

# RLCode와 함께하는 강화학습 실습

RLCode 리더 이용원

# 목차

---

1. 그리드월드
2. 큐러닝 알고리즘 설명
3. 그리드월드와 큐러닝
4. 딥살사 알고리즘 설명
5. 그리드월드와 딥살사

그리드월드

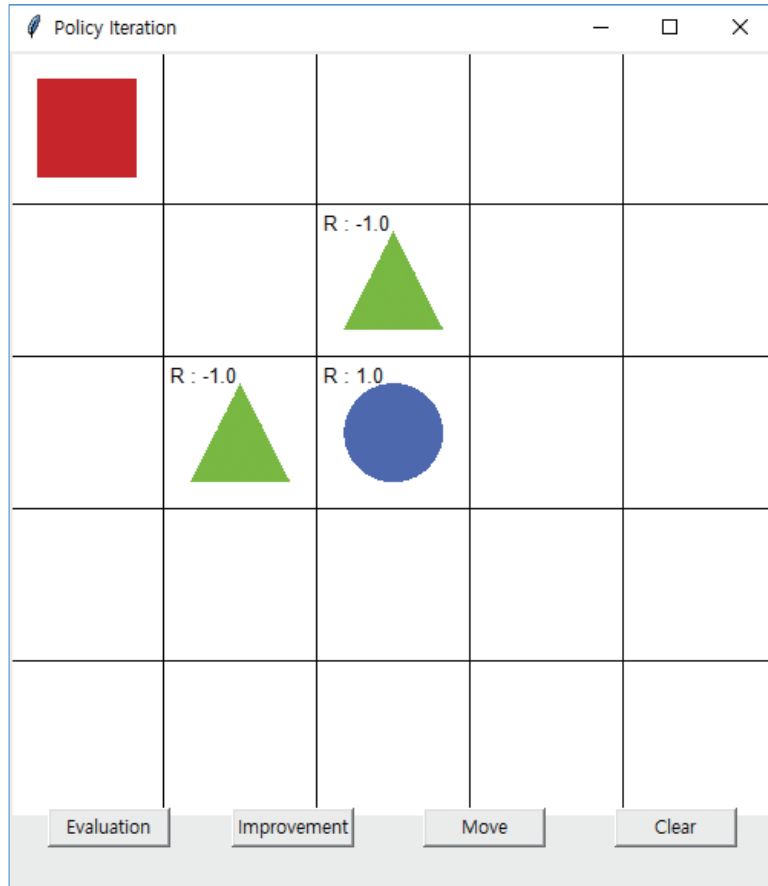
# 실습할 알고리즘

1. Q-Learning
2. DeepSARSA

# 실습할 환경

1. 그리드월드
2. 변형된 그리드월드

# 그리드월드 문제 정의



## 문제의 정의

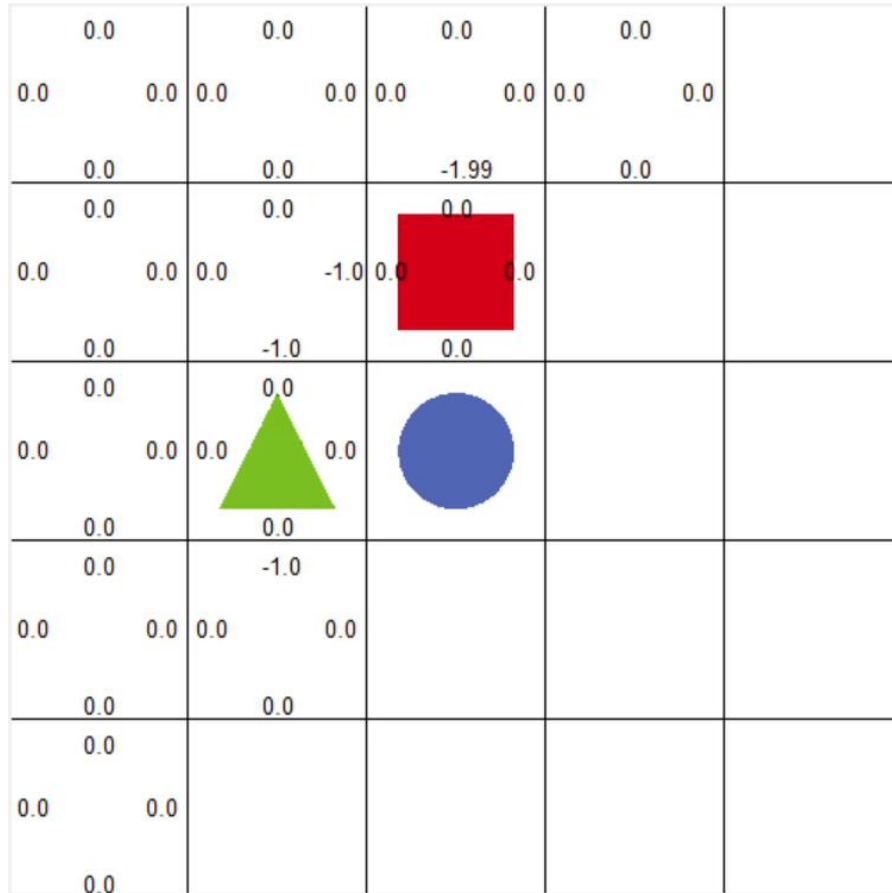
상태:  $[x, y]$  좌표

행동: 상, 하, 좌, 우, 제자리

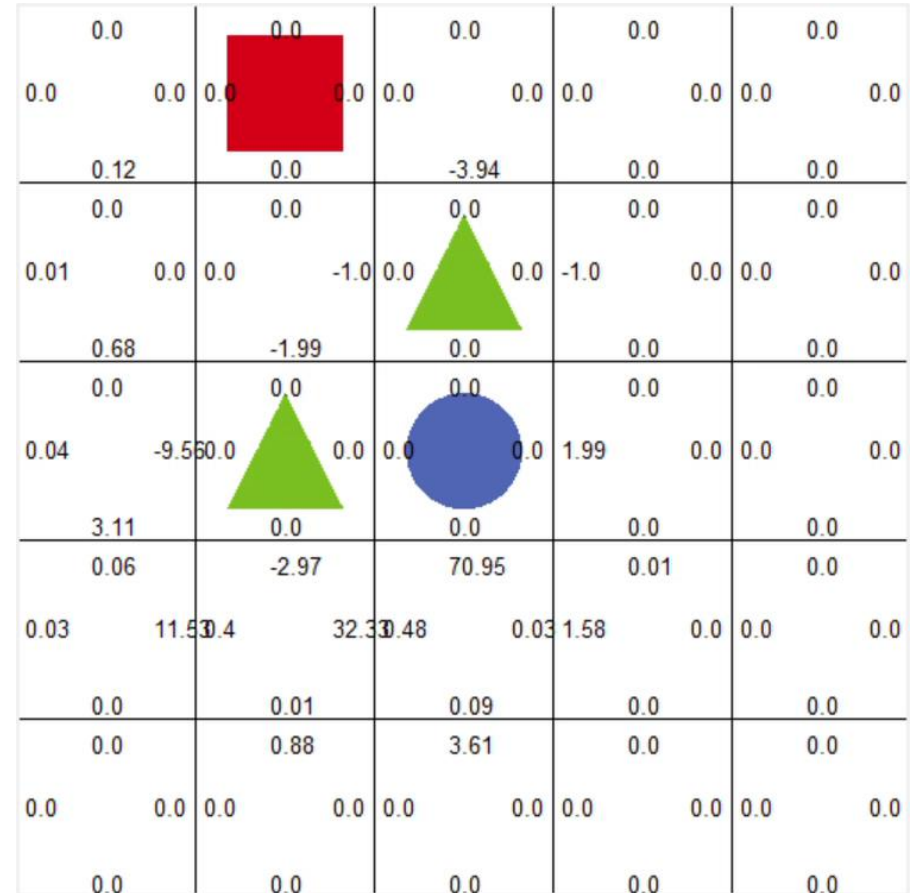
보상: 초록색 세모(-1), 파란색 동그라미(+1)

