Value-based Learning 2

2023. 03. 24

발표자: 김정인

발표자 소개

❖ 이름: 김정인 (Jung In Kim)

- Data Mining & Quality Analytics Lab (김성범 교수님)
- 석박통합과정(2021.09 ~)

❖ 관심 연구 분야

- Reinforcement Learning

❖ E-mail

- Jungin_kim23@korea.ac.kr

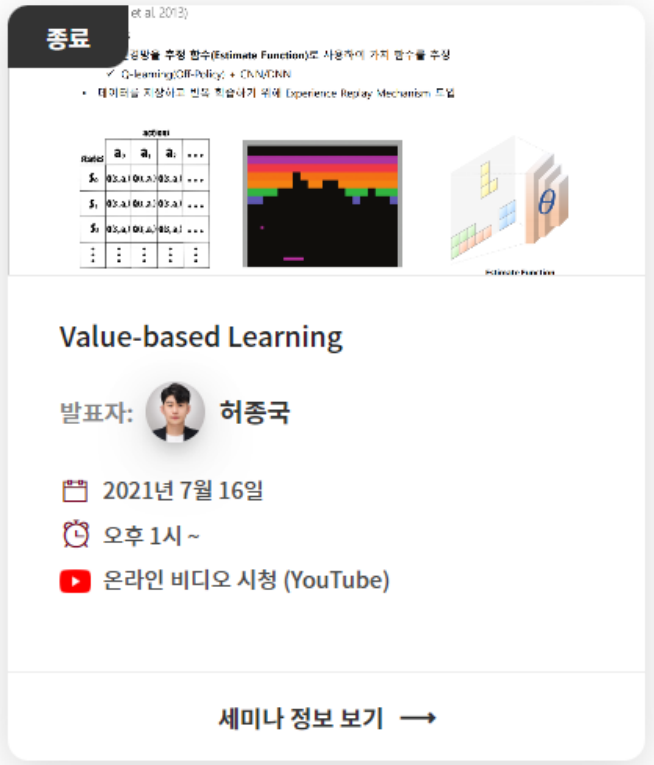
목차

1. Introduction to Reinforcement Learning
2. Deep Reinforcement Learning
   - Deep Q-Network (DQN)
   - Double DQN (DDQN)
   - Dueling Network
   - Prioritized Experience Replay (PER)
3. Conclusion

1. Introduction to Reinforcement Learning (RL)

Related Seminar

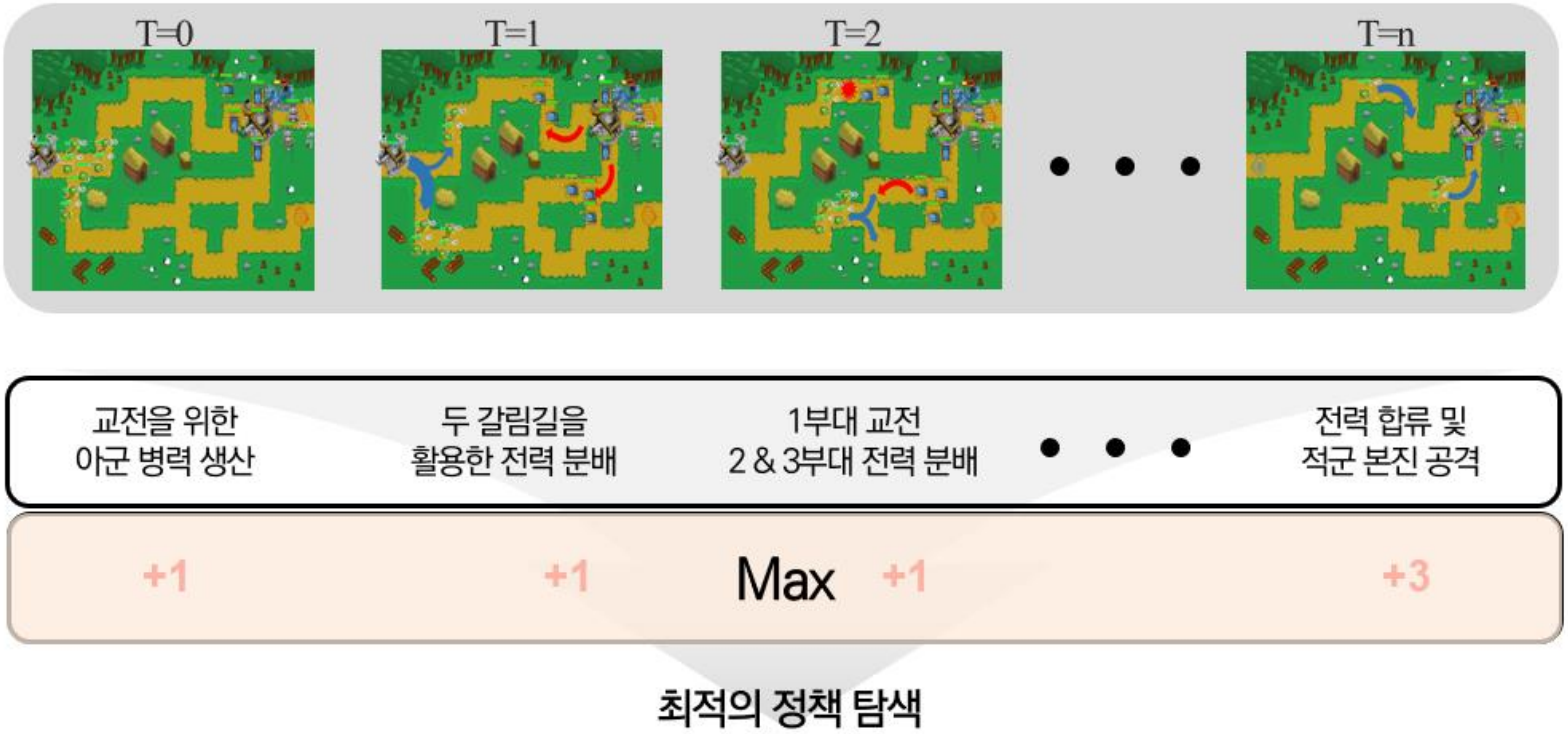
- Value-based Learning: 가치 기반 함수에 관한 전반적인 내용과 DQN, DRQN에 논문 소개
- Basic of Reinforcement Learning: 강화학습을 이해하는데 요구되는 기본 지식에 관한 소개



Introduction to Reinforcement Learning (RL)

❖ Definition of RL

- 순차적인 의사결정 문제에서 에이전트가 환경으로부터 받는 누적 보상 값을 최대화하는 행동 정책을 학습하는 방법
- 정책: 에이전트가 특정 상태에서 어떤 행동을 선택할지 정해주는 함수

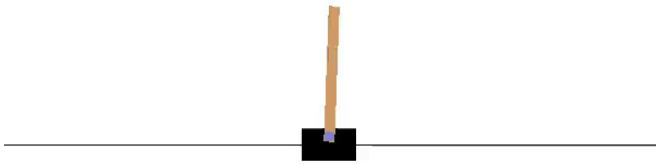


Introduction to Reinforcement Learning (RL)

❖ CartPole Game

- 막대기가 세워진 상태로 게임이 시작되며 넘어지지 않는 것이 목표인 게임

  → Benchmark Game: 단순한 상태 공간, 연속적이지 않은 행동 공간, 분명한 보상 신호 …



환경으로부터 받은 상태 정보를 통해 누적 보상을 최대화하기 위한 행동을 취함

충분한 시간 동안 막대기가 균형을 유지할 때까지 반복적으로 에이전트를 학습

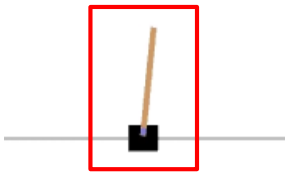
Introduction to Reinforcement Learning (RL)

❖ RL Components: Agent, State, Action, Reward

- Agent: 환경과 상호작용하는 주체이자 인공지능 플레이어
- State: 환경으로부터 받는 카트의 위치, 카트의 속도, 막대기의 각도, 막대기의 각속도 정보
- Action: 좌, 우 이동
- Reward: 각 timestep마다 환경으로부터 받는 스칼라 값

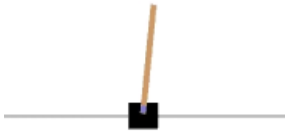
에이전트

- 환경으로부터 받는 상태 정보를 통해 막대기가 넘어지지 않도록 행동(왼쪽 또는 오른쪽)을 취하는 주체



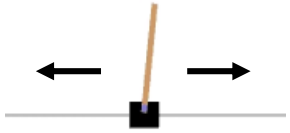
상태

- 환경으로부터 받는 상태 정보
- 카트의 위치 : 가로축에서 카트의 위치
- 카트의 속도: 가로축에서 카트의 속도
- 막대기의 각속도: 극 각속도가 변경되는 속도
- 막대기의 각도: 세로축에 대한 기둥의 각도



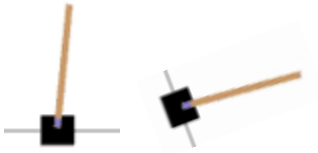
행동

- 환경으로부터 받은 상태 정보를 통해 누적 보상을 최대화하는 행동 선택



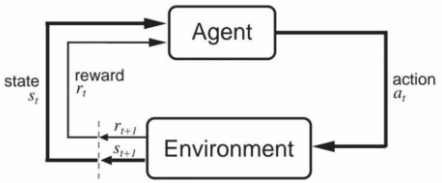
보상

- 각 timestep 마다 환경으로부터 받는 스칼라 값
- 에이전트가 취한 행동의 성공 또는 실패에 기반
- 카트 위에 균형 유지 → +1
- 막대기가 쓰러졌다면 → 0



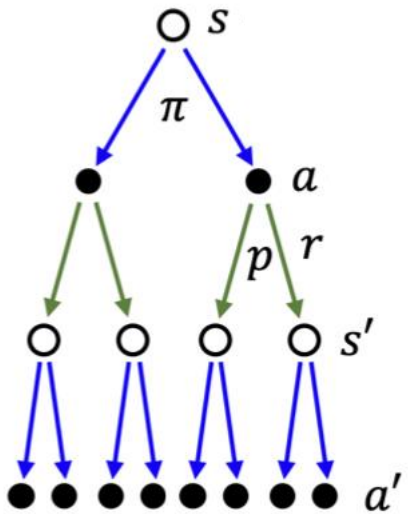
균형 유지 균형 유지 실패

Introduction to Reinforcement Learning (RL)



