

Reinforcement Learning (3)

Bio-Medical Computing Laboratory

JuHyeong, Kim

18th Sep, 2019



Acknowledgement

- This slide is mostly based on the lecture of David Silver , “파이썬과 케라스로 배우는 강화학습”, and “모두의 머신러닝 ”



Bellman Expectation Equation and Policy Evaluation

$$\begin{aligned}
 v_{\pi}(s) &= E_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \dots | S_t = s] \\
 &= E_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s] && \text{(Bellman Expectation Equation)} \\
 &= \sum_{a \in A} \pi(a|s) (R_s^a + \gamma v_{\pi}(s')) && \text{(Policy Evaluation)}
 \end{aligned}$$



Monte Carlo Prediction

$$\begin{aligned}
 v_{\pi}(s) &= E_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \dots | S_t = s] \\
 &= E_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s] && \text{(Bellman Expectation Equation)} \\
 &= \sum_{a \in A} \pi(a|s) (R_s^a + \gamma v_{\pi}(s')) && \text{(Policy Evaluation)}
 \end{aligned}$$

↓ Monte Carlo Prediction

$$v_{\pi}(s) \sim \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i(s)$$



Monte Carlo Prediction

$$\begin{aligned}
 V_{n+1} &= \frac{1}{n} \sum_{i=1}^n G_i \\
 &= \frac{1}{n} (G_n + \sum_{i=1}^{n-1} G_i) \\
 &= \frac{1}{n} + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} G_i \\
 &= \frac{1}{n} (G_n + (n-1)V_n) \\
 &= \frac{1}{n} (G_n + nV_n - V_n) \\
 &= V_n + \frac{1}{n} (G_n - V_n)
 \end{aligned}$$



Monte Carlo Prediction

$$\begin{aligned}
 V_{n+1} &= \frac{1}{n} \sum_{i=1}^n G_i \\
 &= \frac{1}{n} (G_n + \sum_{i=1}^{n-1} G_i) \\
 &= \frac{1}{n} + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} G_i \\
 &= \frac{1}{n} (G_n + (n-1)V_n) \\
 &= \frac{1}{n} (G_n + nV_n - V_n) \\
 &= V_n + \frac{1}{n} (G_n - V_n) \\
 &= V_n + \alpha (G_n - V_n)
 \end{aligned}$$

$\frac{1}{n}$ is the step size.

It is treated as a constant and acts as a weight.



Monte Carlo Prediction

$$\begin{aligned}
 V_{n+1} &= \frac{1}{n} \sum_{i=1}^n G_i \\
 &= \frac{1}{n} (G_n + \sum_{i=1}^{n-1} G_i) \\
 &= \frac{1}{n} + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} G_i \\
 &= \frac{1}{n} (G_n + (n-1)V_n) \\
 &= \frac{1}{n} (G_n + nV_n - V_n) \\
 &= V_n + \frac{1}{n} (G_n - V_n) \\
 &= V_n + \alpha (G_n - V_n)
 \end{aligned}$$

↓ Value function update expression for
Monte Carlo predictions

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



Monte Carlo Prediction & Temporal-Difference Prediction

$$\begin{aligned}
 V_{n+1} &= \frac{1}{n} \sum_{i=1}^n G_i \\
 &= \frac{1}{n} (G_n + \sum_{i=1}^{n-1} G_i) \\
 &= \frac{1}{n} + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} G_i \\
 &= \frac{1}{n} (G_n + (n-1)V_n) \\
 &= \frac{1}{n} (G_n + nV_n - V_n) \\
 &= V_n + \frac{1}{n} (G_n - V_n) \\
 &= V_n + \alpha (G_n - V_n)
 \end{aligned}$$



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

$$\begin{aligned}
 V(S_t) &\leftarrow V(S_t) + \alpha (G_t - V(S_t)) \\
 &\leftarrow V(S_t) + \alpha (R + \gamma V(S_{t+1}) - V(S_t))
 \end{aligned}$$

$$G_t = R + \gamma V(S_{t+1})$$



Monte Carlo Prediction & Temporal-Difference Prediction

$$\begin{aligned}
 V_{n+1} &= \frac{1}{n} \sum_{i=1}^n G_i \\
 &= \frac{1}{n} (G_n + \sum_{i=1}^{n-1} G_i) \\
 &= \frac{1}{n} + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} G_i \\
 &= \frac{1}{n} (G_n + (n-1)V_n) \\
 &= \frac{1}{n} (G_n + nV_n - V_n) \\
 &= V_n + \frac{1}{n} (G_n - V_n) \\
 &= V_n + \alpha (G_n - V_n)
 \end{aligned}$$



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

$$\begin{aligned}
 V(S_t) &\leftarrow V(S_t) + \alpha (G_t - V(S_t)) \\
 &\leftarrow V(S_t) + \alpha (R + \gamma V(S_{t+1}) - V(S_t))
 \end{aligned}$$

↓ Update of Q function in
Temporal-Difference Prediction

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



SARSA

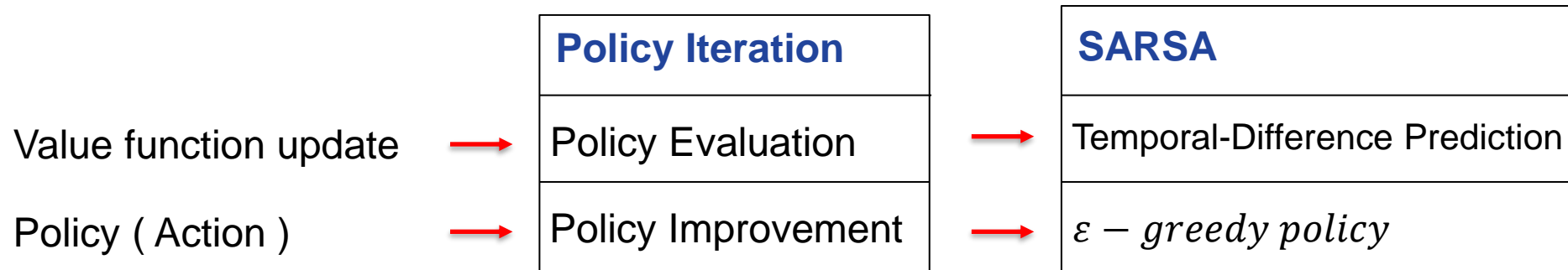
Sarsa is an updated formula from the Temporal-Difference prediction to the Q - function.

The samples used in the above equations are $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$.

Starting from Sarsa, it uses the ε -greedy policy.



Summary of Policy Iteration and SARSA



ε – greedy policy

ε is a very small number :

$$\pi(s) = \begin{cases} a^* = \underset{a \in A_Q}{\operatorname{argmax}}(s, a), & 1 - \varepsilon \\ a \neq a^* \text{ (random action)}, & \varepsilon \end{cases}$$

$1 - \varepsilon$ is a exploit,

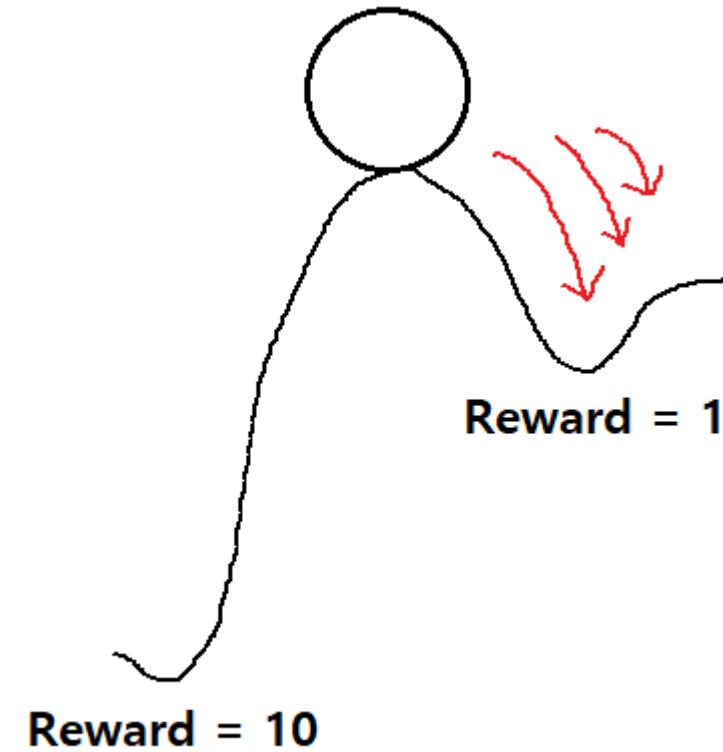
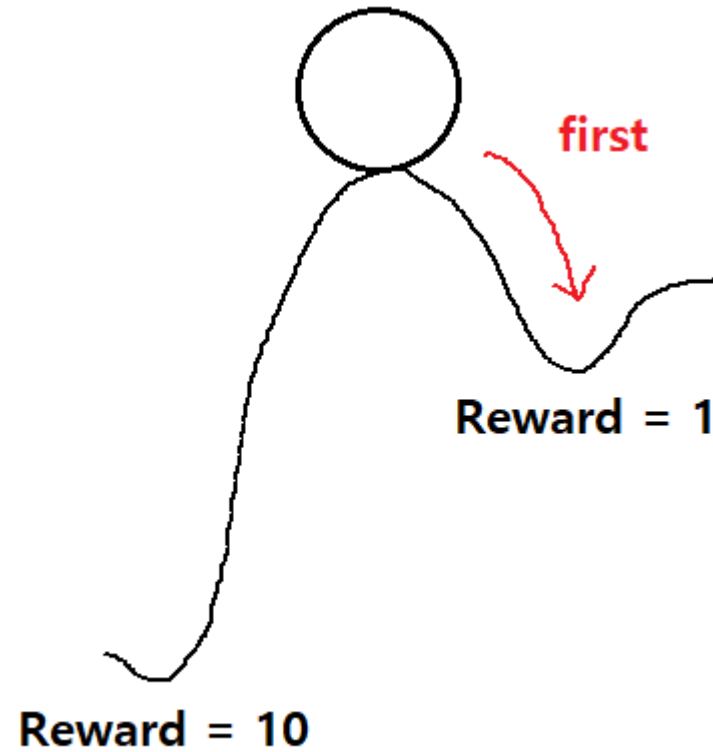
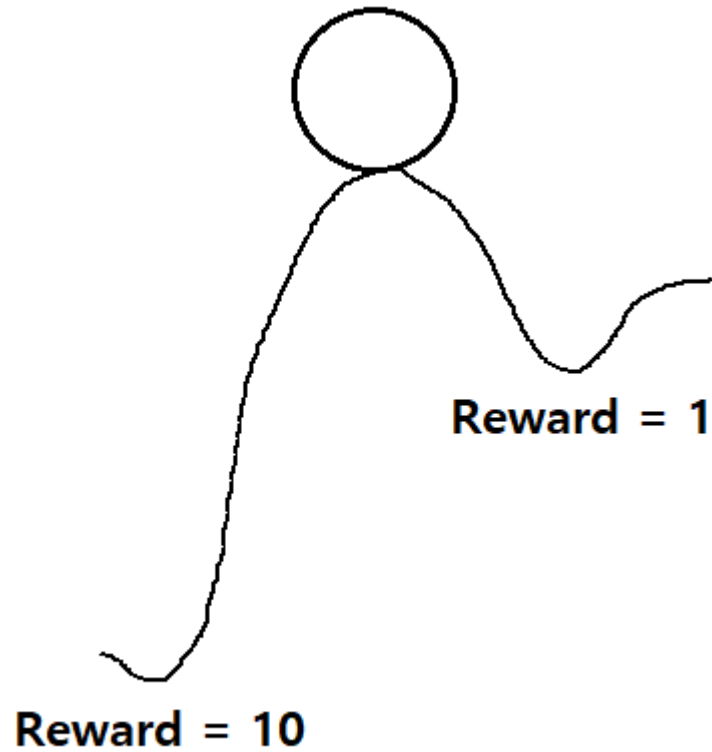
$$a^* = \underset{a \in A_Q}{\operatorname{argmax}}(s, a)$$