목표: 초록색 세모를 피해서 파란색 동그라미로 가기

# 큐러닝으로 학습한 에이전트

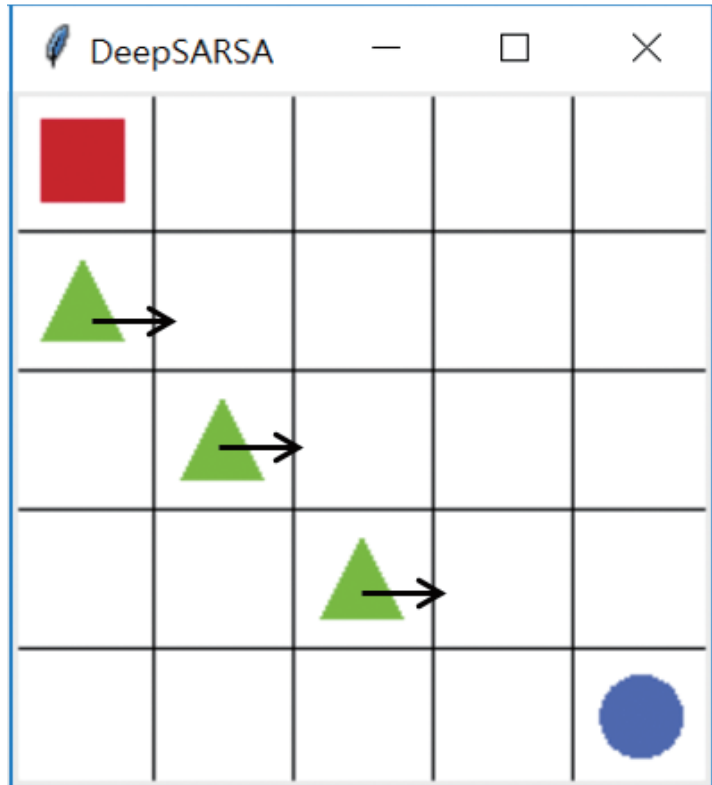


학습 중



학습 후

# 변형된 그리드월드 문제 정의



## 1. 상태의 정의

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

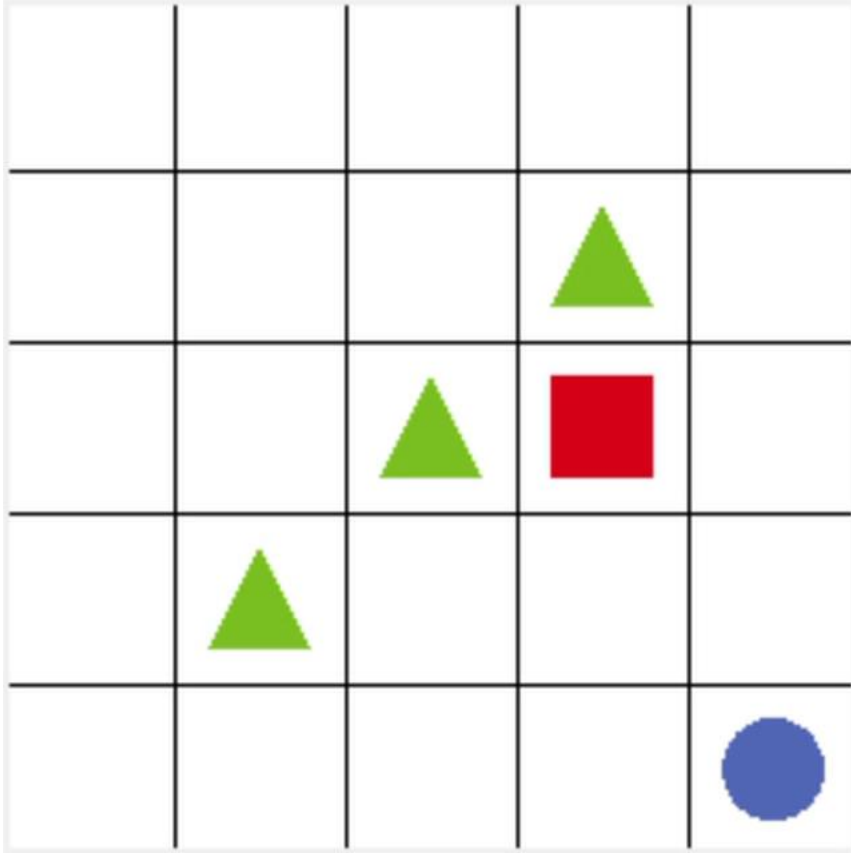
## 2. 행동: 상, 하, 좌, 우, 제자리

## 3. 보상: 초록색 세모(-1), 파란색 동그라미(+1)

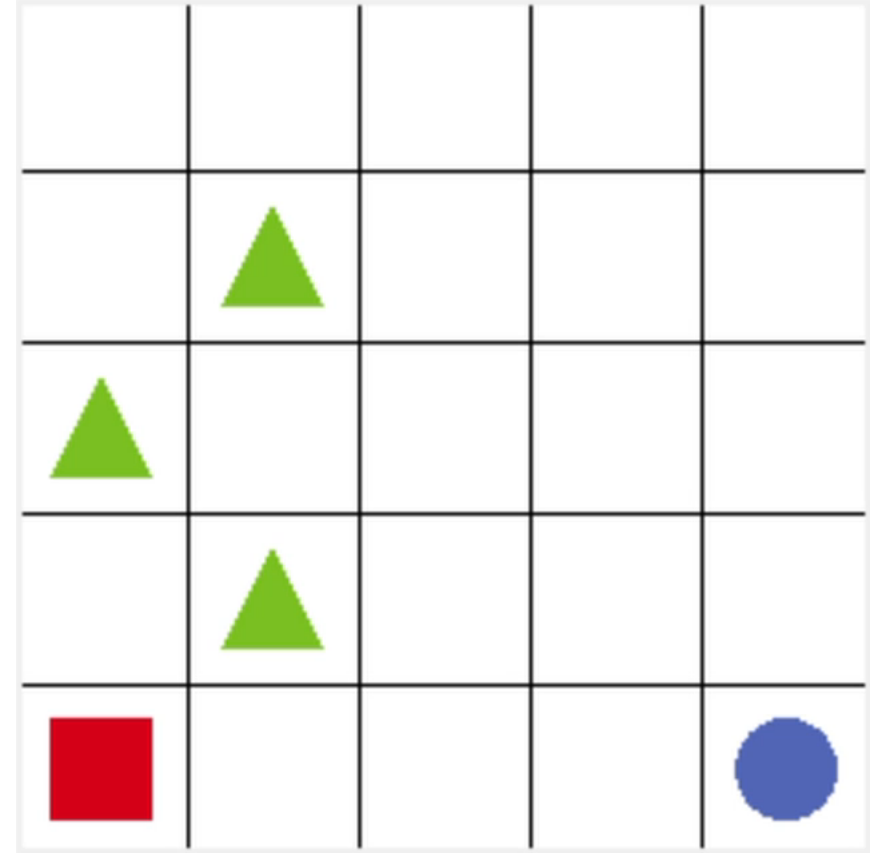
## 4. 목표: 초록색 세모를 피해서 파란색 동그라미로 가기



# 딥살사로 학습한 에이전트



학습 중

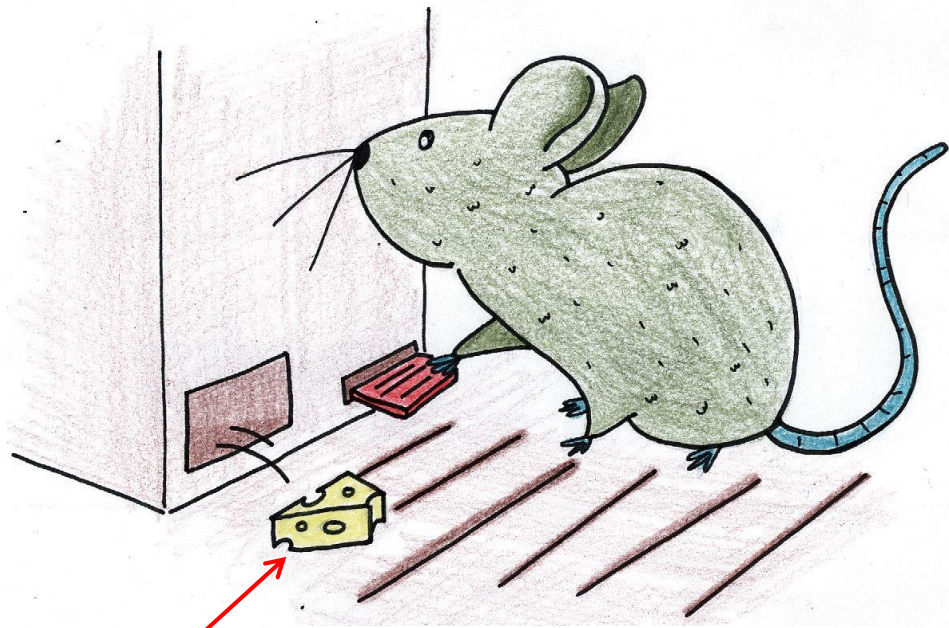


학습 후

# 큐러닝 알고리즘 설명

# 행동심리학의 강화이론

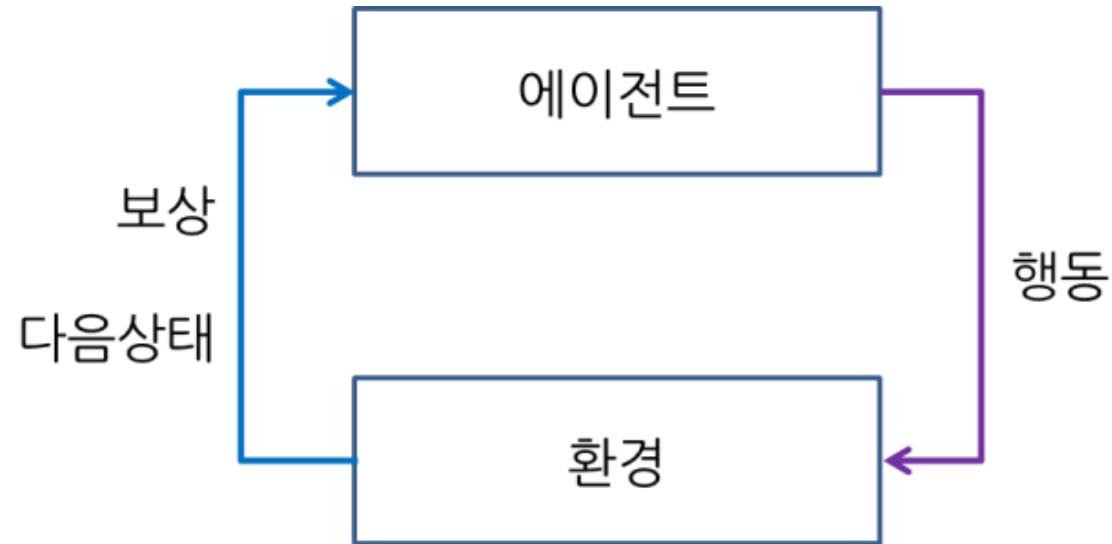
- 스키너의 쥐 실험



**보상**을 받는 행동의 확률 증가 → **강화**

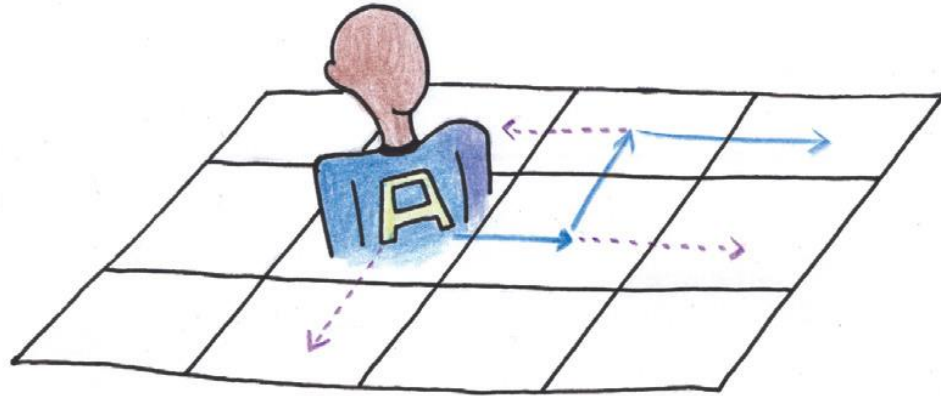
# 학습을 위해 필요한 것

- 핵심은 에이전트와 환경의 상호작용



모르는 단어! [에이전트, 환경] [상태, 행동, 보상]

**에이전트** → 상태를 관찰, 행동을 선택, 목표지향



an autonomous, goal-directed entity which observes and acts upon an environment - 위키피디아

**환경** → 에이전트를 제외한 나머지



판단하는 아이라는 주체를 빼고 길과 자전거와 아이의 몸 또한 환경이 된다

무엇을 관찰?

