Function Approximation & DQN

Slides from

1. 이웅원 외, 파이썬과 케라스로 배우는 강화학습, 주교재
2. 이웅원, 가깝고도 먼 DeepRL, PPT
3. David Silver, Reinforcement Learning, PPT

References

1. Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, MIT Press
2. 유튜브, 전민영, 노승은, 강화학습의 기초 이론, 팡요랩

Contents

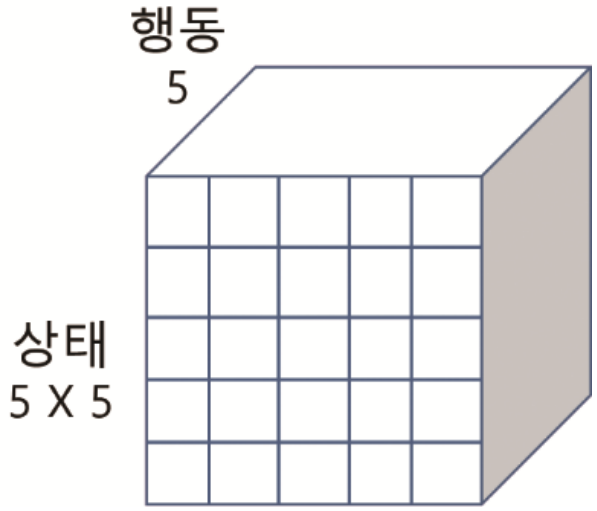
1. 강화학습이 풀고자 하는 문제 : Sequential Decision Problem
2. 문제에 대한 수학적 정의 : MDP & Bellman Equation
3. MDP를 계산으로 푸는 방법 : Dynamic Programming
4. MDP를 학습으로 푸는 방법 : Reinforcement Learning
5. 상태공간이 크고 차원이 높을 때 쓰는 방법 : Function Approximation & DQN
6. 인공신경망으로 정책을 근사하는 방법: Policy Gradient & Actor-Critic

고전 강화학습의 한계

- Tabular Solution Methods
  - 모든 상태의 큐함수를 테이블의 형태로 저장
  - 모든 상태의 큐함수를 방문할 때마다 하나씩 업데이트
  - 모든 상태를 충분히 방문해야 최적의 큐함수에 수렴

➢비효율적인 업데이트 방식

➢적용할 수 있는 문제의 제한
  - 간단한 문제
  - 환경이 변하지 않는 문제

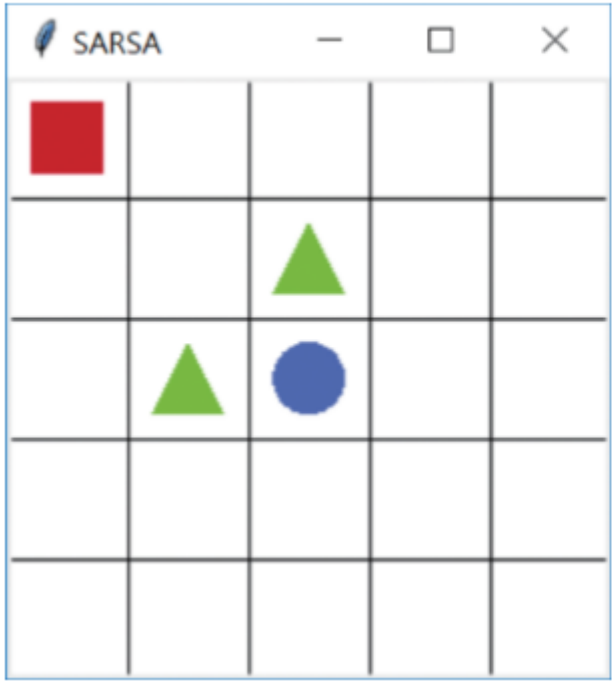


이웅원, 가깝고도 먼 DeepRL, PPT

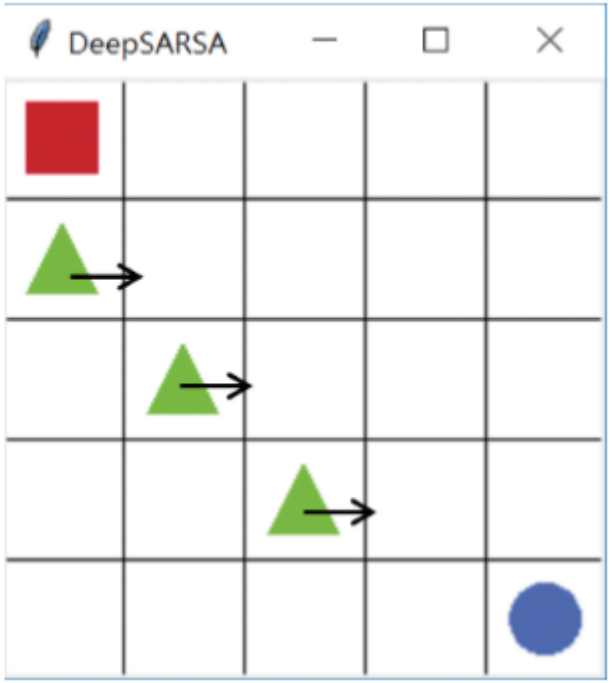
고전 강화학습 알고리즘의 한계

1. 환경이 변하지 않는 Grid world

상태 : 2차원, 25개
(1) 에이전트의 위치 (x, y)



2. 환경이 변하는 Grid world



상태 : 15차원, 18,225,000개
(1) 에이전트에 대한 도착지점의 상대 위치 (x, y)
(2) 도착지점의 레이블
(3) 에이전트에 대한 장애물의 상대 위치 (x, y)
(4) 장애물의 레이블
(5) 장애물의 속도(방향)

