Value-based Learning

2021. 07. 16

발표자: 허종국

발표자 소개

❖ 이름 : 허종국 (Jong Kook Heo)

- Data Mining & Quality Analytics Lab
- M.S. Student (2021.03~)
- 지도 교수 : 김성범 교수님

❖ 관심 연구 분야

- Deep Reinforcement Learning

❖ 연락망

- E-mail : hjkso1406@korea.ac.kr

목 차

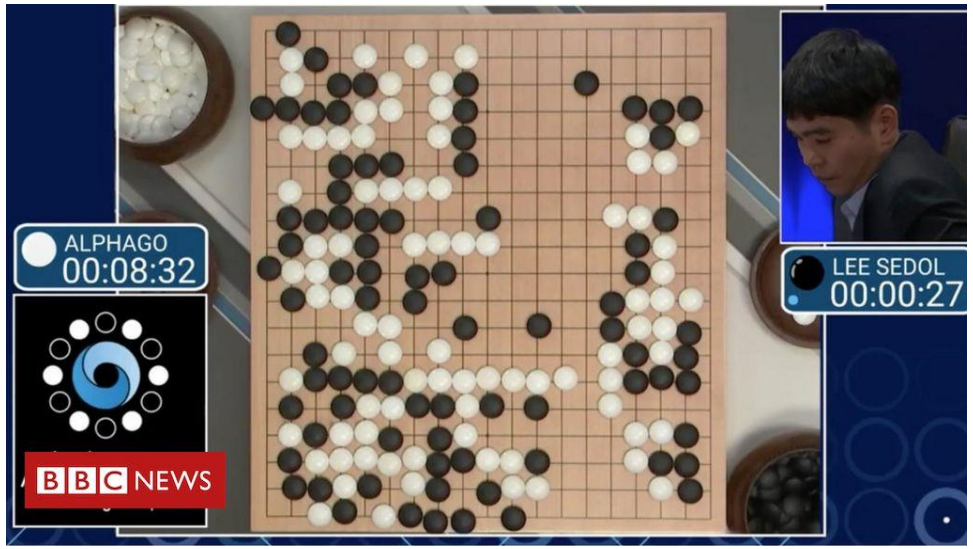
1. Introduction
   - Definition
   - Markov Decision Process
   - Bellman Equation
2. Learning Methods
   - Value-based vs Policy-based
   - MC vs TD
   - On-policy vs Off-policy
3. Value-based Algorithms
   - DQN
   - DRQN
4. Conclusion
5. Appendix
   - Reference
   - Additional Materials

Introduction

Definition

❖ 강화학습의 목적

- Maximize Total Game Score



알파고

자율주행자동차

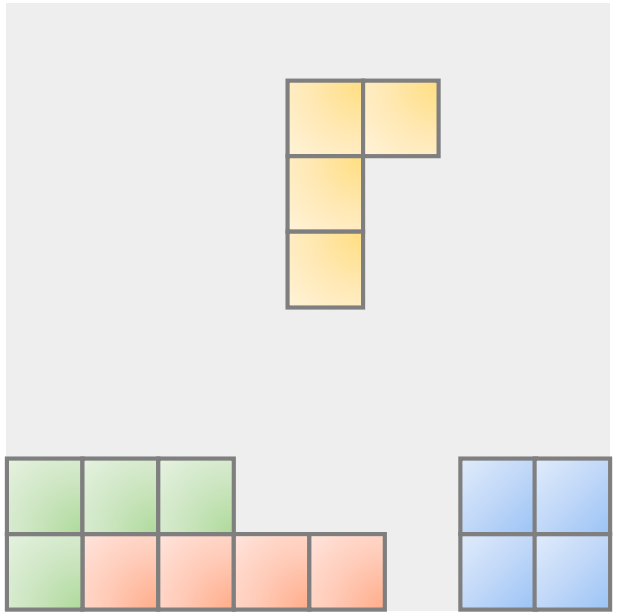
Introduction

Definition

❖ 강화학습의 목적

- Maximize Total Game Score

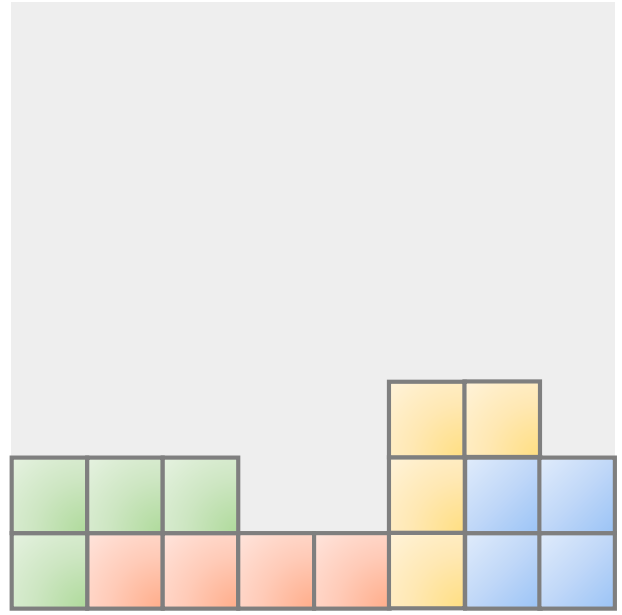
SCORE : 000



Good!

+1

SCORE : 001



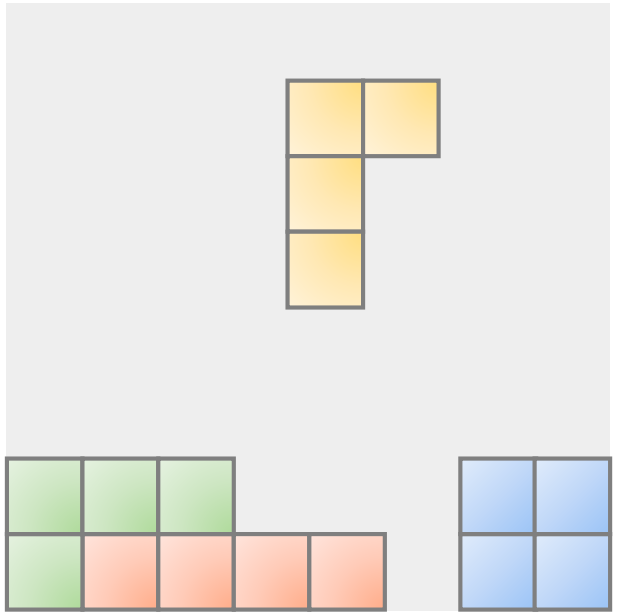
Introduction

Definition

❖ 강화학습의 목적

- Maximize Total Game Score

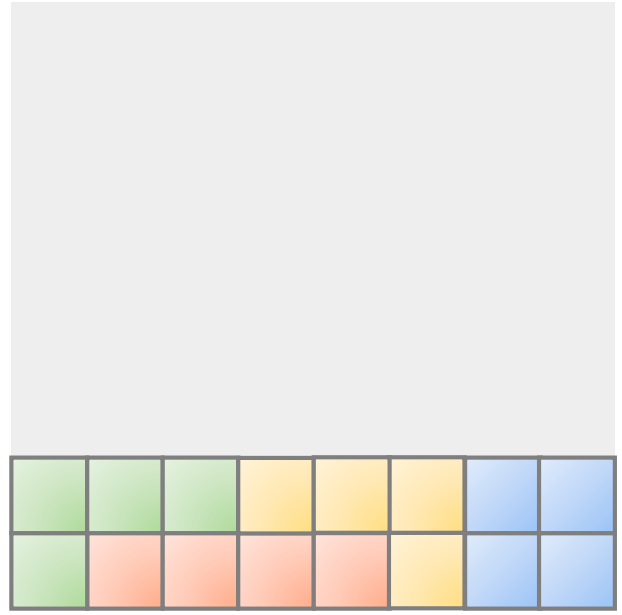
SCORE : 000



Better!

+2

SCORE : 002



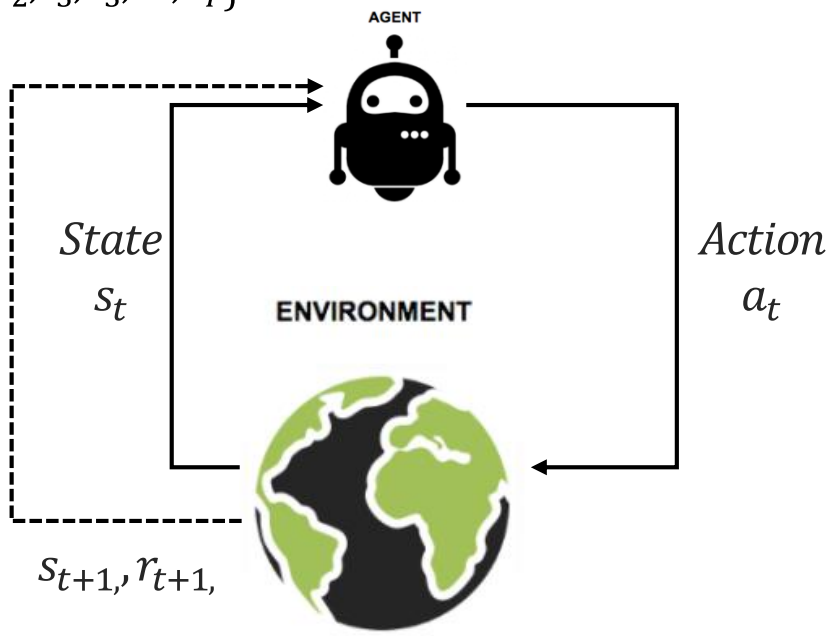
Introduction

Markov Decision Process(MDP)

❖ Markov Decision Process(MDP)

- 5 elements in MDP : <S, A, P, R, γ>
- P : Transition Probability, which satisfies Markov Property : $P[S_{t+1}|S_t] = P[S_{t+1}|S_1, S_2, \dots, S_t]$
- Experience : (s_0, a_0, r_1, s_1),..
- Episode: $\{s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \dots, s_T\}$