ε is a exploration,

$$a \neq a^* \text{ (random action)}$$

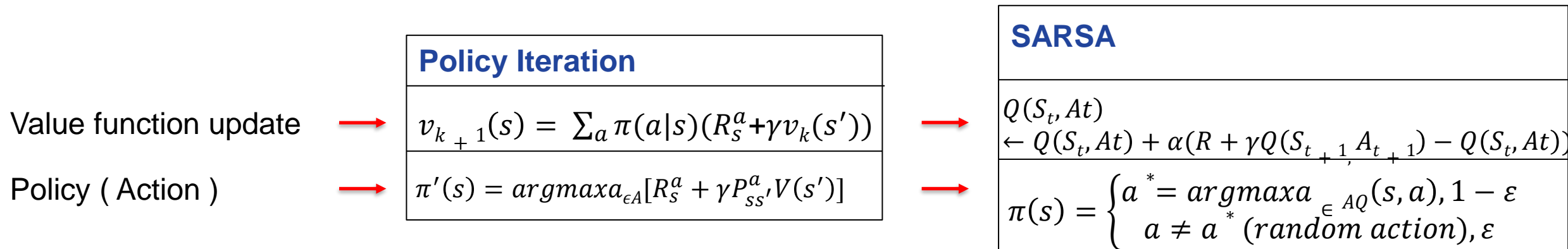
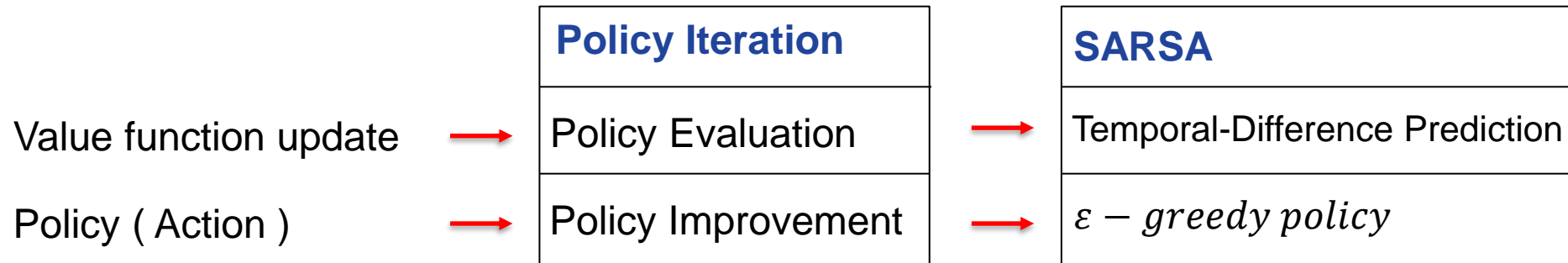


ϵ – greedy policy

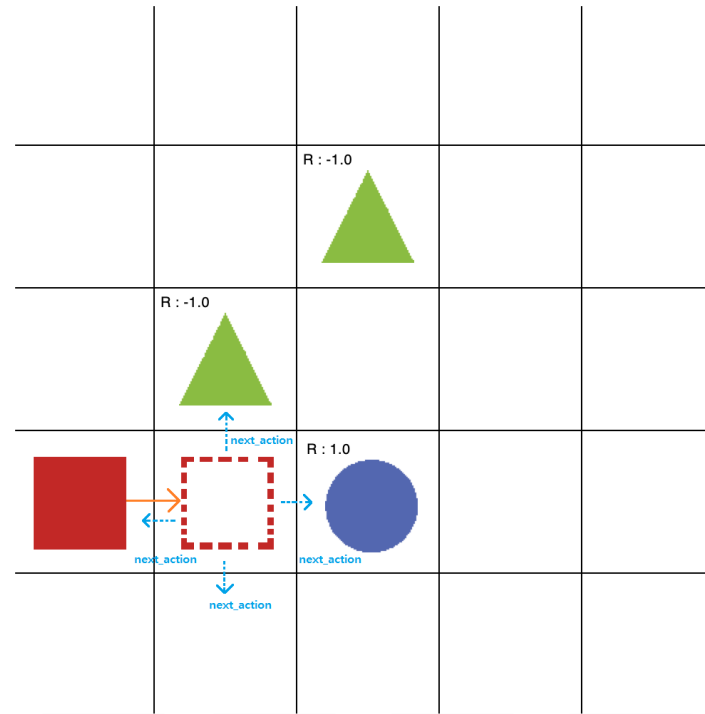
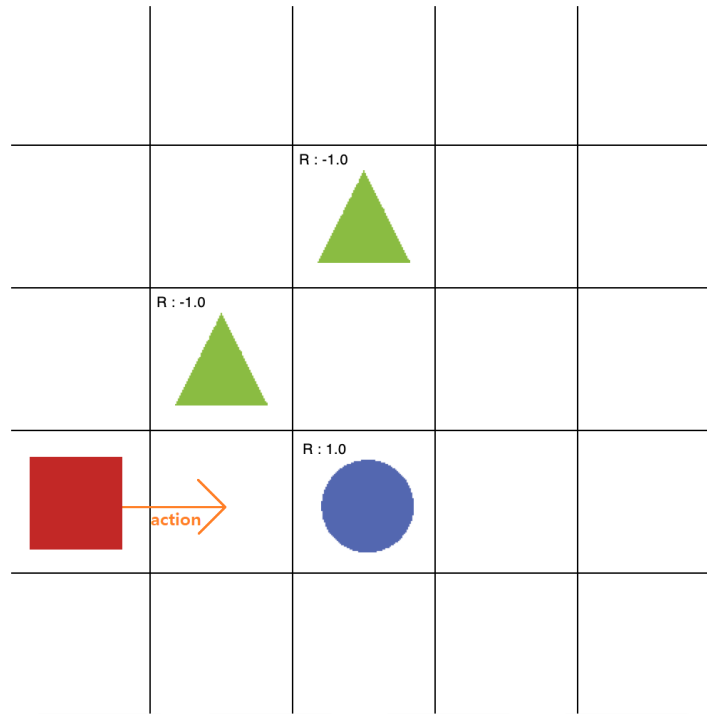




Summary of Policy Iteration and SARSA



SARSA example



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

The Q value for all places is zero.

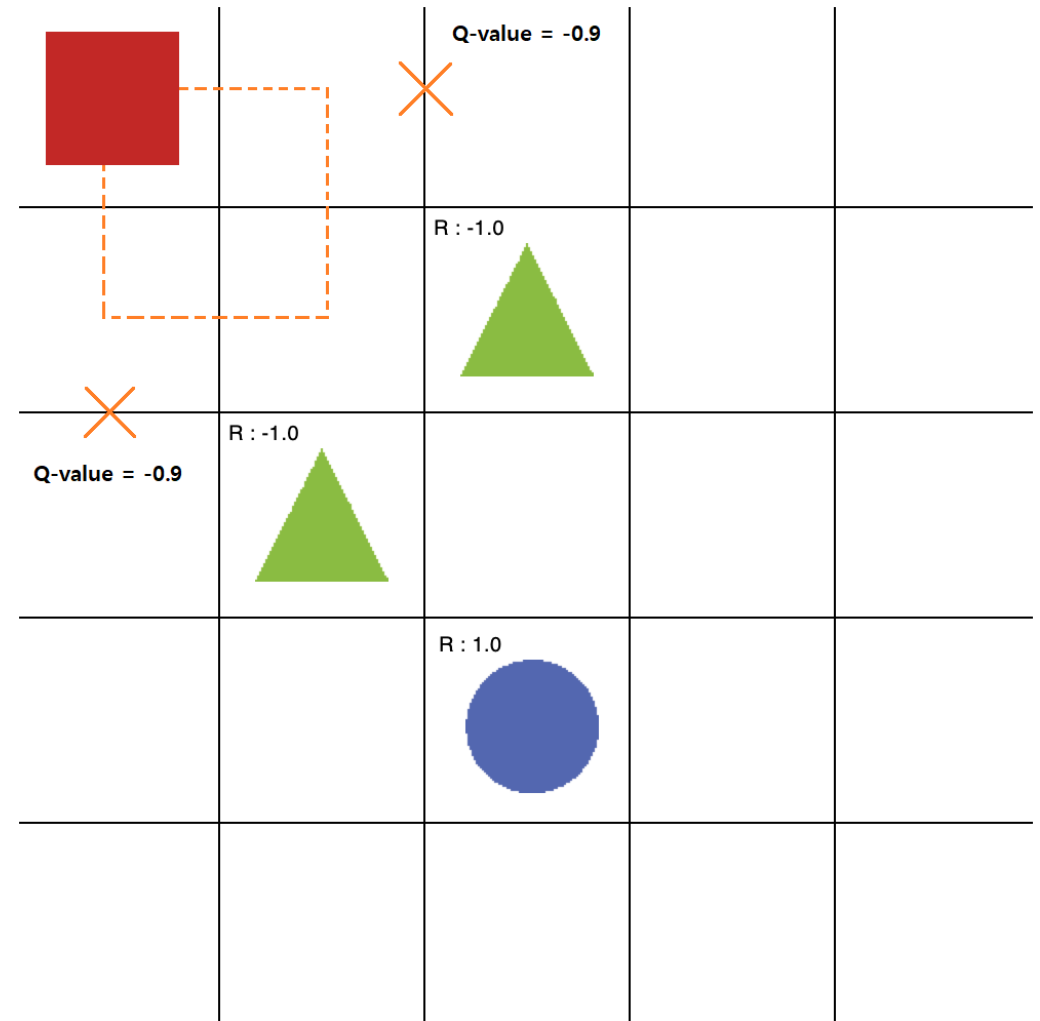
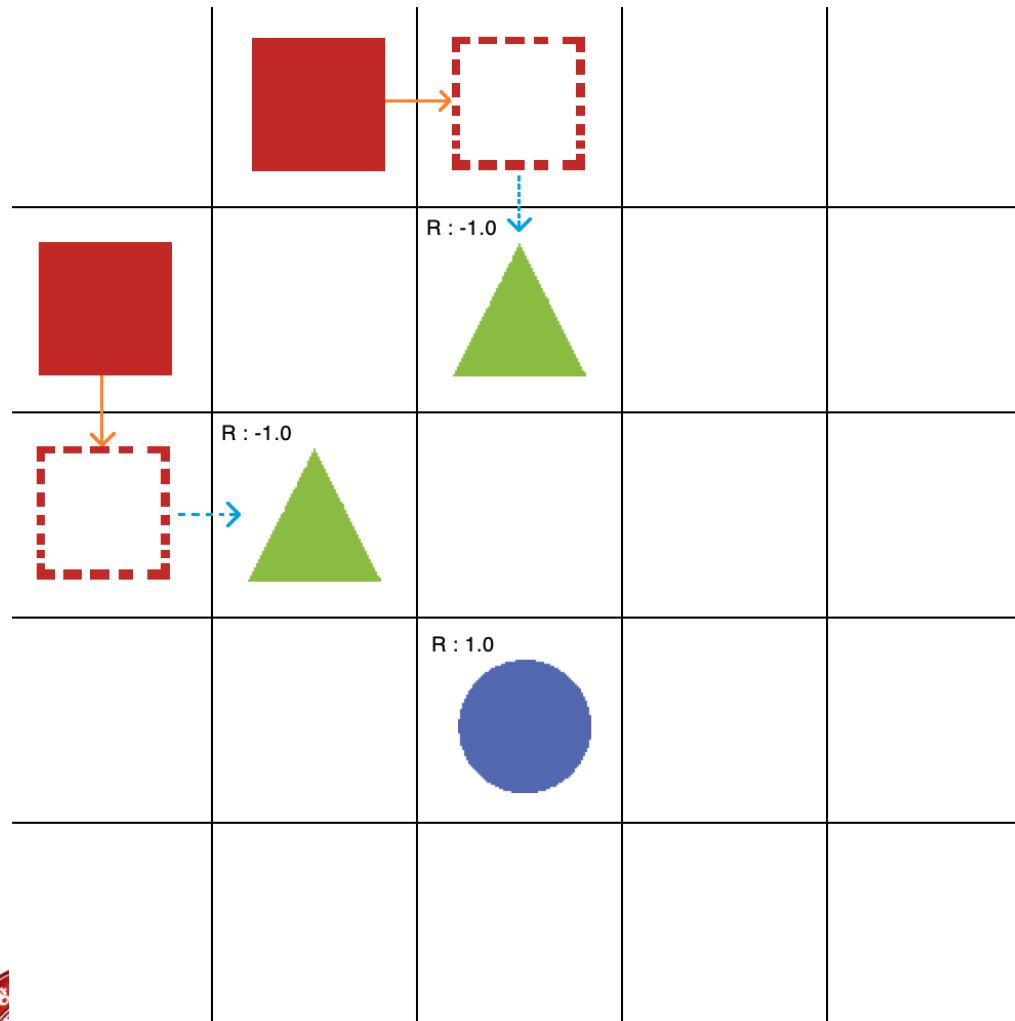
$\alpha = 1, \gamma = 0.9,$