❖ Definition of value function

- 특정 상태에 있거나 특정 행동을 수행하는 것이 얼마나 좋은 지 추정하는 함수
- 에이전트가 특정 상태에서 시작하여 특정 정책에 따라 받을 수 있는 예상 누적 보상(= G_t)으로 정의

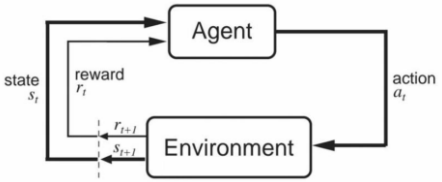


trajectory (or episode): $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

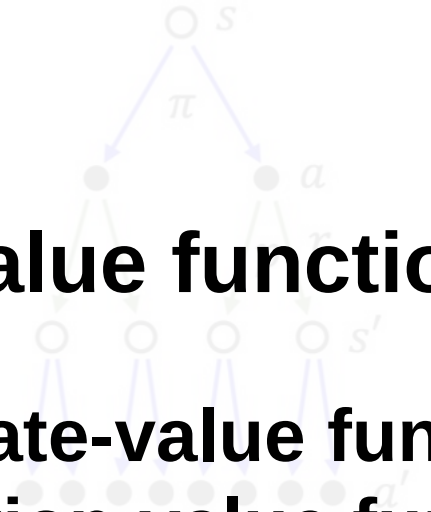
감쇠인자: 미래에 받을 불확실성으로 인해 즉각적인 보상의 우선 순위를 정해 예상되는 보상을 최대화하기 위해 사용되는 파라미터

Introduction to Reinforcement Learning (RL)



❖ Definition of value function

- 특정 상태에 있거나 특정 행동을 수행하는 것이 얼마나 좋은 지 추정하는 함수
- 에이전트가 특정 상태에서 시작하여 특정 정책에 따라 받을 수 있는 예상 누적 보상(= G_t)으로 정의



Value function

1. State-value function
2. Action-value function

trajectory (or episode): $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

감쇠인자: 미래에 받을 불확실성으로 인해 즉각적인 보상의 우선 순위를 정해 예상되는 보상을 최대화하기 위해 사용되는 파라미터

Introduction to Reinforcement Learning (RL)

❖ State value function & Action value function

- 상태 가치 함수: 정책 함수에 따라서 특정 상태(s)에서 에이전트가 받을 수 있는 누적 값을 추정하는 함수
- 행동 가치 함수: 특정 정책 함수에 따라서 특정 상태(s)에서 특정한 행동을 취했을 때 에이전트가 받을 수 있는 누적 값을 추정하는 함수

State value function

$$
\begin{aligned}
& v_\pi(s) \\
&= E_\pi[G_t|S_t = s] \\
&= E_\pi\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \ldots |S_t = s\right] \\
&= E_\pi\left[r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} \ldots)|S_t = s\right] \\
&= E_\pi\left[r_{t+1} + \gamma G_{t+1}|S_t = s\right] \\
&= E_\pi\left[r_{t+1} + \gamma v_\pi(s_{t+1})|S_t = s\right]
\end{aligned}
$$

t+1 시점에 받는 reward → immediate reward, future reward

Action value function

$$
\begin{aligned}
& q_\pi(s,a) \\
&= E_\pi[G_t|S_t = s, A_t = a] \\
&= E_\pi\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \ldots |S_t = s, A_t = a\right] \\
&= E_\pi\left[r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} \ldots)|S_t = s, A_t = a\right] \\
&= E_\pi\left[r_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a\right] \\
&= E_\pi\left[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})|S_t = s, A_t = a\right]
\end{aligned}
$$

t+1 시점에 받는 reward → immediate reward, future reward

Introduction to Reinforcement Learning (RL)

❖ Model-Based vs Model-Free Algorithm

- Model-Based: 보상 함수(r_s^a) 상태 전이 확률($p_{ss'}^a$)을 알고 있는 상황에서 최적 정책을 학습하는 알고리즘
- Model-Free: 보상 함수(r_s^a) 상태 전이 확률($p_{ss'}^a$)을 모르는 상황에서 최적 정책을 학습하기 위한 알고리즘

Model-Based

Model-Free

Introduction to Reinforcement Learning (RL)

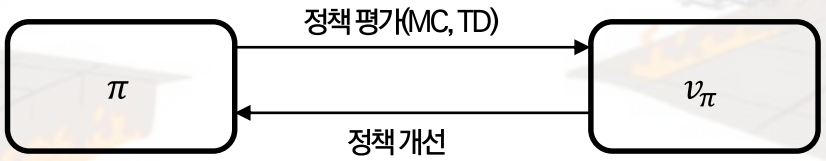
❖ Model-Based vs Model-Free Algorithm

- Model-Based: 보상 함수(r_s^a) 상태 전이 확률($p_{ss'}^a$)을 알고 있는 상황에서 최적 정책을 학습하는 알고리즘
- Model-Free: 보상 함수(r_s^a) 상태 전이 확률($p_{ss'}^a$)을 모를 는 상황에서 최적 정책을 학습하기 위한 알고리즘

Model-Based

Model-Free

Model-Free 에서 누적 보상을 최대화하는 정책 찾기 위한 방법



1. MC control
2. TD control → SARSA / Q-Learning

Introduction to Reinforcement Learning (RL)

❖ Temporal Difference (TD)

- Monte Carlo가 가지는 에피소드가 완전히 종료되는 MDP에서만 가능하다는 한계를 개선
- 상태 전이가 발생하게 되면 바로 이전 시점의 상태의 가치를 평가

$$v_\pi(s_t) = E_\pi\left[\underbrace{r_{t+1} + \gamma v_\pi(s_{t+1})}_{\text{TD target→정답지}} | S_t = s\right] \longrightarrow \text{TD: } V(s_t) = V(s_t) + \alpha^*(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

---

episode

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow \text{Terminate}$$

$$V(s_0) = V(s_0) + \alpha^*(r_1 + \gamma V(s_1) - V(s_0))$$

$$V(s_1) = V(s_1) + \alpha^*(r_2 + \gamma V(s_2) - V(s_1))$$

$$V(s_2) = V(s_2) + \alpha^*(r_3 + \gamma V(s_3) - V(s_2))$$

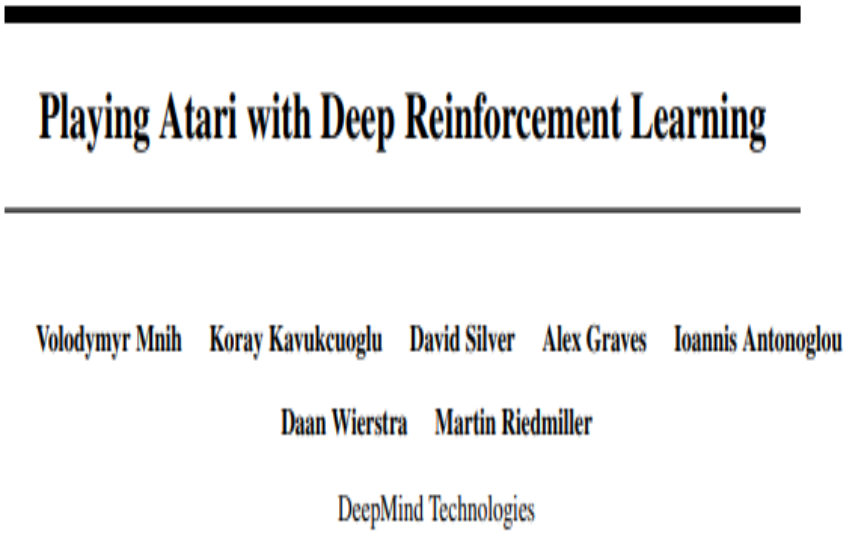
$$V(s_3) = V(s_3) + \alpha^*(r_4 + \gamma V(s_4) - V(s_3))$$

2. Deep Reinforcement Learning

Deep Reinforcement Learning

❖ Deep Q-Network(DQN)

- Google DeepMind에서 연구했으며 2013년, 2015년에 NeurIPS와 Nature에 발표된 방법론
- Deep Neural Network와 Reinforcement Learning을 결합해 인간 수준의 높은 성능을 달성한 첫 알고리즘

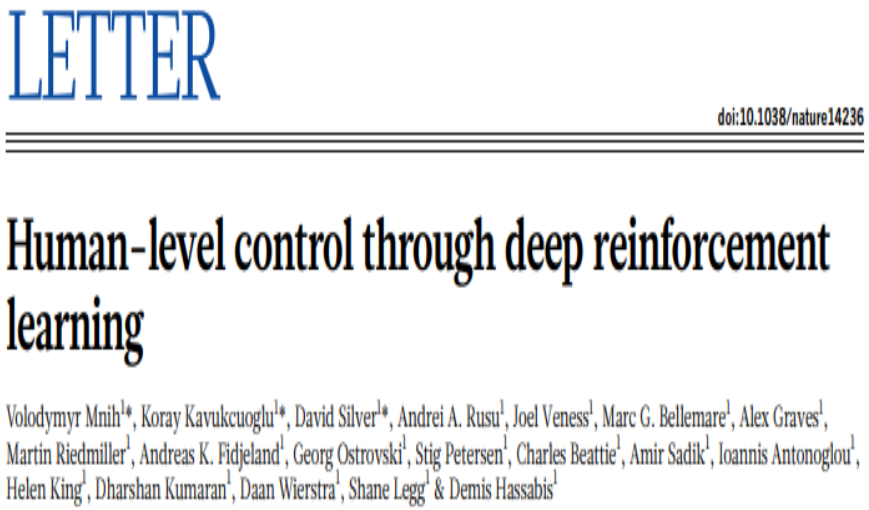


Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou

Daan Wierstra Martin Riedmiller

DeepMind Technologies



LETTER

doi:10.1038/nature14236

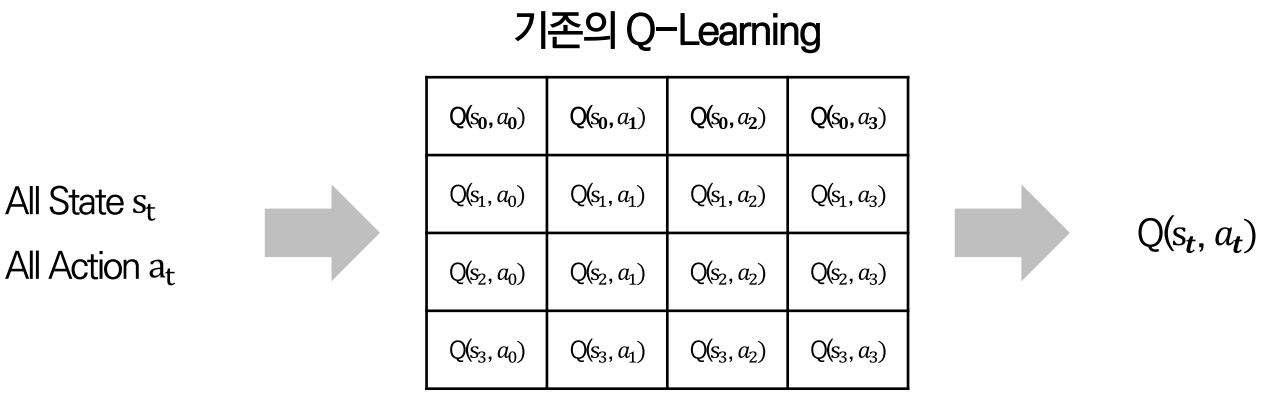
Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