Transition Probability
$P(s', r|s, a)$



Policy
$\pi_\theta(a|s)$

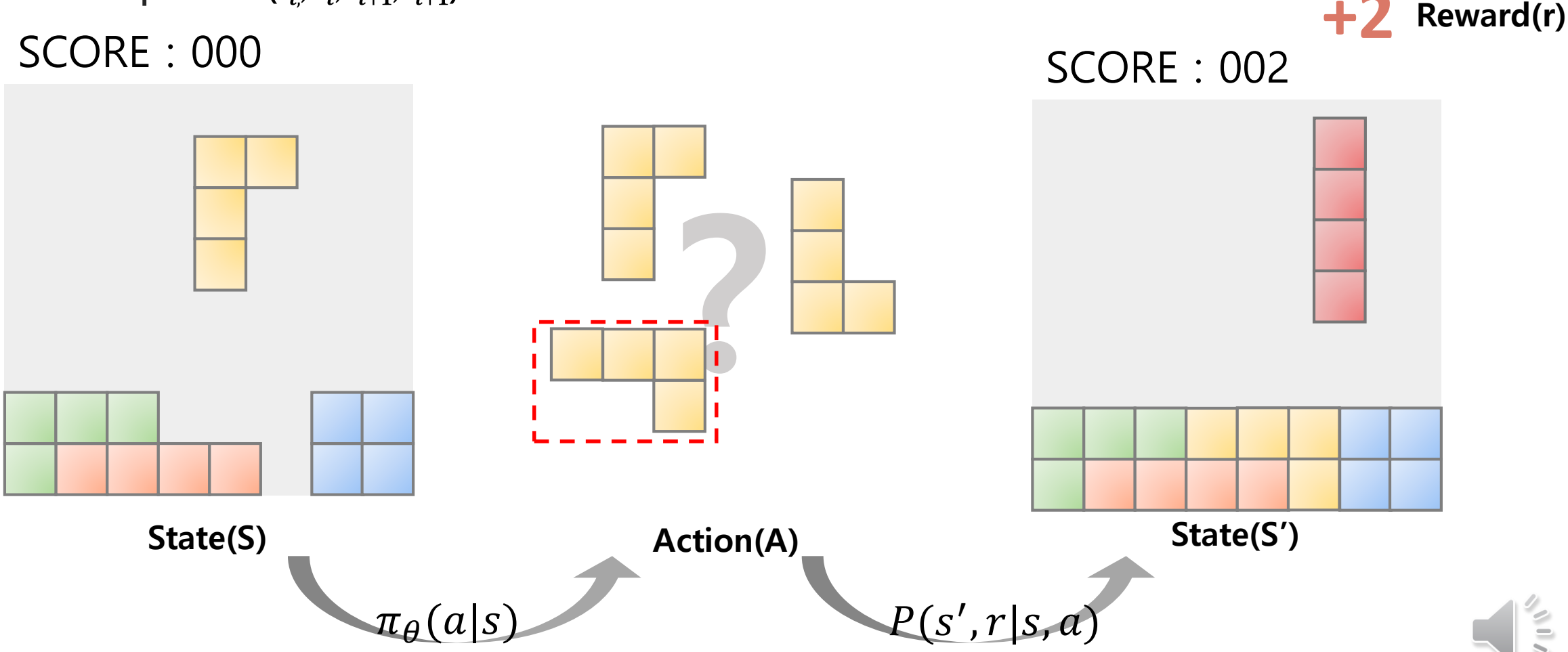
https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

Introduction

Markov Decision Process(MDP)

❖ Markov Decision Process(MDP)

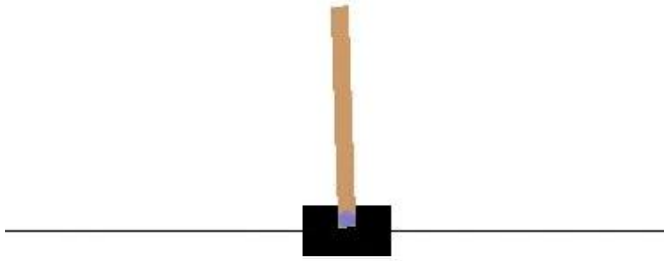
- **Experience : $(s_t, a_t, r_{t+1}, s_{t+1})$**



Introduction

Markov Decision Process(MDP)

❖ Markov Decision Process(MDP)



	CartPole	Breakout
State	카트의 위치, 속력, 막대기 각도, 각속도	게임 화면
Action	좌/우 이동	좌/우 이동
Reward	초마다 +1	깬 블록의 개수

Introduction

Bellman Equation

❖ 가치 함수

- 보상(r)은 단기적인 이득만 나타냄
- 각 상태 및 행동이 얼마나 가치 있는지를 판단하기 위해 누적 보상 값을 예측 할 수 있어야함
- G_t : 현재 시점 t 로부터 받을 수 있는 누적 보상을 수학적으로 정의
  - ✓ $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

단기 보상 R_t

앞으로 얻을 수 있는 누적 보상 G_t

Introduction

Bellman Equation

❖ Bellman Equation(벨만 방정식)

- G_t : 현재 시점 t 로부터 받을 수 있는 누적 보상
- $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- Estimates of G_t
  - ✓ State-Value function (V)
  - ✓ Action-Value function (Q)

얼마나 점수를 더 받을 수 있을까?

State-Value function(상태 가치 함수)

$$V_\pi(s) = E_\pi[G_t|S_t = s]$$
$$= E_\pi[R_{t+1} + \gamma G_t(S_{t+1})|S_t = s]$$
$$= E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s]$$

현재 상태에서
얼마나 점수를 더 받을 수 있을까?

Action-Value function(행동 가치 함수)

$$Q_\pi(s,a) = E[G_t|S_t = s, A_t = a]$$
$$= E[R_{t+1} + \gamma G_t(S_{t+1})|S_t = s, A_t = a]$$
$$= E[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s, A_t = a]$$
$$= E[R_{t+1} + \gamma Q_\pi(S_{t+1}, a')]|S_t = s, A_t = a]$$

현재 상태에서 이 행동을 취하면
얼마나 점수를 더 받을 수 있을까?

Introduction

Bellman Equation

❖ Bellman Equation(벨만 방정식)

- G_t는 앞으로 더 받을 수 있는 보상
  - ✓ $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- V 와 Q는 G_t 에 대한 추정치
  - ✓ $V_\pi(s) = E_\pi[G_t | S_t = s]$
  - ✓ $Q_\pi(s, \mathrm{a}) = E_\pi[G_t | S_t = s, A_t = a]$

$$V_\pi(s_t) = \sum_{a_t \in A} \pi(a_t|s_t) Q_\pi(s_t, a_t)$$

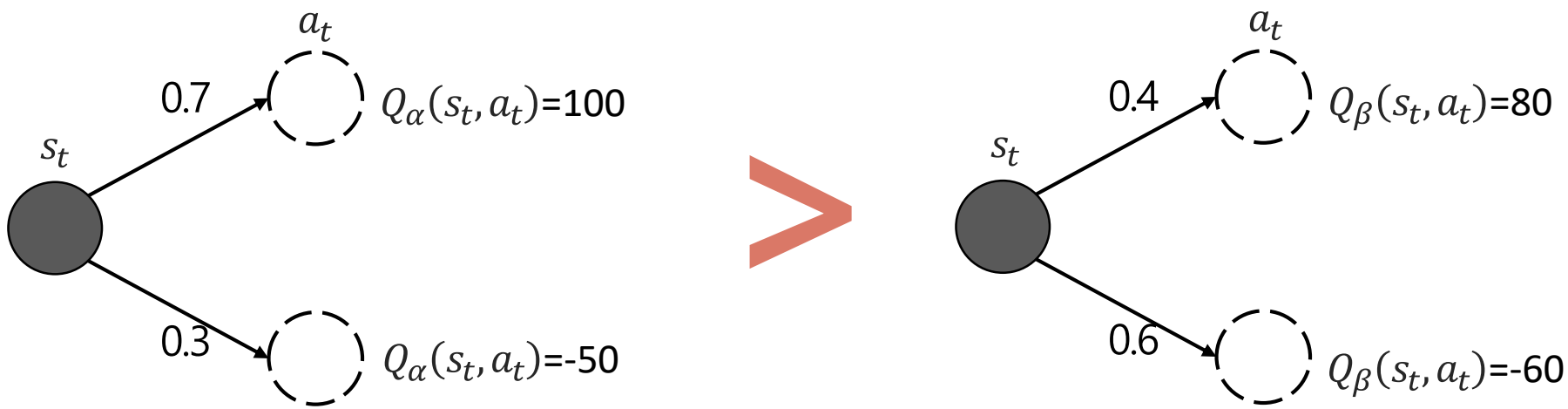
$$E[f(x)] = \sum_{x \in A} p(x) f(x)$$

Introduction

Bellman Equation

❖ Bellman Equation(벨만 방정식)

- G_t는 앞으로 더 받을 수 있는 보상
  - ✓ $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- V 와 Q는 G_t 에 대한 추정치
  - ✓ $V_\pi(s) = E_\pi[G_t | S_t = s]$
  - ✓ $Q_\pi(s, \mathrm{a}) = E_\pi[G_t | S_t = s, A_t = a]$