상태(state), 보상(reward)

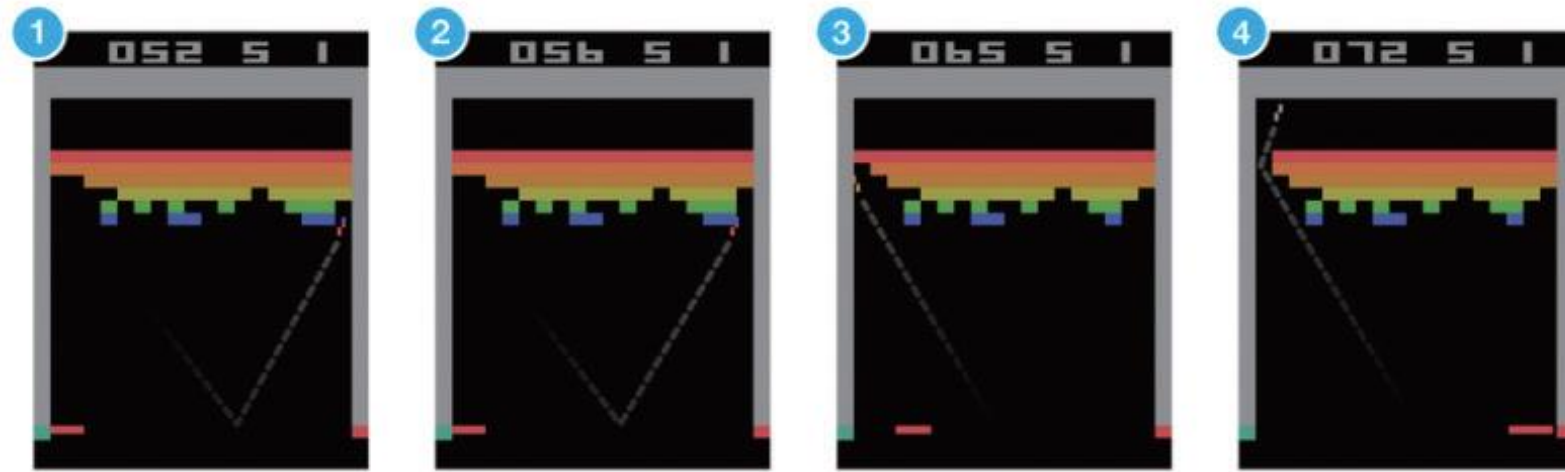
**상태(s)** → 현재 상황을 나타내는 정보



에이전트가 탁구를 치려면 탁구공의 위치, 속도, 가속도와 같은 정보가 필요



**보상(r) → 행동의 좋고 나쁨을 알려주는 정보**

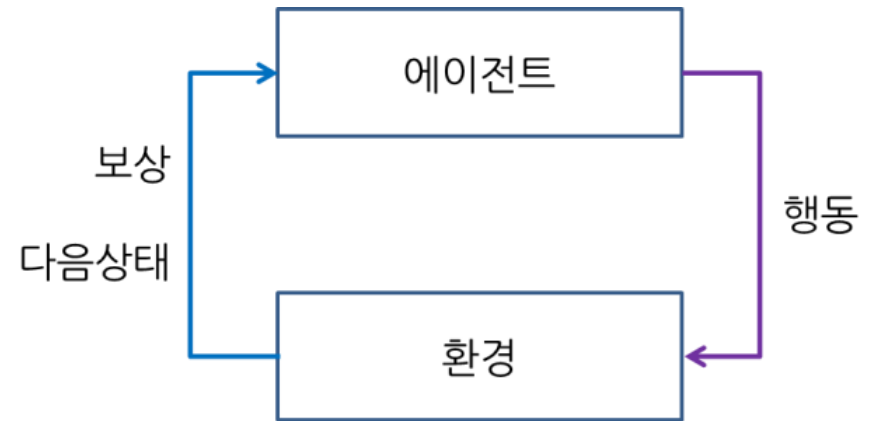


<https://www.intelnervana.com/demystifying-deep-reinforcement-learning/>

**보상은 에이전트가 달성하고자 하는 목표에 대한 정보를 담고 있다**

# 에이전트와 환경의 상호작용 과정

1. 에이전트가 환경에서 자신의 상태를 관찰
2. 그 상태에서 **어떠한 기준**에 따라 행동을 선택
3. 선택한 행동을 환경에서 실행
4. 환경으로부터 다음 상태와 보상을 받음
5. 보상을 통해 에이전트가 가진 정보를 수정함



$$s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_T$$

어떻게 행동을 선택?

판단 기준의 필요 → 가치함수(Value function)

# 가치함수 (Value function)

- 만약 즉각적인 보상만을 고려해서 행동을 선택한다면?



이 행동만이 좋은 행동이고 나머지는 아니다?

# 가치함수 (Value function)

---

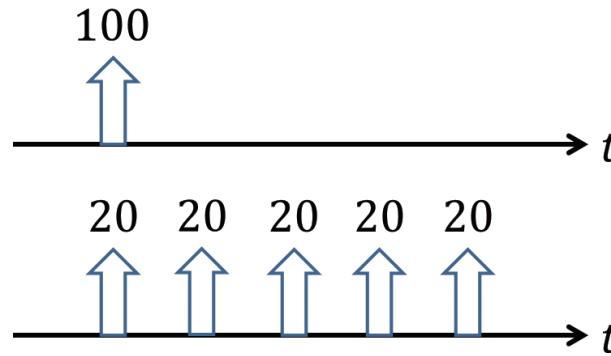
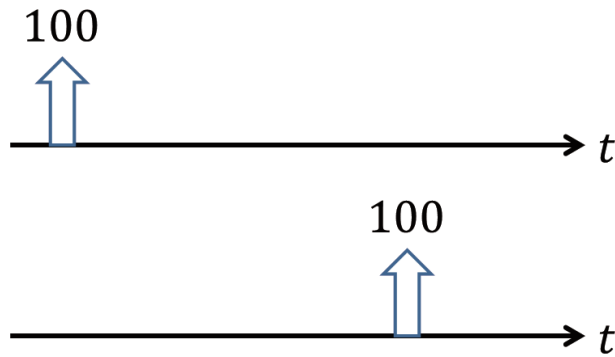
- 보상은 딜레이(delay)된다
- 어떤 행동이 그 보상을 얻게 했는지 명확하지 않다

그렇다면 앞으로 받을 보상을 싹 다 더해보자  
현재 시간 =  $t$

$$\text{보상의 합} = R_{t+1} + R_{t+2} + \cdots + R_T$$

# 가치함수 (Value function)

- 세 가지 문제점 → 감가율의 도입(discount factor)



$$\begin{aligned} 0.1 + 0.1 + \dots &= \infty \\ 1 + 1 + \dots &= \infty \end{aligned}$$

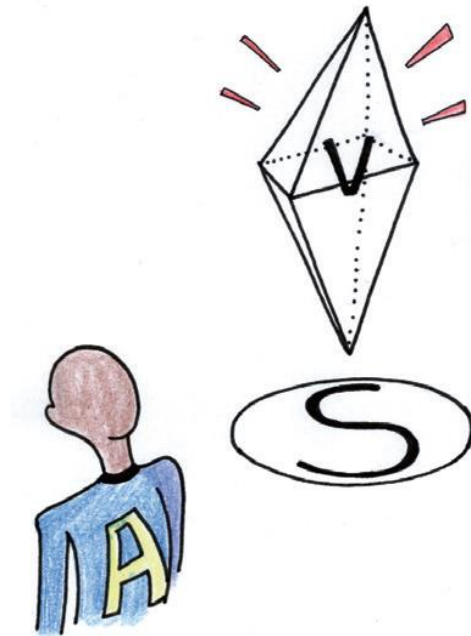
감가율  $0 \leq \gamma \leq 1$

$$\text{보상의 합} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-t-1} R_T$$

# 가치함수(Value function)

- 하지만 아직 보상을 받지 않았는데...? 미래에 받을 보상을 어떻게 알지?

지금 상태에서 미래에 받을 것이라 기대하는 보상의 합 = 가치함수

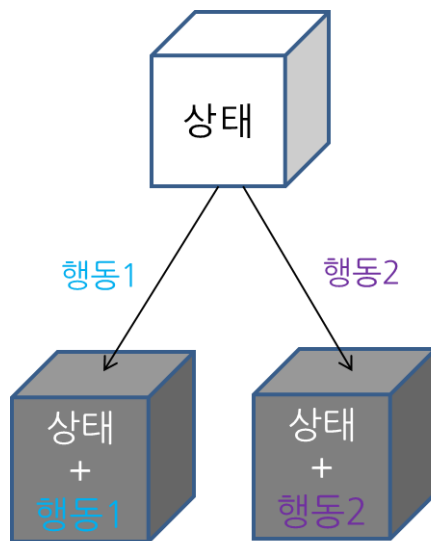


가치함수  $v(s) = E[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s]$

# 큐함수(Q function)

- 하지만 내가 알고 싶은 건 '어떤 행동이 좋은가'인데?