Deep Reinforcement Learning

❖ Deep Q-Network(DQN)

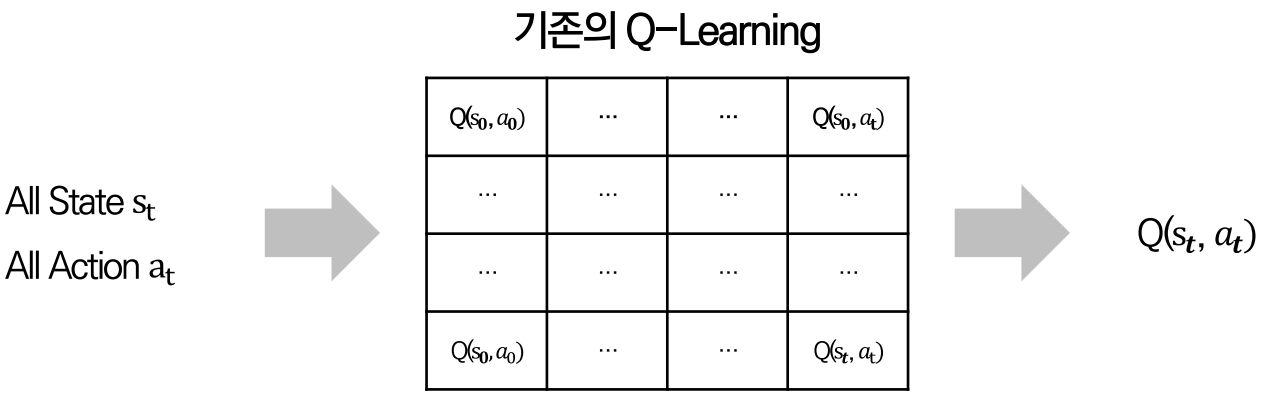
- 기존의 Q-Learning: 모든 상태와 행동에 관한 Q-value function을 계산 해야함
- Deep Q-Learning: Deep Neural Network를 Q-Learning과 결합하여 기존 Q-Learning의 한계 개선



Deep Reinforcement Learning

❖ Deep Q-Network(DQN)

- 기존의 Q-Learning: 모든 상태와 행동에 관한 Q-value function을 계산 해야함
- Deep Q-Learning: Deep Neural Network를 Q-Learning과 결합하여 기존 Q-Learning의 한계 개선

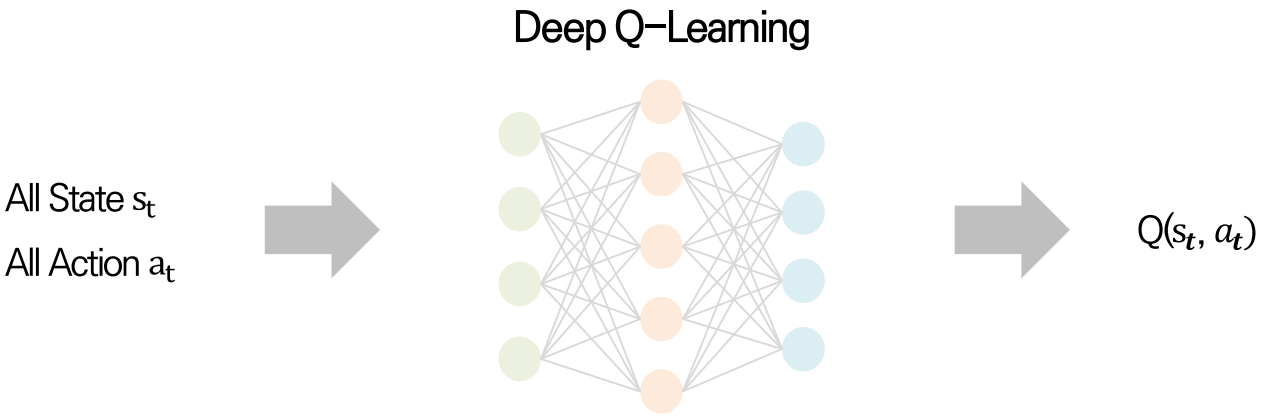


상태와 액션 공간이 커질 수록 많은 메모리가 필요하고 시간이 오래 걸리는 한계 존재

Deep Reinforcement Learning

❖ Deep Q-Network(DQN)

- 기존의 Q-Learning: 모든 상태와 행동에 관한 Q-value function을 계산 해야함
- Deep Q-Learning: Deep Neural Network를 Q-Learning과 결합하여 기존 Q-Learning의 한계 개선



모든 상태와 행동에 관한 Q-value를 사전에 구할 필요가 없음

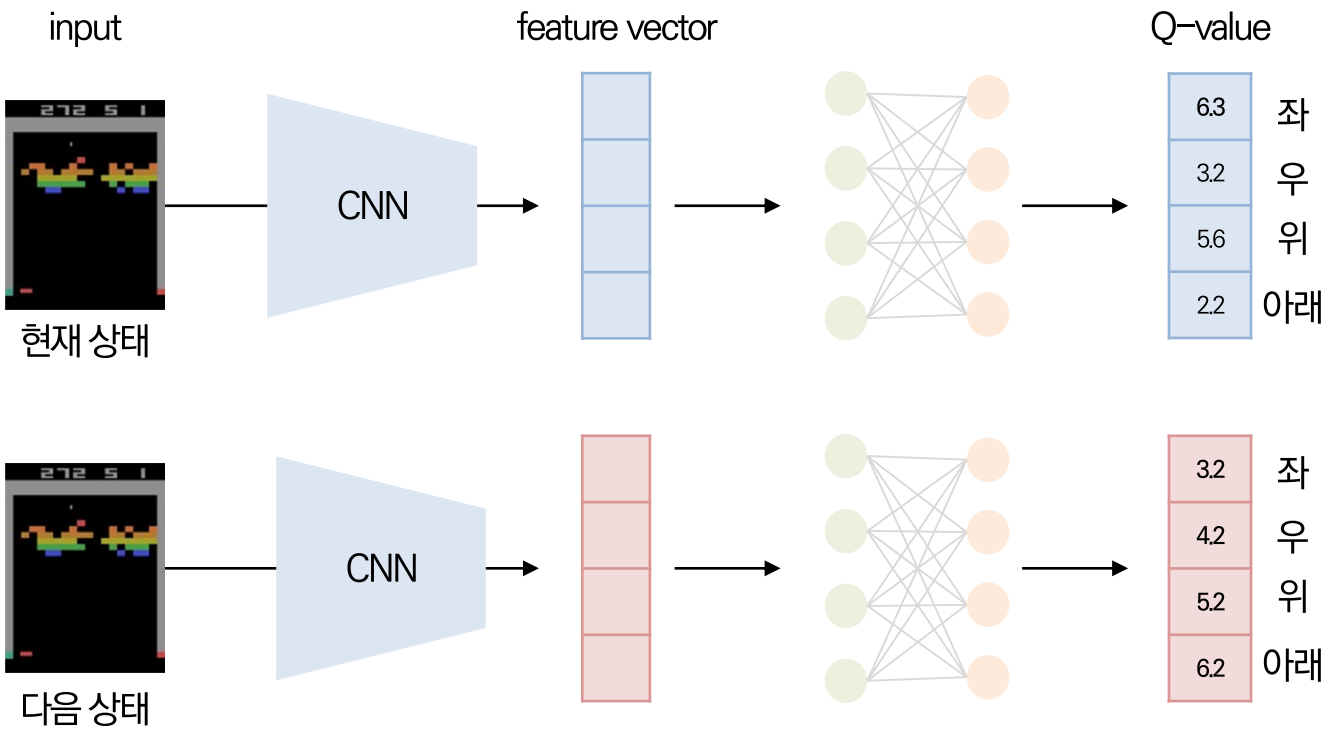
Deep Reinforcement Learning

Target: $r + \gamma \max_{a'} Q(s', a'; \theta)$

loss: $(r + \gamma \max Q(s', a'; \theta) - Q(s, a; \theta))^2$

❖ Deep Q-Network(DQN, 2013)

- Target 값을 구할 때도 동일한 네트워크 사용
  - ✓ gradient descent 따라서 파라미터가 달라지기 때문에 동일한 입력을 넣어도 target 값이 달라질 수 있음 → 학습 불안정성