$V_\alpha(s_t) = 550$

$V_\beta(s_t) = -40$

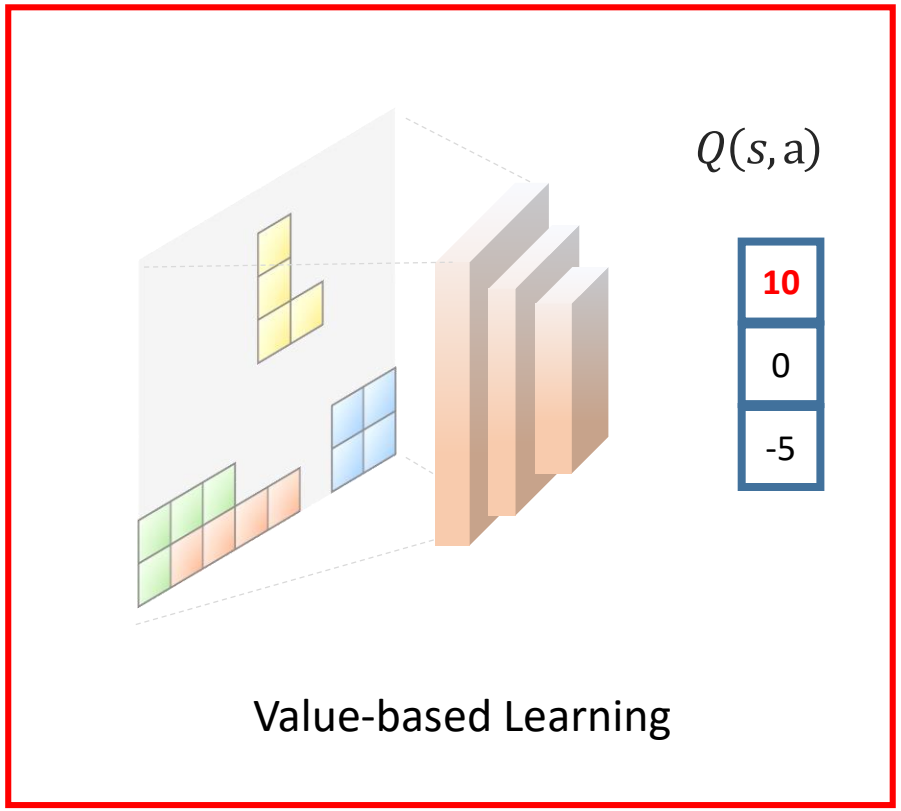
Learning Methods

Value-based vs Policy-based

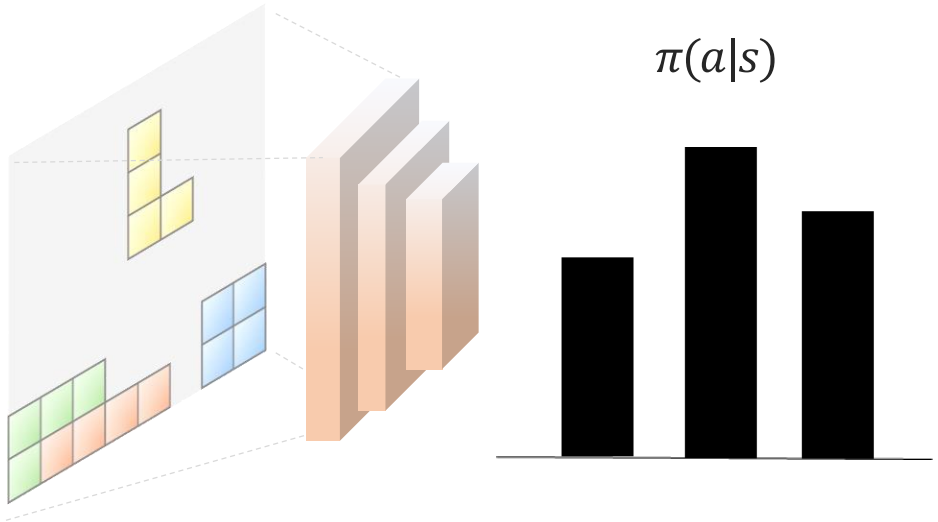
- ✓ $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- ✓ $V_\pi(s) = E_\pi[G_t | S_t = s] = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s]$
- ✓ $V_\pi(s_t) = \sum_{a_t \in A} \pi(a_t|s_t) Q_\pi(s_t, a_t)$
- ✓ $Q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a] = E_\pi[R_{t+1} + \gamma E_\pi[Q(S_{t+1}, a')] | S_t = s, A_t = a]$

❖ 아웃풋에 따른 분류

- Value-based Learning : 상태/행동에 대한 가치함수 추정치
- Policy-based Learning : 행동에 대한 확률 분포



Value-based Learning



Policy-based Learning

Learning Methods

MC vs TD

❖ 업데이트 주기에 따른 분류

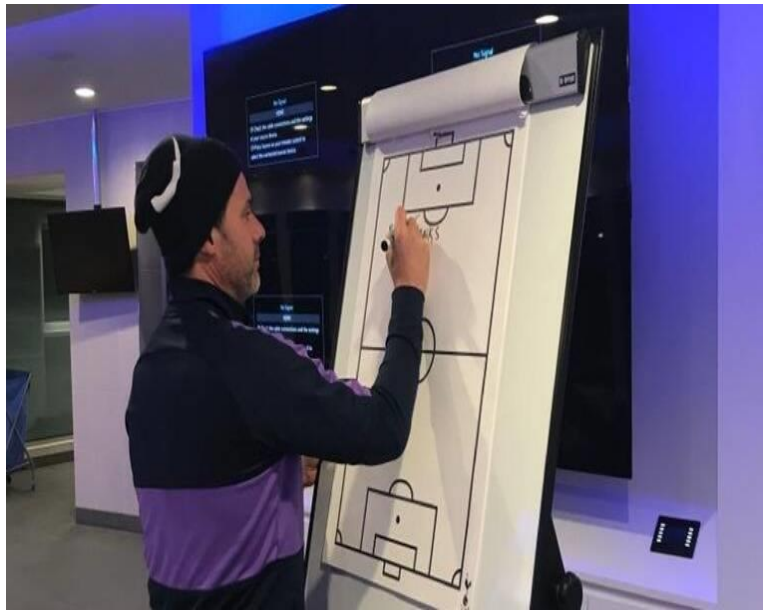
- Monte-Carlo(MC): Episodic Training
- Temporal-difference(TD): Experimental Training

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

✓ $V_\pi(s) = E_\pi[G_t | S_t = s] = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s]$

✓ $V_\pi(s_t) = \sum_{a_t \in A} \pi(a_t | s_t) Q_\pi(s_t, a_t)$

✓ $Q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a] = E_\pi[R_{t+1} + \gamma E_\pi[Q(S_{t+1}, a')] | S_t = s, A_t = a]$



Monte-Carlo Method

Temporal-Difference Method

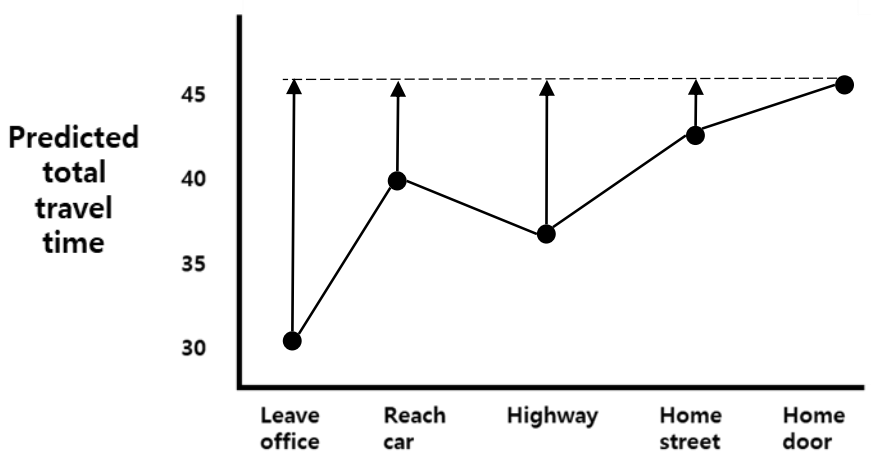
Learning Methods

MC vs TD

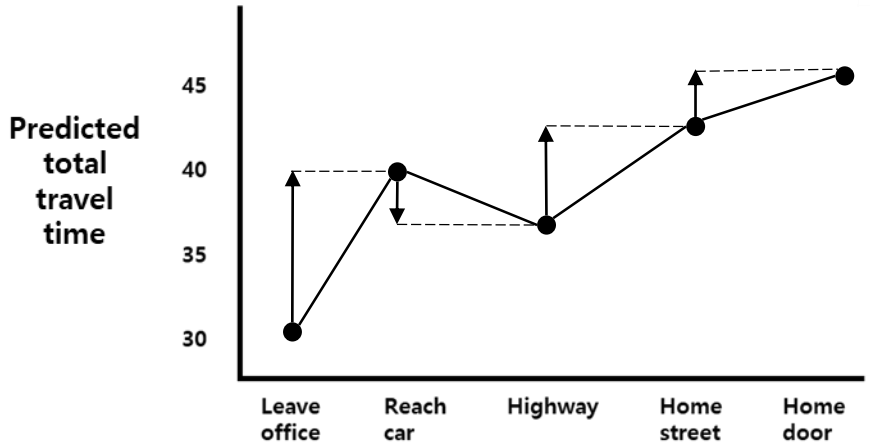
- $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- $V_\pi(s) = E_\pi[G_t | S_t = s] = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s]$
- $V_\pi(s_t) = \sum_{a_t \in A} \pi(a_t | s_t) Q_\pi(s_t, a_t)$
- $Q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a] = E_\pi[R_{t+1} + \gamma E_\pi[Q(S_{t+1}, a')] | S_t = s, A_t = a]$

❖ 업데이트 주기에 따른 분류

- Monte-Carlo(MC): High Variance, Low Bias
- Temporal-difference(TD): Low Variance, High Bias
- Example
  - ✓ Way back home



Monte-Carlo Method



Temporal-Difference Method

https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf

Learning Methods

MC vs TD

❖ 업데이트 주기에 따른 분류

- Monte-Carlo(MC): Update by True value
- Temporal-difference(TD): Update by TD target

$$V(s_t) \leftarrow V(s_t) + \alpha(G_t^{\infty} - V(s_t)) \qquad G_t^{\infty} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots$$ **MC Method**

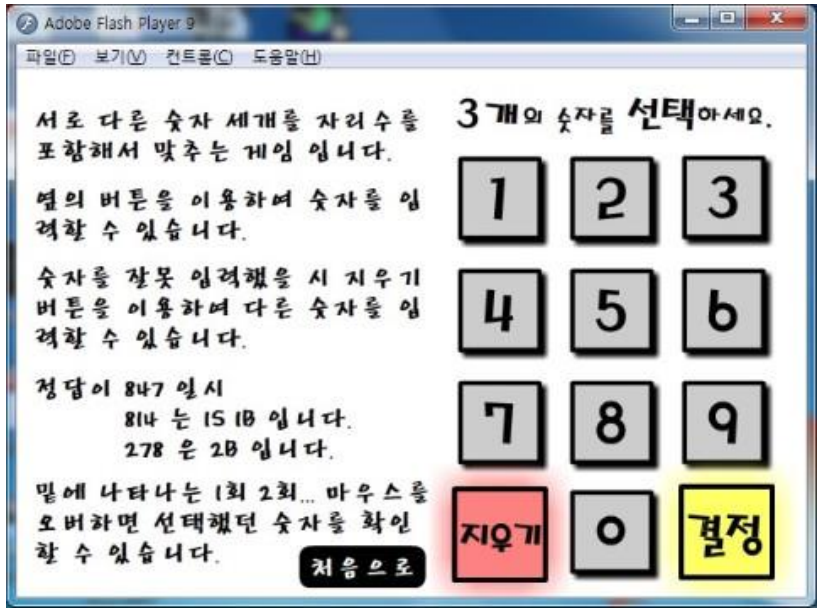
$$V(s_t) \leftarrow V(s_t) + \alpha(G_t^{n} - V(s_t)) \qquad G_t^{n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^{n} V(s_{t+n})$$ **n-step TD Method**

$$V(s_t) \leftarrow V(s_t) + \alpha(G_t^{1} - V(s_t)) \qquad G_t^{1} = R_{t+1} + \gamma V(s_{t+1})$$ **1-step TD Method**