이웅원, 가깝고도 먼 DeepRL, PPT

고전 강화학습 알고리즘의 한계

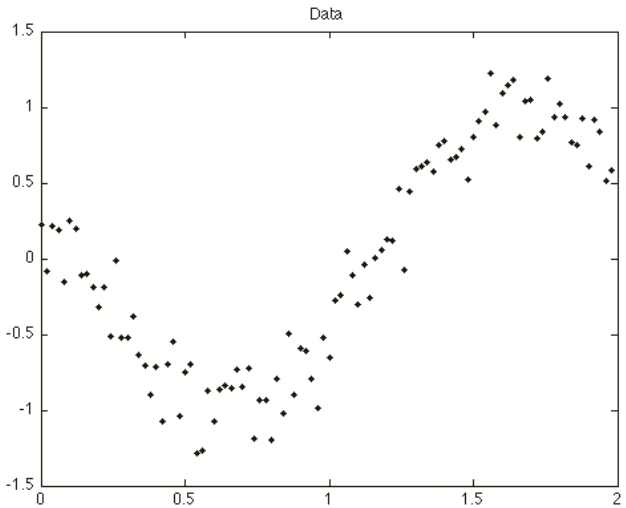
1. TD-gammon이 학습했던 Backgammon의 가능한 state 수
   - 10^{20}개
2. AlphaGo가 학습했던 바둑의 가능한 state 수
   - 10^{270}개
3. Possible Camera image?
4. Robot, Drone -> Continuous state space

근사함수를 통한 일반화

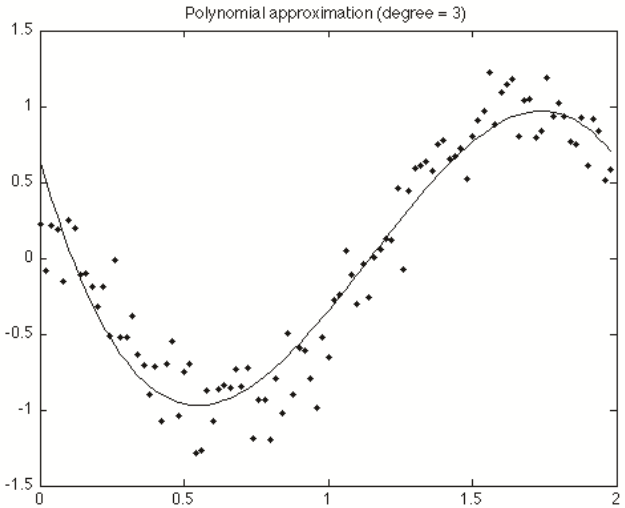
1. 대부분 강화학습을 적용하고 싶은 문제 -> Large state space
2. Large state space : 많은 양의 메모리, 계산량, 데이터 필요
3. 한정된 양의 자원 -> Good approximate solution
4. 비슷한 state는 비슷한 function의 output을 가질 것 -> Generalization!!
   - 어떻게 지금까지 방문한 state들에 대한 경험으로 전체 문제에 대해 generalize 할 수 있을까?
5. Generalization을 하기 위해 Supervised Learning의 기법을 가져다 쓰자!
   - Function Approximation : Target function 이 있고 그 target function 을 approximate 하는 function을 찾기

이웅원, 가깝고도 먼 DeepRL, PPT

매개변수로 근사



$y = ax^3+bx^2+cx+d$



좌측의 데이터를 3차 함수로 근사

- 4개의 매개변수(a,b,c,d)만으로 좌측의 데이터를 대체
- Function Approximation!

