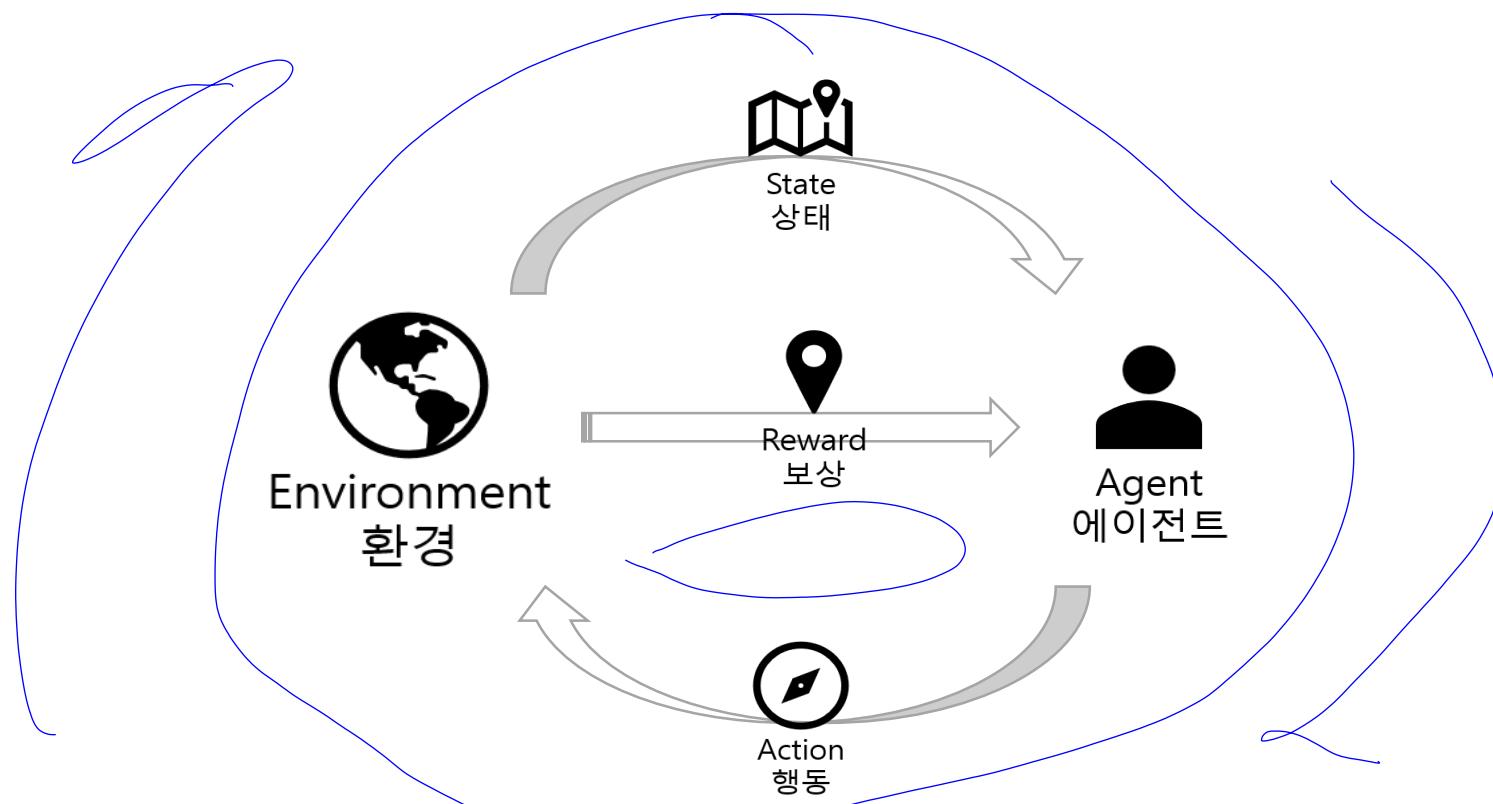
목적 : 보상의 합을 최대화하는 “최적의 행동양식, 정책”을 학습

→ 환경에 대한 지식이 없어도 학습할 수 있다

Unit 01 | Introduction of RL

• Reinforcement Learning

어떤 지도 안에서 정의된 사람이 현재의 위치를 인식하여, 선택 가능한 행동들 중 목적지까지의 거리를 최소화하는 이동 혹은 이동 순서를 선택



Unit 01 | Introduction of RL

- Problems within Reinforcement Learning

- 순차적 행동 결정 문제

행동을 한번만 선택하는 것이 아니라, 계속해서 선택



Unit 01 | Introduction of RL

• Problems within Reinforcement Learning

• MDP ; Markov Decision Process

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

→ State 상태, Action 행동, Reward 보상, Policy 정책

- 상태 : 현재 에이전트의 정보
- 행동 : 에이전트가 어떤 상태에서 취할 수 있는 행동
- 보상 : 에이전트가 한 행동을 평가할 수 있는 방법, 학습할 수 있는 유일한 정보
- 정책 : **목표!** 모든 상태에 대해 에이전트가 어떤 행동을 해야 하는지 정해 놓은 것



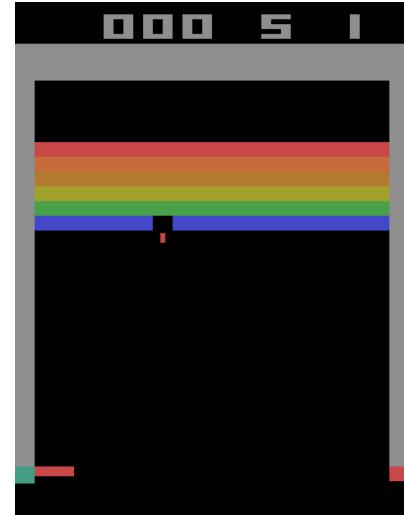
설정된 정책에 따른 행동 확률 분포

Unit 01 | Introduction of RL

- Problems within Reinforcement Learning

- MDP of Breakout

- 상태 : 게임 화면
 - 행동 : 제자리, 왼쪽, 오른쪽, 발사
 - 보상 : 벽돌 하나당 +1, 공을 놓칠 경우 -1, 아무것도 안 하면 0