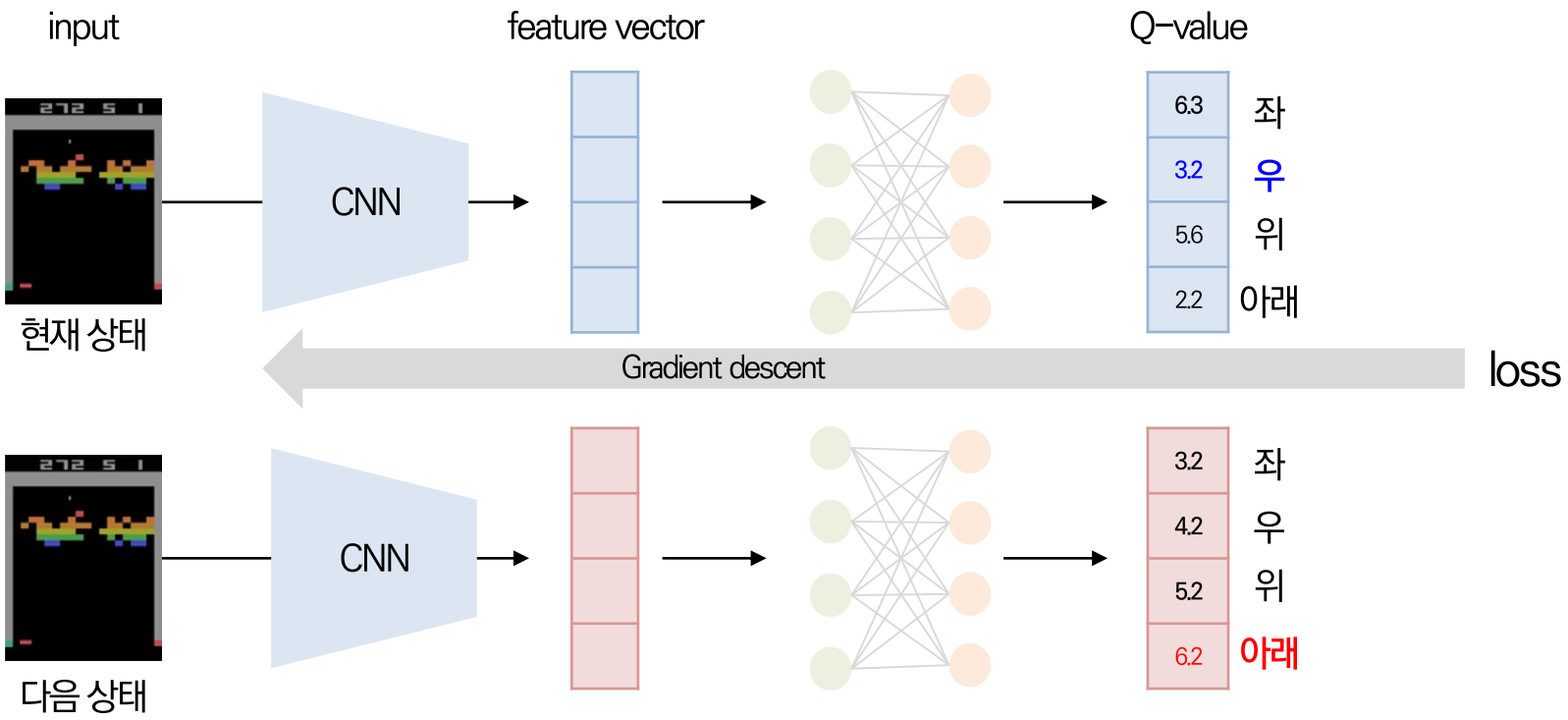
Deep Reinforcement Learning

Target: $r + \gamma \max_{a'} Q(s', a'; \theta)$

loss: $(r + \gamma \max Q(s', a'; \theta) - Q(s, a; \theta))^2$

❖ Deep Q-Network(DQN, 2013)

- Target 값을 구할 때도 동일한 네트워크 사용
  - ✓ gradient descent 따라서 파라미터가 달라지기 때문에 동일한 입력을 넣어도 target 값이 달라질 수 있음 → 학습 불안정성



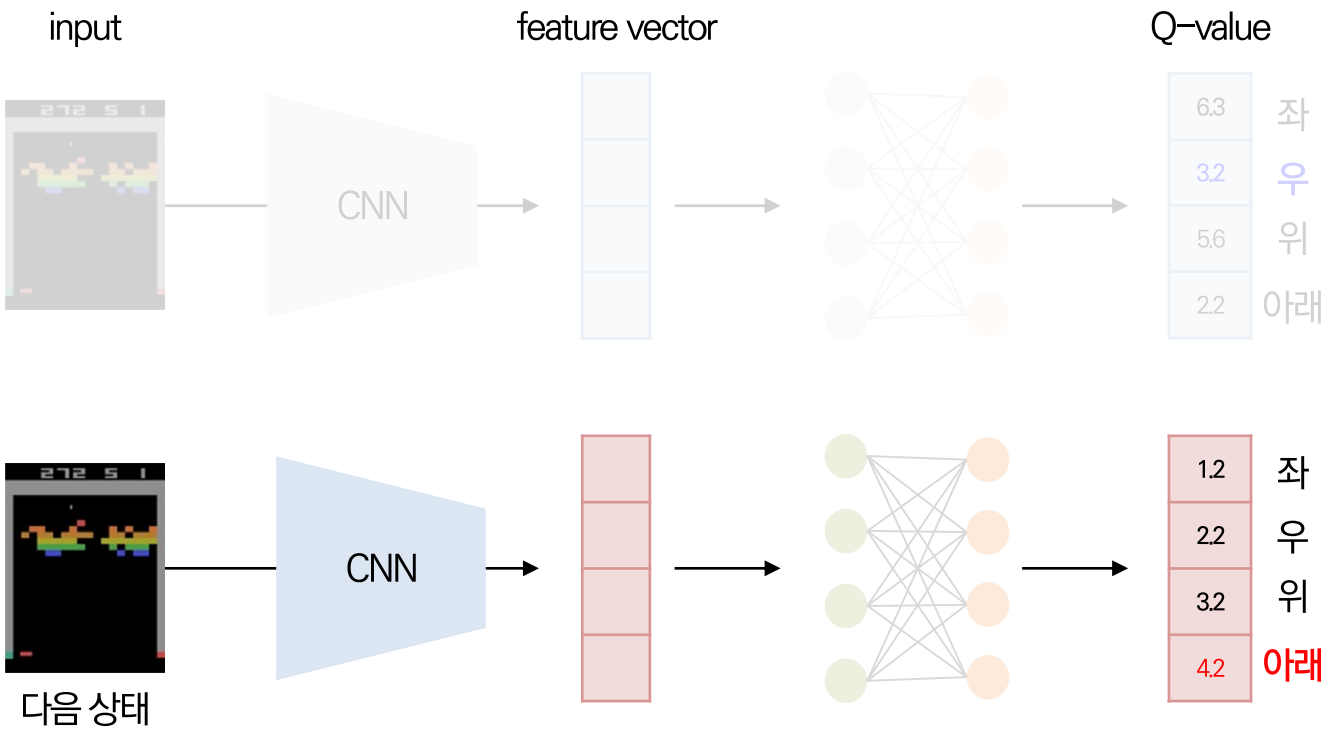
Deep Reinforcement Learning

Target: $r + \gamma \max_{a'} Q(s', a'; \theta)$

loss: $(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta))^2$

❖ Deep Q-Network(DQN, 2013)

- Target 값을 구할 때도 동일한 네트워크 사용
  - ✓ gradient descent 따라서 파라미터가 달라지기 때문에 동일한 입력을 넣어도 target 값이 달라질 수 있음 → 학습 불안정성



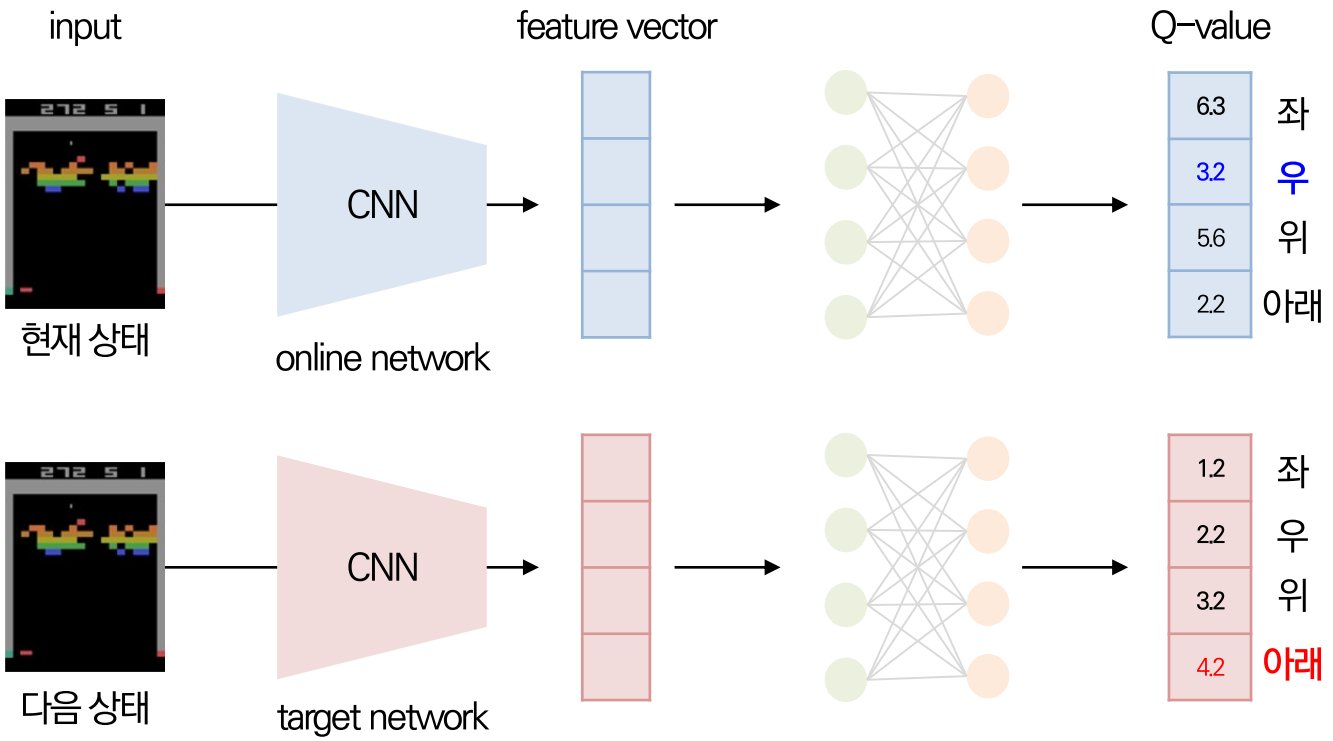
Deep Reinforcement Learning

Target: $r + \gamma \max_{a'} Q(s', a'; \theta^-)$

loss: $(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2$

❖ Deep Q-Network(DQN, 2015)

- DQN의 한계를 개선하기 위해 target network를 추가함
  - ✓ 처음에 초기화한 online network와 동일한 parameter를 가짐
  - ✓ 일정 c step 마다 gradient descent로 업데이트 한 online network의 parameter로 업데이트