지금 상태에서 이 행동을 선택했을 때 미래에 받을 것이라 기대하는 보상의 합  
= 큐함수



$$\text{큐함수 } q(s, a) = E[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s, A_t = a]$$

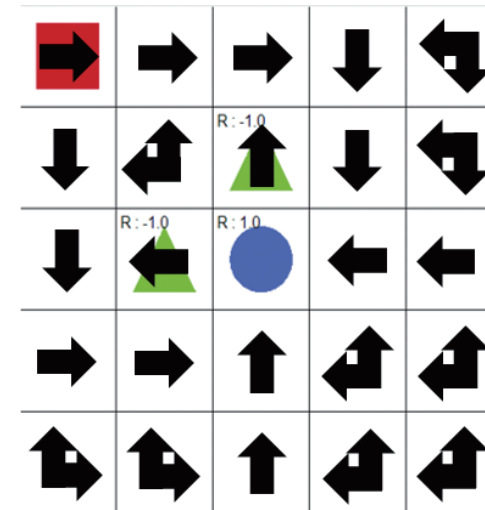


# 정책(Policy)

- 미래에 대한 기대  $\rightarrow$  내가 어떻게 행동할 것인지를 알아야 함
- 각 상태에서 에이전트가 어떻게 행동할 지에 대한 정보

상태  $s$ 에서 행동  $a$ 를 선택할 확률

정책  $\pi(a|s) = P[A_t = a | S_t = s]$



가치함수  $v_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s]$

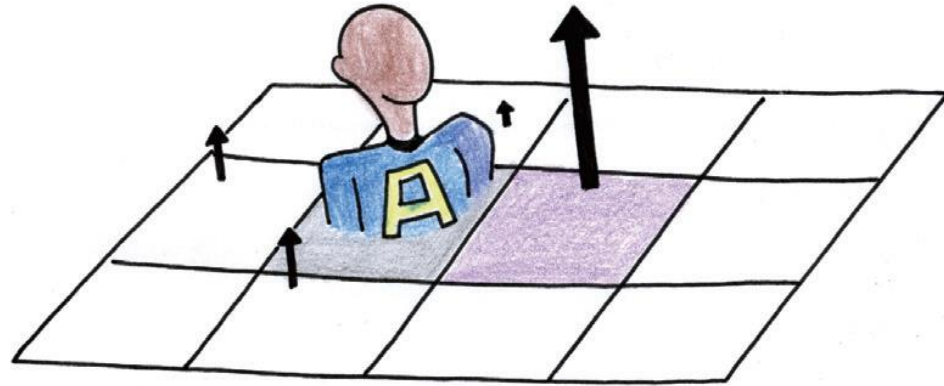
큐함수  $q_{\pi}(s, a) = E_{\pi}[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s, A_t = a]$

**큐함수를 통해 어떻게 행동을 선택?**

**그냥 큰 놈 골라**

# 탐욕정책(greedy policy)

- 지금 상태에서 선택할 수 있는 행동 중에 큐함수가 가장 높은 행동을 선택



탐욕정책  $\pi'(s) = \operatorname{argmax}_a q_{\pi}(s, a)$

어떻게 학습? **큐함수의 업데이트**

# 벨만 방정식(Bellman equation)

- 에이전트는 모든 (상태, 행동)에 대해서 큐함수를 가진다 → 일종의 기억
- 그렇다면 현재의 큐함수를 다음 타임스텝의 큐함수로 표현할 수 있지 않을까?

$$\text{큐함수 } q_{\pi}(s, a) = E_{\pi}[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s, A_t = a]$$

너무 먼 미래에 대해서 기대를 품기보다는 가까운 미래에 대해서 구체적인 기대를 품기로 했다

$$q_{\pi}(s, a) = E_{\pi}[R_{t+1} + \gamma(R_{t+2} + \dots) | S_t = s, A_t = a]$$

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

## 벨만 기대 방정식(Bellman expectation equation)

# 살사(SARSA)

- 벨만 기대 방정식 → 큐함수 업데이트 식

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

미래에 대해서 기대만 하기보다는 실제로 부딪혀보면서 학습하기로 했다

**현재 큐함수  $\leftarrow$  보상 + 감가율  $\times$  다음 큐함수**

$$q(s, a) \leftarrow r + \gamma q_{\pi}(s', a')$$

점진적인 큐함수의 업데이트

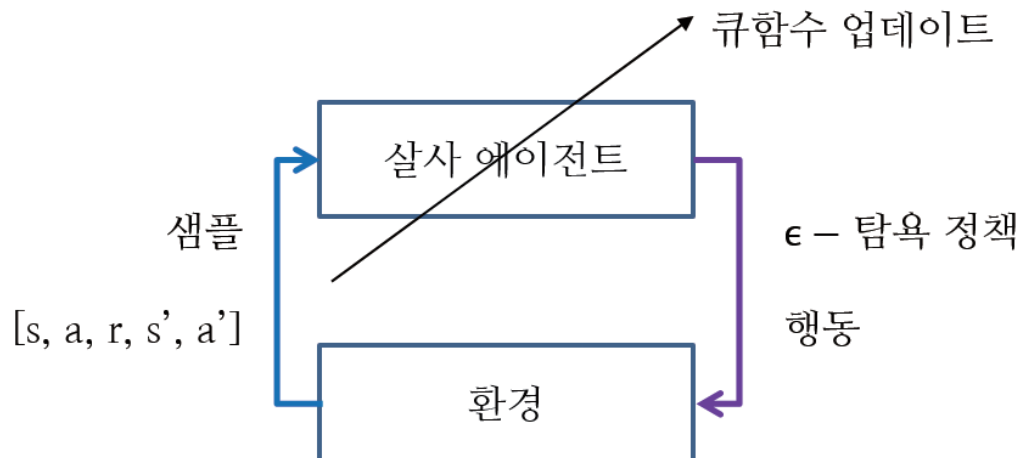
$$q(s, a) \leftarrow q(s, a) + \alpha(r + \gamma q(s', a') - q(s, a))$$

# 살사(SARSA)

- 현재 큐함수를 업데이트하기 위해서는  $(s, a, r, s', a')$ 이 필요

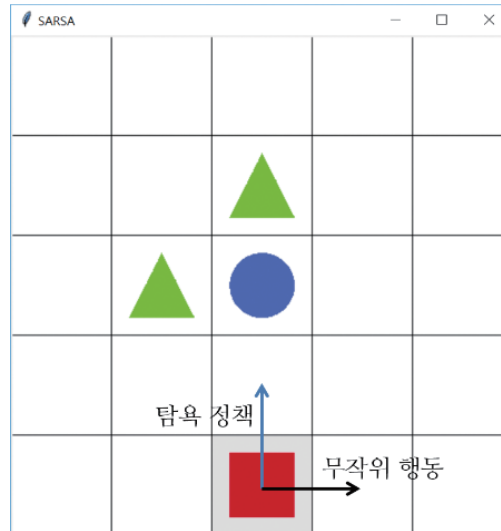
→ 살사(SARSA)

$$q(s, a) \leftarrow q(s, a) + \alpha(r + \gamma q(s', a') - q(s, a))$$



## $\epsilon$ - 탐욕정책

- 탐욕 정책의 Exploration problem → 일정한 확률로 랜덤하게 행동 선택



탐욕정책  $\pi'(s) = \operatorname{argmax}_a q_{\pi}(s, a)$

$\epsilon$  - 탐욕정책  $\pi(s) = \begin{cases} a^* = \operatorname{argmax}_a q(s, a), & 1 - \epsilon \\ a \neq a^*, & \epsilon \end{cases}$