Learning Methods

MC vs TD

- TD learning
  - 한 단계 더 나은 추정치를 향해가기
  - $V(s_t) \leftarrow V(s_t) + \alpha(R_{t+1} + \gamma V(s_{t+1}) - V(s_t))$
  - $Q(s_t,\ a_t) \leftarrow Q(s_t,\ a_t) + \alpha(R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t,\ a_t))$



Learning Methods

MC vs TD

- TD learning
  - 한 단계 더 나은 추정치를 향해가기
  - $V(s_t) \leftarrow V(s_t) + \alpha(R_{t+1} + \gamma V(s_{t+1}) - V(s_t))$
  - $Q(s_t,\ a_t) \leftarrow Q(s_t,\ a_t) + \alpha(R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t,\ a_t))$

Methods

On-policy vs Off-policy

❖ 업데이트 방식에 따른 분류

- On-Policy : Training on data only produced by a current policy ex) Expected SARSA
- Off-Policy : Training on data produced by a differenct policy ex) Q-learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(TD\ target - Q(s_t, a_t))$$

Expected SARSA

$$R_{t+1} + \gamma E_{a\sim\pi}[Q(S_{t+1}, a)]$$

Cannot reuse data

Q-learning

$$R_{t+1} + \gamma Max_a[Q(S_{t+1}, a)]$$

Overestimation

Methods

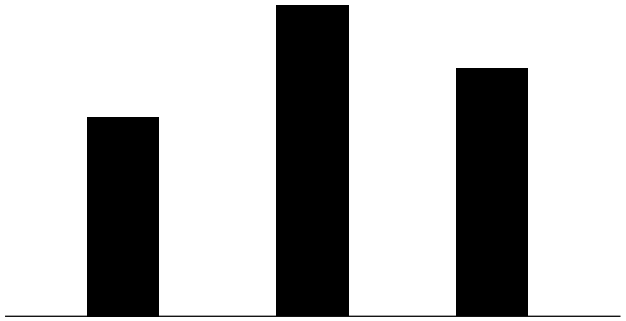
On-policy vs Off-policy

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(TD\ target - Q(s_t, a_t))$$

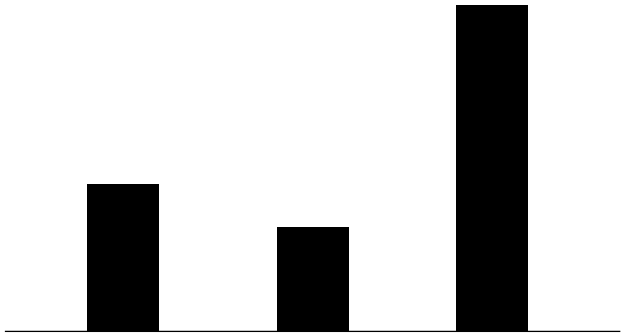
$$R_{t+1} + \gamma E_{a\sim\pi}[Q(S_{t+1}, a)]$$

❖ Expected SARSA

- TD-target : 다음 상태의 행동 가치 함수(Q)의 기대값
- 정책(π)이 업데이트되면 상태(s)에 대한 액션(a)의 확률 분포가 바뀜
- 동일 상태에 대한 타깃값이 계속 바뀌기 때문에 같은 경험(e)을 여러 번 업데이트 할 수 없음



$\pi(a|S_{t+1})$



$\pi(a|S_{t+1})$

Methods

On-policy vs Off-policy

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(TD\ target - Q(s_t, a_t))$$

$$R_{t+1} + \gamma Max_a[Q(S_{t+1}, a)]$$

❖ Q-learning

- TD-target : 다음 상태의 행동 가치 함수(Q)의 최대값
- 정책(π) 변화에 상관없이 해당 상태에서 받을 수 있는 보상의 최대값만 고려
- 행동 가치 함수가 과다추정 될 수 있음

95% 확률로
100만원 받기

1% 확률로
1억원 받기

Value-based Algorithms

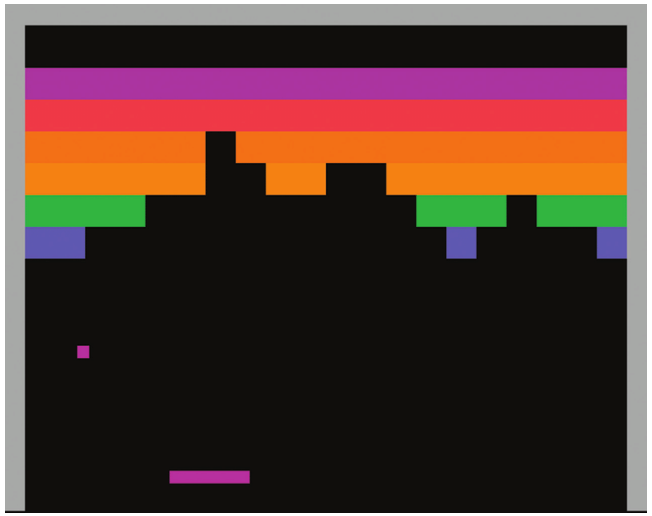
DQN(V Mnih et al. 2013)

❖ Key Points

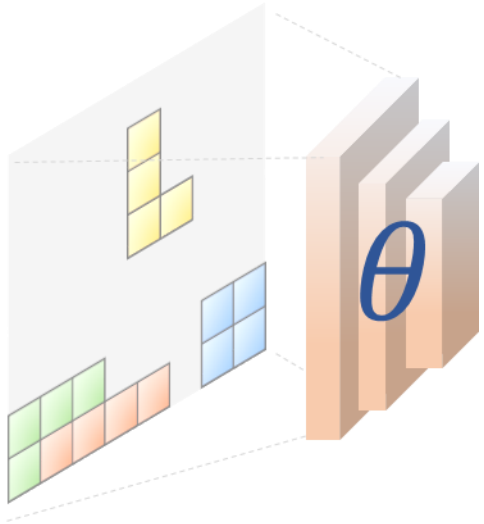
- 인공신경망을 **추정 함수(Estimate Function)**로 사용하여 가치 함수를 추정
  - ✓ Q-learning(Off-Policy) + CNN/DNN
- 데이터를 저장하고 반복 학습하기 위해 Experience Replay Mechanism 도입

	actions			
states	a_0	a_1	a_2	...
s_0	$Q(s_0,a_0)$	$Q(s_0,a_1)$	$Q(s_0,a_2)$	...
s_1	$Q(s_1,a_0)$	$Q(s_1,a_1)$	$Q(s_1,a_2)$	...
s_2	$Q(s_2,a_0)$	$Q(s_2,a_1)$	$Q(s_2,a_2)$	...
⋮	⋮	⋮	⋮	⋮

Tabular Q-learning



Too many states….



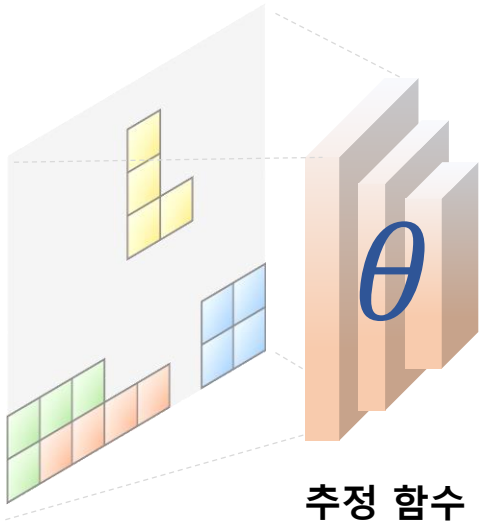
Estimate Function

Value-based Algorithms

DQN(V Mnih et al. 2013)

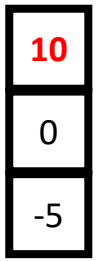
❖ Experience Replay Mechanism

- Transition 을 queue 의 형태로 저장, 추출하여 학습
- 데이터의 효율성 증가 및 샘플 간의 상관성을 제거하여 독립성 보장

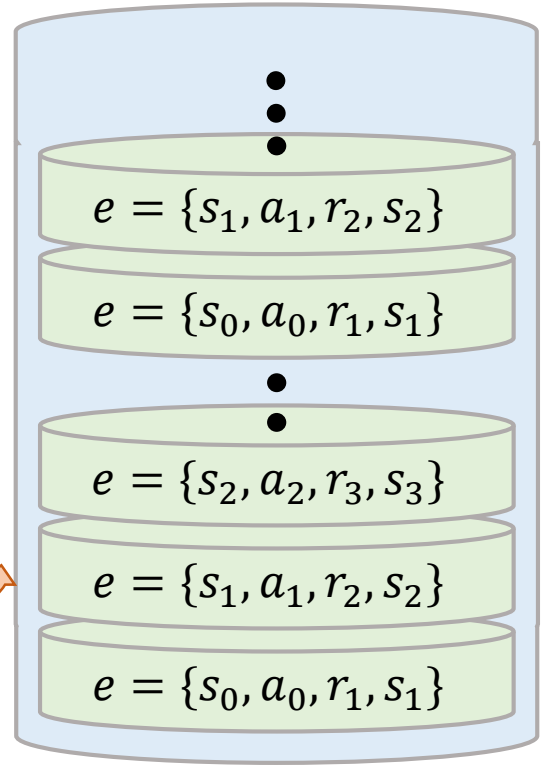


추정 함수
(Estimate Function)