강화학습과 Function approximation

Q-Learning에서

1. Target function : 큐함수
2. 방문한 상태들에 대한 경험 -> 다른 상태들의 큐함수를 generalize
3. Approximate하는 함수의 parameter : θ

$$q_\theta(s,a) \sim q_\pi(s,a)$$

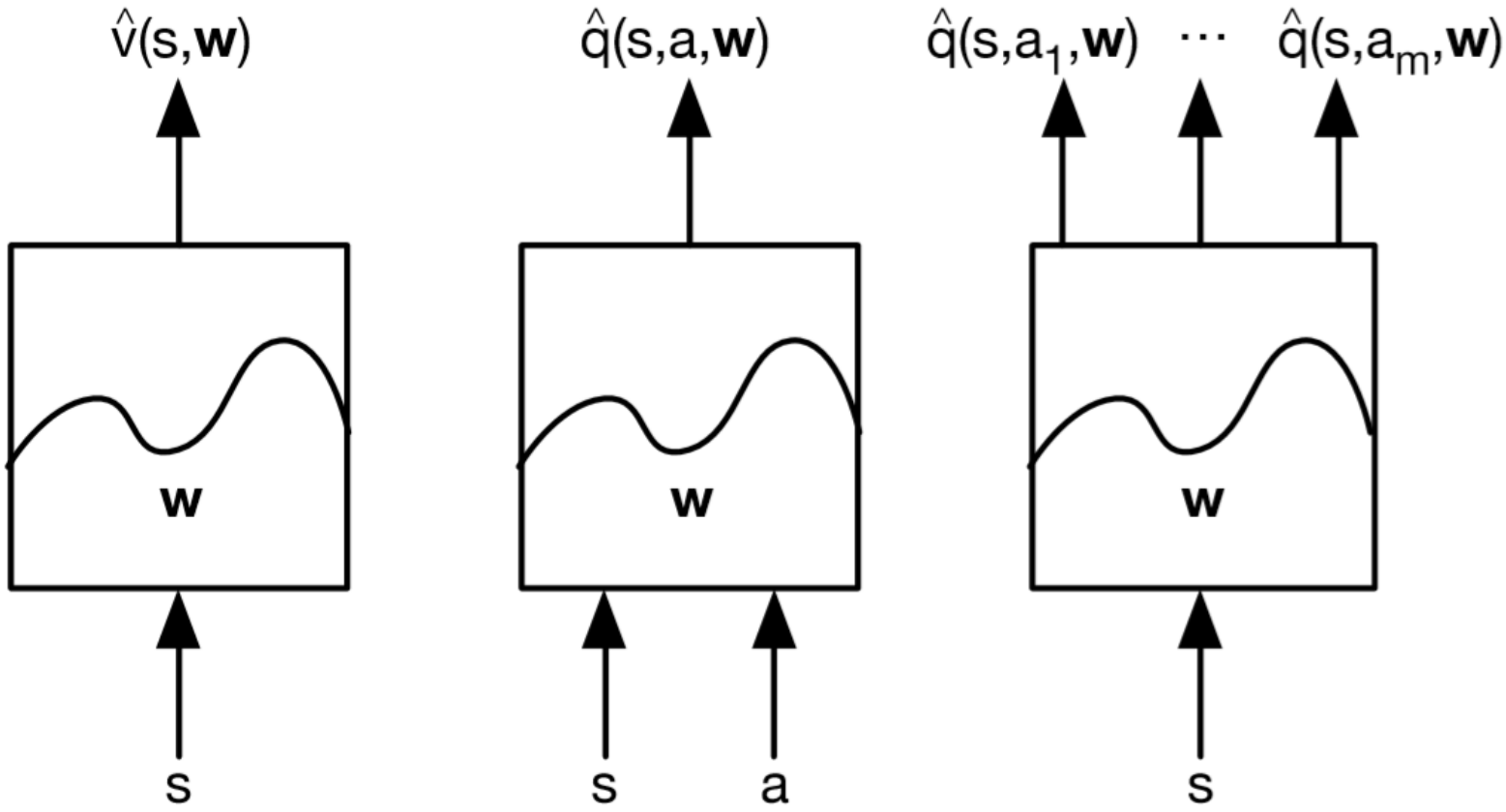
강화학습과 Function approximation

1. Table 형태의 큐함수

A \ S	s_0	s_1	s_2	s_3	s_4	…
a_0	$q(s_0, a_0)$	$q(s_1, a_0)$	$q(s_2, a_0)$	$q(s_3, a_0)$	$q(s_4, a_0)$	…
a_1	$q(s_0, a_1)$	$q(s_1, a_1)$	$q(s_2, a_1)$	$q(s_3, a_1)$	$q(s_4, a_1)$	…
a_2	$q(s_0, a_2)$	$q(s_1, a_2)$	$q(s_2, a_2)$	$q(s_3, a_2)$	$q(s_4, a_2)$	…
a_3	$q(s_0, a_3)$	$q(s_1, a_3)$	$q(s_2, a_3)$	$q(s_3, a_3)$	$q(s_4, a_3)$	…