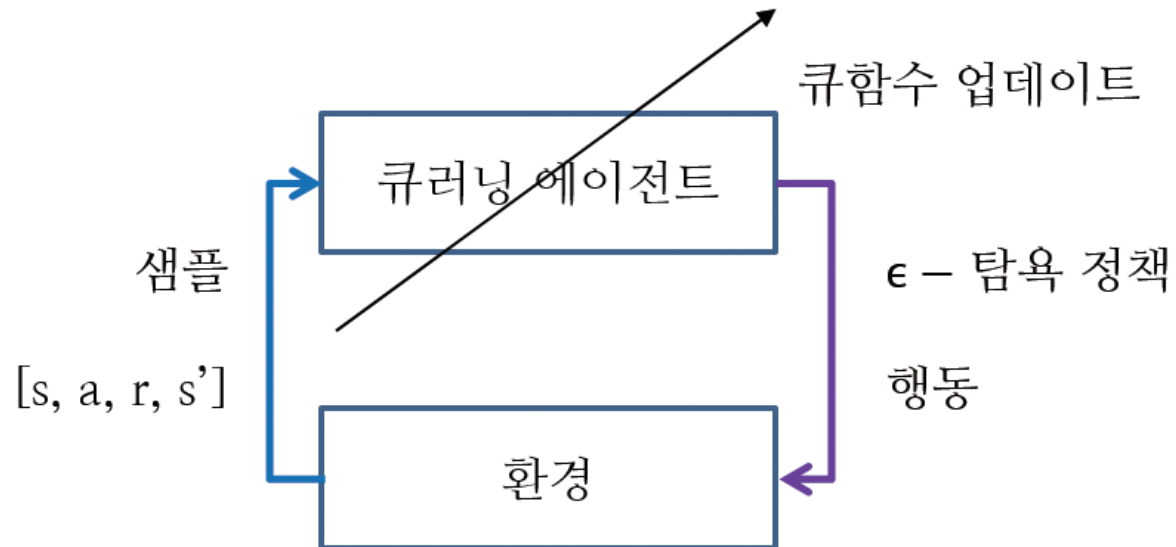


# 큐러닝(Q-Learning)

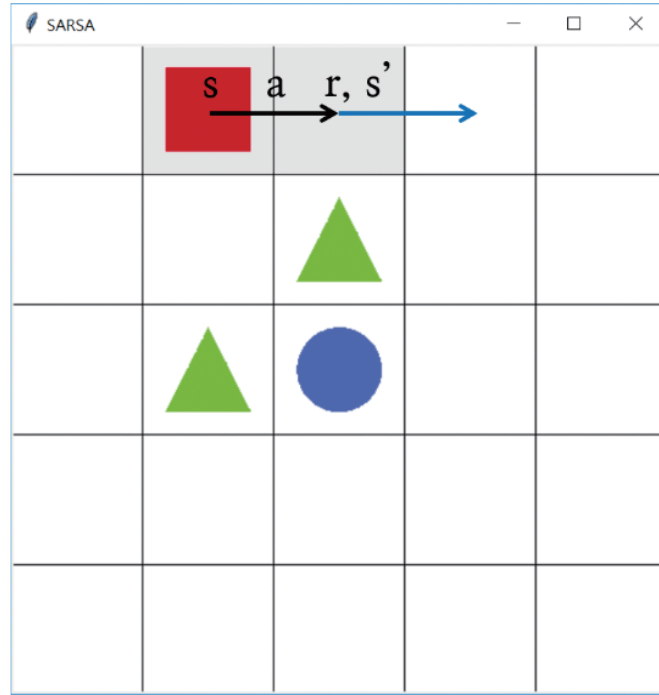
- 기왕 기억을 활용하는 김에 좋은 기억을 활용해보자

→ 다음 큐함수 중에서 가장 값이 큰 큐함수를 이용해서 현재 큐함수를 업데이트  
(Q-Learning)

$$q(s, a) = q(s, a) + \alpha(r + \gamma \max_{a'} q(s', a') - q(s, a))$$



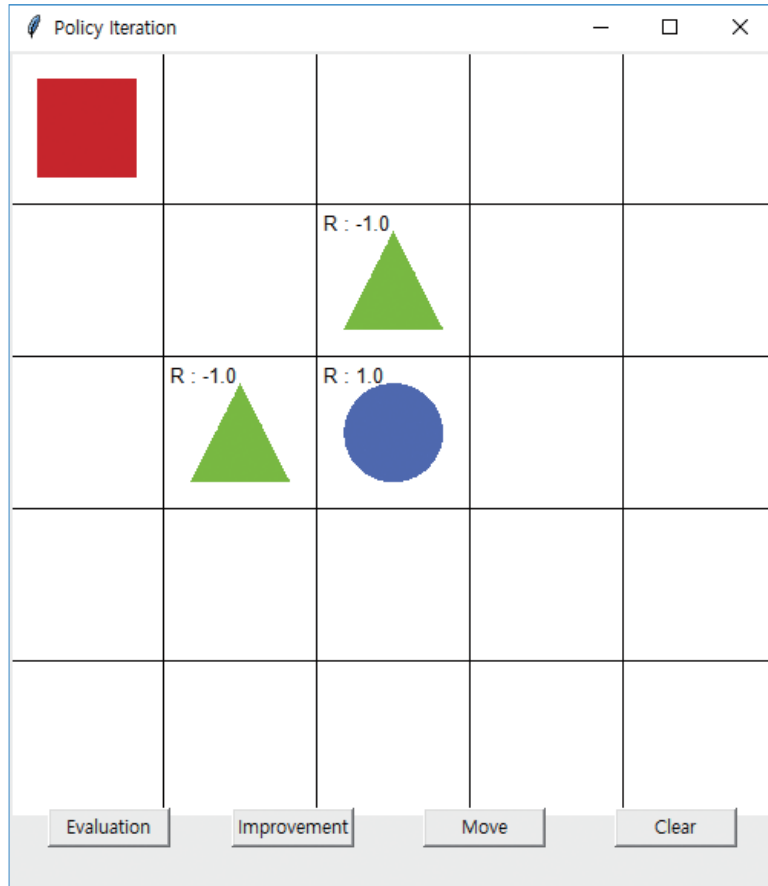
# 큐러닝(Q-Learning)



$$q(s, a) = q(s, a) + \alpha(r + \gamma \max_{a'} q(s', a') - q(s, a))$$

# 그리드월드와 큐러닝

# 그리드월드 문제 정의



## 문제의 정의

상태:  $[x, y]$  좌표

행동: 상, 하, 좌, 우, 제자리

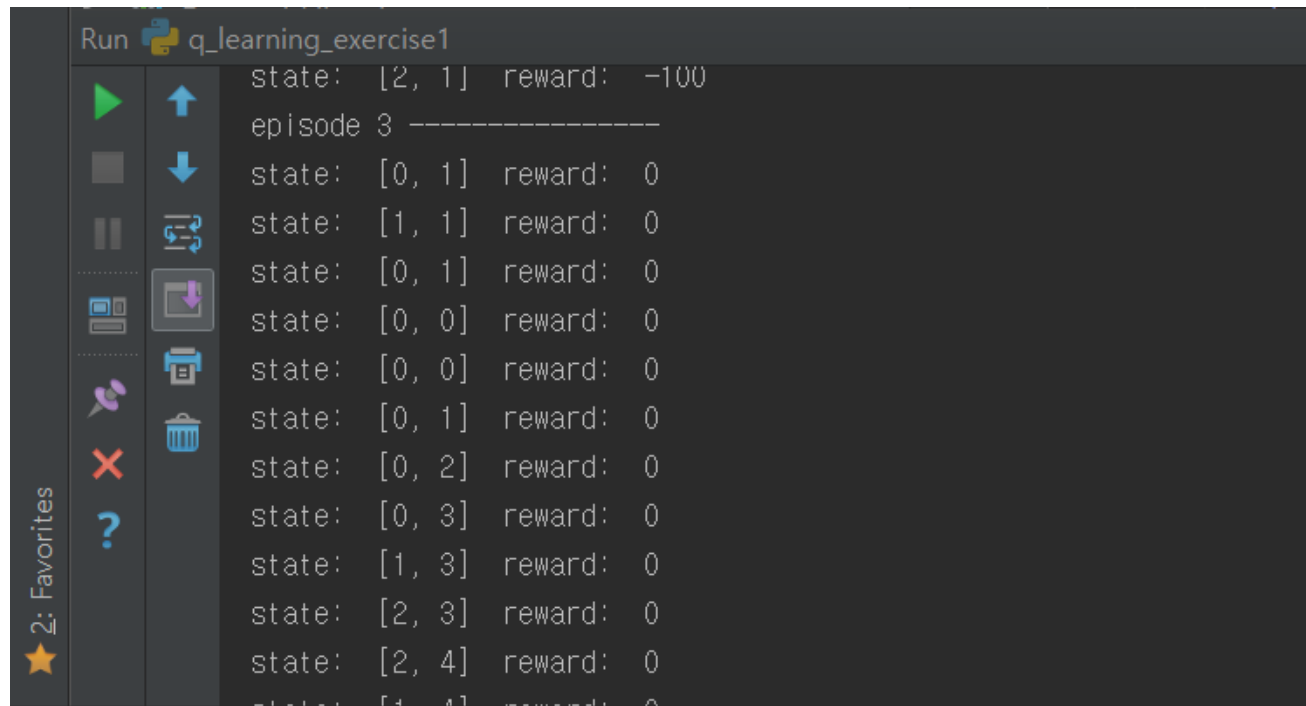
보상: 초록색 세모(-1), 파란색 동그라미(+1)

목표: 초록색 세모를 피해서 파란색 동그라미로 가기

**코드로 실습해봅시다**

# 상태와 보상을 눈으로 확인

1. 랜덤하게 행동을 선택
2. 선택한 행동으로 환경에서 한 스텝 진행
3. 각 스텝마다의 상태와 보상을 출력해보기 → **중요**



```
Run q_learning_exercise1
state: [2, 1] reward: -100
episode 3 -----
state: [0, 1] reward: 0
state: [1, 1] reward: 0
state: [0, 1] reward: 0
state: [0, 0] reward: 0
state: [0, 0] reward: 0
state: [0, 1] reward: 0
state: [0, 2] reward: 0
state: [0, 3] reward: 0
state: [1, 3] reward: 0
state: [2, 3] reward: 0
state: [2, 4] reward: 0
```