1. next_action = up
 $Q(S_t, A_t) = -0.9$
2. next_action = down
 $Q(S_t, A_t) = 0$
3. next_action = right
 $Q(S_t, A_t) = 0.9$
4. next_action = left
 $Q(S_t, A_t) = 0$

***next_action follows the ϵ - Greedy policy.**



Salsa Limits



Q-Learning

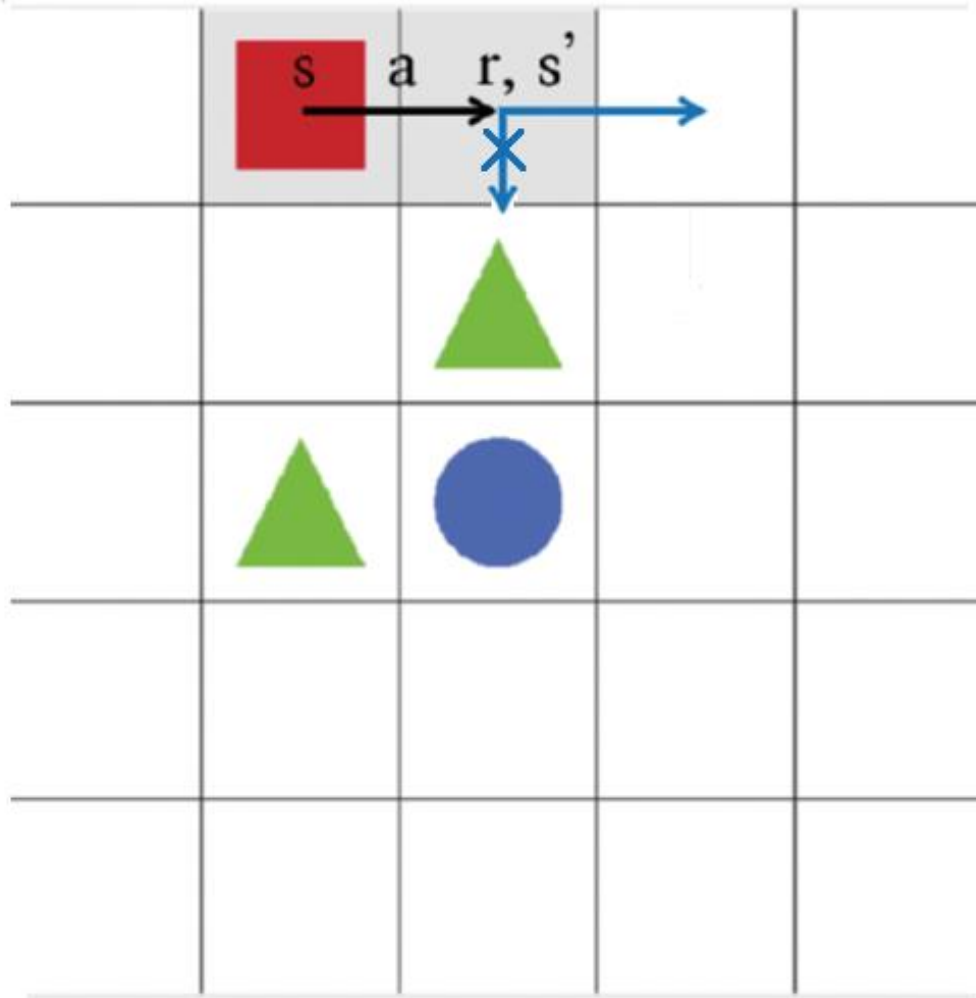
$$\text{SARSA : } Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$