Unit 01 | Introduction of RL

- Problems within Reinforcement Learning

- MDP of Breakout

- 상태 : 게임 화면
- 행동 : 게임을 왼쪽, 오른쪽으로 조작
- 보상 : 별도 하지 않다 +1 점을 놓는 경우 -1 아무것도 안 하면 0

모든 상태를 다 고려할 수 있을까?
→ 상태의 정보를 함수로 근사하는 NN 함께 적용

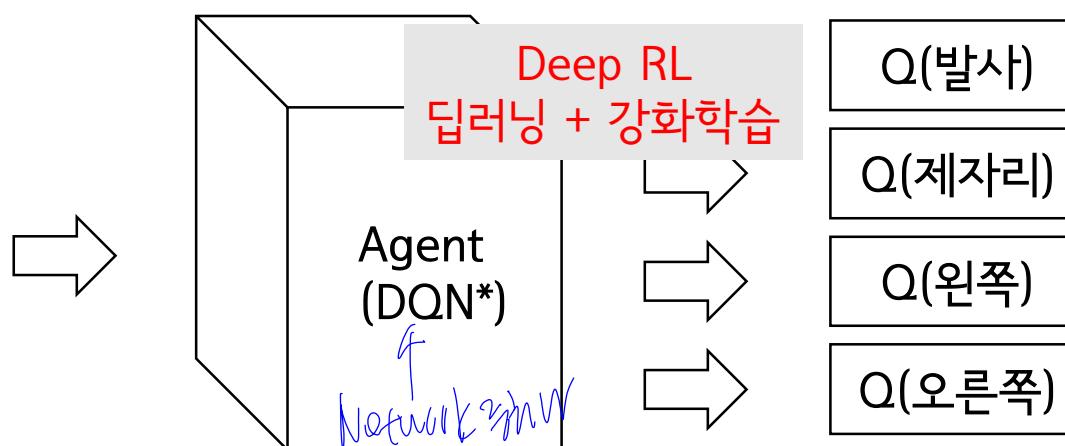
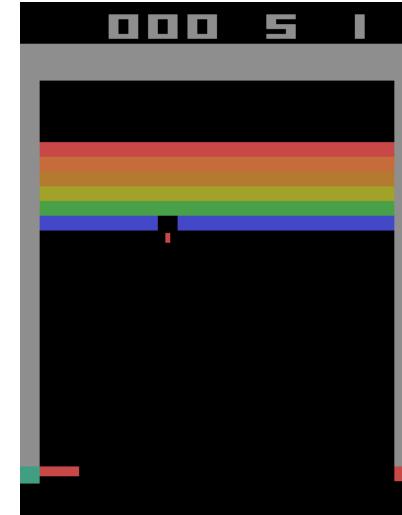


Unit 01 | Introduction of RL

- Problems within Reinforcement Learning

- MDP of Breakout

- 상태 : 게임 화면
- 행동 : 제자리, 왼쪽, 오른쪽, 발사
- 보상 : 벽돌 하나당 +1, 공을 놓칠 경우 -1, 아무것도 안 하면 0



Q-함수 : 각 행동 별 가치

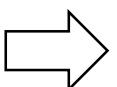
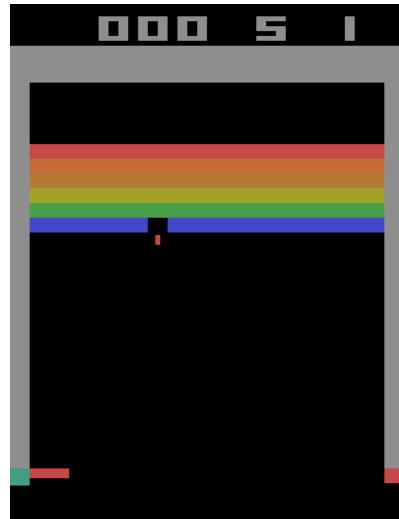
*Playing Atari with Deep Reinforcement Learning – DeepMind

Unit 01 | Introduction of RL

• Problems within Reinforcement Learning

• MDP of Breakout

- 상태 : 게임 화면
- 행동 : 제자리, 왼쪽, 오른쪽, 발사
- 보상 : 벽돌 하나당 +1, 공을 놓칠 경우 -1, 아무것도 안 하면 0



16 8x8 conv, stride 4

32 4x4 conv, stride 2

FC - 256

FC – 4(Q values)

Unit 02 | MDP & Bellman Equation

• MDP ; Markov Decision Process

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

상태 ■ \mathcal{S} is a finite set of states

행동 ■ \mathcal{A} is a finite set of actions

상태변환 확률 ■ \mathcal{P} is a state transition probability matrix,

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

보상함수 ■ \mathcal{R} is a reward function, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$

감가율 ■ γ is a discount factor $\gamma \in [0, 1]$.

A *policy* π is a distribution over actions given states,

정책

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

Unit 02 | MDP & Bellman Equation

• MDP ; Markov Decision Process

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

- S : 상태의 집합

$$\{(1,1), (1,2), (1,3), (1,4), \dots, (4,4)\}$$

- S_t : 시간이 t 일 때 상태

$$S_t = (2,2)$$

- 시간 t 에서의 상태 S_t 가 어떤 상태 s 다

$$S_t = s$$

↓ ↓
상태 행동

<Grid World>

$(1,1)^+$	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	$(4,4)^+$

Unit 02 | MDP & Bellman Equation

• MDP ; Markov Decision Process

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

- A : 행동의 집합

{up, down, left, right}

- A_t : 시간이 t 일 때 행동

$A_t = \text{right}$

- 시간 t 에서의 선택한 행동 A_t 는 a 다

$A_t = a$



<Grid World>

(1,1) ⁺ ¹	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4) ⁺ ¹

Unit 02 | MDP & Bellman Equation

• MDP ; Markov Decision Process

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

myMaze 행동하는게 어떤 시스템이야!!

- R_s^a : 시간 t에서 상태가 s이고, 행동이 a일 때의 보상함수
 $= E[R_{t+1} | S_t = s, A_t = a]$
- $P_{ss'}^a$: 상태 s에서 행동 a를 취했을 때 다음 상태가 s'일 확률
 $= P[S_{t+1} = s' | S_t = s, A_t = a]$
- model ! 환경이 알려주는 다음 상태