$Q(s_t, a_t; \theta)$



$e = \{s_t, a_t, r_{t+1}, s_{t+1}\}$



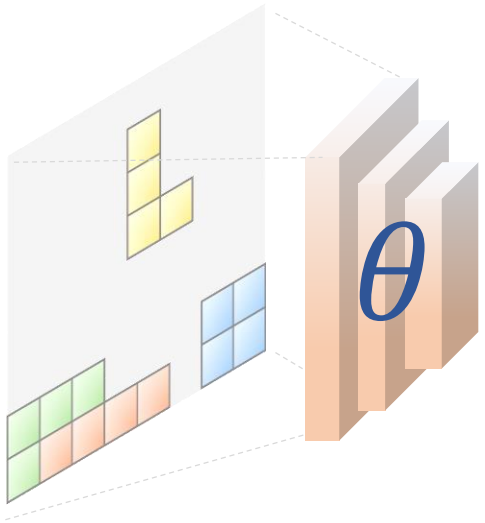
Replay Buffer

Value-based Algorithms

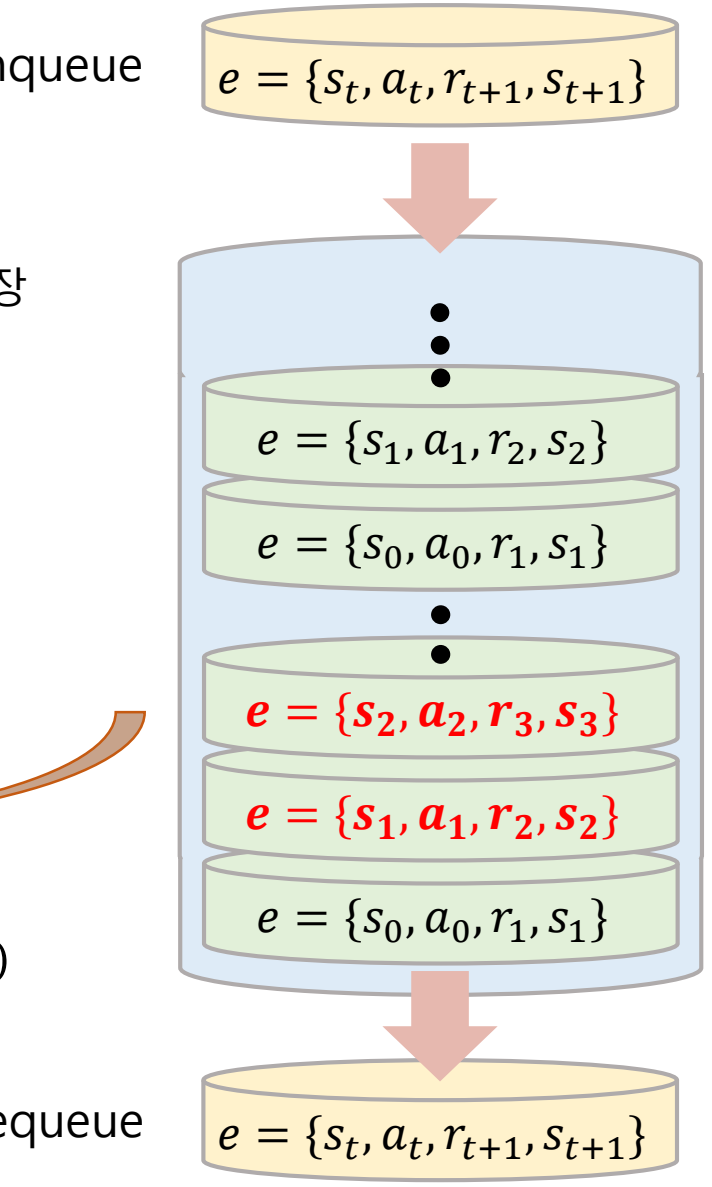
DQN(V Mnih et al. 2013)

❖ Experience Replay Mechanism

- Transition 을 queue 의 형태로 저장, 추출하여 학습
- 데이터의 효율성 증가 및 샘플 간의 상관성을 제거하여 독립성 보장



$$Loss = MSE[Q(s_t, a_t; \theta), TD\ target]$$
$$TD\ target = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta)$$



Value-based Algorithms

DQN(V Mnih et al. 2013)

❖ 문제점

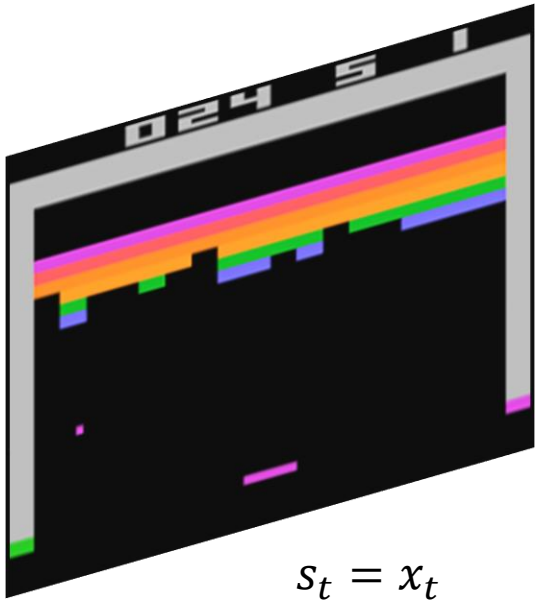
- 하나의 프레임만 봐서는 에이전트의 위치나 속도 등을 파악하기 어렵다.
- 에이전트의 exploration 이 보장 되지 않는다.
- parameter 가 바뀌면 target 값 또한 바뀌기 때문에 학습이 불안정적이다.

Value-based Algorithms

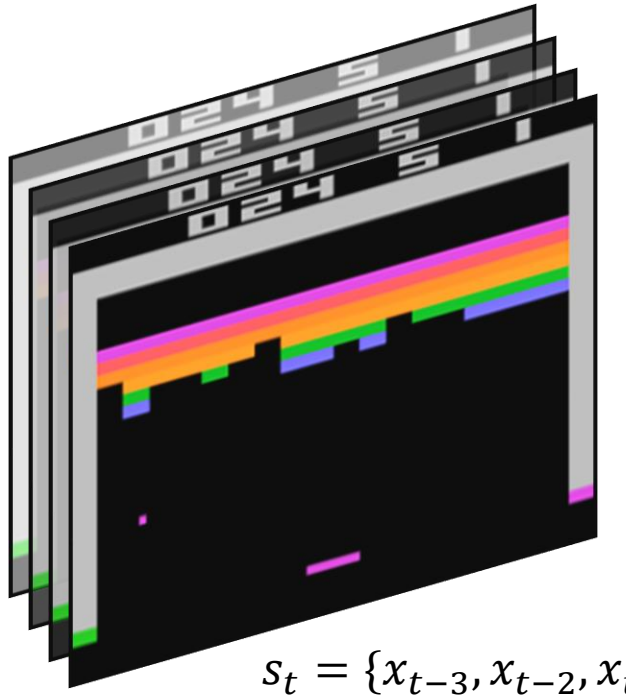
DQN(V Mnih et al. 2013)

❖ 문제점

- **하나의 프레임만 봐서는 에이전트의 위치나 속도 등을 파악하기 어렵다.**
  - ✓ 4개의 연속적인 프레임을 쌓아 input 으로 사용
- 에이전트의 exploration 이 보장 되지 않는다.
- parameter 가 바뀌면 target 값 또한 바뀌기 때문에 학습이 불안정적이다.



$s_t = x_t$



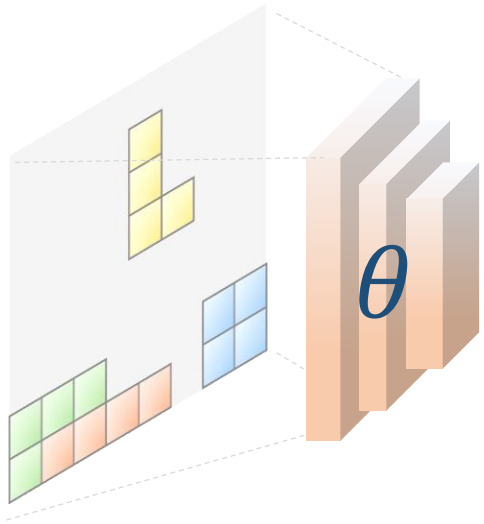
$s_t = \{x_{t-3}, x_{t-2}, x_{t-1}, x_t\}$

Value-based Algorithms

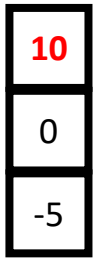
DQN(V Mnih et al. 2013)

❖ 문제점

- 하나의 프레임만 봐서는 에이전트의 위치나 속도 등을 파악하기 어렵다.
- **에이전트의 exploration 이 보장 되지 않는다.**
  - ✓ ϵ-greedy policy 를 통해 다양한 행동(a) 를 보장
- parameter 가 바뀌면 target 값 또한 바뀌기 때문에 학습이 불안정적이다.