$$Q \text{ Learning: } Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$



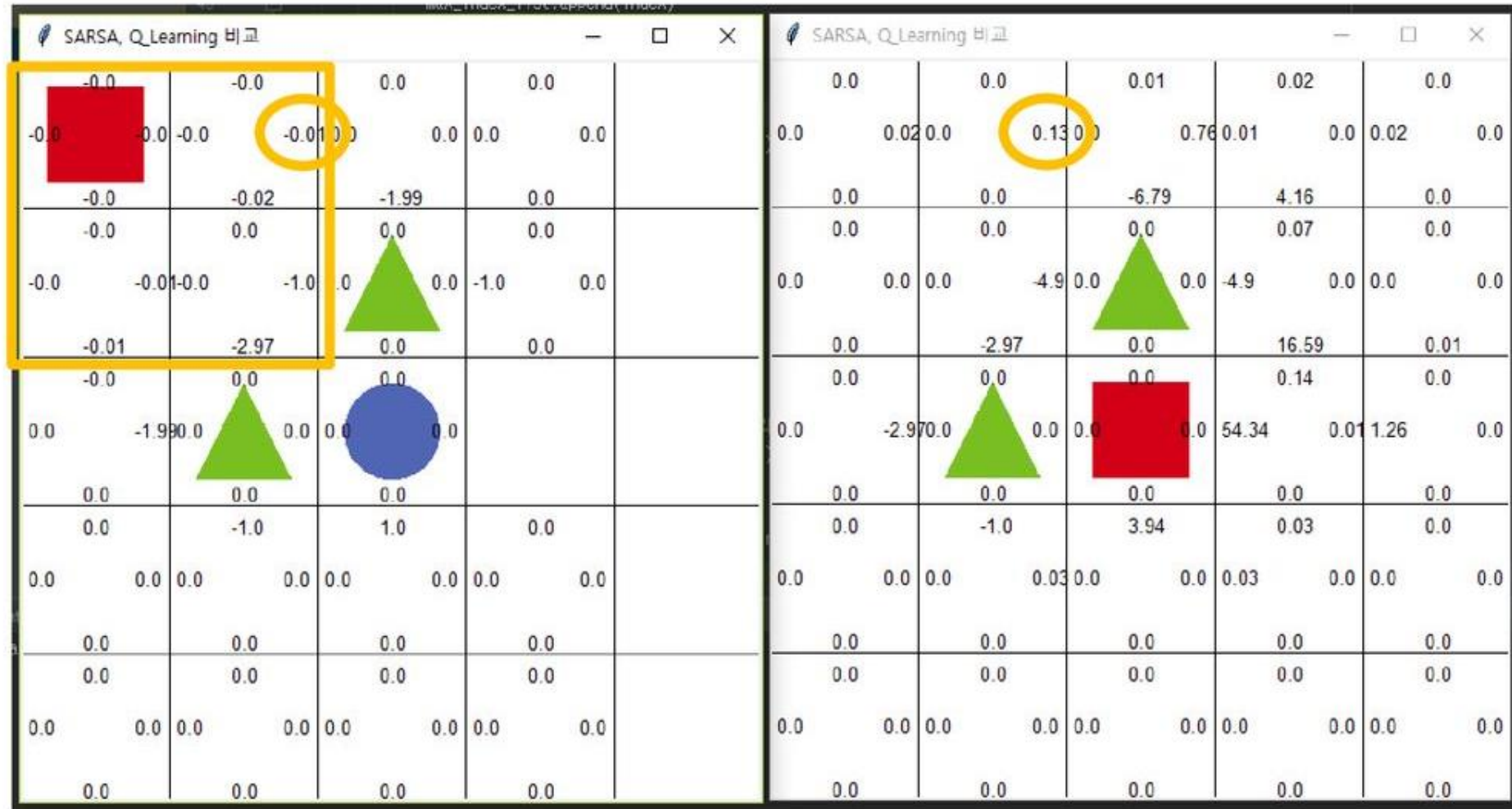
Q-Learning



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



comparison of SARSA and Q-Learning



< SARSA (left) and Q-Learning (right) >



Problems with Dynamic Programming

1. Calculation Complexity
2. Curse of Dimension
3. Complete information about the environment



Problems with Dynamic Programming


1. Calculation Complexity

2. Curse of Dimension

3. Complete information about the environment → Solve (Monte Carlo Prediction , SARSA, Q-learning)

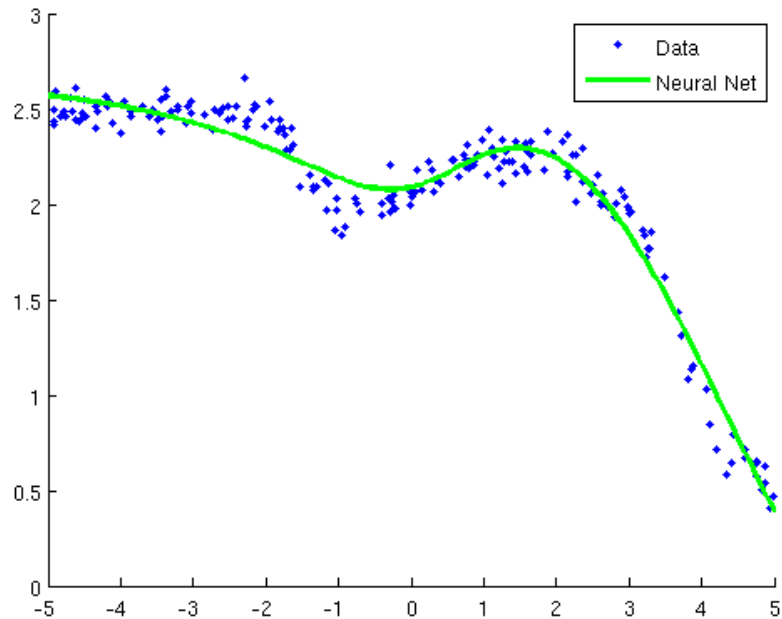
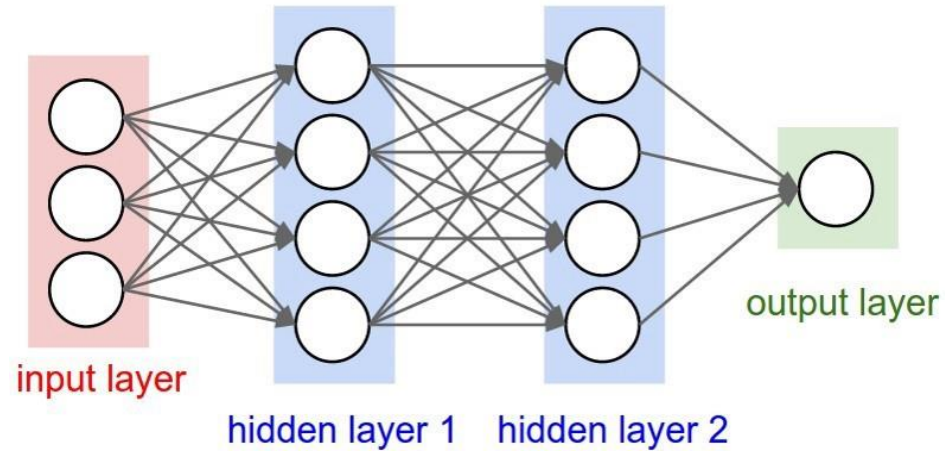


Problems with Dynamic Programming

- 
1. Calculation Complexity
 2. Curse of Dimension
- }
- Approximate the Q function \longrightarrow Solve (Deep Q-Learning, Deep Q-network)
3. Complete information about the environment \longrightarrow Solve (Monte Carlo Prediction , SARSA, Q-learning)



Deep Q-Learning



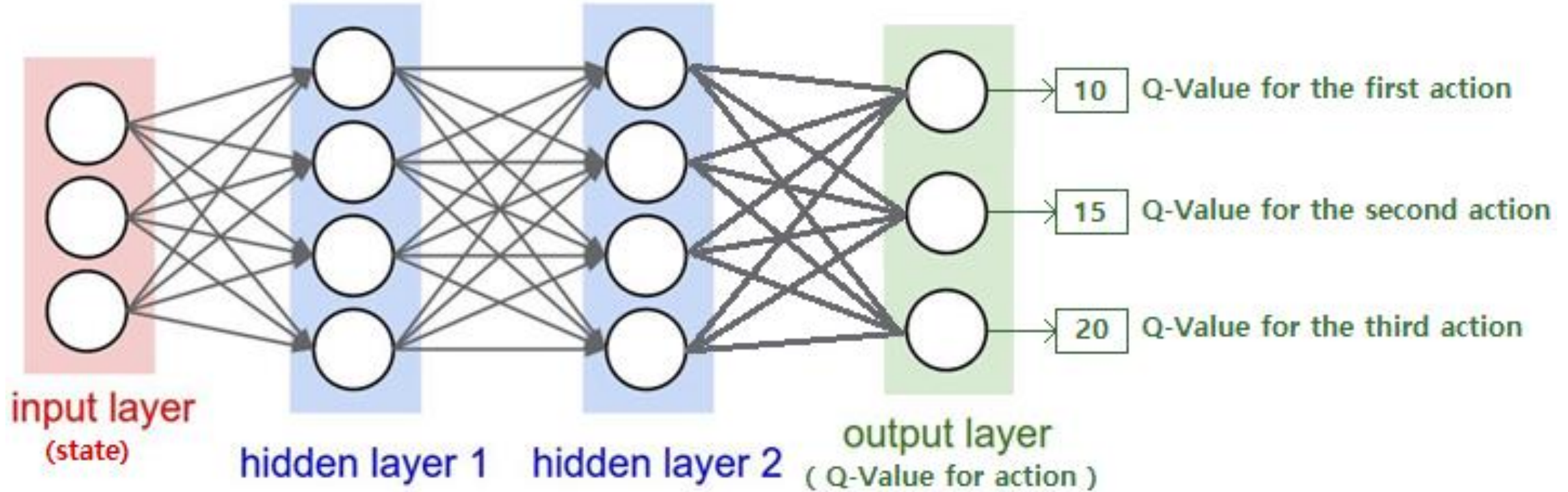
input = state

output = values of Q-function by action

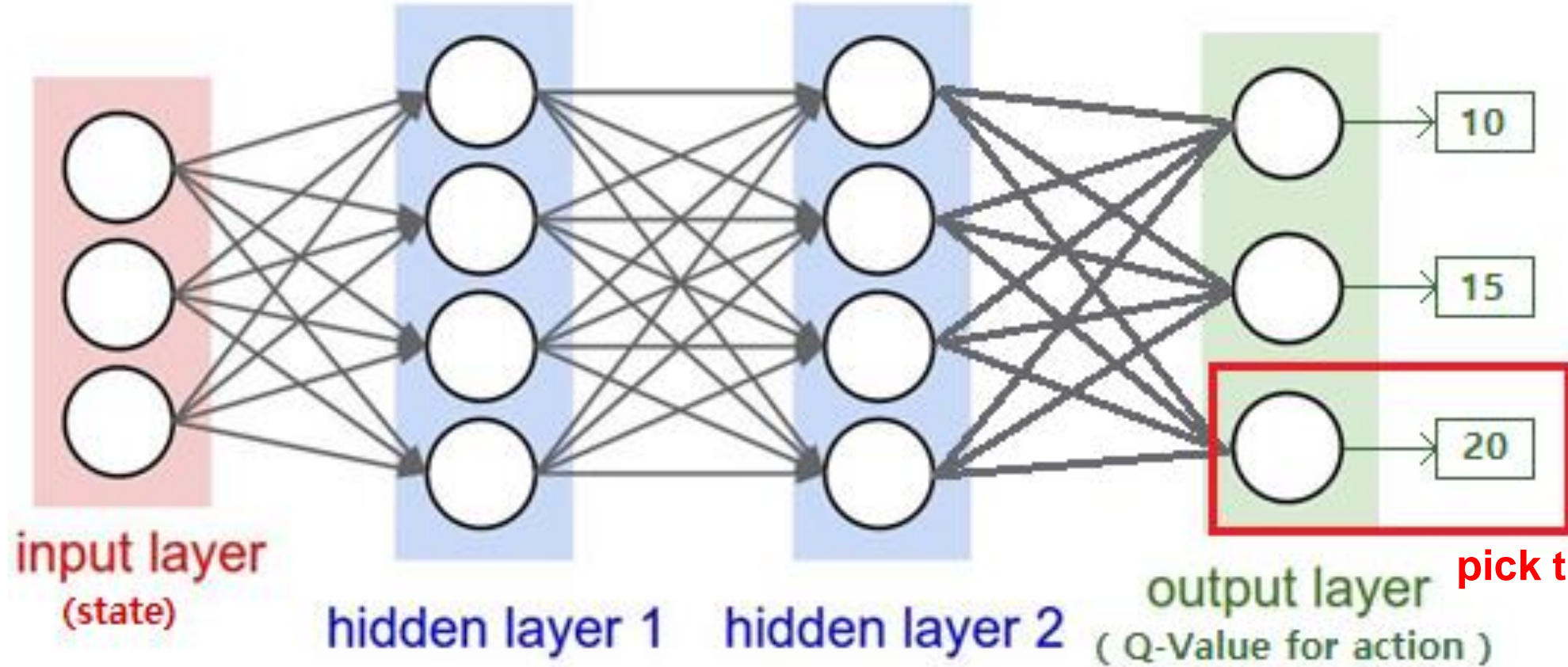
*If there are three actions, the output returns the Q-value for each action.

The next action is the one with the highest Q value.

Deep Q-Learning example

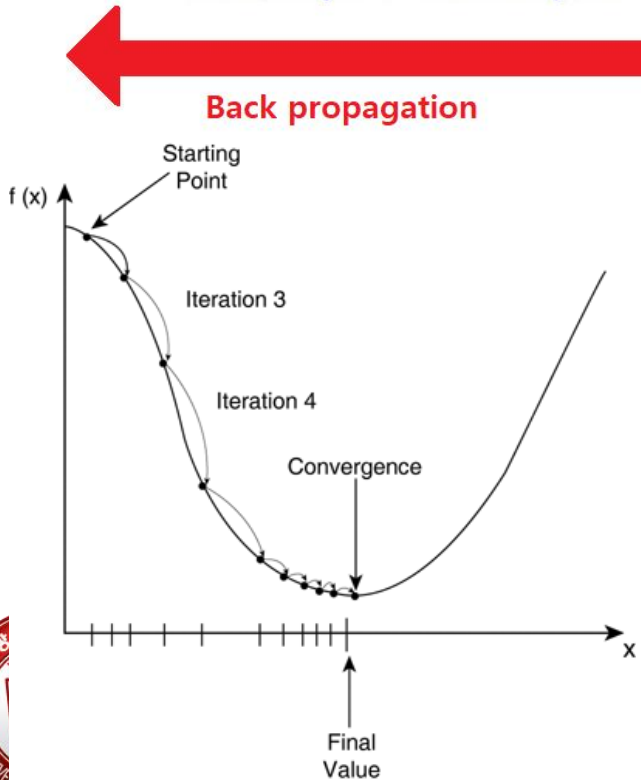
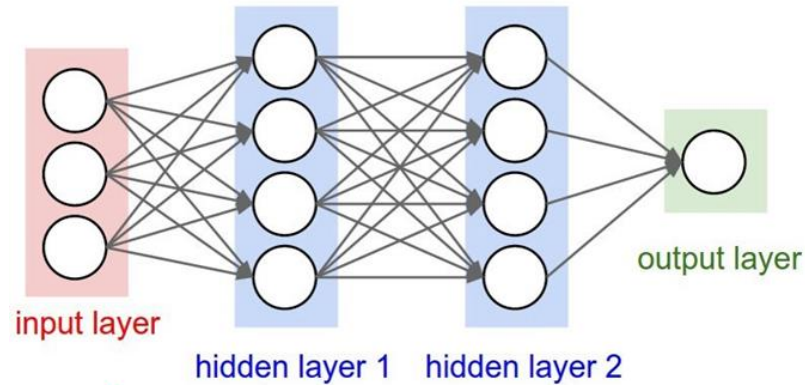


Deep Q-Learning example



pick the action = 2

update of Deep Q-Learning



MSE in Deep Learning:

$$\text{MSE} = (\text{answer} - \text{prediction})^2 = \frac{1}{n} \sum (y_i - \hat{y}_i)^2$$

MSE in Deep Q-Learning:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R + \gamma \max_{a'} Q(s', a') - Q(S_t, A_t))$$

This can be found in Q-function, corresponding to the role of correctness and prediction.