<Grid World>

(1, 1) ⁺¹	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	(4, 4) ⁺¹

Unit 02 | MDP & Bellman Equation

• MDP ; Markov Decision Process

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

- γ : 나중에 받는 보상의 가치를 감소시키는 정도, 감가율

$$\gamma \in [0, 1]$$

- $\gamma^{k-1} R_{t+k}$: 현재 시간 t 로부터 k 가 지난 후 받는 보상 R_{t+k} 의 현재 가치

\downarrow
 $\gamma^{k-1} R_{t+k}$

<Grid World>

(1, 1) ¹	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	(4, 4) ¹

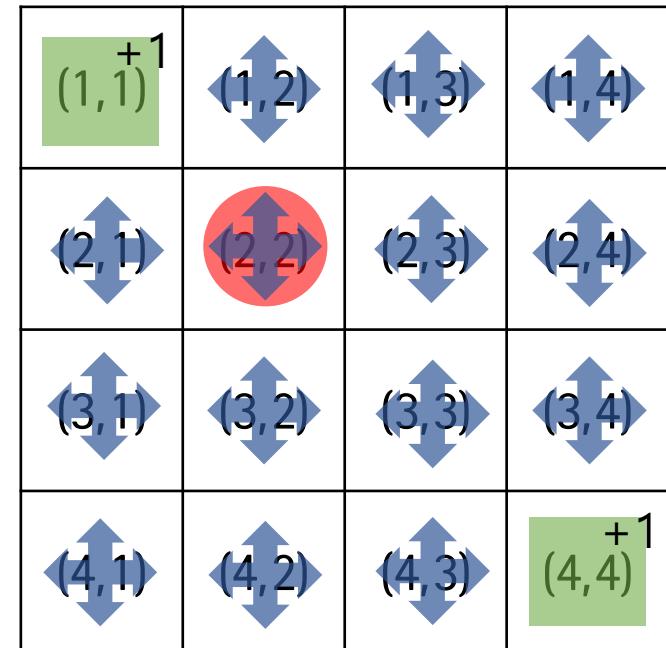
Unit 02 | MDP & Bellman Equation

• MDP ; Markov Decision Process

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

- $\pi(a|s)$: 모든 상태에서 에이전트가 할 행동, 정책
 $= P[A_t = a | S_t = s]$

<Grid World>



Unit 02 | MDP & Bellman Equation

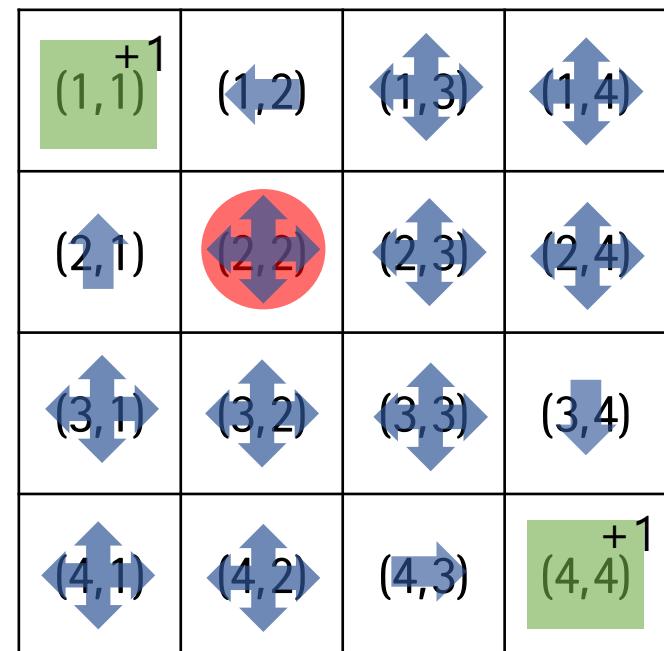
• MDP ; Markov Decision Process

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

- $\pi(a|s)$: 모든 상태에서 에이전트가 할 행동, 정책
 $= P[A_t = a | S_t = s]$



<Grid World>



Unit 02 | MDP & Bellman Equation

• MDP ; Markov Decision Process

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

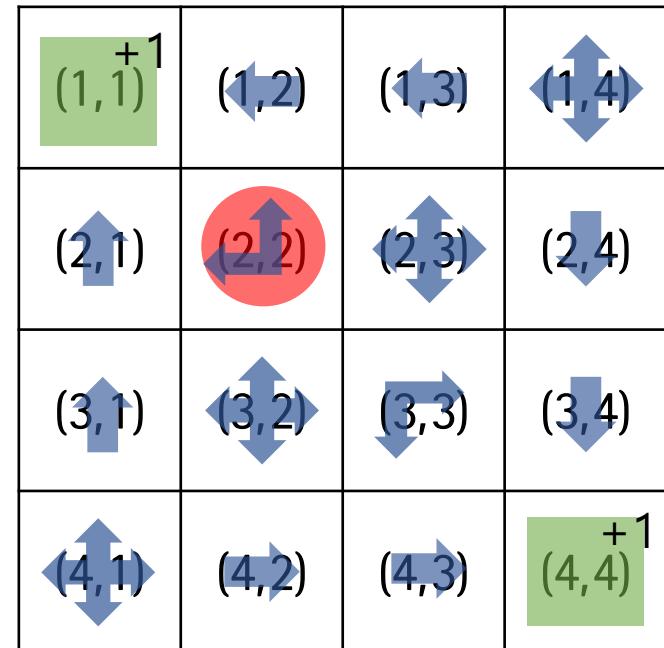
- $\pi(a|s)$: 모든 상태에서 에이전트가 할 행동, 정책
 $= P[A_t = a | S_t = s]$

→ 최적 정책이 되도록 학습해 나가는 과정



: 강화학습

<Grid World>



Unit 02 | MDP & Bellman Equation

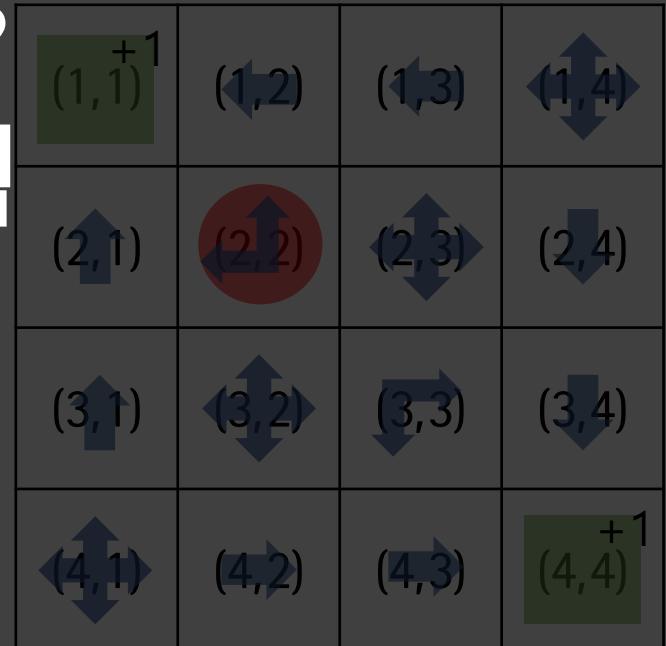
• MDP ; Markov Decision Process

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

- $\pi(a|s)$: 모든 상태에서 에이전트가 한 행동 정책
 $= P[A_t = a | S_t = s]$
- 어떤 정책이 좋은 정책?
 → 보상이 가장 큰 정책
 → 최적 정책이 되도록 학습해 나가는 과정**

: 강화학습

<Grid World>



Unit 02 | MDP & Bellman Equation

• Value Function

가감수

어떤 정책이 더 좋은 정책인지 판단하는 기준, **앞으로 받을 거라 예상되는 보상의 합**

- G_t : 앞으로 받을 보상의 현재 t시점에서 가치, 반환값 Return

$$= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots$$

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

$$\gamma = 1 - \text{할인율}$$

$$\gamma R_{t+1} = 0.1 \text{ 일 때 예상되는 } \\ \text{보상의 합은 } 1.111111111111111$$

$$G_1 = R_2 + \gamma^1 R_3 + \gamma^2 R_4 + \gamma^3 R_5$$

$$G_2 = R_3 + \gamma^1 R_4 + \gamma^2 R_5$$

$$G_3 = R_4 + \gamma^1 R_5$$

$$G_4 = R_5$$

마지막 가치는 0.1111111111111111

25 50% 할인
 100 100%

100% 할인
 100% 할인

$\Rightarrow \gamma \text{는 약 } 0.999\dots$

* $\gamma = 0.999\dots$ 일 때 R_{t+1}, R_{t+2}, \dots 을 계산할 때

$t=1$ 부터 4까지 진행할 경우

Unit 02 | MDP & Bellman Equation

• Value Function

어떤 정책이 더 좋은 정책인지 판단하는 기준



- state - value function 어떤 상태 s 에서 받을 반환값의 기댓값 → 가치가 가장 높은 상태 선택 가능

$$v(s) = E[G_t \mid S_t = s]$$

$$= E[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \quad \text{반환값으로 표현}$$

$$= E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s] \quad \text{가치함수로 표현}$$

다음 시점의 상태

$$v_\pi(s) = E[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \quad \text{정책을 고려한 가치함수로 표현}$$