# 코드의 큰 구조 → 환경과의 상호작용

---

1. 현재 상태에서  $\epsilon$ -탐욕 정책에 따라 행동을 선택

→ `action = get_action(state)`

2. 선택한 행동으로 환경에서 한 타임스텝을 진행

→ `env.step(action)`

3. 환경으로부터 보상과 다음 상태를 받음

→ `reward, next_state`

4.  $(s, a, r, s')$ 을 통해 큐함수를 업데이트

→ `learn(state, action, reward, next_state)`

# 딥살사 알고리즘 설명



# 그리드월드의 변형

- 장애물이 움직임 → 기존 큐함수를 저장하는 방식이 힘들

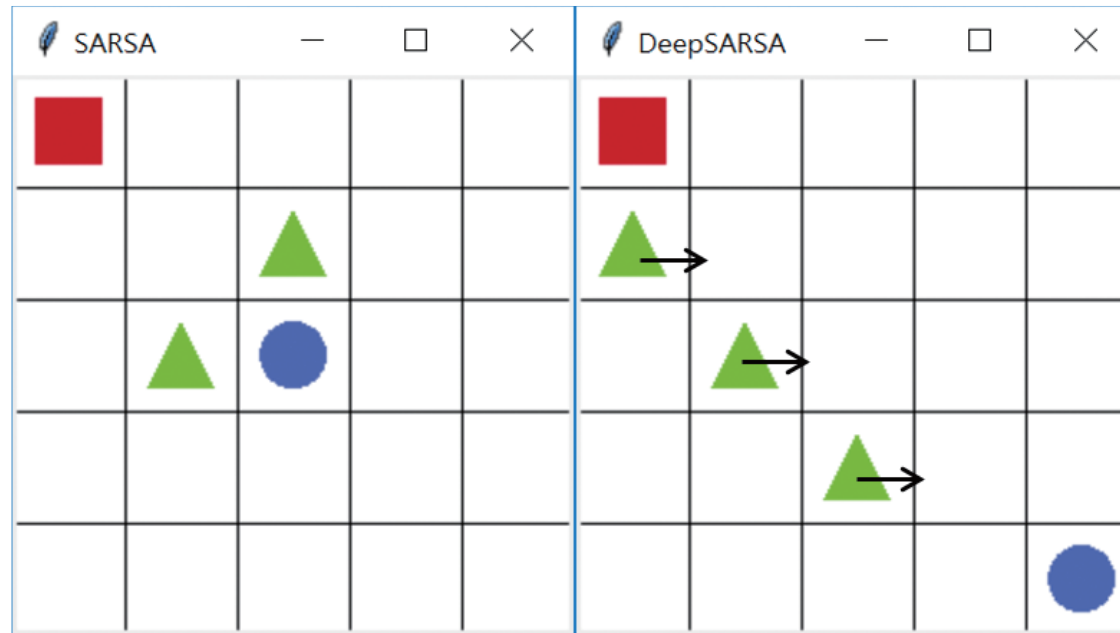
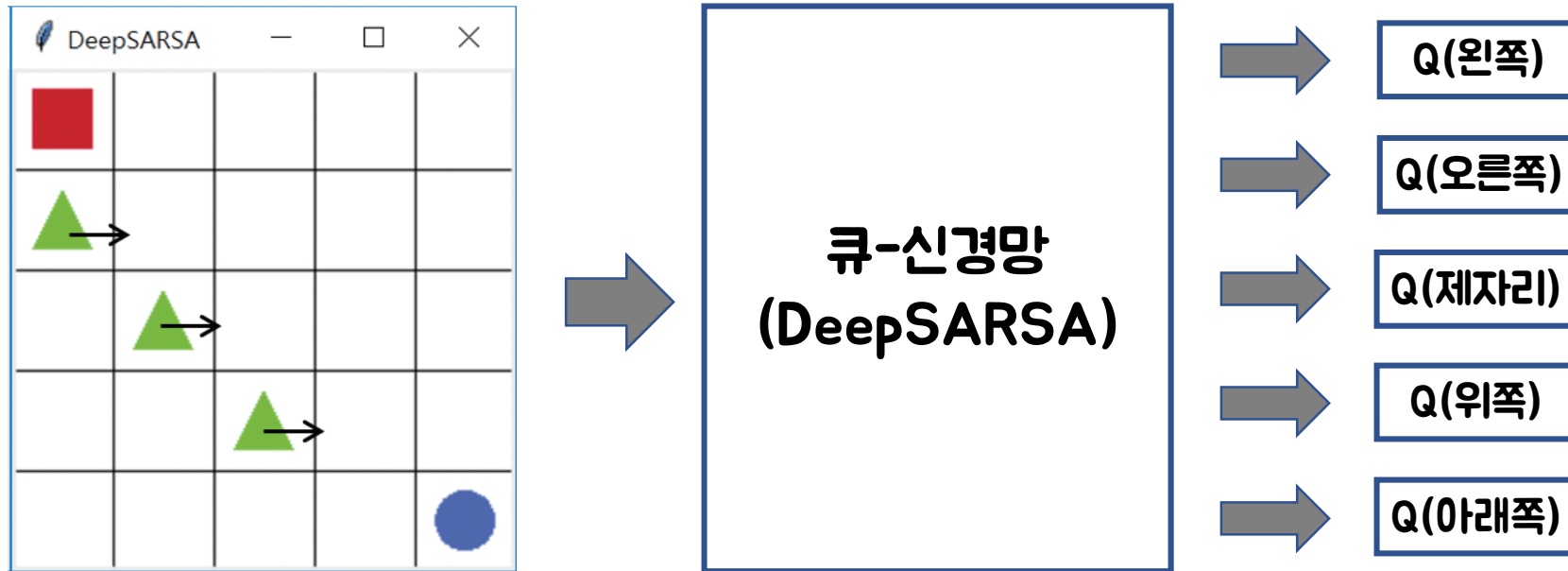


Table 형태로 모든 상태에 대해 큐함수를 저장하기 힘들

# 딥살사(DeepSARSA)

- 큐함수를 인공지능망으로 근사하자!

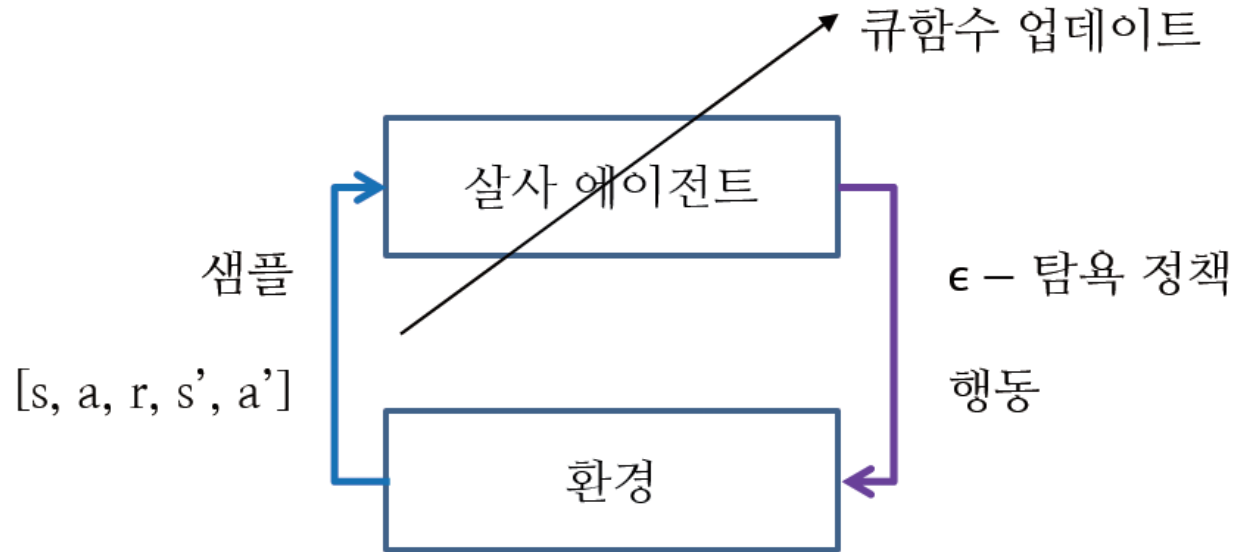


상태(히스토리) → 큐-신경망 → 각 행동에 대한 큐함수 값

# 인공신경망 업데이트

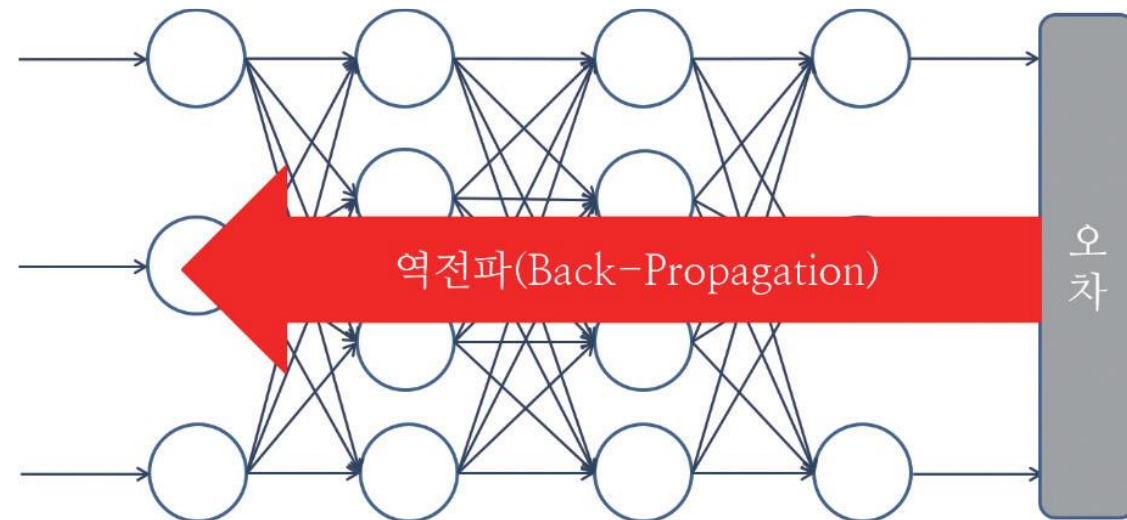
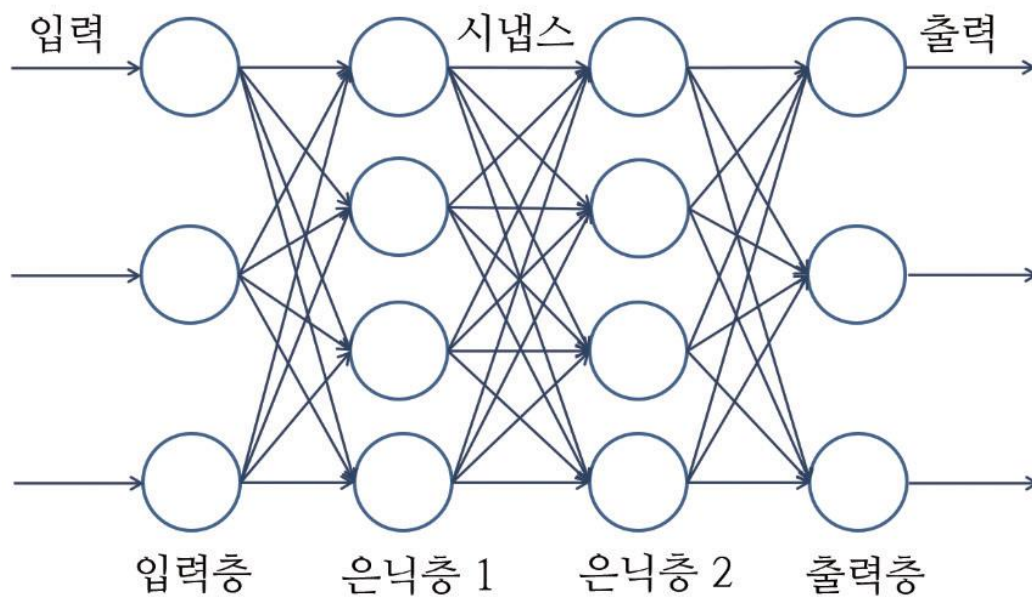
- 살사(SARSA)의 큐함수 업데이트 식

$$q(s, a) \leftarrow q(s, a) + \alpha(r + \gamma q(s', a') - q(s, a))$$



# 인공신경망 업데이트

- (정답 - 예측)의 오차를 이용해서 인공신경망에 역전파  
→ 인공신경망의 업데이트



# 인공신경망 업데이트

- 살사의 큐함수 업데이트 식

$$q(s, a) \leftarrow q(s, a) + \alpha(r + \gamma q(s', a') - q(s, a))$$

- 큐함수를 인공신경망(parameter  $\theta$ )로 근사

$$q_{\theta}(s, a) \leftarrow q_{\theta}(s, a) + \alpha(r + \gamma q_{\theta}(s', a') - q_{\theta}(s, a))$$