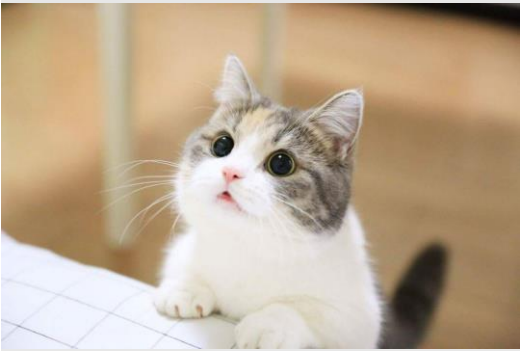
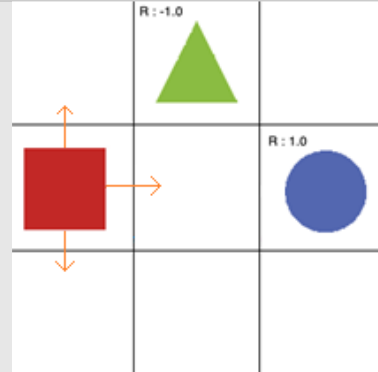

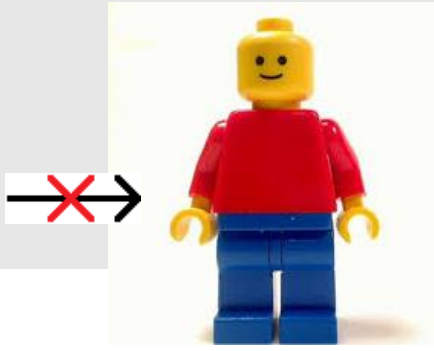
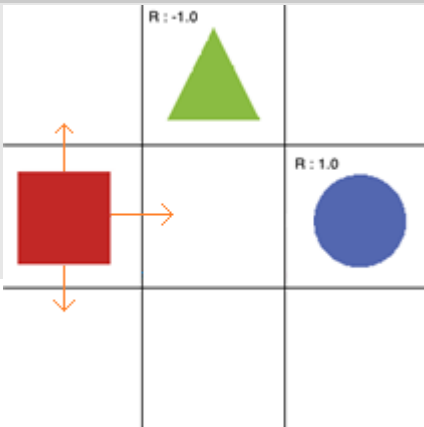
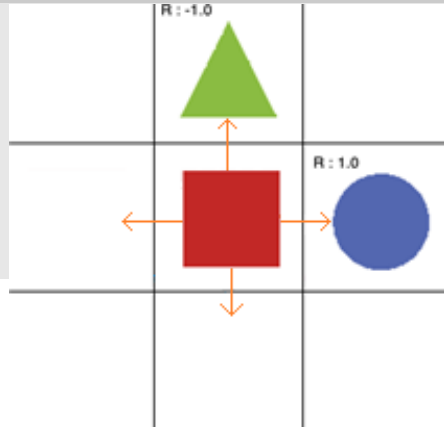
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\underbrace{R + \gamma \max_{a'} Q(s', a')}_{\text{answer}} - \underbrace{Q(S_t, A_t)}_{\text{prediction}})$$

Therefore, MSE in Deep Q-Learning are as follows.

$$\text{MSE} = (\text{answer} - \text{prediction})^2 = (R + \gamma \max_{a'} Q(s', a') - Q(s, a))^2$$

Problems with Deep Q-Learning

The problem with deep Q-Learning comes from the gap with the Deep Neural Network.

General deep Learning		Deep Q-Learning	
Given a label (correct answer)		The label (correct answer) is unclear	
	Answer = Cat		Answer = right. Best action??
Data is independent		Data is dependent	
			

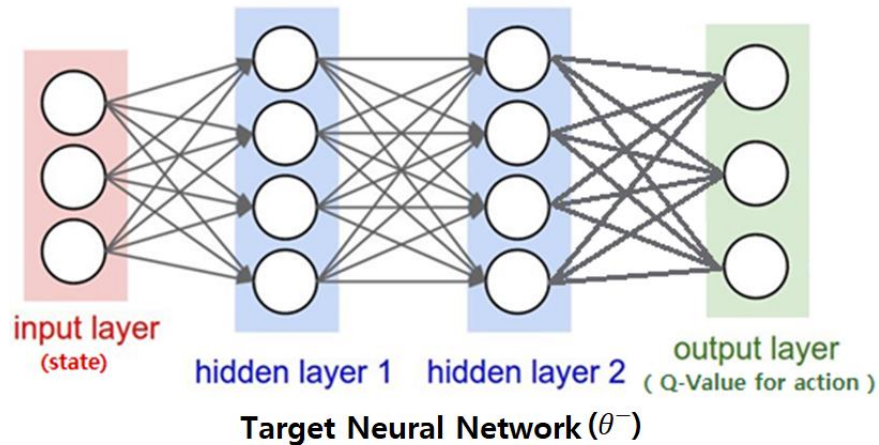
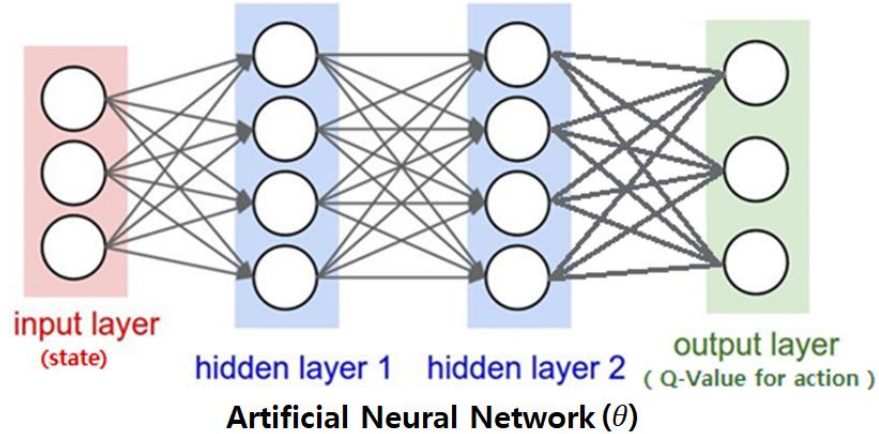
Problems with Deep Q-Learning

1. The label (correct answer) is unclear
Each time Deep Q-Learning updates, the correct answers and predictions that target the update continue to change.
→ The correct answer is obtained by adding a target neural network whose value does not change over time
2. Samples (MDP) are dependent on each other
Deep Q-Learning can fall into one trend. As a result, correlations between samples can be generated, allowing only bad situations to be learned.
→ The replay memory is used to update the artificial neural network with random samples.

→ To solve the above problem, Deep Q-network was developed.



Deep Q-Network



$$\begin{aligned} \text{MSE of Deep Q-Learning} &= (\text{answer} - \text{prediction})^2 \\ &= (R + \gamma \max_{a'} Q(s', a', \theta) - Q(s, a, \theta))^2 \end{aligned}$$

*The problem is that the answer that is the target of the update is constantly changing.

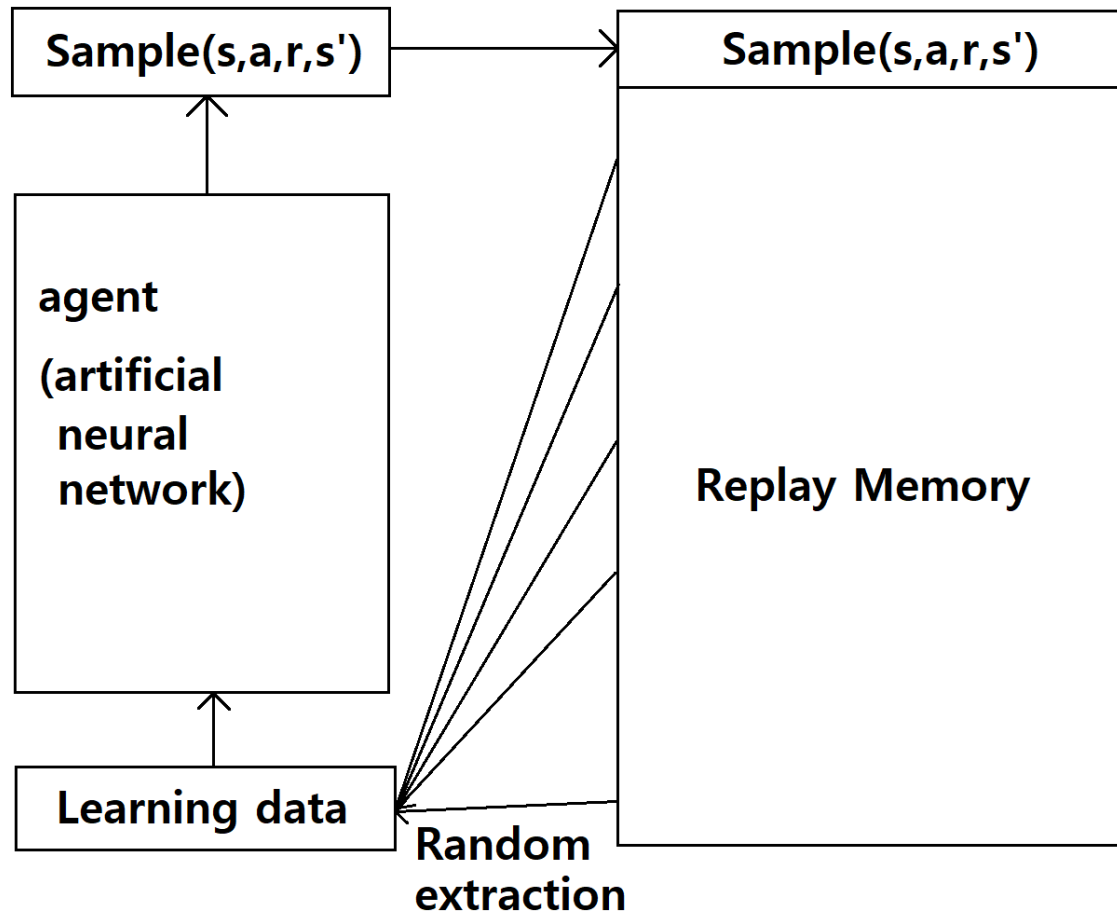
In neural networks, the problem gets worse if the answer keeps changing. Therefore, we need another neural network that does not change the answer for a certain period of time.