이웅원, 가깝고도 먼 DeepRL, PPT

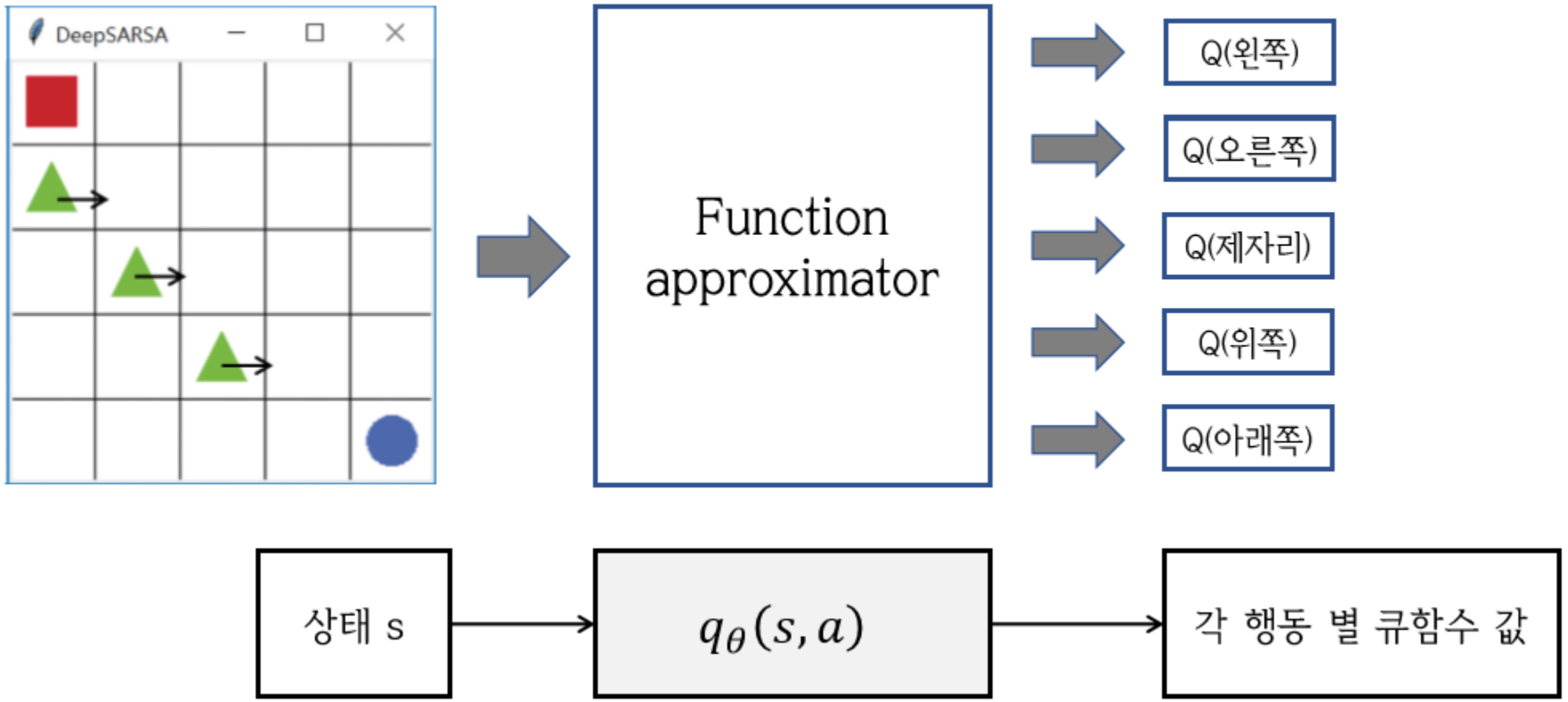
Types of Value Function Approximation



David Silver, Reinforcement Learning, PPT

강화학습과 Function approximation

2. 함수 형태로 근사한 큐함수



이웅원, 가깝고도 먼 DeepRL, PPT

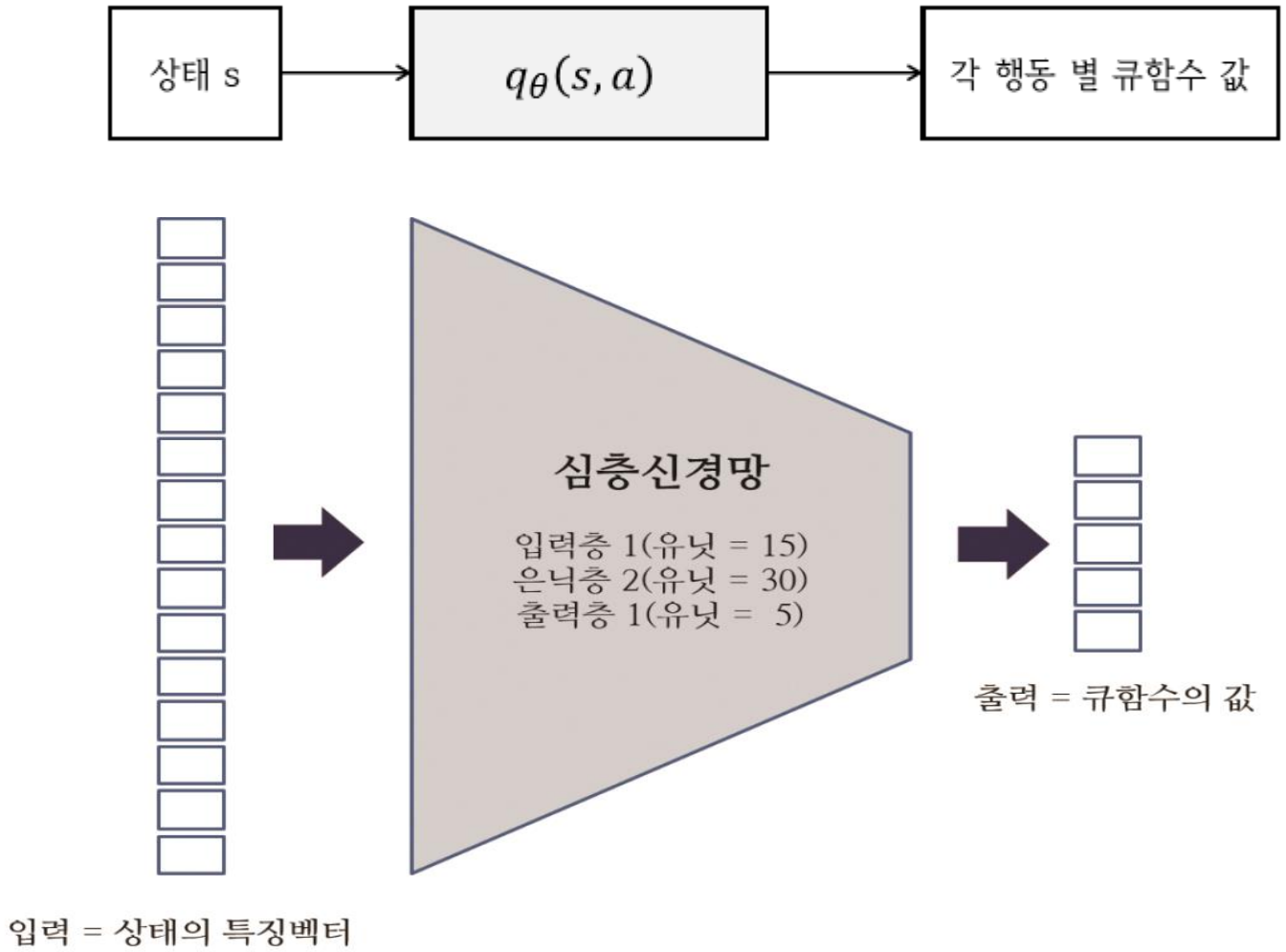
Which Function Approximator?

There are many function approximators, e.g.

- Linear combinations of features
- Neural network
- Decision tree
- Nearest neighbour
- Fourier / wavelet bases
- ...

David Silver, Reinforcement Learning, PPT

근사함수로 인공신경망을 이용



인공신경망의 학습

- Function approximation으로 인공신경망을 사용하므로 지도 학습의 기법들을 가져올 수 있음
- 지도학습의 학습 데이터

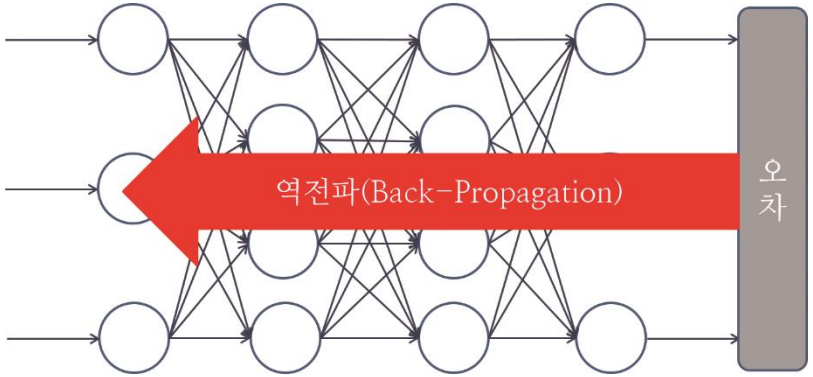
- 인공신경망의 출력 : 큐함수에 대한 예측
  - ➢큐함수의 값은 continuous -> Regression
- 오차함수 (Loss Function)
  - ➢Mean square error : 오차 = $(정답 - 예측)^2$
- 인공신경망의 학습
  - ➢오차함수를 최소화 하도록 인공신경망의 파라미터를 업데이트