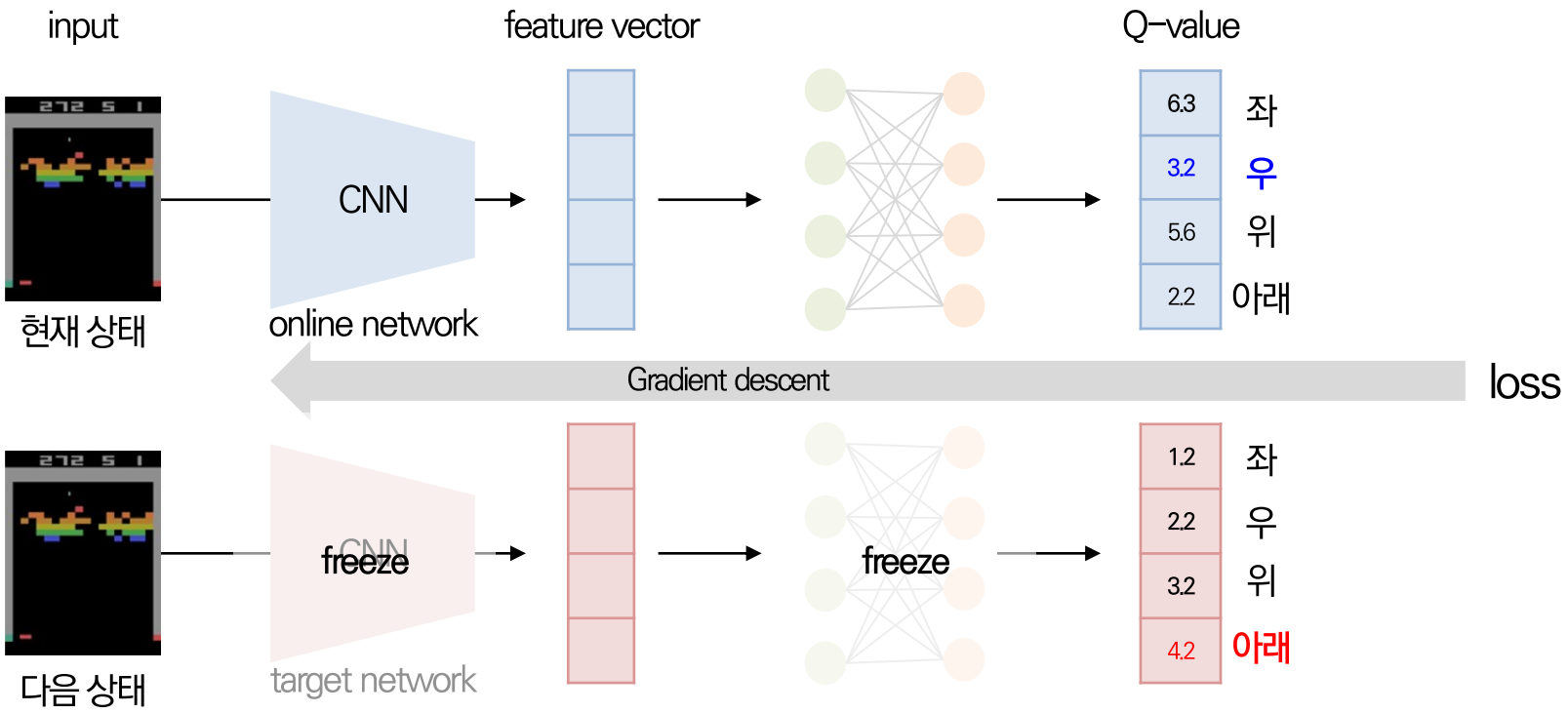
Deep Reinforcement Learning

Target: $r + \gamma \max_{a'} Q(s', a'; \theta^-)$

loss: $(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2$

❖ Deep Q-Network(DQN, 2015)

- DQN의 한계를 개선하기 위해 target network를 추가함
  - ✓ 처음에 초기화한 online network와 동일한 parameter를 가짐
  - ✓ 일정 c step 마다 gradient descent로 업데이트 한 online network의 parameter로 업데이트



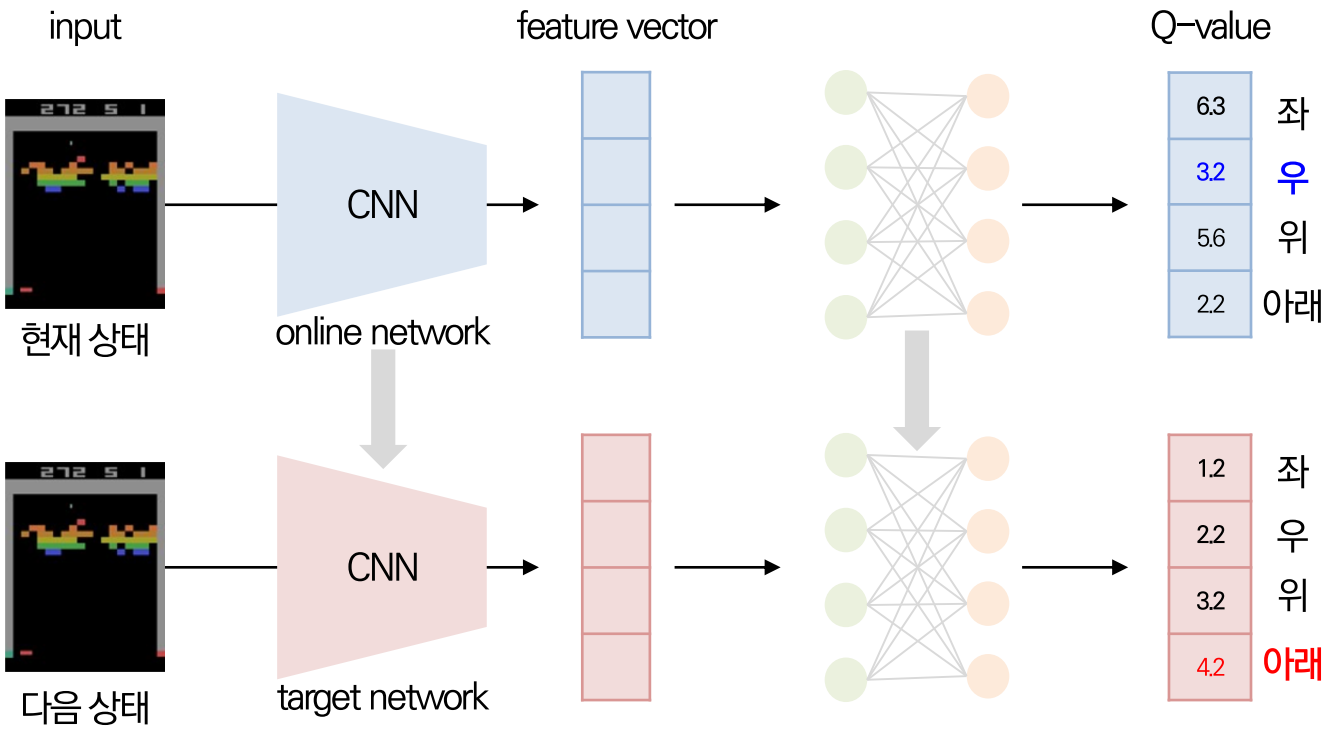
Deep Reinforcement Learning

Target: $r + \gamma \max_{a'} Q(s', a'; \theta^-)$

loss: $(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2$

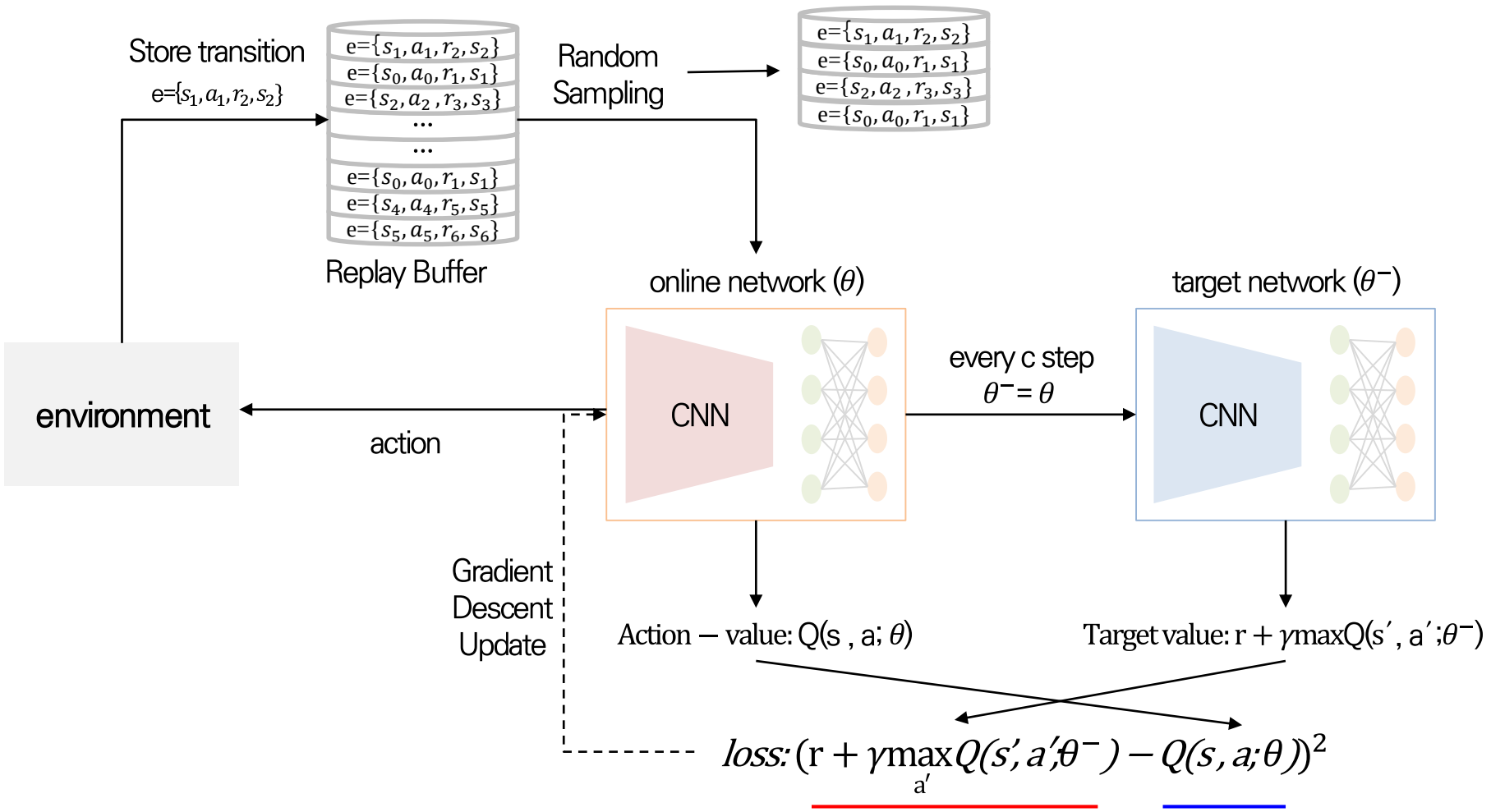
❖ Deep Q-Network(DQN, 2015)