This is called the target neural network, and θ^- is used to distinguish it.

$$\begin{aligned} \text{MSE of Deep Q-Network} &= (\text{answer} - \text{precision})^2 \\ &= (R + \gamma \max_{a'} Q(s', a', \theta^-) - Q(s, a, \theta))^2 \end{aligned}$$



Deep Q-Network



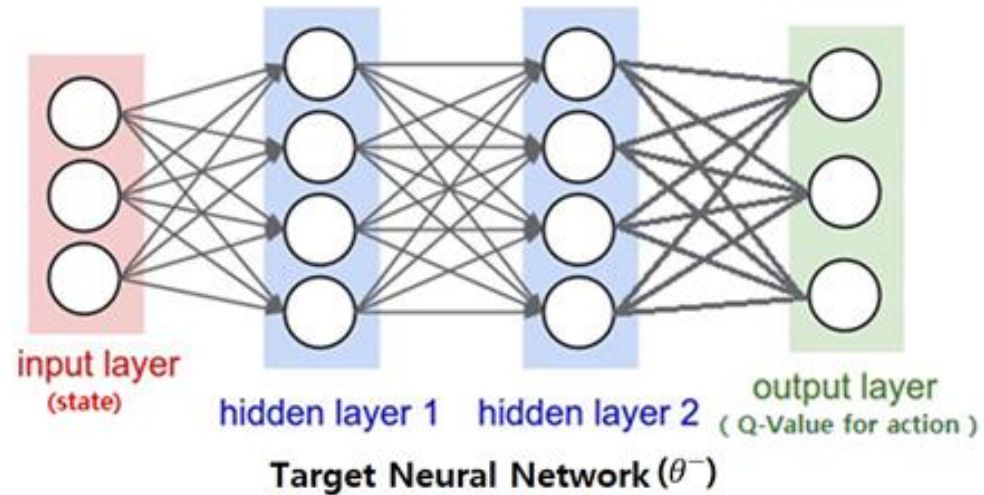
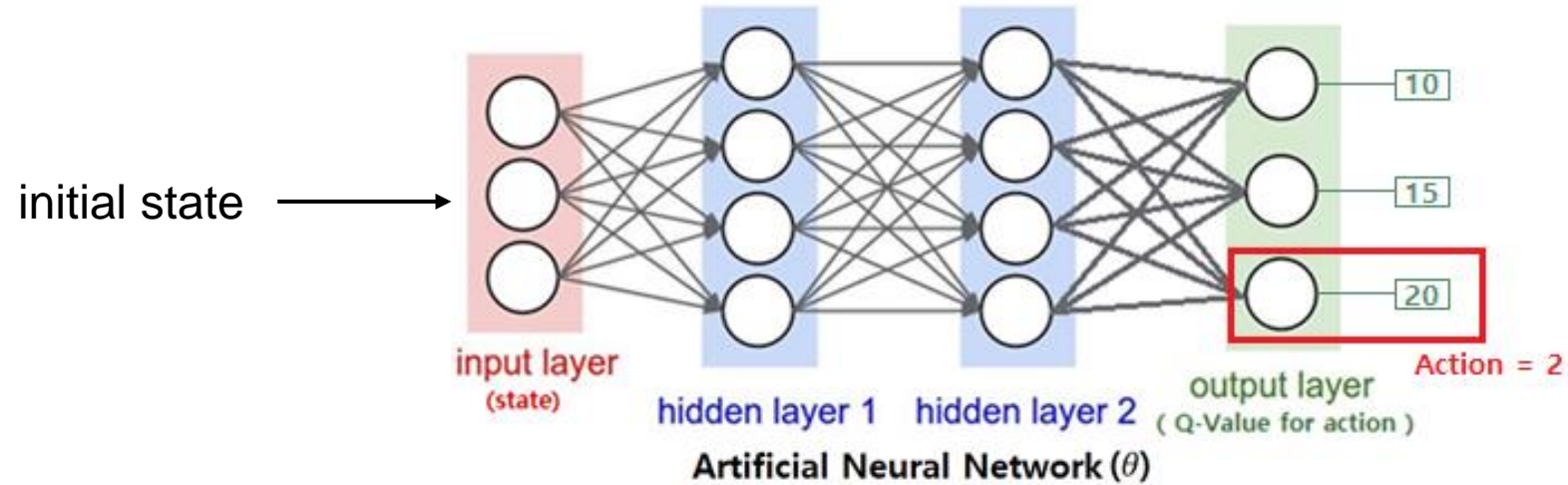
Agents explore → Get a sample(s,a,r,s') → Save to Replay Memory → Randomly extract data from Replay Memory → Agents explore → ...

It can prevent learning only in bad situations.

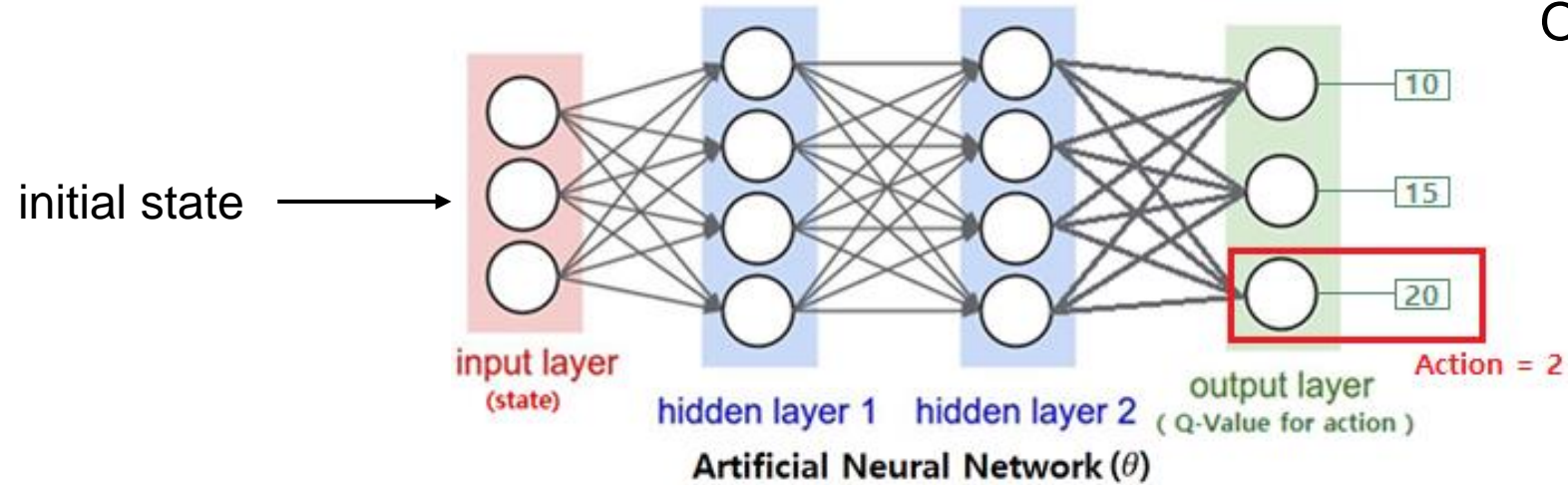
Learning is stable because the neural network is updated using multiple samples that are not time-related.



Deep Q-Network example

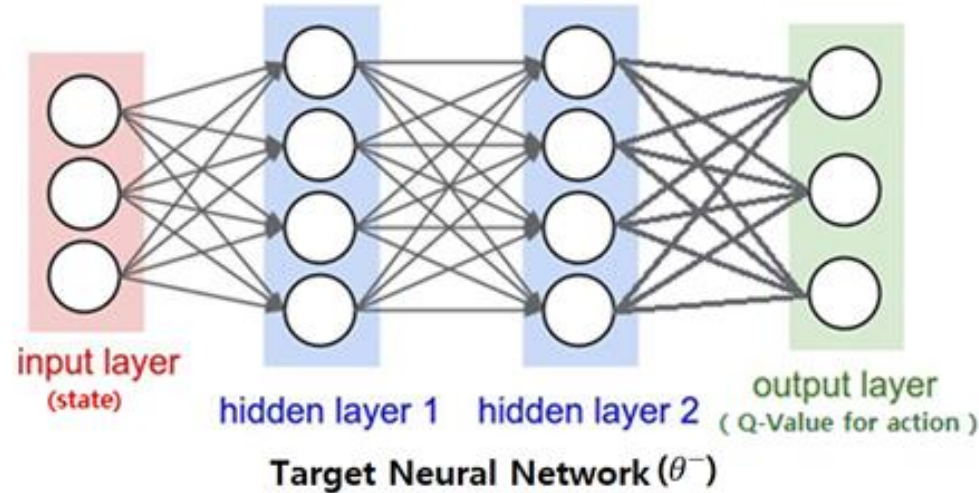


Deep Q-Network example

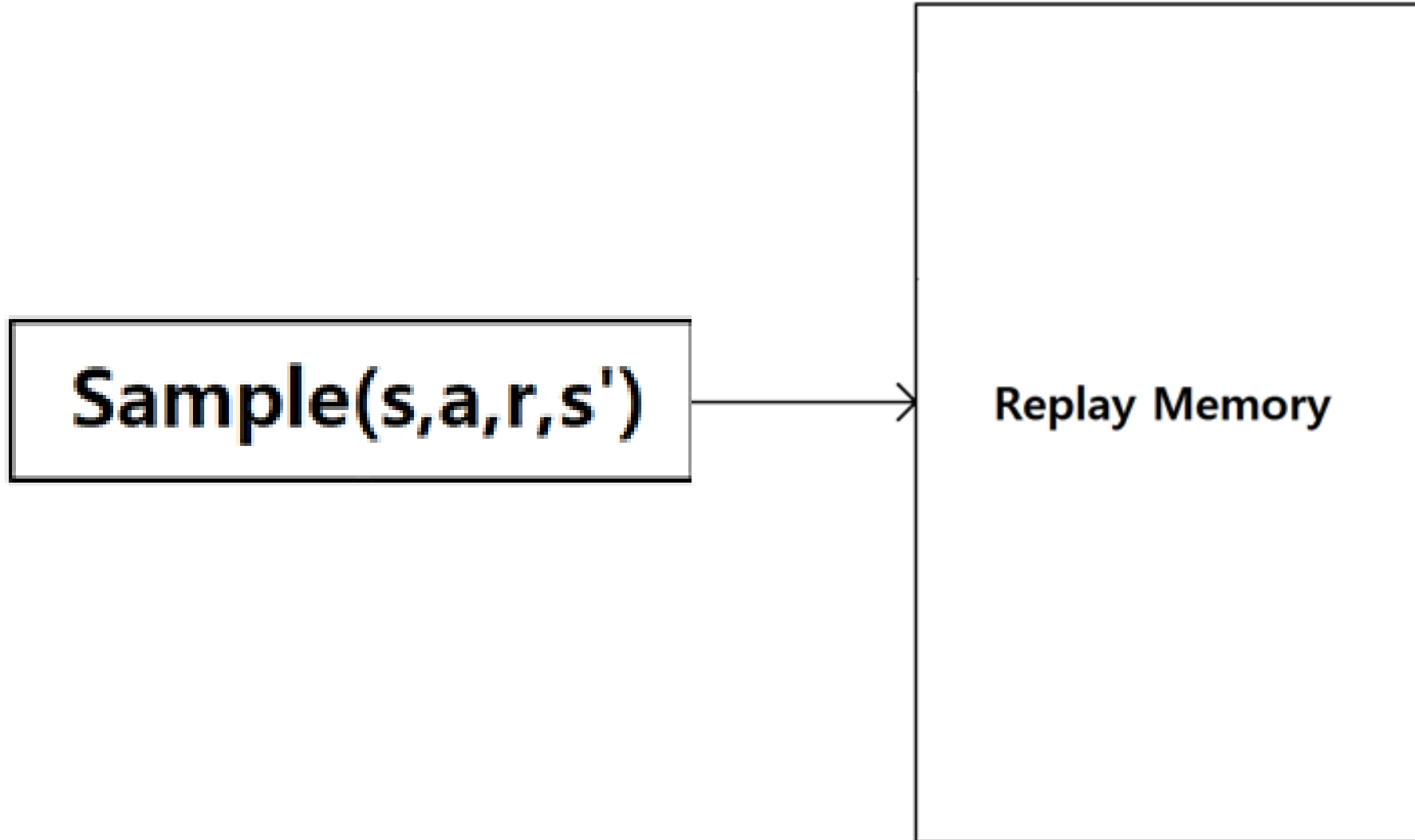


One time step...