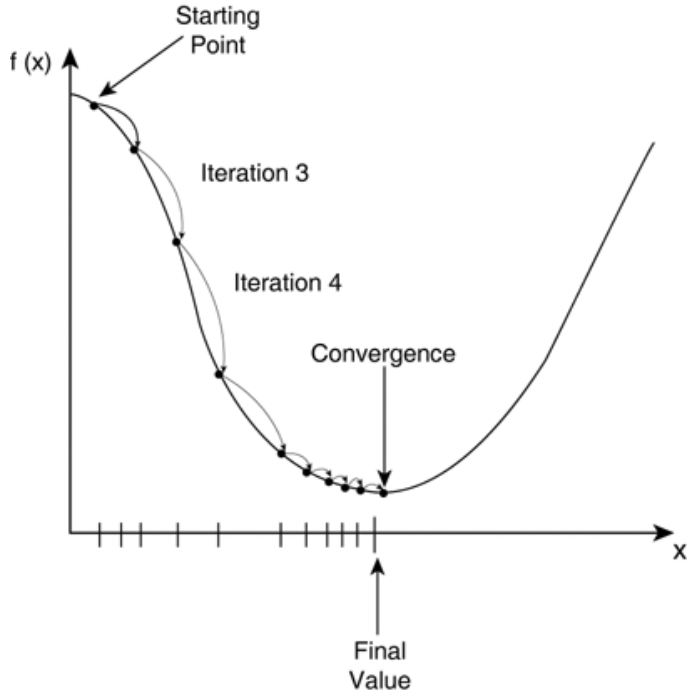
인공신경망의 학습

- 업데이트 대상 : 각 큐함수의 값 -> 큐함수의 parameter 값

$$q_\theta(s,a) \text{ -> } \theta$$

- 업데이트 방법 : Back-propagation & Gradient descent
  - ➢업데이트 값 ∝ 오차 × 오차기여도





강화학습과 Neural Network

1. Nonlinear function approximation -> Neural Network
   - Neural Network의 activation function이 nonlinear
2. Neural Network를 이용한 큐함수 근사
   - $q_\theta(s,a)$ 의 θ가 Neural Network의 parameter
3. 큐함수의 업데이트
   - MSE error에 대한 gradient
     - SARSA : $\nabla_\theta\left(r+\gamma q_\theta(s',a')-q_\theta(s,a)\right)^2$
     - Q-Learning : $\nabla_\theta\left(r+\gamma\max_{a'} q_\theta(s',a')-q_\theta(s,a)\right)^2$
   - 계산한 gradient를 backpropagation

이웅원, 가깝고도 먼 DeepRL, PPT

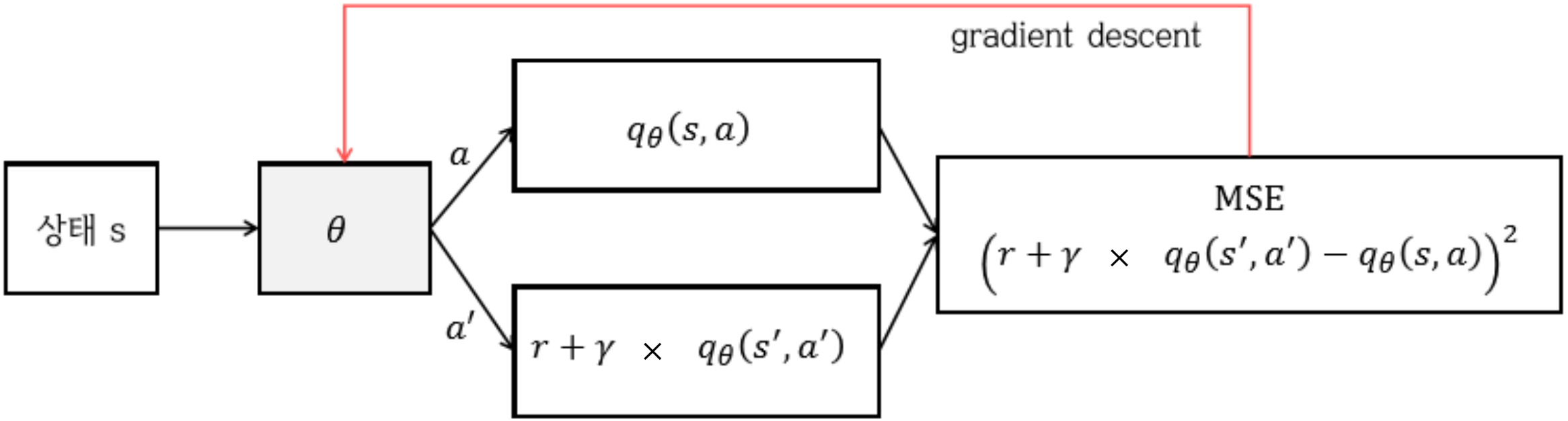
SARSA with function approximation

- SARSA의 큐함수 업데이트 식

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

- 경사하강법
  - 정답 역할 : $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$
  - 예측 역할 : $Q(S_t, A_t)$
  - MSE = (정답 − 예측)^2 = $(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))^2$