※ 시기고려

우리 강당 가능한 제일 높은 상태로 선택

: 보상 받거나 끝나거나

Unit 02 | MDP & Bellman Equation

• Value Function

어떤 정책이 더 좋은 정책인지 판단하는 기준

- action - value function 어떤 상태 s 에서 행동 a 를 했을 때 받을 반환값의 기댓값

$$\begin{aligned} q(s,a) &= E[G_t \mid S_t = s, A_t = a] \\ &= E[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \quad \text{반환값으로 표현} \\ &= E[R_{t+1} + \gamma q(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \quad \text{가치함수로 표현} \end{aligned}$$

$$q_\pi(s,a) = E[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \quad \text{정책을 고려한 가치함수로 표현}$$

Unit 02 | MDP & Bellman Equation

• Value Function

어떤 정책이 더 좋은 정책인지 판단하는 기준

- state - value function 어떤 상태 s 에서 받을 반환값의 기댓값

$$v_{\pi}(s) = E[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s]$$

- action - value function 어떤 상태 s 에서 행동 a 를 했을 때 받을 반환값의 기댓값

$$q_{\pi}(s, a) = E[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

$$\begin{aligned} v_{\pi}(s) &= \sum_{a \in A} \pi(a|s) q_{\pi}(s, a) \\ q_{\pi}(s, a) &= R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s') \end{aligned}$$

Unit 02 | MDP & Bellman Equation

• Bellman Expectation Equation → 다음 상태로 전환될 확률

특정 정책을 따라갔을 때, 현재 상태의 가치함수와 다음 상태의 가치함수 사이의 관계식

S'로 쌍방향 DP

현재 가치함수 값을 업데이트하면서 계산

- state - value function 어떤 상태 s 에서 받을 반환값의 기댓값

$$\begin{aligned} v_{\pi}(s) &= E[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s] \\ &= \sum_{a \in A} \pi(a|s) (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s')) \end{aligned}$$

- action - value function 어떤 상태 s 에서 받을 반환값의 기댓값

$$q_{\pi}(s,a) = E[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

$$= R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s') q_{\pi}(s', a')$$

상태 s 를 이용해
수식 바꿔서 사용

$$\begin{aligned} v_{\pi}(s) &= \sum_{a \in A} \pi(a|s) q_{\pi}(s,a) \\ q_{\pi}(s,a) &= R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s') \end{aligned}$$

Unit 02 | MDP & Bellman Equation

- Bellman Optimality Equation

~~*최적~~ 정책을 따라갔을 때, 현재 상태의 가치함수와 다음 상태의 가치함수 사이의 관계식

- state - value function 어떤 상태 s 에서 받을 반환값의 기댓값

$$v_*(s) = \max_a q_*(s, a)$$

(가장 큰 가치의 a 를 주는 것의 확률)

$$v_*(s) = \max_a (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s'))$$

* v^* 는 π^* 로 표기
 ↪ 확률 가능

- action - value function 어떤 상태 s 에서 받을 반환값의 기댓값

$$q_*(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')$$

$$q_*(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_a q_*(s', a')$$

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s, a)$$

$$q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$

Unit 02 | MDP & Bellman Equation

- **Bellman Optimality Equation**

최적 정책을 따라갔을 때, 현재 상태의 가치함수와 다음 상태의 가치함수 사이의 관계식

- **Optimal Policy** 최적정책, 각 상태 s 에서 최적의 큐함수를 최대화하는 행동 a 을 선택

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underbrace{\operatorname{argmax}}_{\text{최적 큐함수}} q_*(s,a) \\ 0 & \text{otherwise} \end{cases}$$

* argmax :

Unit 02 | MDP & Bellman Equation

- **Bellman Optimality Equation**

최적 정책을 따라갔을 때, 현재 상태의 가치함수와 다음 상태의 가치함수 사이의 관계식

- **Optimal Policy** 최적정책, 각 상태 s 에서 최적의 큐함수를 최대화하는 행동 a 을 선택

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_a q_*(s, a) \\ 0 & \text{otherwise} \end{cases}$$

Solving Bellman Equation

Unit 03 | Dynamic Programming

방정식 풀는 법

• Dynamic Programming

큰 문제의 해답에 작은 문제들의 해답이 포함되어 있어 연산시에 중복이 발생하는 경우,
큰 문제를 작은 문제로 나누어 해결함으로써 중복되는 연산을 해결하는 문제 해결 전략

• Prediction

MDP 정보를 통해 주어진 정책에 대한 가치함수 예측

방법 → Policy Evaluation

• Control

MDP를 통해 가치함수를 최적화하고 정책을 발전시켜 나가서 최적 정책 제어

→ Policy Iteration, Value Iteration

Unit 03 | Dynamic Programming

• Policy Evaluation

k번째 step에서의 정책에 대한 가치 함수를 구하는 과정

한 step에서 모든 상태에 대해 동시에 한 번씩 벨만 방정식을 계산하고 업데이트

$\leftarrow \text{update} \rightarrow \text{predict}$

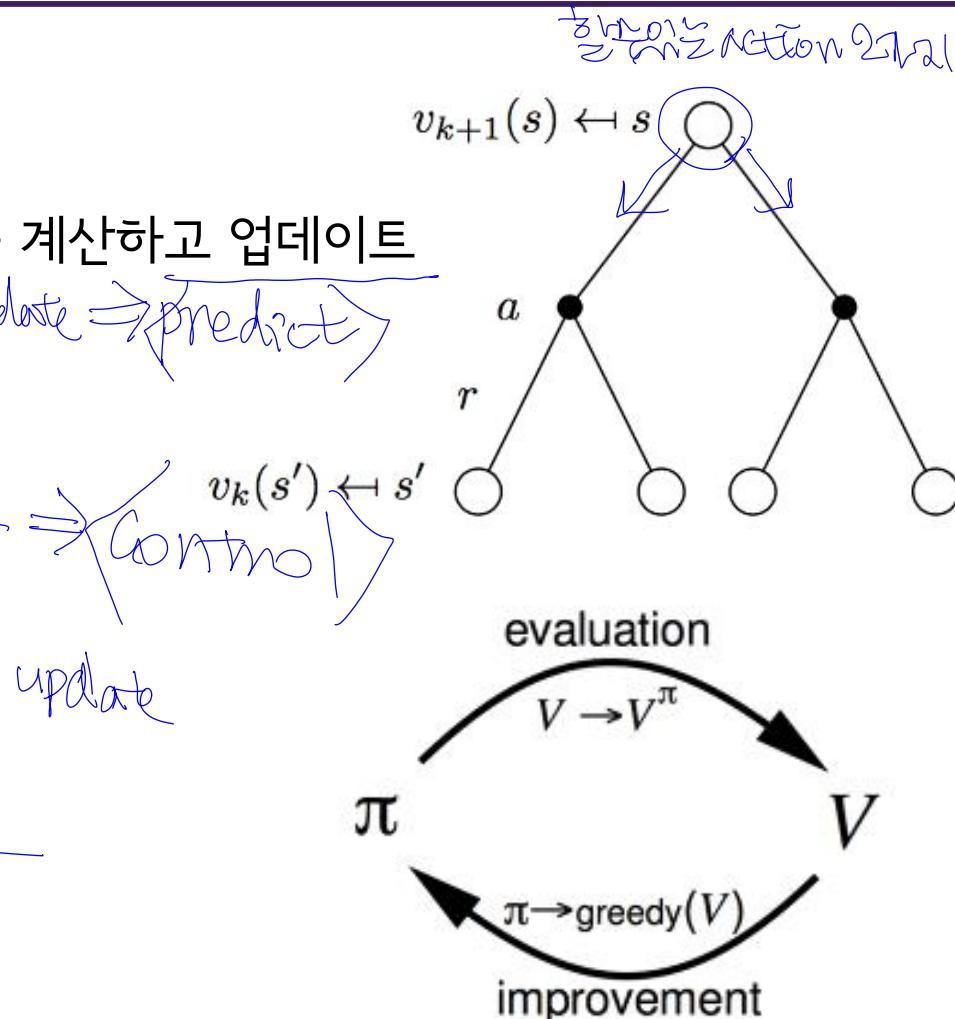
무작위 행동, 어느 순간 수렴'을 가리

• Policy Iteration

벨만 기대 방정식 사용해서 evaluation과 improvement 반복

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_k(s') \right)$$

기대 방정식을 통해 가치를 update



• Value Iteration

벨만 최적 방정식 사용해서 evaluation과 improvement 반복

$$v_{k+1}(s) = \max_{a \in \mathcal{A}} \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_k(s') \right)$$