- DQN의 한계를 개선하기 위해 target network를 추가함
  - ✓ 처음에 초기화한 online network와 동일한 parameter를 가짐
  - ✓ 일정 c step 마다 gradient descent로 업데이트 한 online network의 parameter로 업데이트



Deep Reinforcement Learning

❖ Deep Q-Network(DQN, 2015) overall framework



Deep Reinforcement Learning

❖ Deep Reinforcement Learning with Double Q-Learning (DDQN)

- Google DeepMind에서 연구했으며, 2015년 AAAI에서 발표된 방법론
- 기존 DQN의 target 값이 특정 조건에서 overestimate된다는 문제 발생
  - ✓ Target value를 추정할 때 Q-function을 사용하는데, 이때 수식상 max operator로 인해 overestimate된다고 주장
  - ✓ Overestimation 자체가 최종 학습된 정책에 부정적인 영향을 미침

Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Deep Reinforcement Learning with Double Q-Learning

Hado van Hasselt , Arthur Guez, and **David Silver**
Google DeepMind

Deep Reinforcement Learning

❖ Double Deep Q-Network(DDQN, 2015)

- 기존 DQN에서 max 연산자를 사용해 target 값을 추정하였을 때 overestimation 문제 발생
  - ✓ Target 값을 추정할 때 max 연산자는 Q-network가 쫓아가야 할 target을 최적 값보다 크게 추정하게 됨

$$y_t^{DQN} = r_t + \gamma \max_{a'} Q(s', a'; \theta^-)$$

Target Network에서 다음 상태에 대한 행동을 선택할 때와 행동 가치를 추정할 때 같은 파라미터를 사용함
➔ Overoptimistic value estimate

Deep Reinforcement Learning

❖ Double Deep Q-Network(DDQN, 2015)

- 알고리즘 측면에서 target value 추정식에 max operator가 아닌 argmax operator를 사용 → overestimation 개선
- DDQN에서는 target value를 추정할 때 행동을 선택하고 평가하는 것을 분리시킴
  - ✓ Online network($\boldsymbol{\theta}$)를 사용해 Q-value를 최대화하는 행동을 선택
  - ✓ Target network($\boldsymbol{\theta^-}$)를 사용해 target value에서 사용하는 Q-value 추정

$$y_t^{DQN} = r_t + \gamma \max_{a'} Q(s', a'; \theta^-)$$

$$y_t^{DDQN} = r_t + \gamma Q(s', \mathit{argmax}_{a'} Q(s', a'; \theta); \theta^-)$$

Deep Reinforcement Learning

❖ Dueling Network Architectures for Deep Reinforcement Learning

- Google DeepMind에서 연구했으며, 2015년 ICML에서 발표한 방법론
- 기존 DQN의 네트워크에서 하나의 estimator를 두 개의 estimator로 분리한 것이 특징
  - ✓ 각 estimator는 상태 가치 함수(state-value function)와 상대에 의존적인 이득 함수(advantage function)을 추정
  - ✓ Advantage function (A(s,a)): 주어진 상태에서 특정 행동이 다른 행동보다 얼마나 더 좋은 지 보여주는 함수

Dueling Network Architectures for Deep Reinforcement Learning

---

Ziyu Wang ZIYU@GOOGLE.COM
Tom Schaul SCHAUL@GOOGLE.COM
Matteo Hessel MTTHSS@GOOGLE.COM
Hado van Hasselt HADO@GOOGLE.COM
Marc Lanctot LANCTOT@GOOGLE.COM
Nando de Freitas NANDODEFREITAS@GOOGLE.COM
Google DeepMind, London, UK

Deep Reinforcement Learning

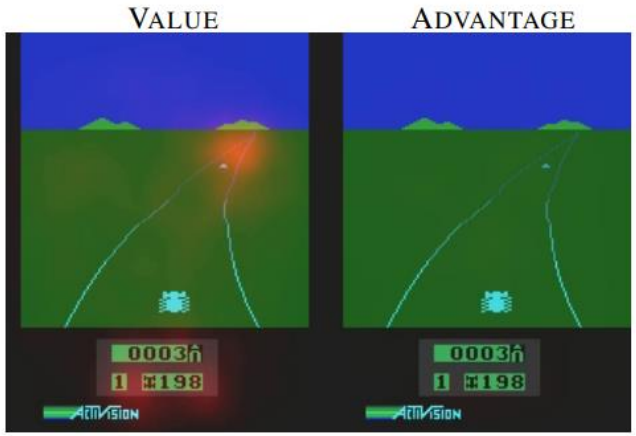
❖ Dueling Network Architectures for Deep Reinforcement Learning

- 본 연구에서 Dueling Network의 이점을 설명하기 위한 실험으로 Atari의 Enduro 게임을 활용함
  - ✓ 차를 좌,우로 움직이며 길을 따라 계속 가게 되면 점수가 쌓이고 차 또는 길에 부딪히면 점수가 쌓이는 속도가 줄어드는 게임

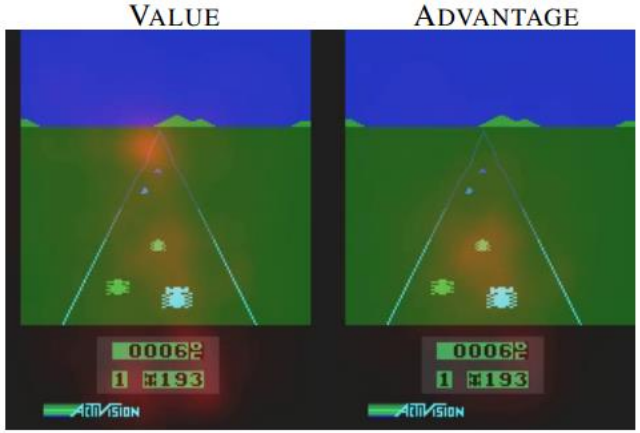
Deep Reinforcement Learning

❖ Dueling Network Architectures for Deep Reinforcement Learning

- 상태 가치 함수와 이득 함수에 관한 saliency map을 활용하여 Dueling Network의 이점을 설명한 실험
- 에이전트가 상태마다 행동을 해보면서 추정하는 Q-value를 학습하지 않더라도 어떤 상태가 가치 있는지 확인 가능
  - ✓ 기존에는 특정 상태에서 모든 행동에 대한 Q-value를 추정하면서 최적 정책을 탐색해야 함



전방에 장애물이 없는 상황



전방에 장애물이 있는 상황

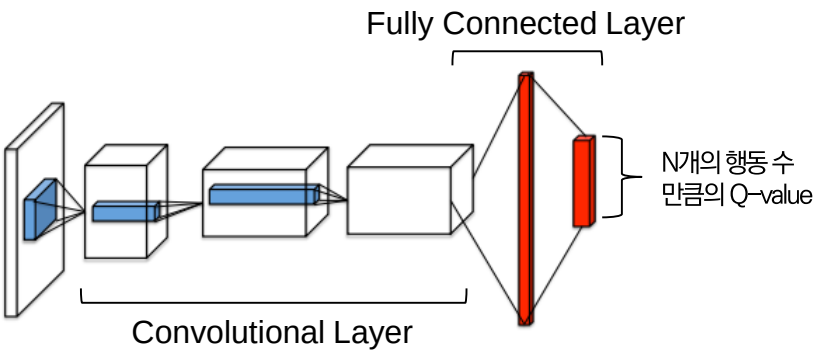
만약, 상태와 행동 둘 중 어느 하나라도 추정된 Q-value에 부정적인 영향을 준다면 해당 Q-value로 최적 정책을 탐색하는데 방해

Dueling Network는 정책을 평가하는 동안에 올바른 행동을 더 빠르게 확인할 수 있음

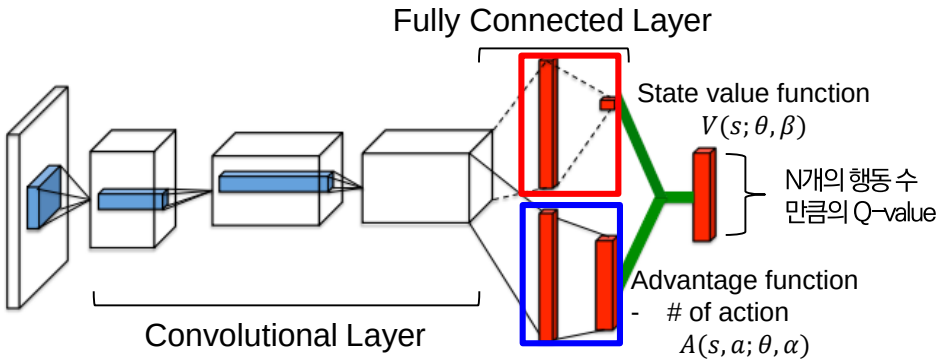
Deep Reinforcement Learning

❖ Dueling Network Architectures for Deep Reinforcement Learning

- DDQN에서의 target value 추정하는 학습 방법을 사용
- CNN을 통과해 나온 특징 벡터를 two stream을 통해 각각 상태 가치 함수와 이득 함수를 추정함
- 최종적으로, 추정된 두 값을 합함으로써 각 행동 개수만큼의 Q-value 추정