$Q(s_t, a_t; \theta)$



$$95\% : a = argmax_{a'} Q(s_t, a_t; \theta)$$
$$5\% : a = random$$

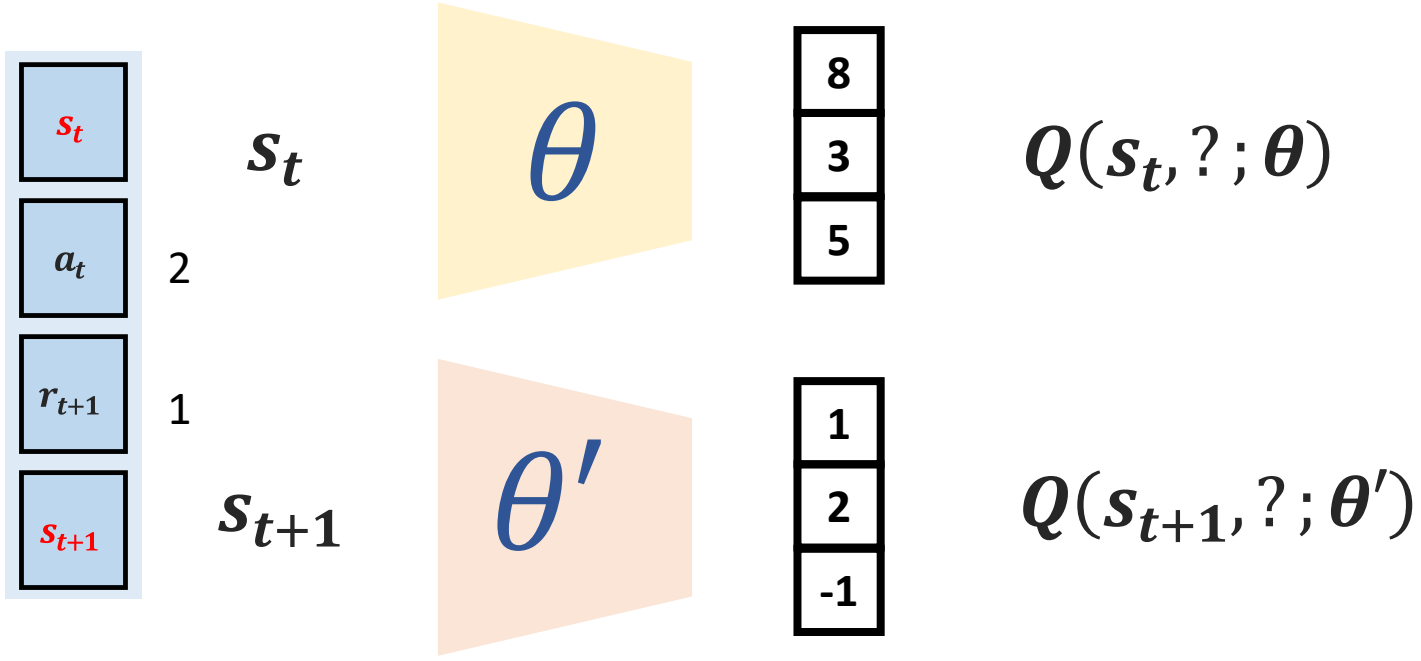
Value-based Algorithms

DQN(V Mnih et al. 2013)

$$Loss = MSE[Q(s_t, a_t; \theta), TD\ target]$$
$$TD\ target = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta')$$

❖ 문제점

- 하나의 프레임만 봐서는 에이전트의 위치나 속도 등을 파악하기 어렵다.
- 에이전트의 exploration 이 보장 되지 않는다.
- **parameter 가 바뀌면 target 값 또한 바뀌기 때문에 학습이 불안정적이다.**
  - ✓ 안정적인 학습을 위해 target network 도입



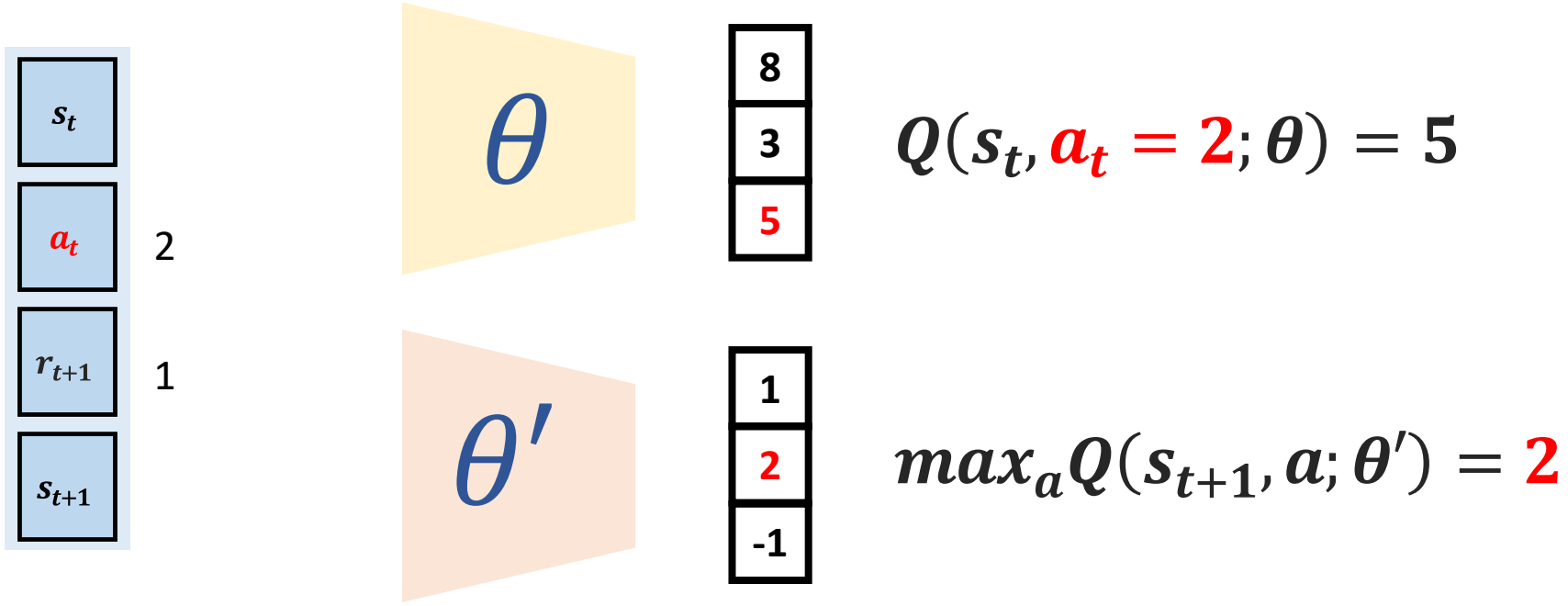
Value-based Algorithms

DQN(V Mnih et al. 2013)

$$Loss = MSE[Q(s_t, a_t; \theta), TD\ target]$$
$$TD\ target = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta')$$

❖ 문제점

- 하나의 프레임만 봐서는 에이전트의 위치나 속도 등을 파악하기 어렵다.
- 에이전트의 exploration 이 보장 되지 않는다.
- **parameter 가 바뀌면 target 값 또한 바뀌기 때문에 학습이 불안정적이다.**
  - ✓ 안정적인 학습을 위해 target network 도입



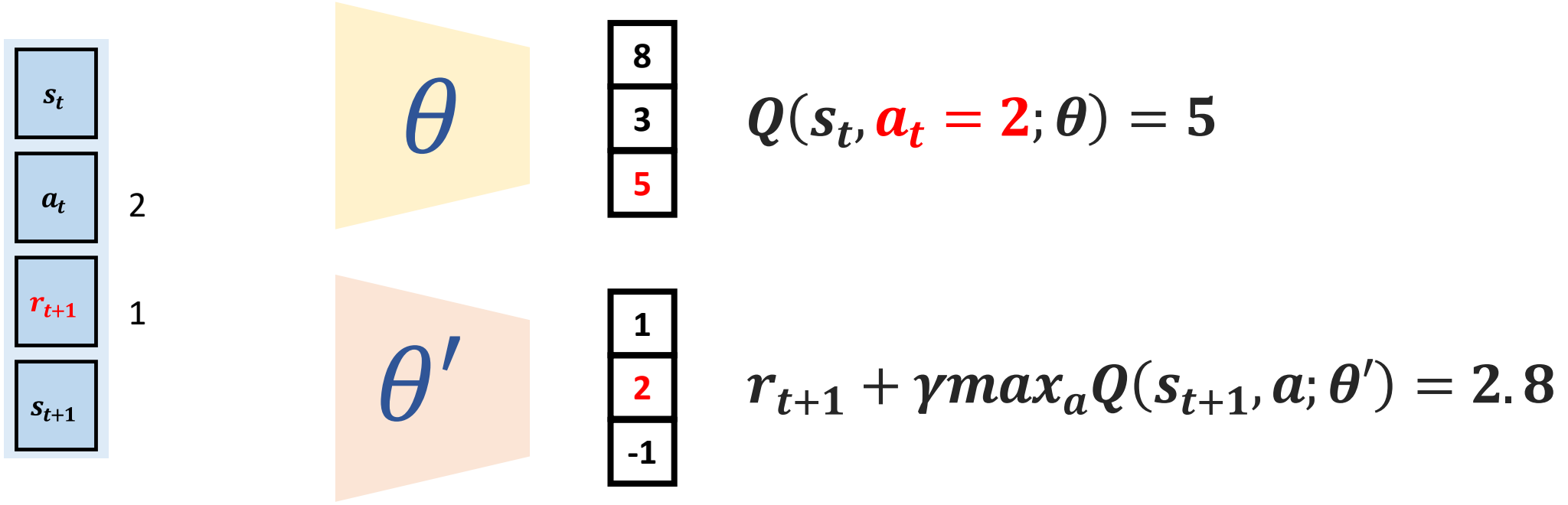
Value-based Algorithms

DQN(V Mnih et al. 2013)

$$Loss = MSE[Q(s_t, a_t; \theta), TD\ target]$$
$$TD\ target = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta')$$

❖ 문제점

- 하나의 프레임만 봐서는 에이전트의 위치나 속도 등을 파악하기 어렵다.
- 에이전트의 exploration 이 보장 되지 않는다.
- **parameter 가 바뀌면 target 값 또한 바뀌기 때문에 학습이 불안정적이다.**
  - ✓ 안정적인 학습을 위해 target network 도입