Unit 03 | Dynamic Programming

• Dynamic Programming

큰 문제의 해답에 작은 문제들의 해답이 포함되어 있어 연산시에 중복이 발생하는 경우,
큰 문제를 작은 문제로 나누어 해결함으로써 중복되는 연산을 해결하는 문제 해결 전략

$$\bullet v_{k+1}(s) = E[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s]$$

If k → ∞, v 는 v_* 로 수렴하고 π 는 π_* 로 수렴

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s'))$$

→ 두 번 계산하지 않기! 강화학습은 많

Unit 03 | Dynamic Programming

• MDP ; Markov Decision Process - Review

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

- R_s^a : 시간 t에서 상태가 s이고, 행동이 a일 때의 보상함수
 $= E[R_{t+1} | S_t = s, A_t = a]$

- $P_{ss'}^a$: 상태 s에서 행동 a를 취했을 때 다음 상태가 s'일 확률
 $= P[S_{t+1} = s' | S_t = s, A_t = a]$

→ model! 환경이 알려주는 다음 상태

문제 2.
 → 현실에서 적용하기

<Grid World>

$(1, 1)^{+1}$	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	$(4, 4)^{+1}$

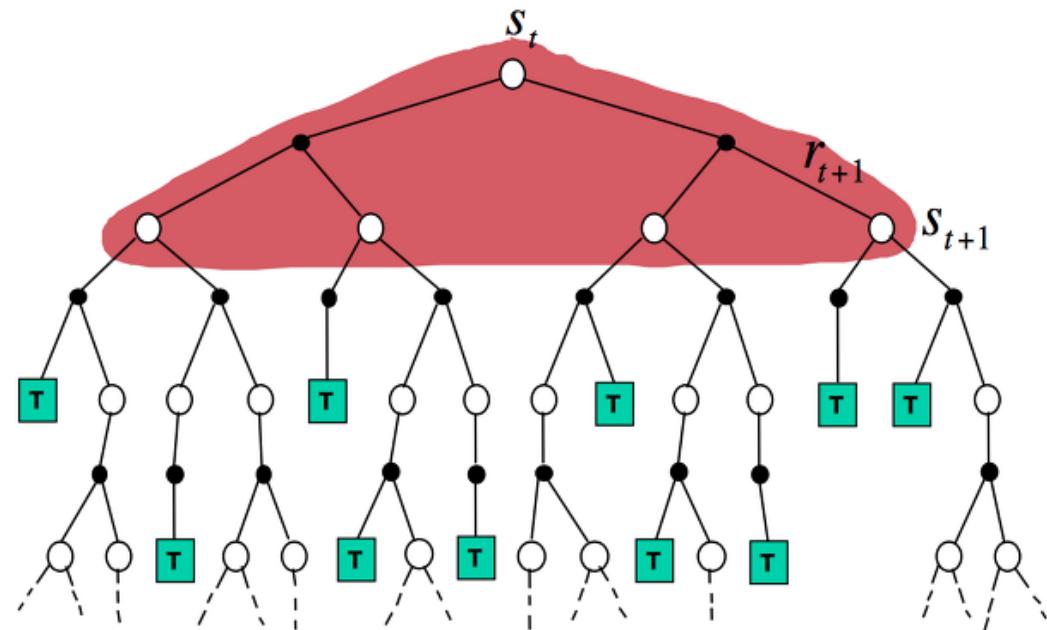
Unit 03 | Dynamic Programming

• Dynamic Programming

큰 문제의 해답에 **작은 문제들의 해답**이 포함되어 있어 연산시에 중복이 발생하는 경우,
큰 문제를 작은 문제로 나누어 해결함으로써 중복되는 연산을 해결하는 문제 해결 전략

$$V(S_t) \leftarrow \mathbb{E}_\pi [R_{t+1} + \gamma V(S_{t+1})]$$

<R.L>
모델학습
Input → Output



Unit 03 | Dynamic Programming

- MDP ; Markov Decision Process - Review

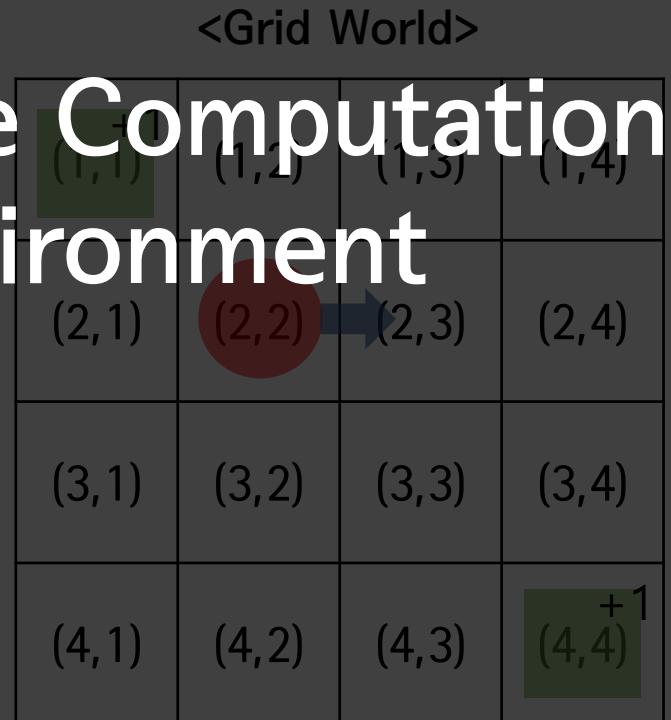

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

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

P1. Full-width backup \rightarrow Expensive Computation

- $P_{ss'}^a$: 상태 s 에서 행동 a 를 취했을 때 다음 상태가 s' 일 확률
 $= P[S_{t+1} = s' | S_t = s, A_t = a]$

\rightarrow model ! 환경이 알려주는 다음 상태



Unit 03 | Dynamic Programming

• MDP ; Markov Decision Process - Review

순차적 행동 결정 문제를 수학적으로 정의했을 때 구성 요소들

- R_s^a : 시간 t에서 상태가 s이고, 행동이 a일 때의 보상함수
 $= E[R_{t+1} | S_t = s, A_t = a]$

Model Free 학습 → 강화학습

모델 없이 환경과의 상호작용을 통해 입력과 출력 사이의 관계를 학습

- $P_{ss'}^a$: 상태 s에서 행동 a를 취했을 때 다음 상태가 s'일 확률
 $= P[S_{t+1} = s' | S_t = s, A_t = a]$

→ model ! 환경이 알려주는 다음 상태

<Grid World>

	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4) +1

Unit 03 | Dynamic Programming

- **Dynamic Programming - Review**

큰 문제의 해답에 작은 문제들의 해답이 포함되어 있어 연산시에 중복이 발생하는 경우,
큰 문제를 작은 문제로 나누어 해결함으로써 중복되는 연산을 해결하는 문제 해결 전략

- **Prediction**

MDP 정보를 통해 주어진 정책에 대한 가치함수 예측

→ Policy Evaluation

- **Control**

MDP를 통해 가치함수를 최적화하고 정책을 발전시켜 나가서 최적 정책 제어

→ Policy Iteration, Value Iteration

Unit 04 | Model Free

DP와 RL의 차이!