SARSA with function approximation



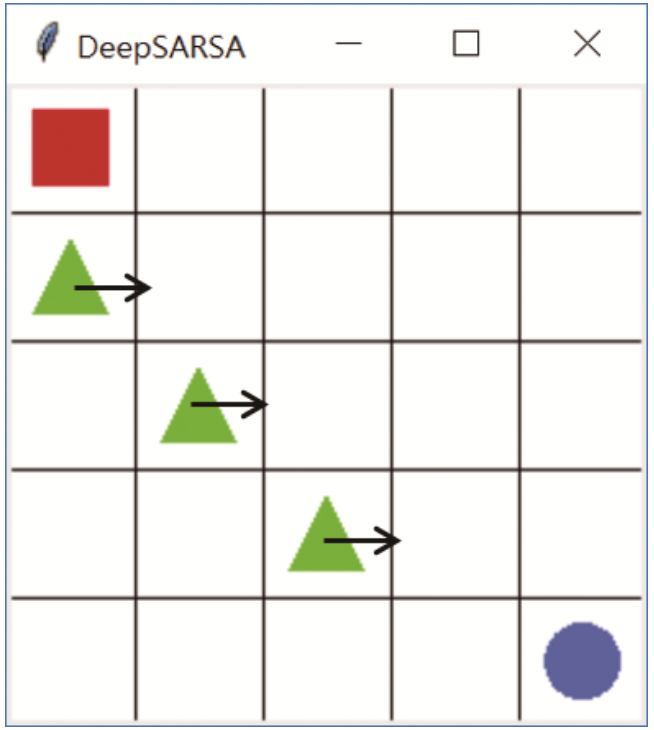
이웅원, 가깝고도 먼 DeepRL, PPT

SARSA with function approximation

- 신경망을 이용한 SARSA 학습 과정

1. 상태 관찰
2. 신경망을 이용하여 행동 선택 -> $q_\theta(s, a)$
3. 행동하고 다음 상태와 보상을 받음 -> $r + \gamma q_\theta(s', a')$
4. TD error의 gradient를 따라 큐함수의 parameter를 업데이트
   -> $\left(r + \gamma q_\theta(s', a') - q_\theta(s, a)\right)^2$

Deep SARSA

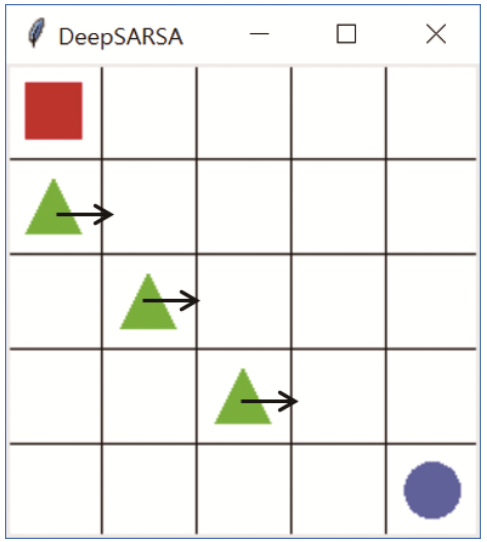


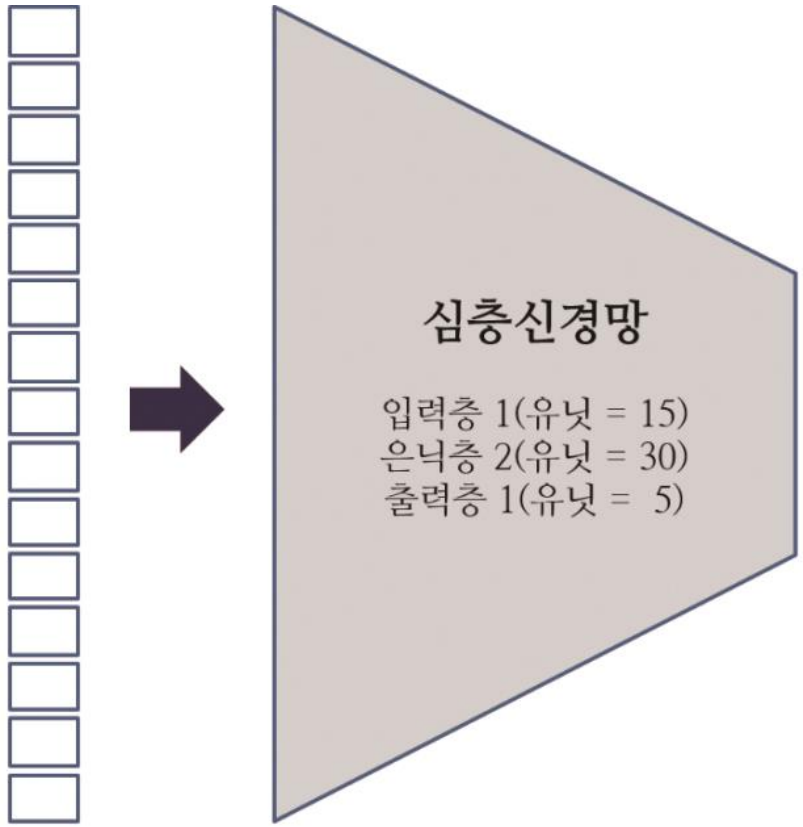
MDP 상태 정의

(1) 에이전트에 대한 도착지점의 상대 위치 (x, y)
(2) 도착지점의 레이블
(3) 에이전트에 대한 장애물의 상대 위치 (x, y)
(4) 장애물의 레이블
(5) 장애물의 속도(방향)

➢ (1)+(2)+3*{(3)+(4)+(5)} = 15 특징벡터

Deep SARSA





입력 = 상태의 특징벡터

출력 = 큐함수의 값

Q(왼쪽)

Q(오른쪽)

Q(제자리)