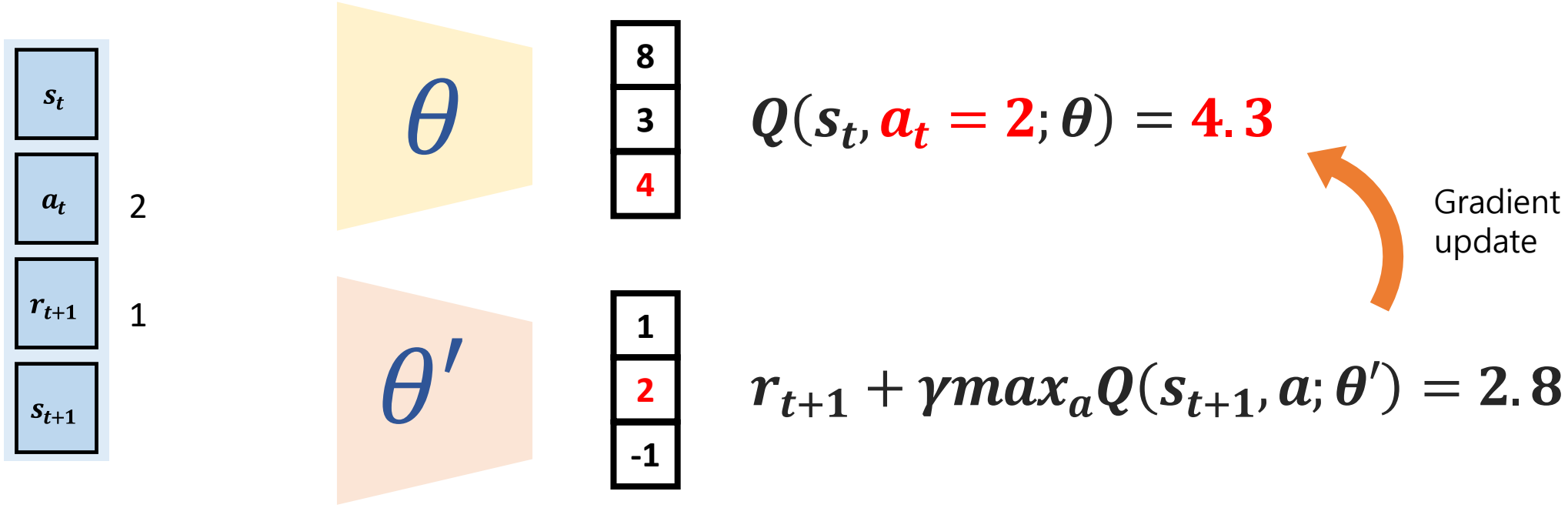
Value-based Algorithms

DQN(V Mnih et al. 2013)

$$Loss = MSE[Q(s_t, a_t; \theta), TD\ target]$$
$$TD\ target = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta')$$

❖ 문제점

- 하나의 프레임만 봐서는 에이전트의 위치나 속도 등을 파악하기 어렵다.
- 에이전트의 exploration 이 보장 되지 않는다.
- **parameter 가 바뀌면 target 값 또한 바뀌기 때문에 학습이 불안정적이다.**
  - ✓ 안정적인 학습을 위해 target network 도입



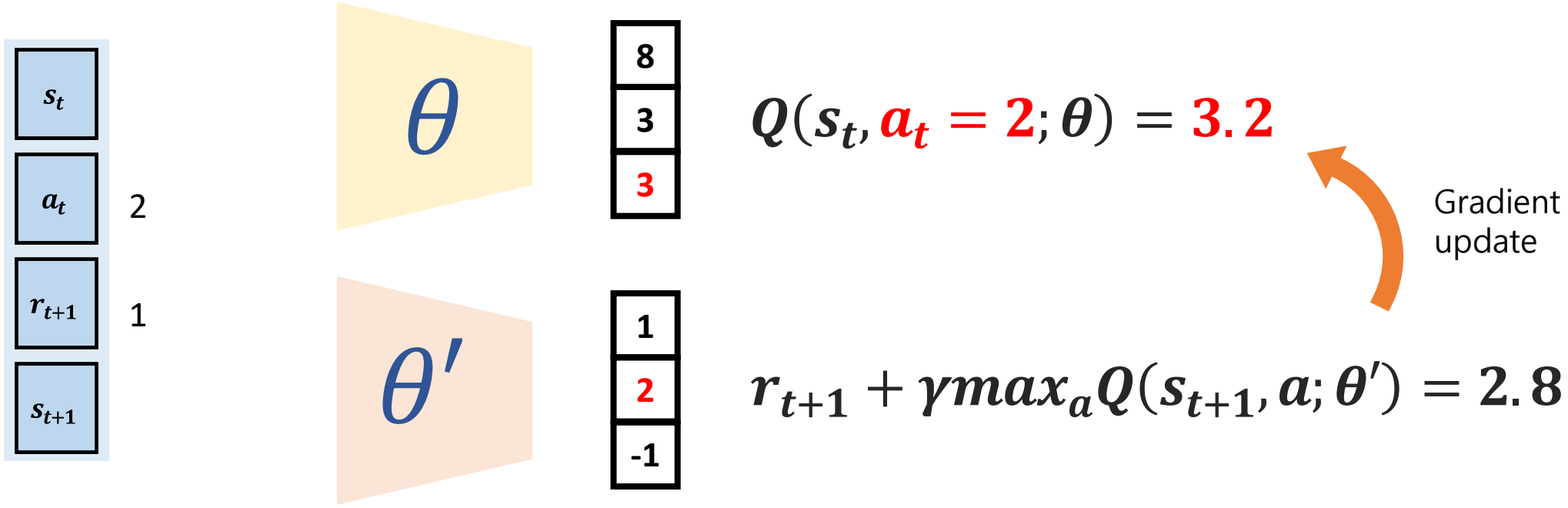
Value-based Algorithms

DQN(V Mnih et al. 2013)

$$Loss = MSE[Q(s_t, a_t; \theta), TD\ target]$$
$$TD\ target = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta')$$

❖ 문제점

- 하나의 프레임만 봐서는 에이전트의 위치나 속도 등을 파악하기 어렵다.
- 에이전트의 exploration 이 보장 되지 않는다.
- **parameter 가 바뀌면 target 값 또한 바뀌기 때문에 학습이 불안정적이다.**
  - ✓ 안정적인 학습을 위해 target network 도입

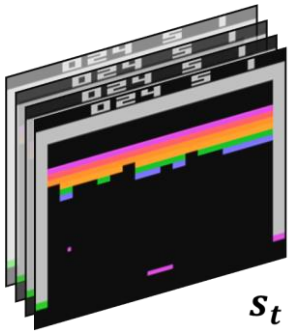


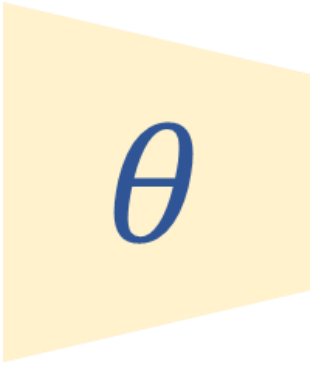
Value-based Algorithms

DQN(V Mnih et al. 2013)

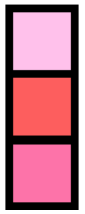
❖ Total Process

- 4개의 프레임을 합쳐 상태(s)로 입력을 받는다.
- Eval Net 을 통해 각 행동(a) 에 대한 행동가치함수의 추정값을 얻는다.
- ϵ-greedy policy 를 통해 행동(a)를 선택한다.
- 다음 상태(s')와 보상(r)을 환경으로부터 전달받는다.





$Q(s_t, a_t; \theta)$

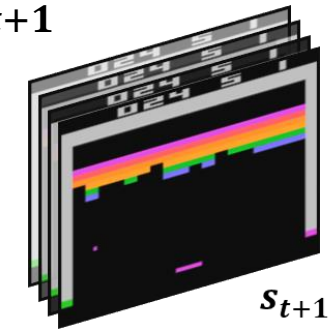


$95\% : a = argmax_{a'} Q(s_t, a_t; \theta)$

$5\% : a = random$

Choose action by ϵ-greedy policy

Gain r_{t+1}

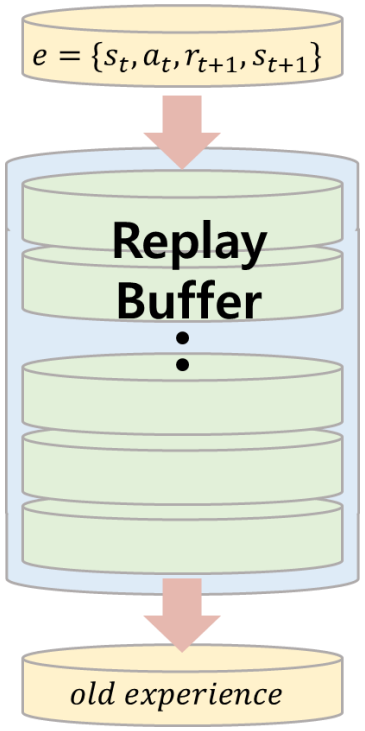


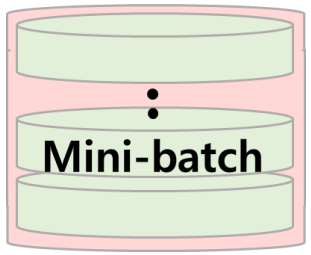
Value-based Algorithms

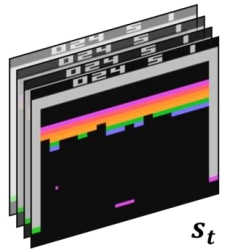
DQN(V Mnih et al. 2013)

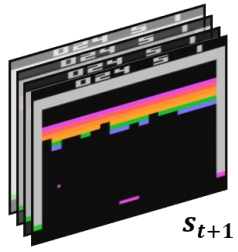
❖ Total Process

- Replay Buffer(Queue)에 경험을 저장한다.
- Replay Buffer 에서 mini-batch 를 샘플링하고 Q 값을 구한다.
- **Target network 의 parameter 는 일정 주기마다 current network 의 parameter 로 업데이트한다!!**









θ

θ'