• Reinforcement Learning

• Prediction

에이전트와 환경과의 상호작용을 통해 주어진 정책에 대한 가치함수 예측

→ Monte-Carlo Prediction, TD Prediction

• Control

가치함수를 토대로 정책을 발전시켜 나가서 최적 정책을 제어

→ SARSA, Q-Learning

Unit 04 | Model Free

• Monte-Carlo Prediction

에이전트가 한 번 환경에서 에피소드(sample)를 진행하고, 에피소드의 평균으로 가치함수 값 추정

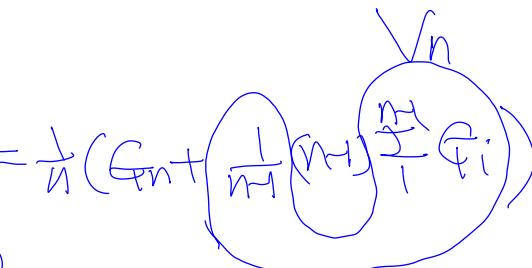
*몬테카를로 근사 : sample을 통해서 원래의 값과 비슷하게 추정

- $v_{\pi}(s) \sim \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i(s)$ $N(s)$: 상태 s 를 방문한 에피소드의 수, G_i : i 번째 에피소드에서 s 의 반환값
- $v_{k+1}(s) = v_k(s) + \alpha(G_k(s) - v_k(s))$

* α 는 모든 s 에 대해서 같은데

MCP는 특정 episode로! (모든 걸 계산하는 X)

$$\begin{aligned} V_{n+1} &= \frac{1}{n} \sum_i^n G_i \\ &= \frac{1}{n} (G_n + \sum_i^m G_i) = \frac{1}{n} (G_n + \left(\frac{1}{m} \sum_i^m G_i \right) \sum_i^m G_i) \\ &= \frac{1}{n} (G_n + nV_n - V_n) \\ &\Rightarrow \pi + \frac{1}{n} (G_n - V_n) \end{aligned}$$



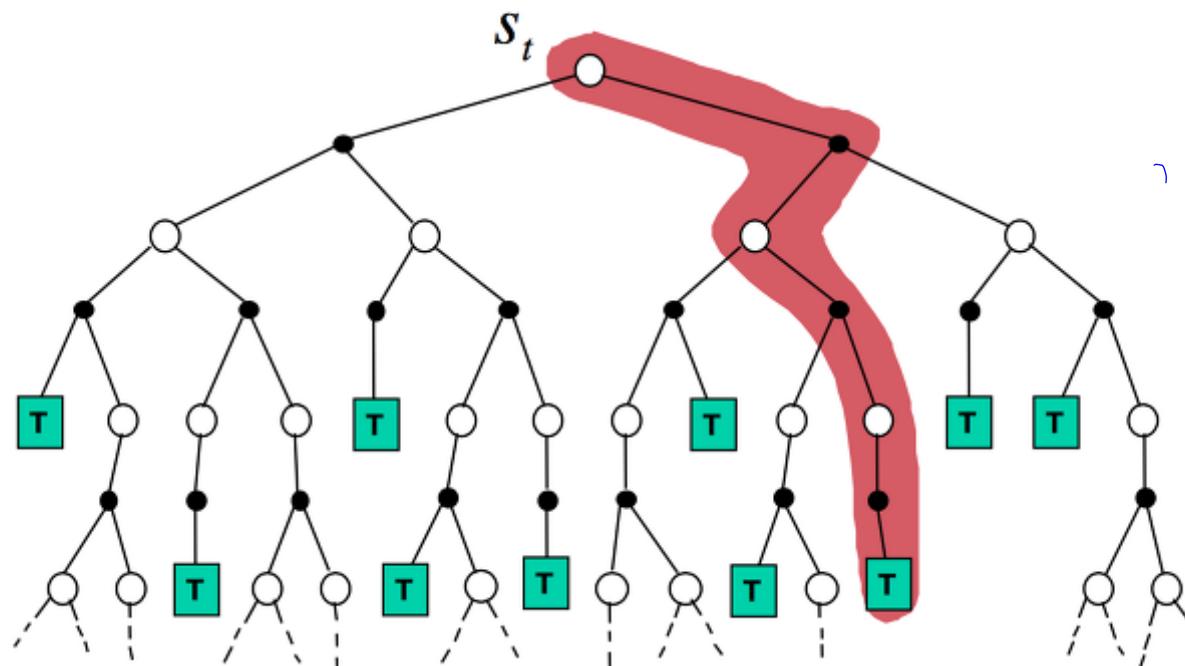
→ 한 에피소드가 끝나면 에피소드 동안 방문했던 모든 상태의 가치함수를 업데이트하고, 다시 시작 상태부터 새로운 에피소드를 진행

Unit 04 | Model Free

• Monte-Carlo Prediction

에이전트가 한 번 환경에서 에피소드(sample)를 진행하고, 에피소드의 평균으로 가치함수 값 추정

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$



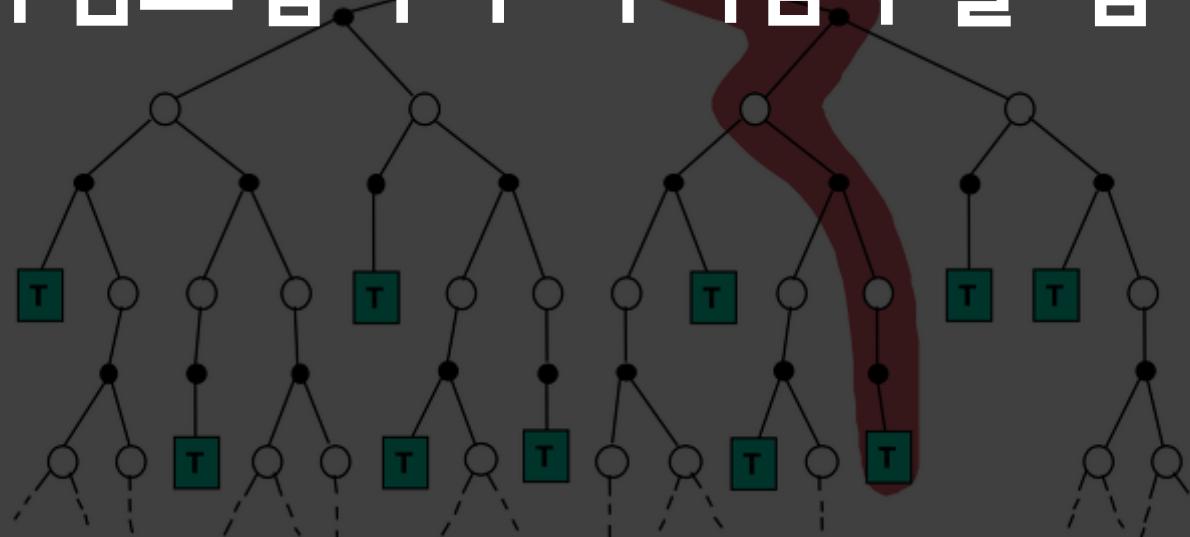
Unit 04 | Model Free

• Monte-Carlo Prediction

에이전트가 한 번 환경에서 에피소드(sample)를 진행하고, 에피소드의 평균으로 가치함수 값 추정

단정

한 에피소드가 끝나야지 업데이트 가능
→ 타임스텝마다 가치함수를 업데이트



Unit 04 | Model Free

(설명 대신 사용)