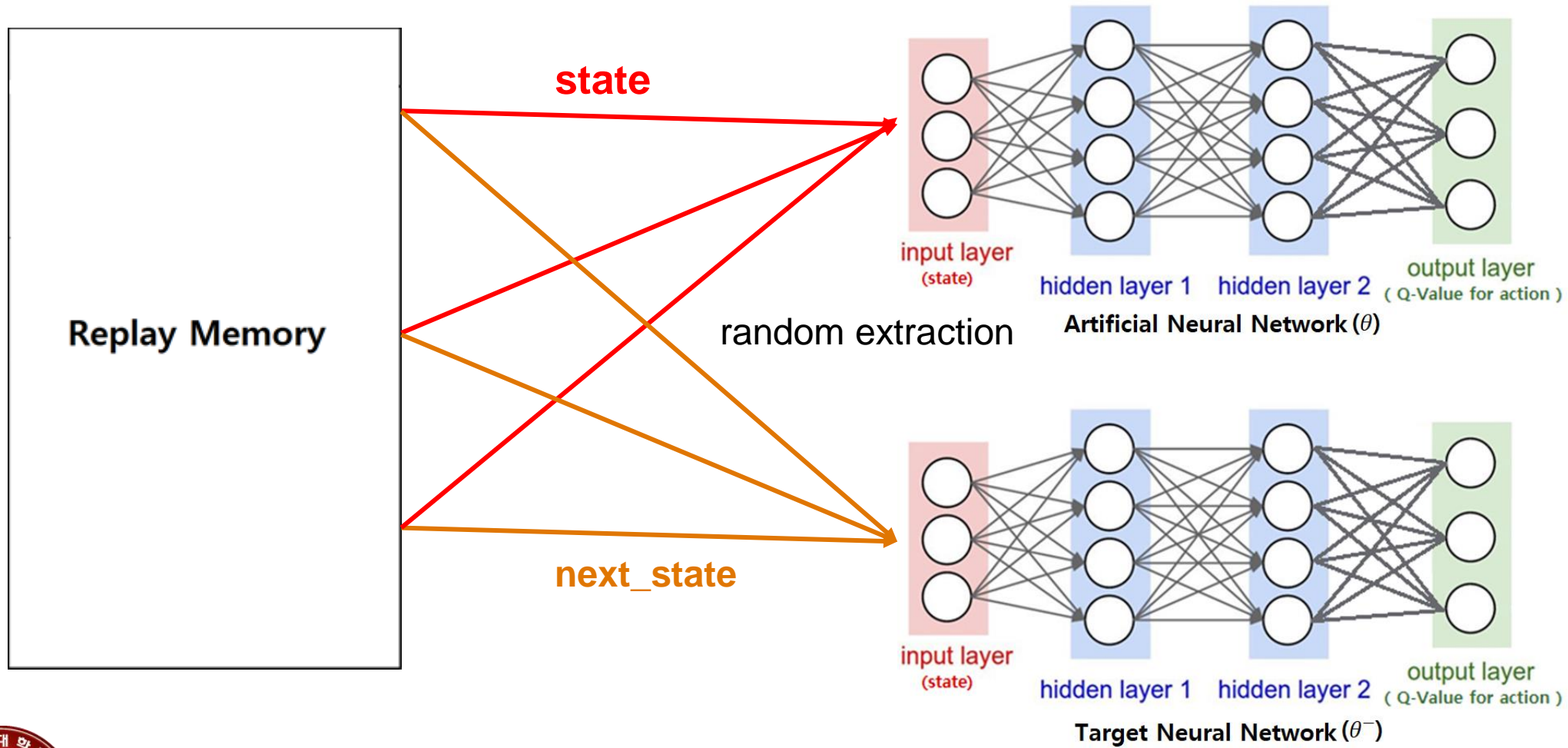
get Reward, Next_state



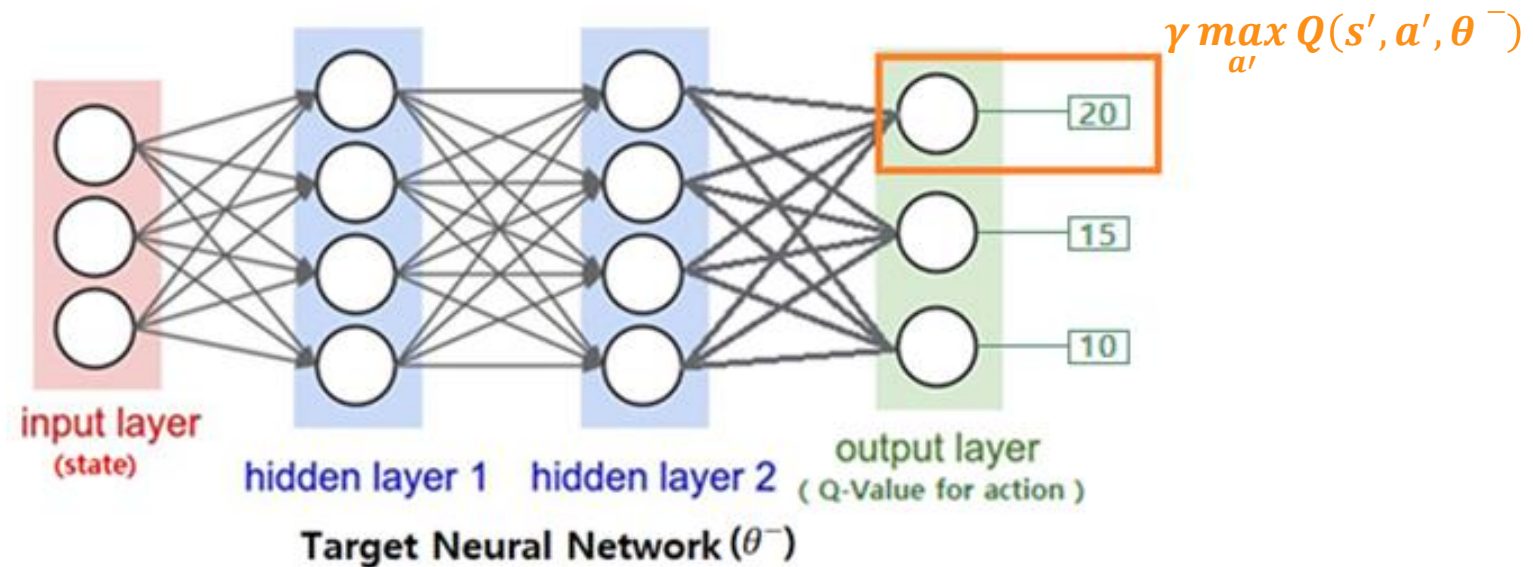
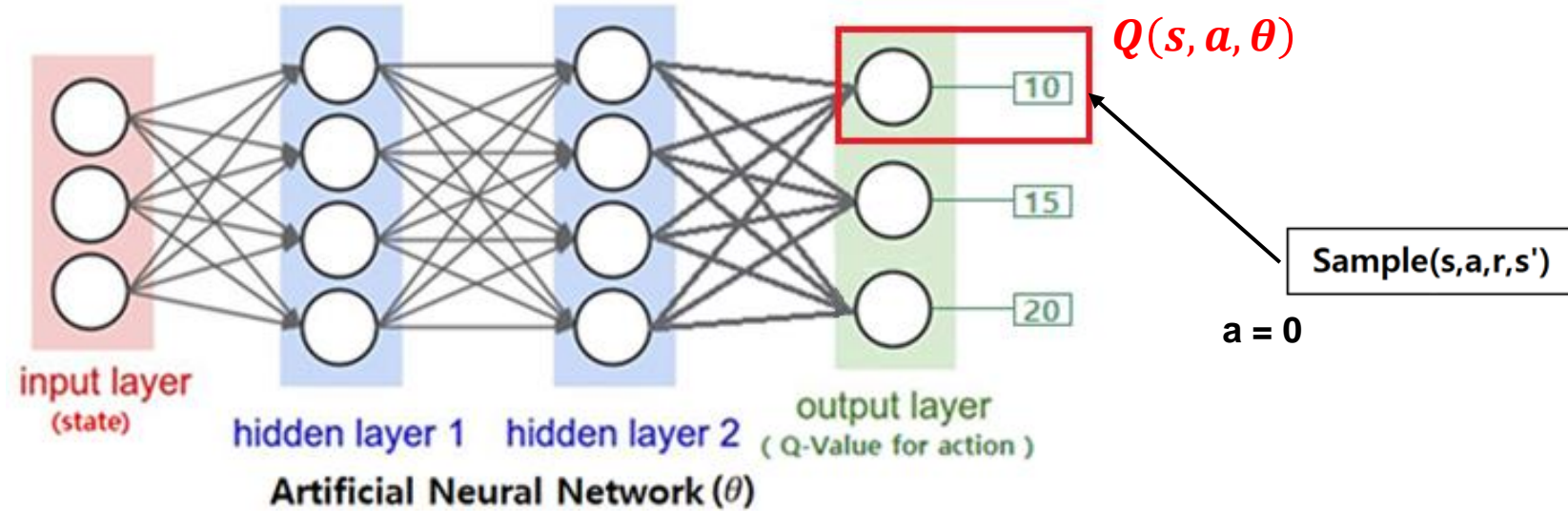
Deep Q-Network example