Single Stream Q-Network

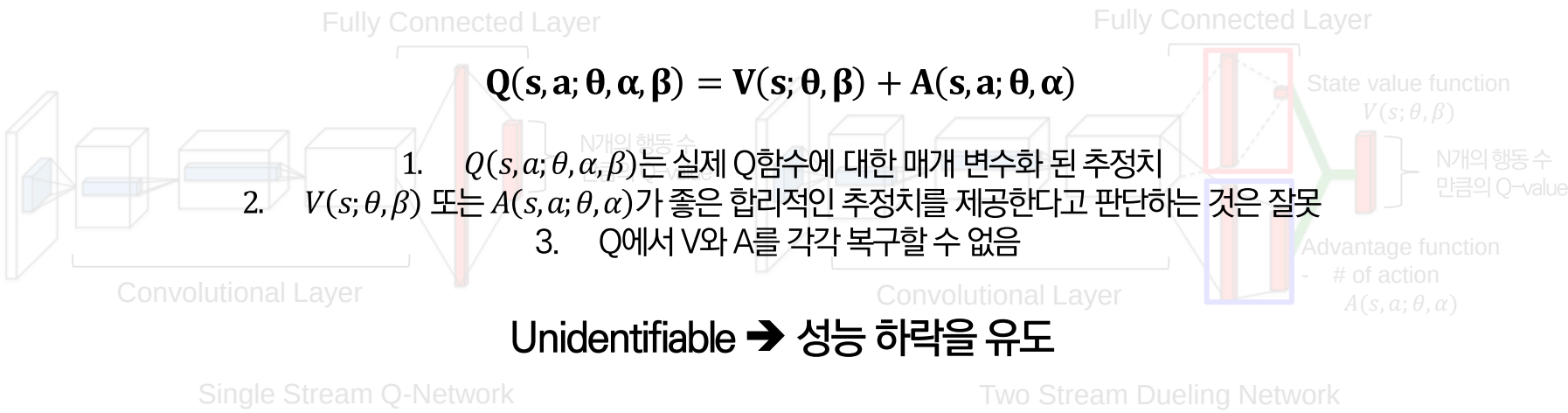


Two Stream Dueling Network

Deep Reinforcement Learning

❖ Dueling Network Architectures for Deep Reinforcement Learning

- DDQN에서의 target value 추정하는 학습 방법을 사용
- CNN을 통과해 나온 특징 벡터를 two stream을 통해 각각 상태 가치 함수와 이득 함수를 추정함
- 최종적으로, 추정된 두 값을 합함으로써 각 행동 개수만큼의 Q-value 추정



$$\mathbf{Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha)}$$

1. $Q(s,a;\theta,\alpha,\beta)$는 실제 Q함수에 대한 매개 변수화 된 추정치
2. $V(s;\theta,\beta)$ 또는 $A(s,a;\theta,\alpha)$가 좋은 합리적인 추정치를 제공한다고 판단하는 것은 잘못
3. Q에서 V와 A를 각각 복구할 수 없음

Unidentifiable ➔ 성능 하락을 유도

Deep Reinforcement Learning

❖ Dueling Network Architectures for Deep Reinforcement Learning

- 이러한 unidentifiable 문제를 개선하기 위한 기준점을 위한 방법 2가지 제시 → 이득 함수의 2가지 성질 사용

① $\boldsymbol{E_{a\sim\pi(s)}[A^{\pi}(s,a)] = 0}$

$Q^{\pi}(s,a) = V^{\pi}(s) + A^{\pi}(s,a)$ ← 우리가 얻고자 하는 값 Q

$\boldsymbol{V^{\pi}(s) = E_{a\sim\pi(s)}[Q^{\pi}(s,a)]}$ ← 상태 가치 함수와 행동 가치 함수의 관계

$\boldsymbol{E_{a\sim\pi(s)}}[Q^{\pi}(s,a)] = \boldsymbol{E_{a\sim\pi(s)}}[V^{\pi}(s)] + \boldsymbol{E_{a\sim\pi(s)}}[A^{\pi}(s,a)]$

$\boldsymbol{E_{a\sim\pi(s)}}[Q^{\pi}(s,a)] = \boldsymbol{E_{a\sim\pi(s)}}[V^{\pi}(s) + A^{\pi}(s,a)]$

$\boldsymbol{E_{a\sim\pi(s)}}[Q^{\pi}(s,a)] = V^{\pi}(s) + \boldsymbol{E_{a\sim\pi(s)}}[A^{\pi}(s,a)]$

$V^{\pi}(s) = V^{\pi}(s) + \boldsymbol{E_{a\sim\pi(s)}}[A^{\pi}(s,a)]$

$\boldsymbol{E_{a\sim\pi(s)}}[A^{\pi}(s,a)] = 0$

② $\boldsymbol{A(s,a^{*}) = 0}$

If, $Q^{\pi}(s,a)$이 가장 큰 값을 도출하는 행동만을 선택하는 정책 → Deterministic Policy

$a^{*} = argmax_{a'\in A}Q(s,a')$

$\boldsymbol{V^{\pi}(s) = E_{a\sim\pi(s)}[Q^{\pi}(s,a)] == V(s) = Q(s,a')}$

$Q^{\pi}(s,a) = V^{\pi}(s) + A^{\pi}(s,a)$

$Q(s,a') - V(s) = A(s,a')$

$\boldsymbol{A(s,a') = 0}$

Deep Reinforcement Learning

❖ Dueling Network Architectures for Deep Reinforcement Learning

- 이러한 unidentifiable 문제를 개선하기 위한 기준점을 위한 방법 2가지 제시 → 이득 함수의 2가지 성질 사용

① $\boldsymbol{E_{a\sim\pi(s)}[A^{\pi}(s,a)]=0}$

$$Q(s,a;\theta,\alpha,\beta)=V(s;\theta,\beta)+A(s,a;\theta,\alpha)$$

$$Q(s,a;\theta,\alpha,\beta)=V(s;\theta,\beta)+\underbrace{\left(A(s,a;\theta,\alpha)-\frac{1}{|A|}\sum_{a'}A(s,a';\theta,\alpha)\right)}_{=0}$$

② $\boldsymbol{A(s,a^{*})=0}$

$$Q(s,a;\theta,\alpha,\beta)=V(s;\theta,\beta)+A(s,a;\theta,\alpha)$$

↓

$$Q(s,a;\theta,\alpha,\beta)=V(s;\theta,\beta)+\underbrace{\left(A(s,a;\theta,\alpha)-max_{a'\in|A|}A(s,a';\theta,\alpha)\right)}_{=0\ (\text{Optimal action } a^{*} \text{ 선택했을 때})}$$

$\mathbf{A(s,a;\theta,\alpha)}$가 $\boldsymbol{\alpha}$로 파라미터화 되어 있기 때문에 Advantage function의 성질을 따르도록 변형

➔Unidentifiable 문제 해결

Deep Reinforcement Learning

❖ Prioritized Experience Replay (PER, 2016)

- Google DeepMind에서 연구했으며, 2016년 ICLR에서 발표한 방법론
- 기존 방법론에서는 Replay Buffer에 보관했던 transition을 random sampling 진행 → 중요도를 고려하지 않음
  - ✓ 중요한 transition이 더 빈번하게 추출되도록 각 transition에 중요도를 부여 → 더 효과적으로 학습 가능

Published as a conference paper at ICLR 2016

PRIORITIZED EXPERIENCE REPLAY

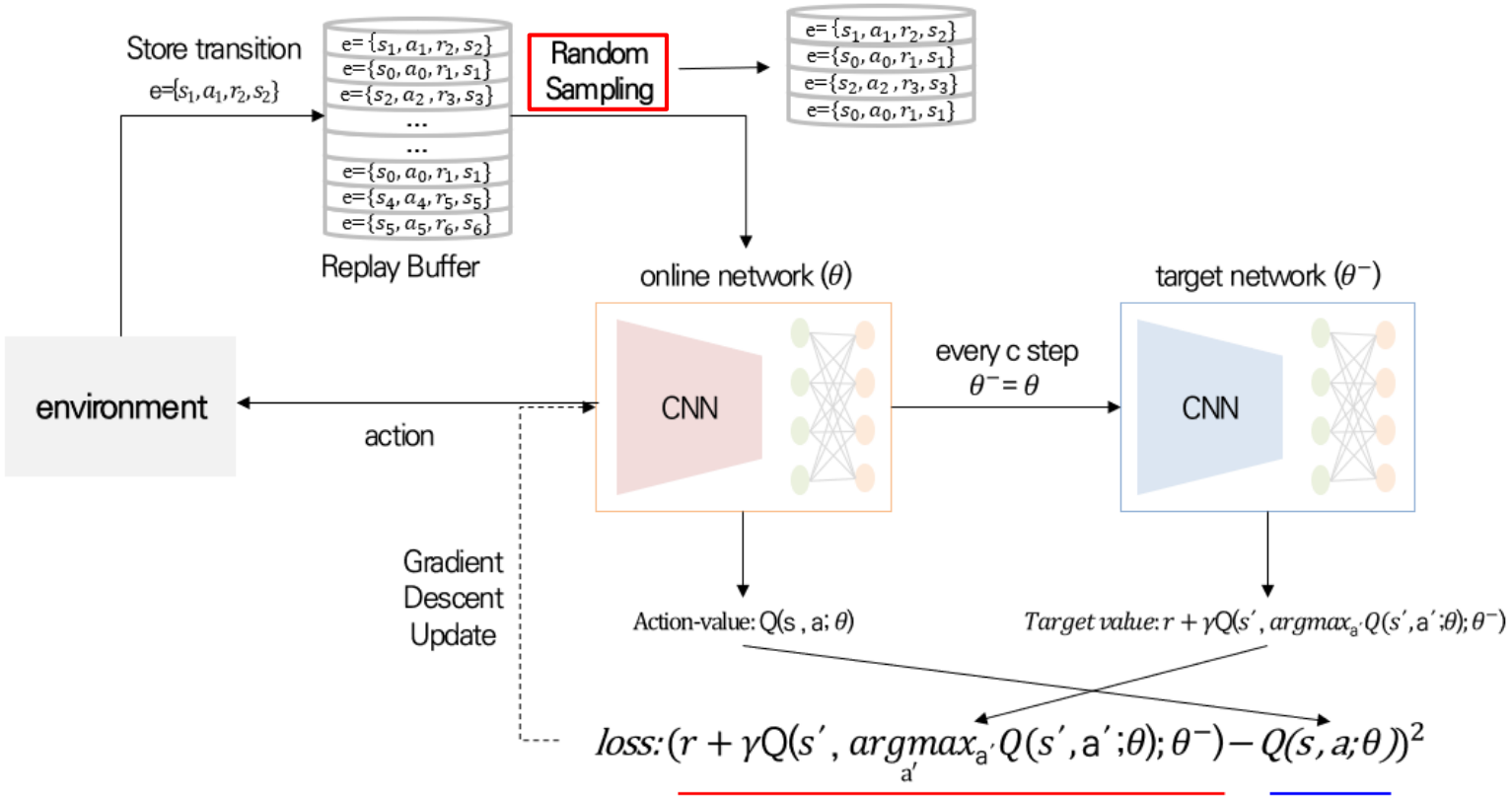
Tom Schaul, John Quan, Ioannis Antonoglou and David Silver
Google DeepMind
{schaul,johnquan,ioannisa,davidsilver}@google.com