8
3
5

$Q(s_t, a_t; \theta) = 5$

$a_t = 2$

Gradient update

max

1
2
-1

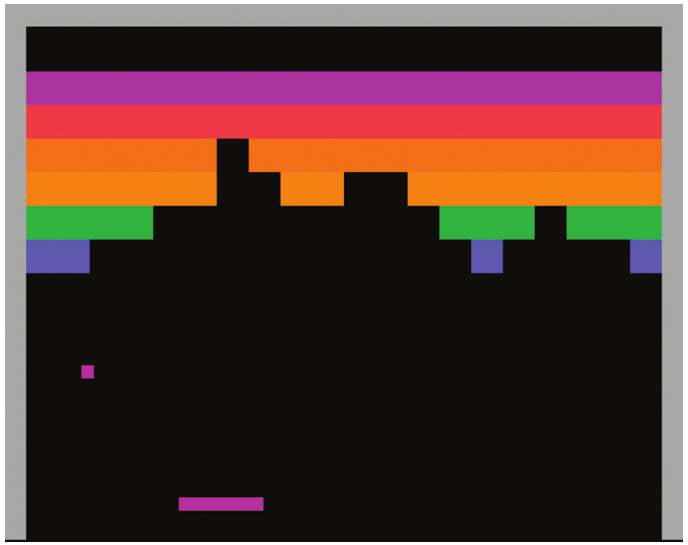
$\times \gamma + r_{t+1} = 2.8$

Value-based Algorithms

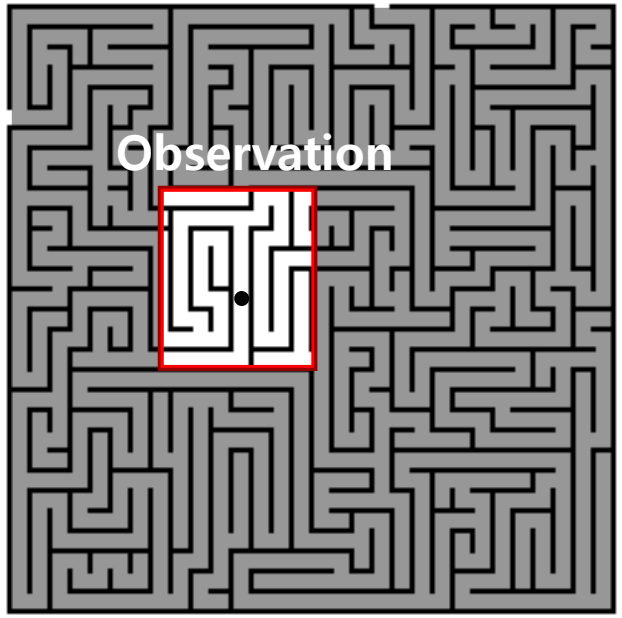
DRQN(M Hausknecht et al. 2015)

❖ Key Points

- POMDP(Partially Observable MDP)
  - ✓ 실제로 우리가 보는 것은 상태(S) 가 아니라 관측치(O).
  - ✓ What's the difference between 'State' and 'Observation'
- DRQN 은 POMDP 가정에서 DQN 을 사용하기 위해 LSTM 을 도입



Velocity?? Direction??



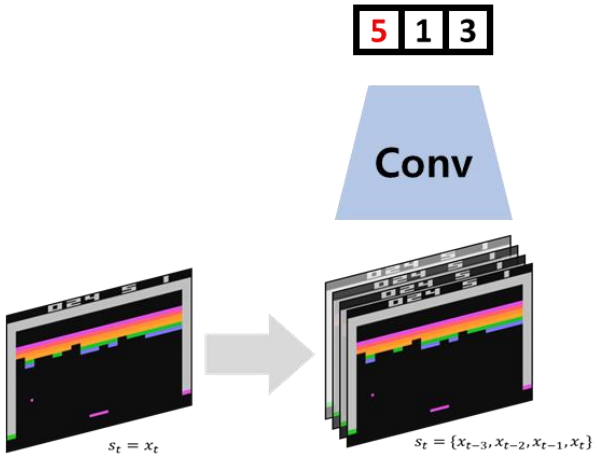
State

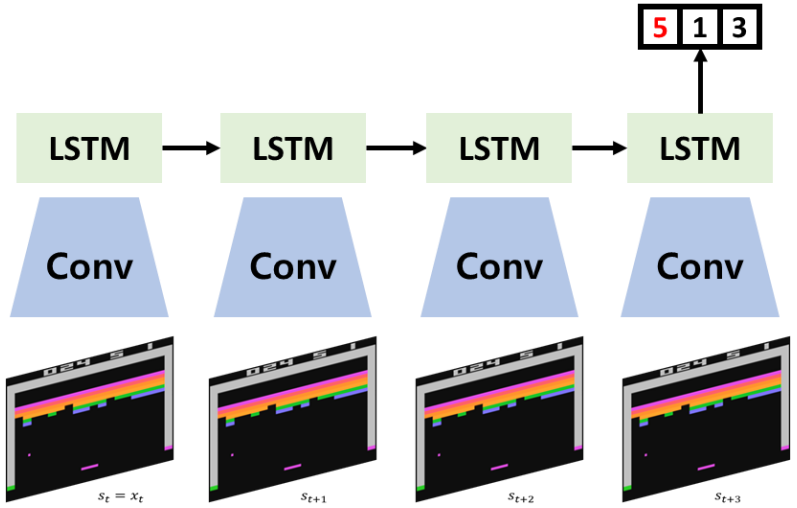
Value-based Algorithms

DRQN(M Hausknecht et al. 2015)

❖ Key Points

- 기존의 DQN 은 4개의 frame 을 병합함으로써 POMDP 보다는 MDP 에 가까워짐
- DRQN 은 LSTM 을 이용하여 POMDP 상황을 극복하고자 함





Value-based Algorithms

DRQN(M Hausknecht et al. 2015)

- Experimental Results
  - DRQN 이 DQN 을 항상 뛰어넘는 성능을 보이는 것은 아님
  - Frostbite 에서는 DQN 보다 훨씬 좋았으나 Beam Rider 에서는 성능이 뒤떨어짐

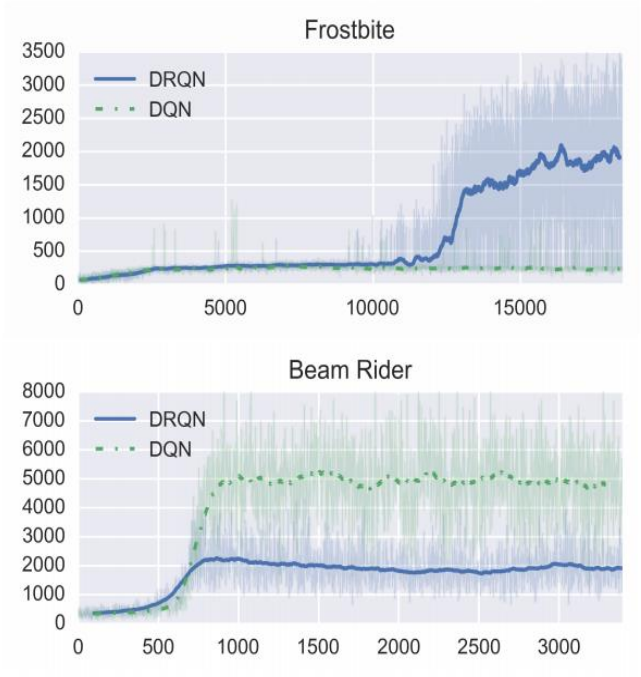


Figure 4: Frostbite and Beam Rider represent the best and worst games for DRQN. Frostbite performance jumps as the agent learns to reliably complete the first level.

Frostbite

Beam Rider

Value-based Algorithms

DRQN(M Hausknecht et al. 2015)

- When game is flickering…
  - DRQN 은 frame 의 일부 손실에도 성능의 하락이 DQN 보다 적음을 보여줌

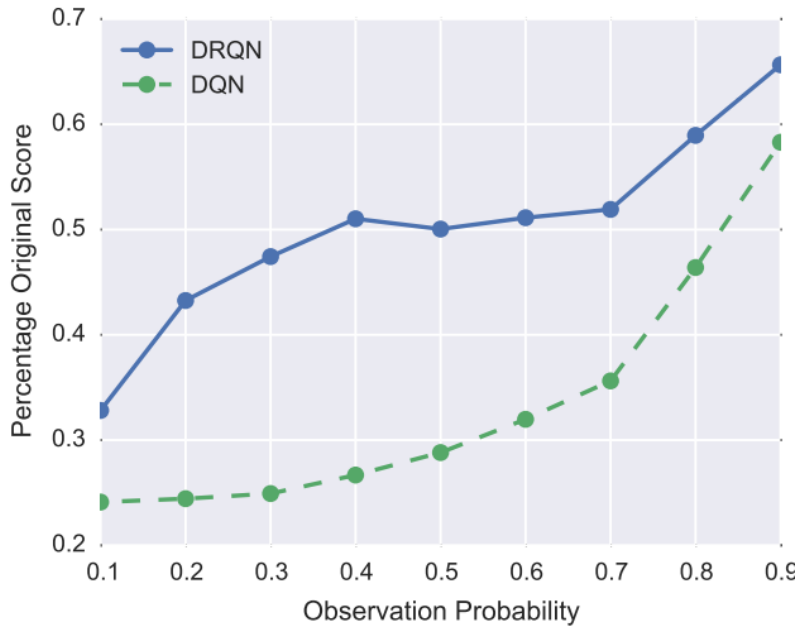


Figure 5: When trained on normal games (MDPs) and then evaluated on flickering games (POMDPs), DRQN's performance degrades more gracefully than DQN's. Each data point shows the average percentage of the original game score over all 9 games in Table 1.

Conclusion

❖ Summary

- 강화학습은 미래에 대한 누적 보상을 최대화 하는 것이다.

- 누적 보상은 어떻게 구할까?? 추정치로!!
  - ✓ 상태가치함수 : 현재 상태에 대한 평가
  - ✓ 행동가치함수 : 현재 상태에서 행동에 대한 평가
  - ✓ **수 많은 경우의 수에 대한 추정치를 따로 구할 수 없으니 추정 함수를 사용하자!!**

- 추정 함수는 어떻게 학습하지??
  - ✓ Monte-Carlo & Temporal-Difference
  - ✓ On-Policy & Off-Policy

Appendix

Reference

❖ Papers

- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.
- Hausknecht, M., & Stone, P. (2015, September). Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*.

❖ Sites & Codes

- https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html
- https://github.com/qfettes/DeepRL-Tutorials
- https://github.com/seungeunrho/minimalRL

Appendix

Additional Materials

❖ Papers

- Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 30, No. 1).
- Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, June). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning* (pp. 1995-2003). PMLR.
- Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.
- Bellemare, M. G., Dabney, W., & Munos, R. (2017, July). A distributional perspective on reinforcement learning. In International Conference on Machine Learning (pp. 449-458). PMLR.
- Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., ... & Legg, S. (2017). Noisy networks for exploration. arXiv preprint arXiv:1706.10295.
- Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., ... & Silver, D. (2018, April). Rainbow: Combining improvements in deep reinforcement learning. In Thirty-second AAAI conference on artificial intelligence.

❖ Sites & Codes

- https://gym.openai.com/
- https://stable-baselines3.readthedocs.io/en/master/