• TD Prediction ; Temporal-Difference

에이전트는 현재 상태 s 에서 행동을 선택하고 받는 보상 R 과 다음 상태 s' 을 이용해 타임스텝마다 바로 가치함수 업데이트 : MC + DP

$\xrightarrow{\text{Sampling}}$ \rightarrow 에피소드 종료까지 대기 필요X

$$\cdot v_{\pi}(s) = E[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s]$$

$$\cdot v_{k+1}(s) = v_k(s) + \alpha(R + \gamma v_k(s') - v_k(s))$$

$\xrightarrow{\text{Each Step로 바로 update}}$

→ 에피소드가 끝날 때까지 기다릴 필요가 없이 바로 가치함수 업데이트 가능

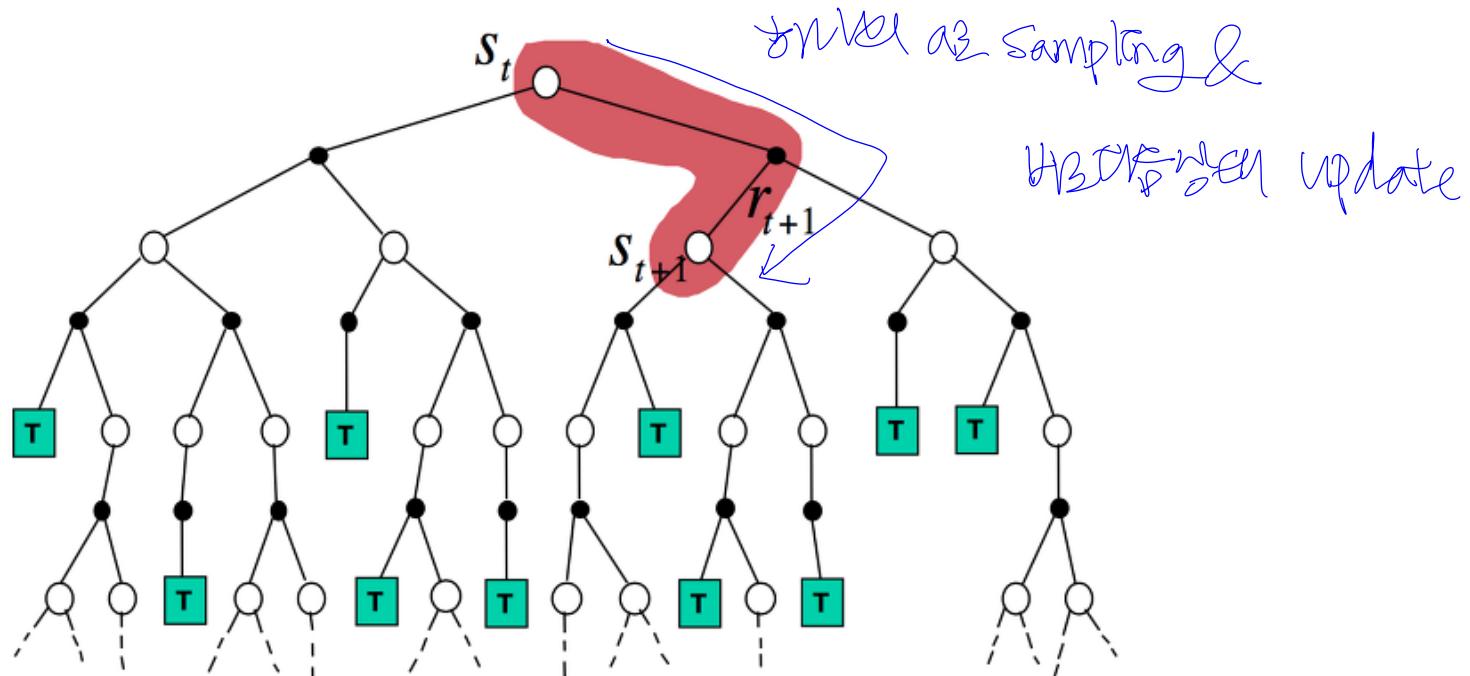
→ 대부분의 경우 MC 보다 ~~자~~ 효율적으로 빠른 시간 안에 수렴

Unit 04 | Model Free

• TD Prediction ; Temporal-Difference

에이전트는 현재 상태 s 에서 행동을 선택하고 받는 보상 R 과 다음 상태 s' 을 이용해 타임스텝마다 바로 가치함수 업데이트

$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



Unit 04 | Model Free

• SARSA : TD Control

SARSA a역시

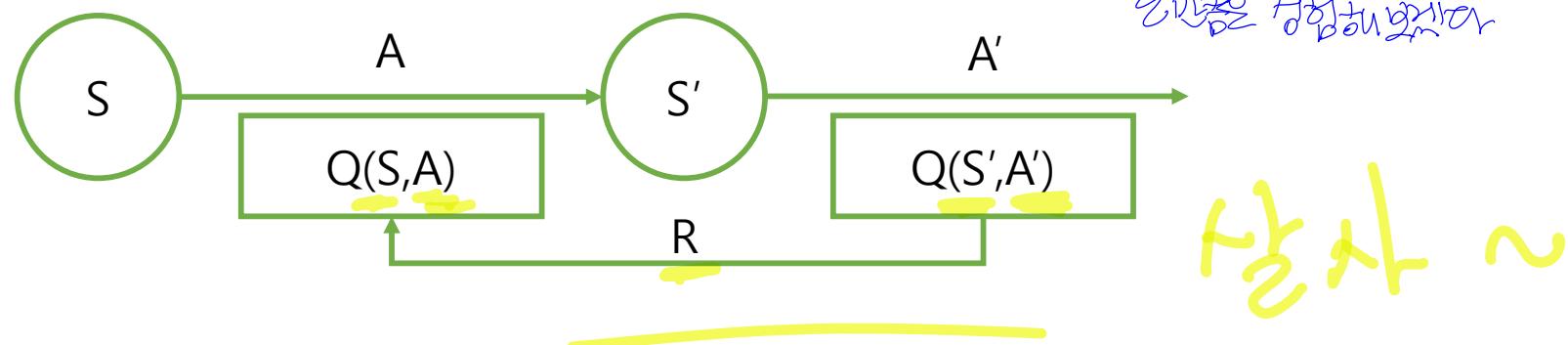
TD prediction의 결과로 계산된 가치함수를 ϵ -greedy 정책에 따라 정책을 업데이트

$$\cdot q_{k+1}(s, a) = q_k(s, a) + \alpha(R + \gamma q_k(s', a') - q_k(s, a))$$

$$\cdot \pi(s) = \begin{cases} 1 & \text{if } a = \operatorname{argmax} q(s, a) \\ 0 & \text{otherwise} \end{cases} \rightarrow \pi(s) = \begin{cases} 1 - \epsilon & \text{if } a = \operatorname{argmax} q(s, a) \\ \epsilon & \text{otherwise} \end{cases}$$

탐험

*보상이 알리움



할사 ~

*보상을 고지해 놓았음.