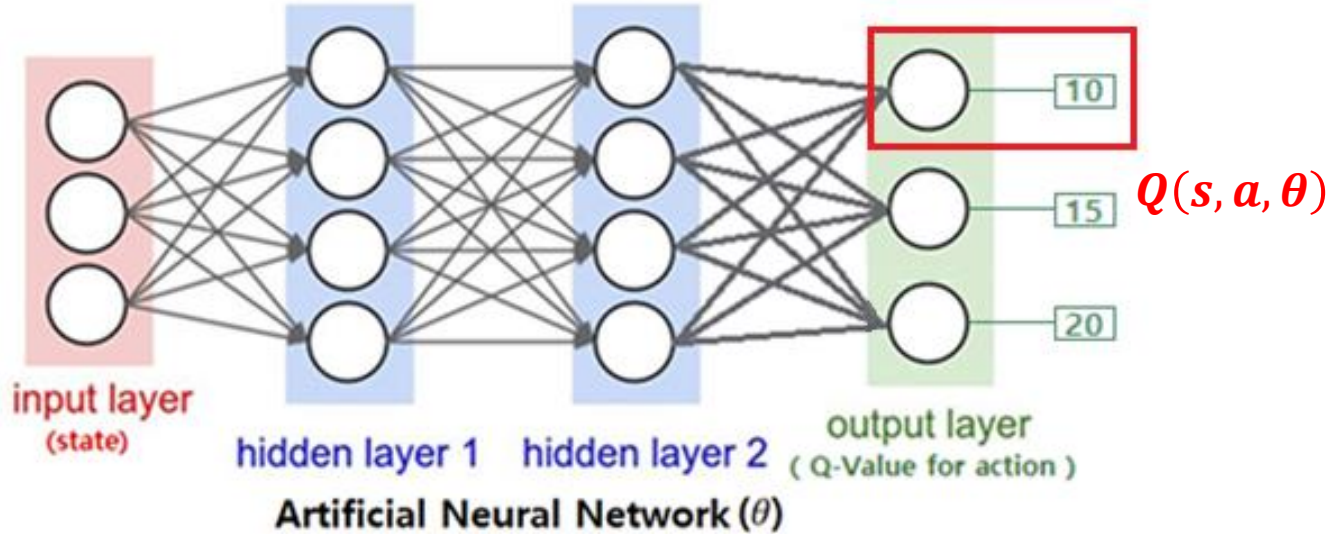
Deep Q-Network example



Deep Q-Network example

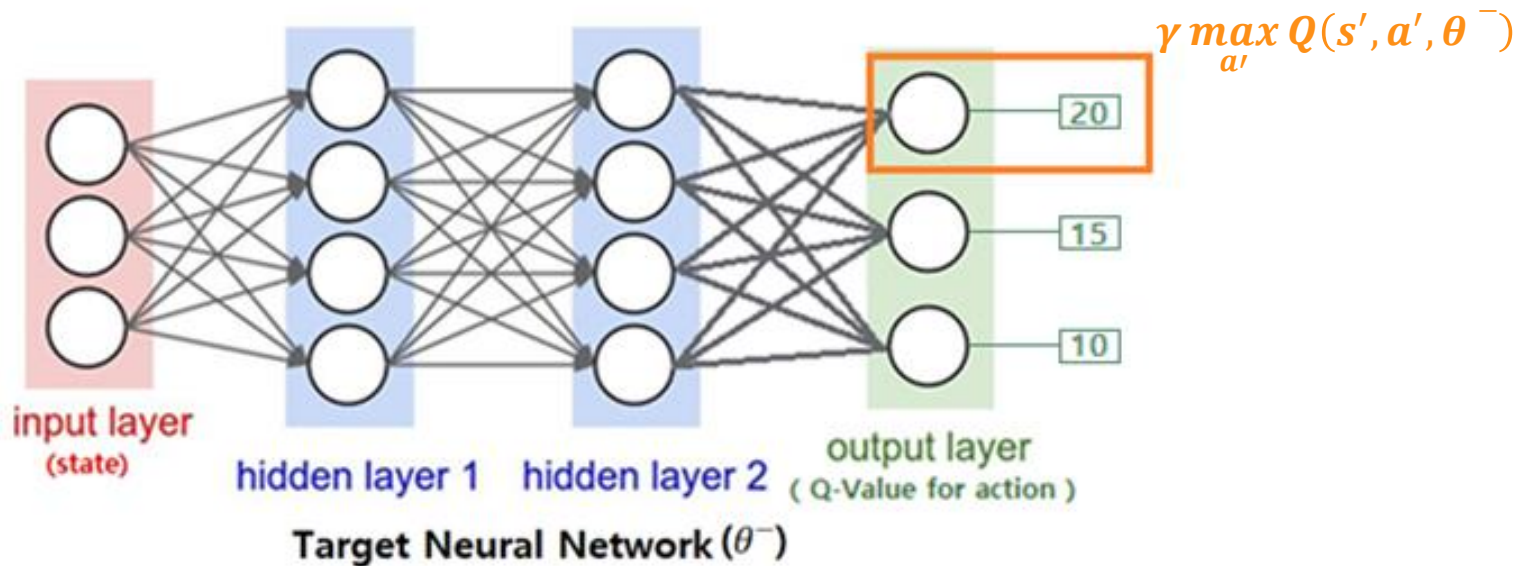


Deep Q-Network example

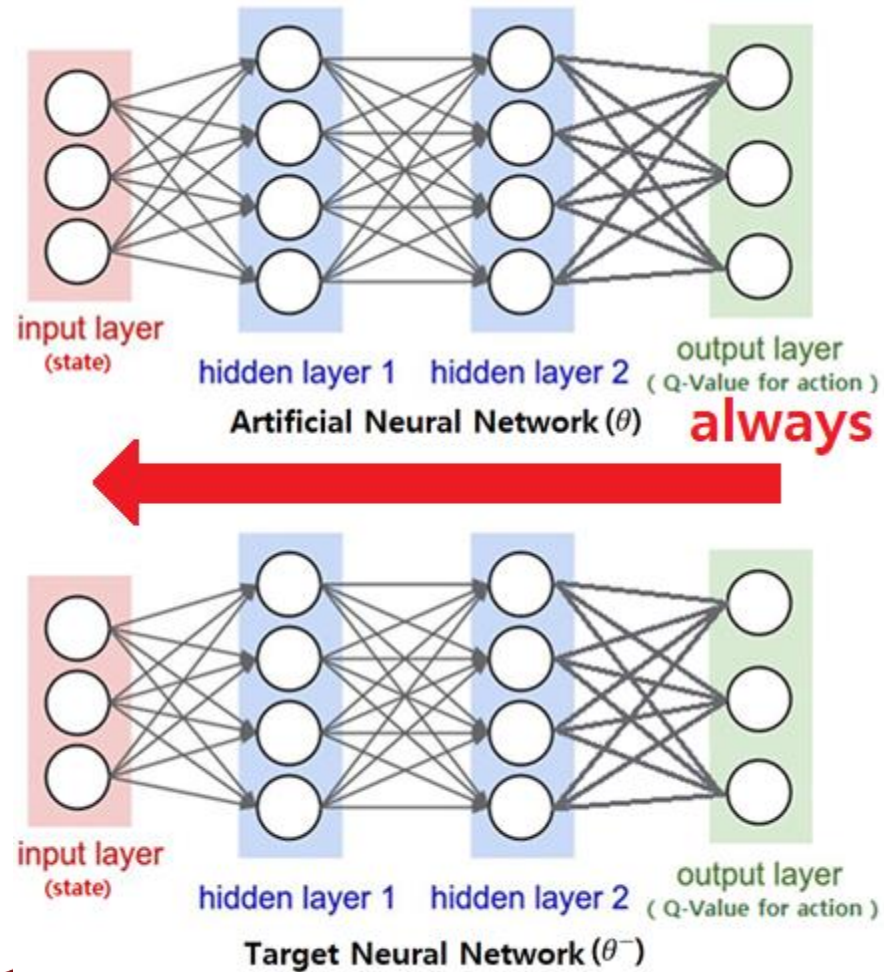


$$\text{MSE} = (\text{answer} - \text{prediction})^2$$

$$= (R + \gamma \max_{a'} Q(s', a', \theta^-) - Q(s, a, \theta))^2$$



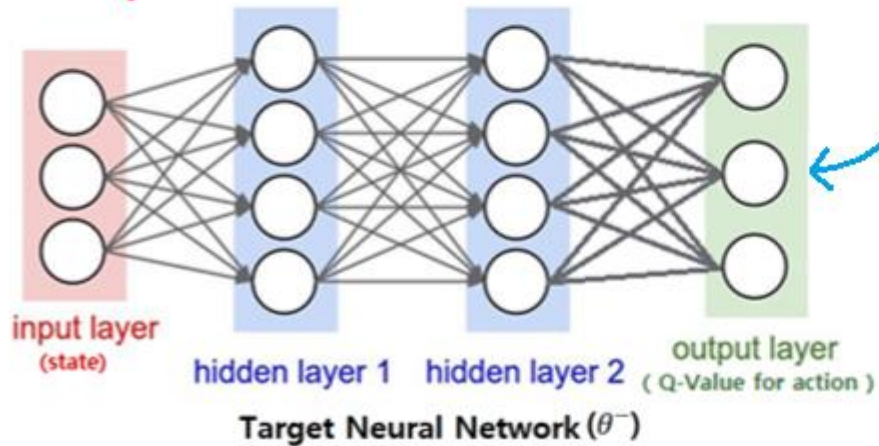
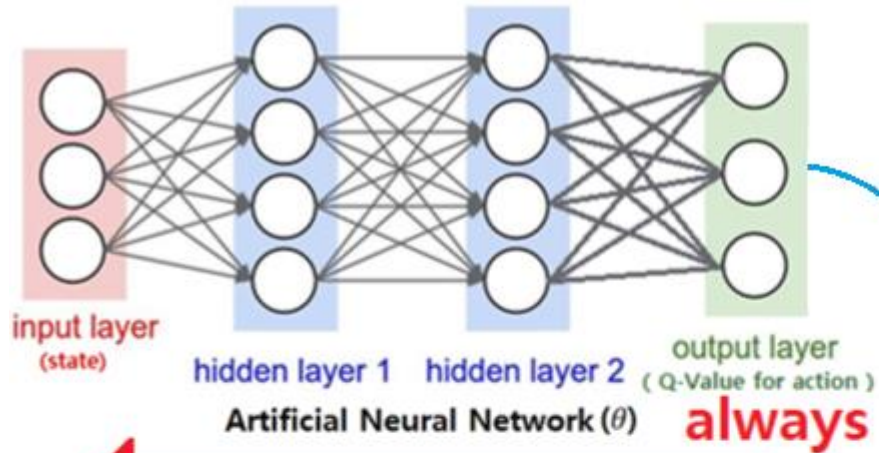
Deep Q-Network example



$$\text{MSE} = (\text{answer} - \text{prediction})^2 = (R + \gamma \max_{a'} Q(s', a', \theta^-) - Q(s, a, \theta))^2$$

The update of the Artificial neural network updates every time it is operated.

Deep Q-Network example



Copy all weights

$$\text{MSE} = (\text{answer} - \text{prediction})^2 = (R + \gamma \max_{a'} Q(s', a', \theta^-) - Q(s, a, \theta))^2$$

After a certain number of times, the target neural network is copied with the weight of the artificial neural network.