- 큐함수가 아닌 큐함수를 근사한 인공신경망을 업데이트(MSE loss function)

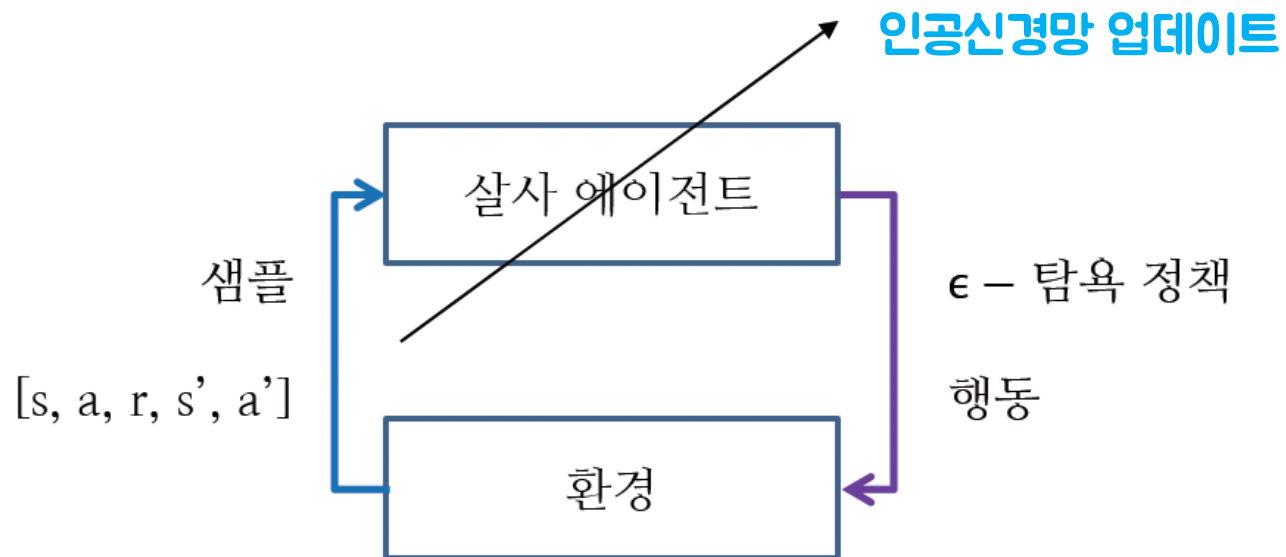
$$MSE = \left( \underbrace{r + \gamma q_{\theta}(s', a')}_{\text{정답}} - \underbrace{q_{\theta}(s, a)}_{\text{예측}} \right)^2$$

정답

예측

# 인공신경망 업데이트

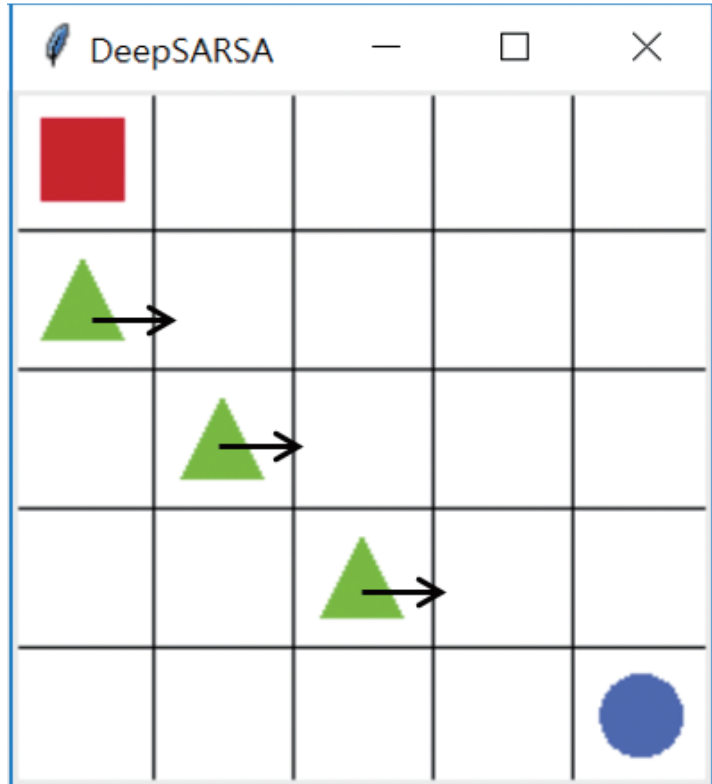
- 큐-신경망 업데이트



$$MSE = \left( r + \gamma q_{\theta}(s', a') - q_{\theta}(s, a) \right)^2$$

그리드월드와 **딥살사**

# 변형된 그리드월드 문제 정의



## 1. 상태의 정의

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

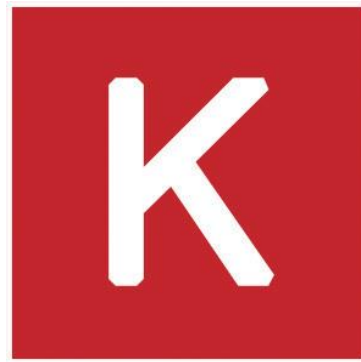
## 2. 행동: 상, 하, 좌, 우, 제자리

## 3. 보상: 초록색 세모(-1), 파란색 동그라미(+1)

## 4. 목표: 초록색 세모를 피해서 파란색 동그라미로 가기



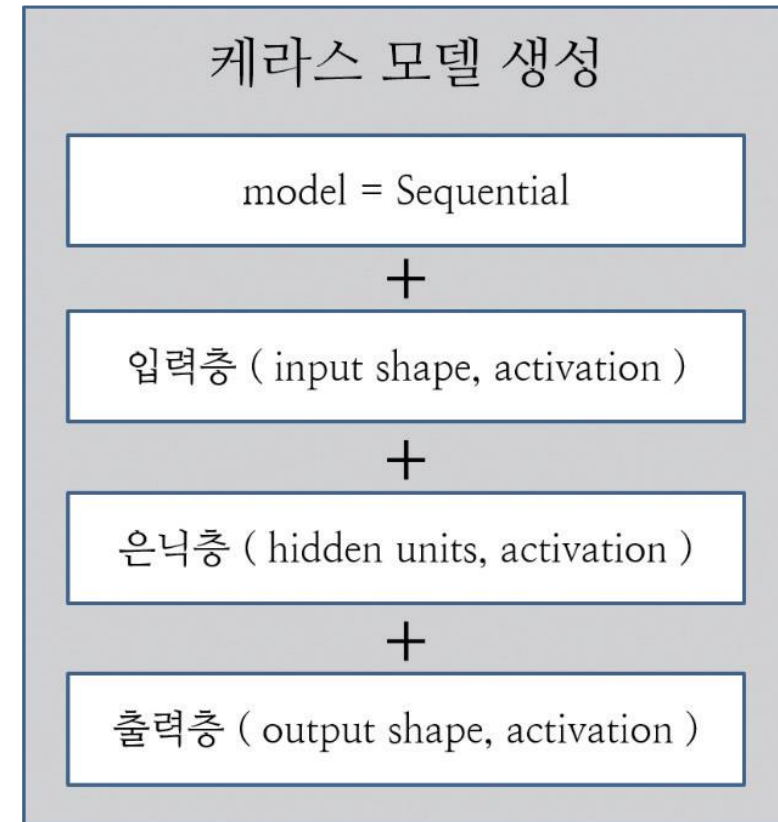
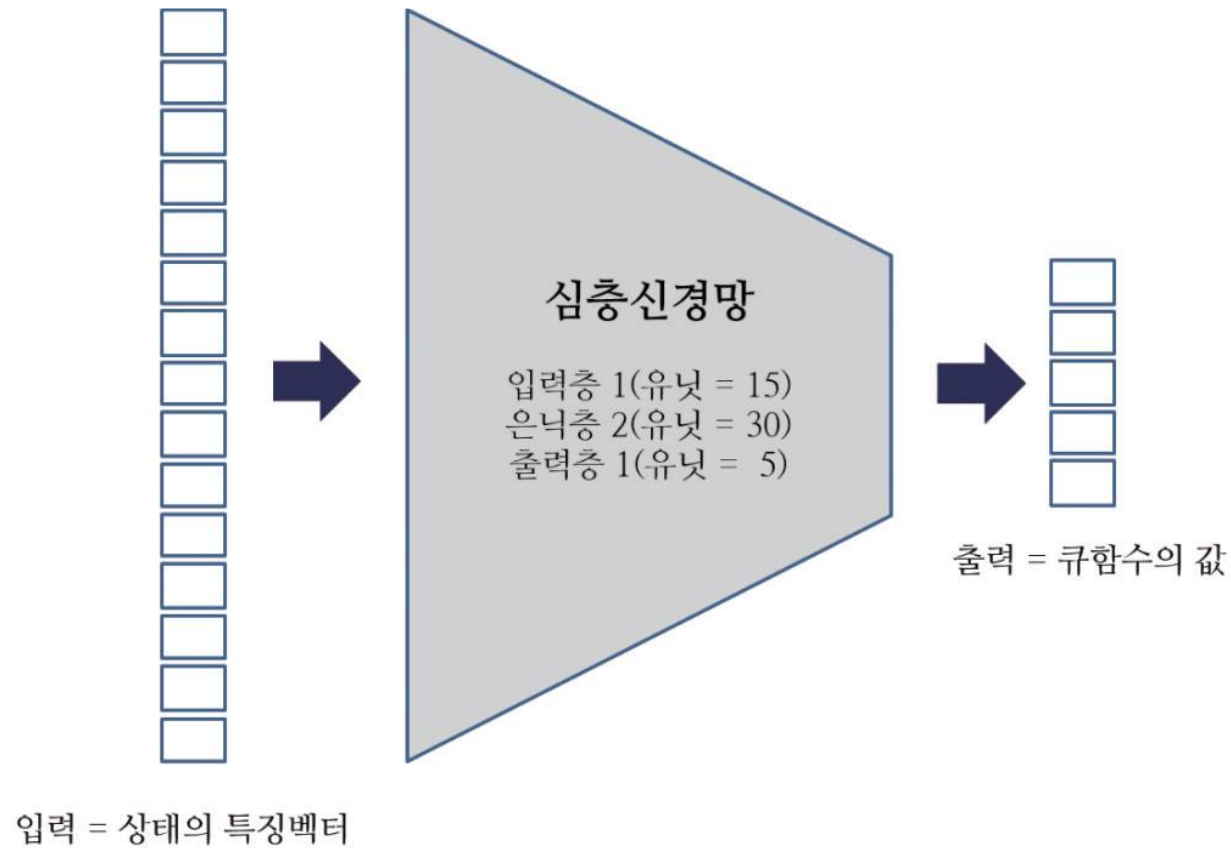
**케라스 = 딥러닝 프레임워크**