Q(위쪽)

Q(아래쪽)

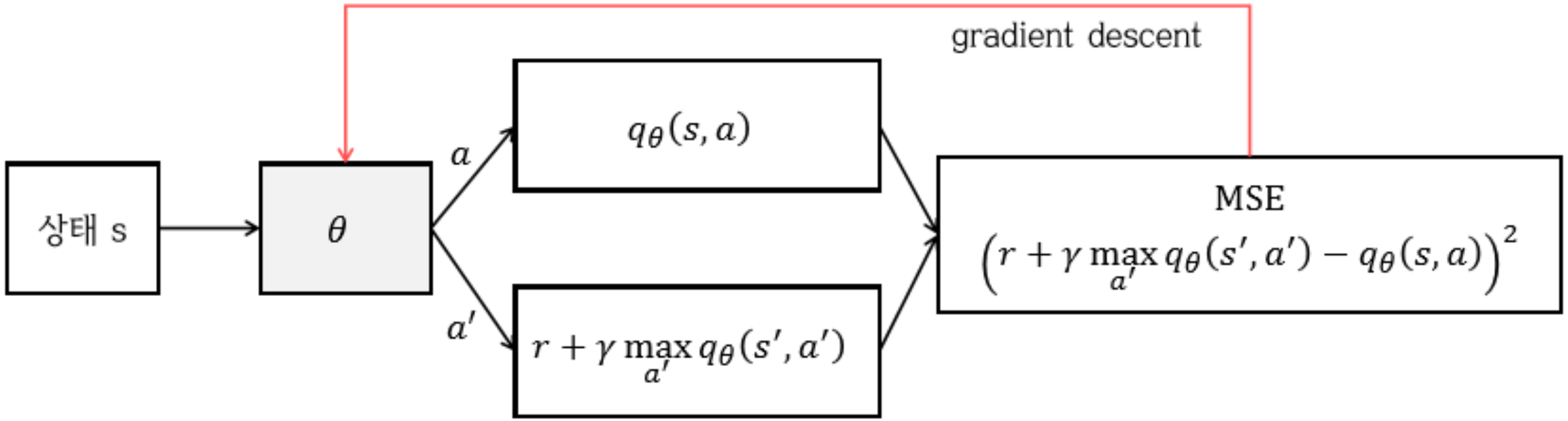
Q-Learning with function approximation

- Q-Learning에서 큐함수의 업데이트 식

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right)$$

- 경사하강법
  - 정답 역할 : $R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$
  - 예측 역할 : $Q(S_t, A_t)$
  - MSE = (정답 − 예측)^2 = $\left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right)^2$

Q-Learning with function approximation



이웅원, 가깝고도 먼 DeepRL, PPT

Q-Learning with function approximation

- 신경망을 이용한 Q-Learning 학습 과정

1. 상태 관찰
2. 신경망을 이용하여 행동 선택 -> $q_\theta(s, a)$
3. 행동하고 다음 상태와 보상을 받음 -> $r + \gamma \max_{a'} q_\theta(s', a')$
4. TD error의 gradient를 따라 큐함수의 parameter를 업데이트
   -> $\left(r + \gamma \max_{a'} q_\theta(s', a') - q_\theta(s, a)\right)^2$

Deep Q-Learning

- Deep Reinforcement Learning = Reinforcement Learning + Deep Learning

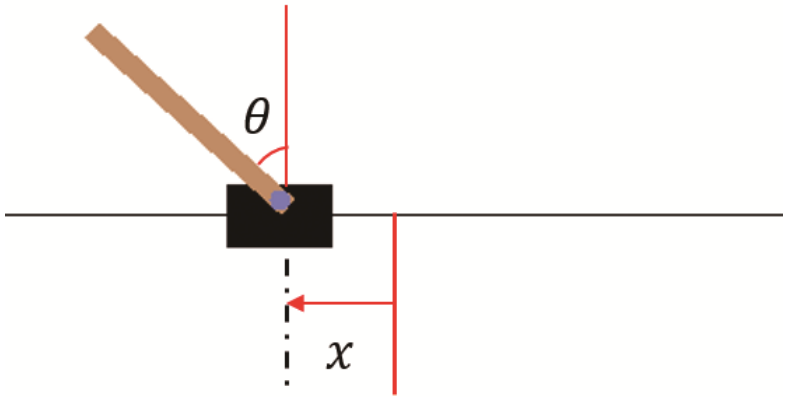
- DQN (Deep Q-Network)
  - “Playing Atari with Deep Reinforcement Learning”-Mnih, 2013
  - DeepMind의 초창기 논문
  - Atari game을 화면으로부터 학습
  - 화면은 high-dimension -> Deep Learning을 사용