온전히 경험해보게 되어

Unit 04 | Model Free

• SARSA : TD Control

TD prediction의 결과로 계산된 가치함수를 ε -greedy 정책에 따라 정책을 업데이트

$$\bullet q_{k+1}(s, a) = q_k(s, a) + \alpha(R + \gamma q_k(s', a') - q_k(s, a))$$

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

Repeat (for each episode):

 Initialize S

 Choose A from S using policy derived from Q (e.g., ε -greedy) 예제

 Repeat (for each step of episode):

 Take action A , observe R, S'

 Choose A' from S' using policy derived from Q (e.g., ε -greedy) 예제

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$$

$$S \leftarrow S'; A \leftarrow A';$$

 until S is terminal

for update

Unit 04 | Model Free

• Q - Learning : Off-Policy TD Control

ϵ -greedy 정책의 문제점 : 탐험을 위해 선택한 행동으로 최적 정책을 학습 할 수 없음

→ 현재 행동하는 정책과 독립적으로 학습하는 방법, 행동하는 정책과 학습하는 정책을 따로 분리

ϵ -greedy

Bellman optimality equation

$$q_{k+1}(s, a) = q_k(s, a) + \alpha(R + \gamma \max_{a'} q_k(s', a') - q_k(s, a))$$

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

Repeat (for each episode):

 Initialize S

 Repeat (for each step of episode):

 Choose A from S using policy derived from Q (e.g., ϵ -greedy)

 Take action A , observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$$

$S \leftarrow S'$;

 until S is terminal

방문한 State는 0이

수집한 R은 update



TD

RL

Unit 04 | Model Free

• Q - Learning : Off-Policy TD Control

ε -greedy 정책의 문제점 : 탐험을 위해 선택한 행동으로 최적 정책을 학습 할 수 없음

→ 현재 행동하는 정책과 독립적으로 학습하는 방법, 행동하는 정책과 학습하는 정책을 따로 분리

~~문제점!~~

ε -greedy

Bellman optimality equation

$$q_{k+1}(s, a) \leftarrow q_k(s, a) + \alpha(R + \gamma \max_{a'} q_k(s', a') - q_k(s, a))$$

모든 상태를 테이블 형식으로 저장 불가능

Initialize $Q(s, a)$, $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

Repeat (for each episode):

Initialize S

Repeat (for each step of episode):

Choose A from S using policy derived from Q (e.g., ε -greedy)

Take action A , observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$$

$S \leftarrow S'$;

until S is terminal

ex) 알파고는
한국어로 말하
는가요?

Unit 05 | Deep RL



Deep Reinforcement Learning

RL + NN, 가치함수를 w 파라미터를 사용해 근사시키고, 업데이트를 반복

- $\hat{q}(s, a, w) \approx q(s, a)$: function approximation, generalization

- Loss function : MSE 

$$L(w) = \{R_{t+1} + \gamma \max_{a'} \hat{q}(s', a', w) - \underbrace{\hat{q}(s, a, w)}_{\text{목표값}}\}^2$$

- Optimization : SGD

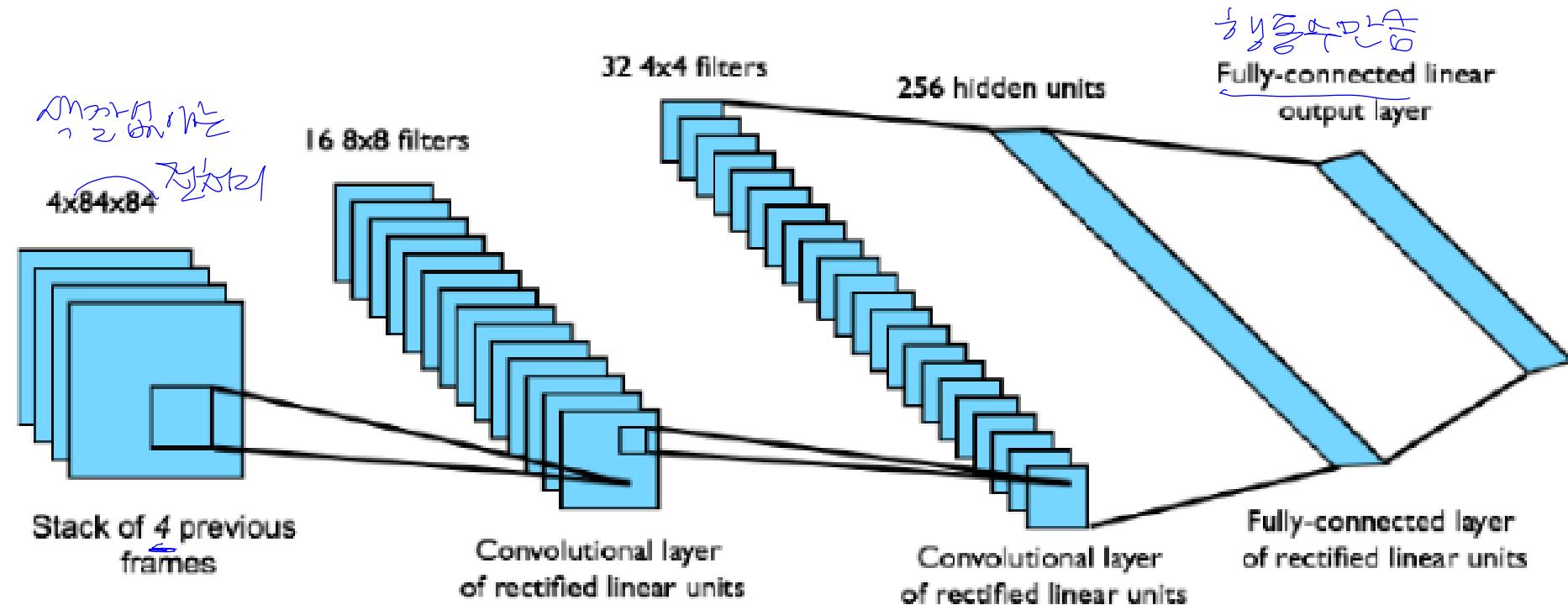
$$\nabla_w L(w) \quad \underline{\text{backpropagation}}$$



Unit 05 | Deep RL

- DQN ; Deep Q-Network in Atari

Deep RL + CNN



Unit 05 | Deep RL

- Deep RL

- Value-based RL *(지향형 RL)*
- Policy-based RL *SARSA 학습 알고리즘*
각 행동할 확률인 정책을 근사
: policy gradient ; REINFORCE
- Actor - Critic
Value-based RL \cap Policy-based RL
결합 *OX 알고리즘*

Unit 00 | Assignment

각자 궁금한 부분 찾아보기

Q & A

들어주셔서 감사합니다.