Deep Reinforcement Learning

❖ Prioritized Experience Replay (PER, 2016)

- Deep Q-Network의 목표는 Deep Neural Network가 최적 Q-function에 근사하는 것

→ Target Network에서 추출된 target 값과 Online Network에서 추출된 Q-value 간에 차이가 0에 수렴하도록

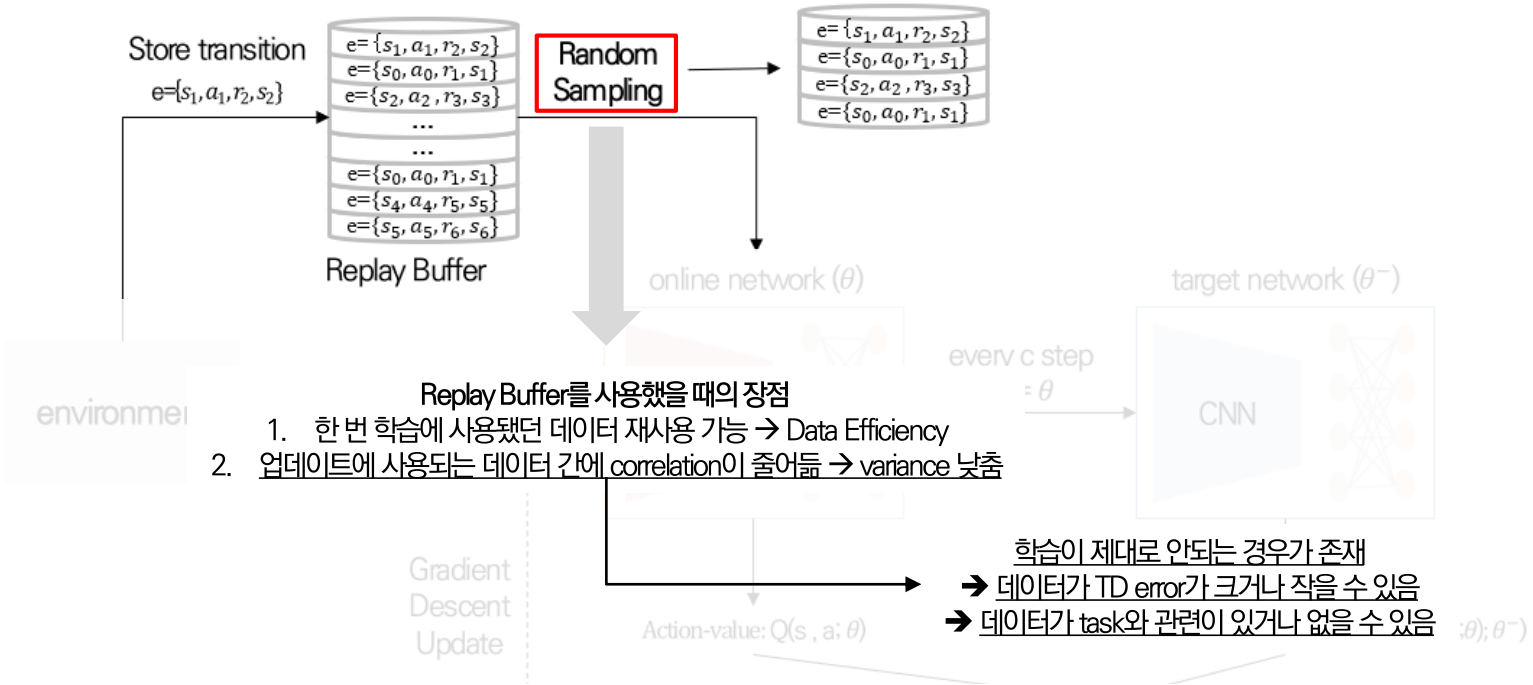


Deep Reinforcement Learning

❖ Prioritized Experience Replay (PER, 2016)

- Deep Q-Network의 목표는 Deep Neural Network가 최적 Q-function에 근사하는 것

  → Target Network에서 추출된 target 값과 Online Network에서 추출된 Q-value 간에 차이가 0에 수렴하도록



TD error가 큰 transition에 우선 순위를 부여하여 더 자주 학습에 사용되도록 하기 위한 방법 제시

→ Prioritizing with TD error

Deep Reinforcement Learning

❖ Prioritized Experience Replay (PER, 2016)

- Prioritizing with TD error: TD 에러가 큰 transition을 replay buffer로부터 우선적으로 샘플링하는 방법
  - ✓ Q-learning 업데이트 진행 시, TD 에러에 비례하도록 업데이트
  - ✓ 알려진 TD 에러가 없는 새로운 transition의 경우 maximal priority를 적용해 메모리에 적재
- 하지만, 해당 방법은 3 가지 문제가 존재
  1. 제일 큰 TD 에러를 지닌 transition을 찾기 위해 replay buffer 내에 모든 transition을 뒤지면서 발생하는 비용문제 &
     샘플링 된 transition에만 TD error가 업데이트 진행되는데 이 때 낮은 TD 에러를 지닌 transition은 오랫동안 사용되지 않을 수 있음
  2. 노이즈에 민감
  3. TD 에러가 높은 transition만 자주 샘플링 되면 극히 일부분에만 집중하는 문제 발생 → overfitting에 빠질 수 있음

Deep Reinforcement Learning

❖ Prioritized Experience Replay (PER, 2016)

- Prioritizing with TD error: TD 에러가 큰 transition을 replay buffer로부터 우선적으로 샘플링하는 방법
  - ✓ Q-learning 업데이트 진행 시, TD 에러에 비례하도록 업데이트
  - ✓ 알려진 TD 에러가 없는 새로운 transition의 경우 maximal priority를 적용해 메모리에 적재
- 하지만, 해당 방법은 3 가지 문제가 존재
  1. 제일 큰 TD 에러를 지닌 transition을 찾기 위해 replay buffer 내에 모든 transition을 뒤지면서 발생하는 비용문제 &
     샘플링 된 transition에만 TD error가 업데이트 진행되는데 이 때 낮은 TD 에러를 지닌 transition은 오랫동안 사용되지 않을 수 있음
  2. 노이즈에 민감
  3. TD 에러가 높은 transition만 자주 샘플링 되면 극히 일부분에만 집중하는 문제 발생 → overfitting에 빠질 수 있음

Stochastic Prioritization

Deep Reinforcement Learning

❖ Prioritized Experience Replay (PER, 2016)

- Stochastic Prioritization: 확률을 기반으로 우선 순위를 부여하는 샘플링하는 방법
  - ✓ Pure greed Prioritization과 Uniform Random Sampling을 보간한 방법

$$\underbrace{P(i)}_{\text{확률}} = \frac{p_i^{\alpha}}{\sum_k p_k^{\alpha}}$$

α가 0에 가까우면 Uniform Random Sampling
→ Prioritization 강도 조절해주는 파라미터

① Proportional prioritization

$$p_i = |\underbrace{\delta_i}_{\text{TD error}}| + \epsilon$$

TD error가 0일 때 0이 되지 않게 해주는 작은 상수 값

예시)

$p_0=12, p_1=10, p_2=2$

$P(0)=\frac{12}{24}, P(1)=\frac{10}{24}, P(2)=\frac{2}{24}$

② rank-based prioritization

$$p_i = \frac{1}{rank(i)}$$

Replay buffer 내에 TD error에 따라 정렬 했을 때 각 transition의 순위

예시)

$TD_0=12, TD_1=10, TD_2=2$

$p_0=1, p_2=2, p_3=3$

$P(0)=\frac{1}{6}, P(1)=\frac{2}{6}, P(2)=\frac{3}{6}$

Deep Reinforcement Learning

❖ Prioritized Experience Replay (PER, 2016)

- 기존 DQN에서는 Uniform Sampling(Random Sampling)을 통해 expectation에 대한 Uniform Distribution 보장

  → Correlation 완화

- 하지만, Stochastic Prioritization 방식은 정형화되어 있지 않기 때문에 bias 발생

  → Importance Sampling weights 추가 → Prioritized하게 뽑지만 Uniform을 따르게 해줌

Uniform Probability

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta}$$

Non Uniform Probability

1에 가까울 수록 Uniform distribution 따르는 것을 최대한 고려

본 논문에서는 β를 초기값에서 1까지 점진적으로 늘리면서 실험 진행

3. Conclusion

Conclusion

❖ Deep Reinforcement Learning

- Deep Q-Network (DQN)
  - ✓ 기존 Q-Learning의 한계점을 Deep Neural Network의 도입으로 개선함
  - ✓ 2013년도에서는 단일 Q-Network를 사용하여 gradient descent에 따라서 동일한 입력을 넣어도 값이 달라지는 학습 불안정성이 존재 → 2015년도에 Target Network를 추가하여 이를 안정시킴
- Double DQN (DDQN)
  - ✓ 기존 DQN이 가지는 Overoptimistic 문제를 개선하기 위해 기존 target value를 구하는 데 사용되었던 max 연산자 대신에 argmax를 사용하여 액션을 선택하고 평가하는 것을 분리함
- Dueling Network
  - ✓ 정책 탐색에 불필요한 상태 또는 행동 정보를 보는 것은 불필요하기 때문에 상태 가치 함수와 이득 함수를 도출하는 two stream Network를 도입하여 정책을 평가하는 동안에 올바른 행동을 더 빠르게 확인할 수 있게 됨
- Prioritized Experience Replay (PER)
  - ✓ 중요한 transition이 더 빈번하게 sampling되게 하기 위해서 우선 순위를 부여하여 sampling하는 방법론 제안

감사합니다.