<https://keras.io/>

# 케라스 간단 설명

- Sequential model



**코드로 실습해봅시다**

# 케라스 간단 설명

- Sequential model

```
1  from keras.layers import Dense
2  from keras.models import Sequential
3  from keras.optimizers import Adam
4  import numpy as np
5
6  state_size = 15
7  action_size = 5
8  batch_size = 100
9
10 x_train = np.random.rand(batch_size, state_size)
11 y_train = np.random.rand(batch_size, action_size)
12
13 model = Sequential()
14 model.add(Dense(30, input_dim=state_size, activation='relu'))
15 model.add(Dense(30, activation='relu'))
16 model.add(Dense(action_size, activation='linear'))
17 model.compile(loss='mse', optimizer=Adam(lr=0.001))
18 model.summary()
19
20 model.fit(x_train, y_train, batch_size=32, epochs=1)
21
```

# 상태와 보상 확인

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

```
episode 0 -----
state: [1, 1, -1, -1, 2, 2, -1, -1, 3, 3, -1, -1, 4, 4, 1] action: 0 reward: 0
state: [0, 1, -1, -1, 1, 2, -1, -1, 2, 3, -1, -1, 3, 4, 1] action: 2 reward: 0
state: [1, 0, -1, -1, 2, 1, -1, -1, 3, 2, -1, -1, 3, 3, 1] action: 1 reward: 0
state: [1, 0, -1, -1, 2, 1, -1, -1, 3, 2, -1, -1, 3, 3, 1] action: 4 reward: 0
state: [2, -1, -1, -1, 3, 0, -1, -1, 2, 1, -1, 1, 3, 2, 1] action: 1 reward: 0
state: [2, 0, -1, -1, 3, 1, -1, -1, 2, 2, -1, 1, 3, 3, 1] action: 0 reward: 0
state: [3, 1, -1, -1, 2, 2, -1, 1, 1, 3, -1, 1, 3, 4, 1] action: 0 reward: 0
state: [3, 1, -1, -1, 2, 2, -1, 1, 1, 3, -1, 1, 3, 4, 1] action: 0 reward: 0
state: [2, 0, -1, 1, 1, 1, -1, 1, 0, 2, -1, 1, 3, 3, 1] action: 1 reward: 0
state: [3, 0, -1, 1, 2, 1, -1, 1, 1, 2, -1, 1, 4, 3, 1] action: 3 reward: 0
state: [2, -1, -1, 1, 1, 0, -1, 1, 0, 1, -1, 1, 4, 2, 1] action: 1 reward: 0
```

# 코드의 큰 구조 → 환경과의 상호작용

---

1. 현재 상태에서  $\epsilon$ -탐욕 정책에 따라 행동을 선택

→ `action = get_action(state)`

2. 선택한 행동으로 환경에서 한 타임스텝을 진행

→ `env.step(action)`

3. 환경으로부터 보상과 다음 상태를 받음

→ `reward, next_state`

4. 다음 상태에서  $\epsilon$ -탐욕 정책에 따라 다음 행동을 선택

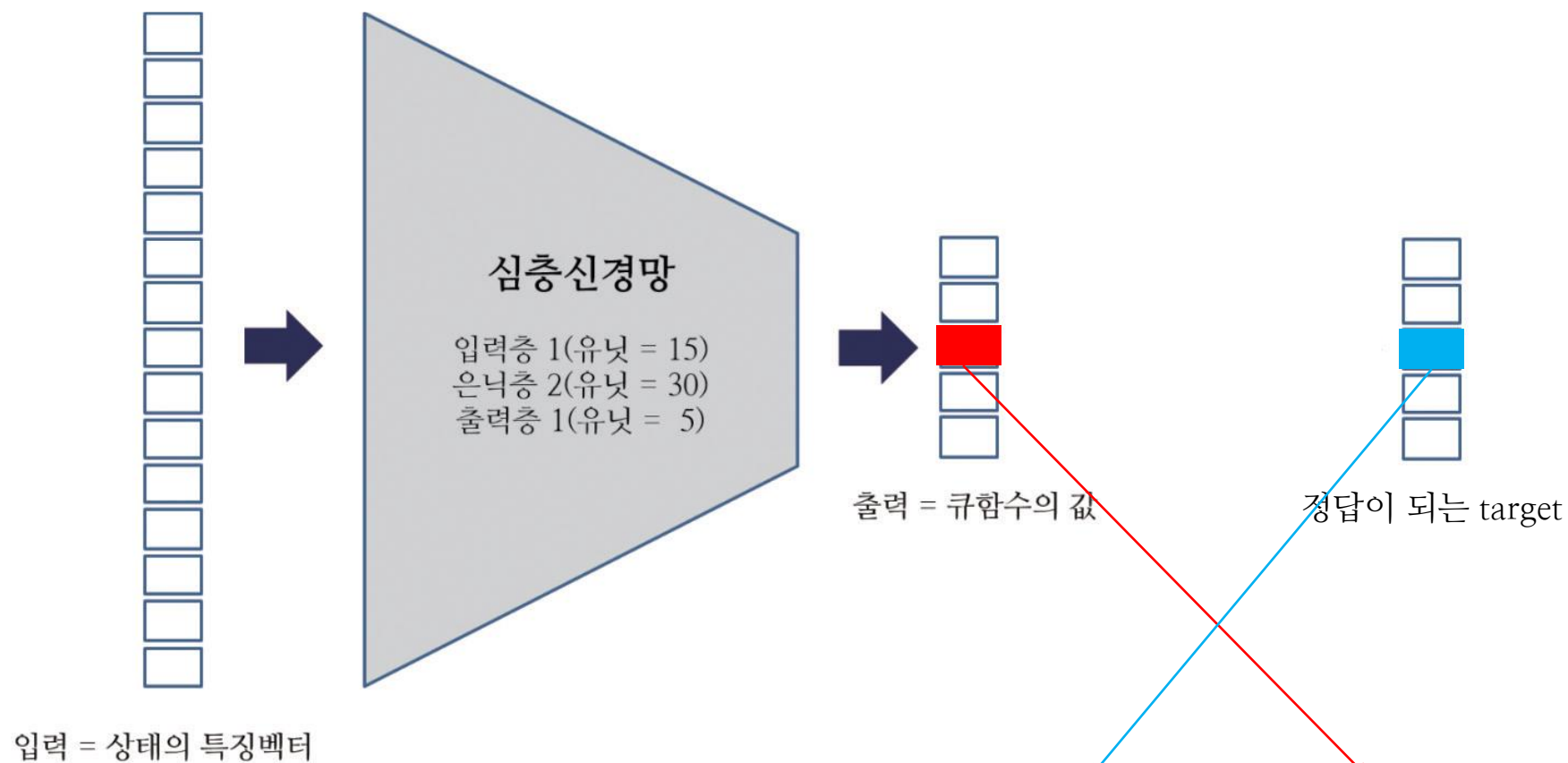
→ `next_action = get_action(next_state)`

5.  $(s, a, r, s')$ 을 통해 큐함수를 업데이트

→ `train_model(state, action, reward, next_state, next_action)`

# MSE 에러와 model.fit()

- Sequential model



$$MSE = \left( r + \gamma q_{\theta}(s', a') - q_{\theta}(s, a) \right)^2 \longrightarrow \text{model.fit()}$$

# 파이썬과 케라스로 배우는 강화학습



<http://wikibook.co.kr/reinforcement-learning/>



# 책이 다루는 내용

---

- 강화학습의 배경과 개념
- 강화학습을 위한 기본적인 이론: MDP, 벨만 방정식, 다이내믹 프로그래밍
- 고전 강화학습 알고리즘: 몬테카를로, 살사, 큐러닝
- 인공지능망을 이용한 강화학습 알고리즘: 딥살사, REINFORCE, DQN, 액터-크리틱, A3C
- 강화학습 알고리즘 구현 및 설명: 그리드월드, 카트폴, 아타리게임

감사합니다