- DQN의 핵심 개념
  - Experience Replay
  - Target Network

Cart-Pole

- 카트폴
  - 카트폴이 일정 각도 이상으로 떨어지거나 화면을 벗어나면 에피소드가 끝남
  - 학습목표 : 카트폴을 5초 동안 세우는 것

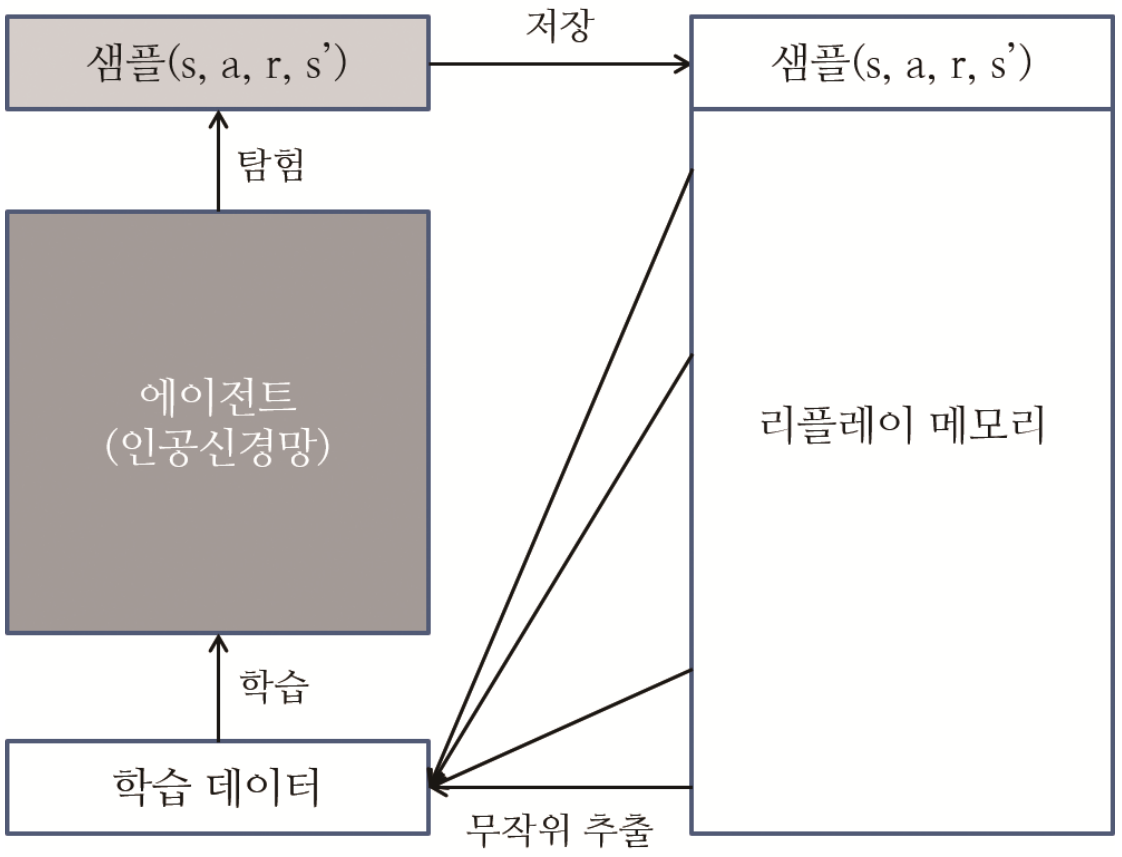
- MDP 상태 정의
  - 카트의 위치 x
  - 카트의 속도 $\dot{x}$
  - 폴의 각도 θ
  - 폴의 각속도 $\dot{\theta}$



$$\text{에이전트의 상태} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

DQN - Experience Replay

- Off-Policy
- Experience Replay
  - Sample 들의 상관관계를 깸 -> Neural Network의 안정적인 학습
  - 일정한 크기를 가지는 memory (FIFO)
- Online update with stochastic gradient descent
  - 매 스텝마다 replay memory에서 추출한 mini-batch로 Q-function 업데이트
  - 점진적으로 변하는 Q-function에 대해 $\varepsilon-greedy\ policy$로 행동 선택



이웅원, 가깝고도 먼 DeepRL, PPT

DQN - Target Network

- Q-Learning에서 큐함수의 업데이트

$$\text{MSE} = (\text{정답} - \text{예측})^2 = \left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a', \theta) - Q(S_t, A_t, \theta)\right)^2$$

- Target Network
  - Target network θ^-의 사용 : update의 target이 계속 변하는 문제를 개선
  - 일정주기마다 현재의 network θ^- 를 θ로 업데이트

$$\text{MSE} = (\text{정답} - \text{예측})^2 = \left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a', \theta^-) - Q(S_t, A_t, \theta)\right)^2$$

이웅원, 가깝고도 먼 DeepRL, PPT

DQN의 학습과정

1. exploration
2. append sample to replay memory
3. random sampling, training
4. target Q-network update

DQN의 학습과정

1. exploration
   - 정책은 큐함수에 대한 $\varepsilon-greedy\ policy$
   - ε은 time-step에 따라서 decay -> 점점 수렴
   - ε은 1에서 시작해서 0.1까지 decay, 0.1을 계속 유지 -> 지속적 탐험

2. append sample to replay memory
   - 에이전트는 $\varepsilon-greedy\ policy$에 따라 샘플 [s, a, r, s']을 생성
   - 샘플을 replay memory에 append
   - Replay memory가 다 차면 오래된 sample부터 하나씩 빼고 새로운 sample을 memory에 넣기

DQN의 학습과정

3. random sampling, training
   - Mini-batch (32개) 샘플을 추출
   - 샘플로부터 target값과 prediction 값을 구하기 (32개)
     - ❖$MSE\ error : (target - prediction)^2$
     - ❖Target : $r + \gamma \max_{a'} q_{\theta^-}(s', a')$
     - ❖Prediction : $q_\theta(s, a)$
   - MSE error에 대한 gradient backpropagation

4. target Q-network update : 일정 주기마다 target Q-network를 현재 Q-network로 업데이트

이웅원, 가깝고도 먼 DeepRL, PPT

DQN의 학습과정 정리

1) 상태에 따른 행동 선택
2) 선택한 행동으로 환경에서 한 타임스텝을 진행
3) 환경으로부터 다음 상태와 보상을 받음
4) 샘플(s,a,r,s')을 리플레이 메모리에 저장
5) 리플레이 메모리에서 무작위 추출한 샘플로 학습
6) 일정 주기마다 Target network 업데이트

Summary

- 매개변수로 큐함수를 근사
- 근사함수로 인공신경망을 이용
- Deep SARSA
  - MSE = (정답 – 예측)^2 = $(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))^2$
- DQN (Deep Q-Network)
  - Experience Replay
  - Target Network
  - MSE = (정답 – 예측)^2 = $\left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a', \theta^-) - Q(S_t, A_t, \theta)\right)^2$