

UNIST HeXA 여름 특강

# 강화학습 기초

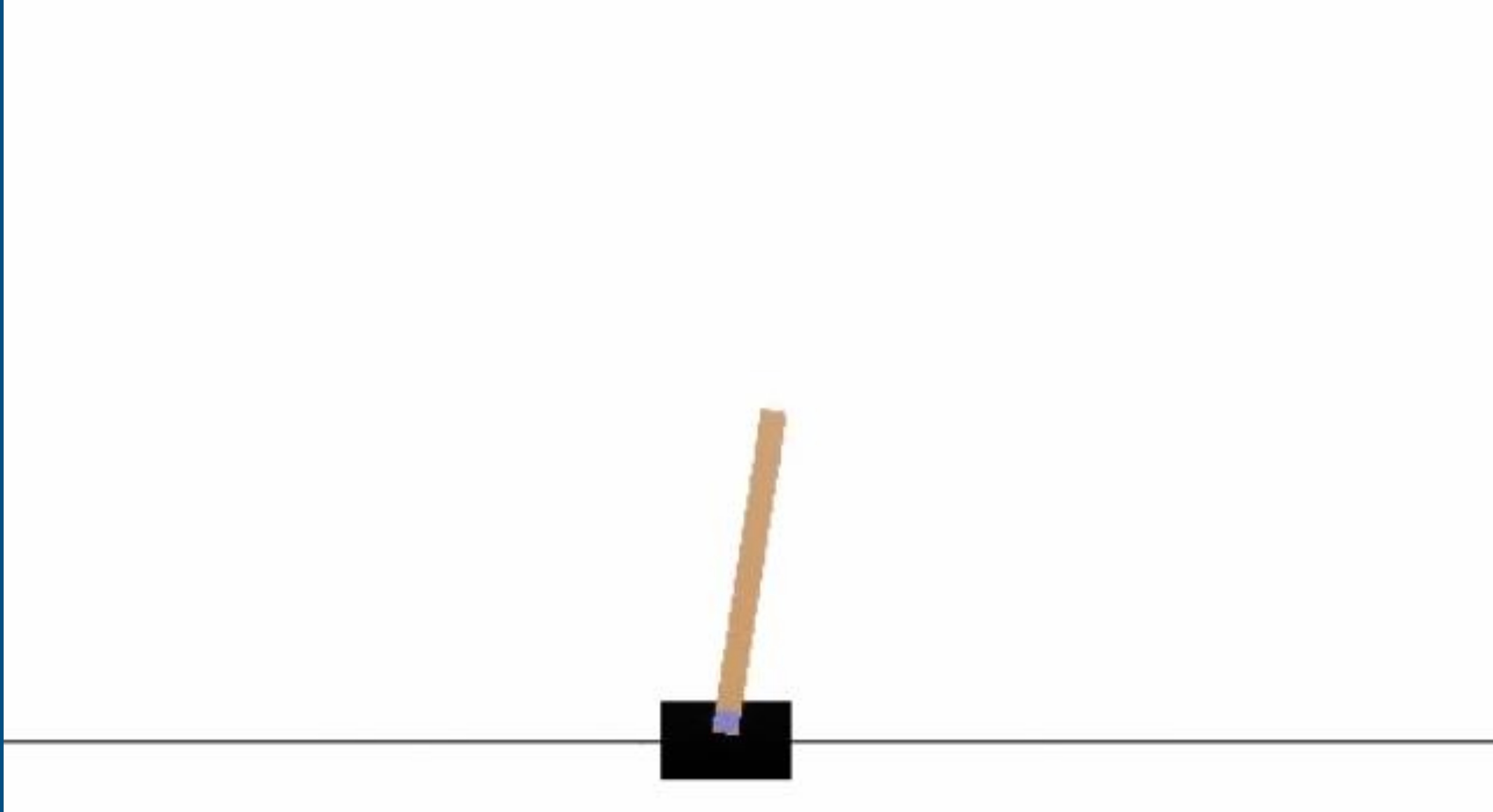
Chris Ohk

utilForever@gmail.com

# 카트폴 (Cartpole)

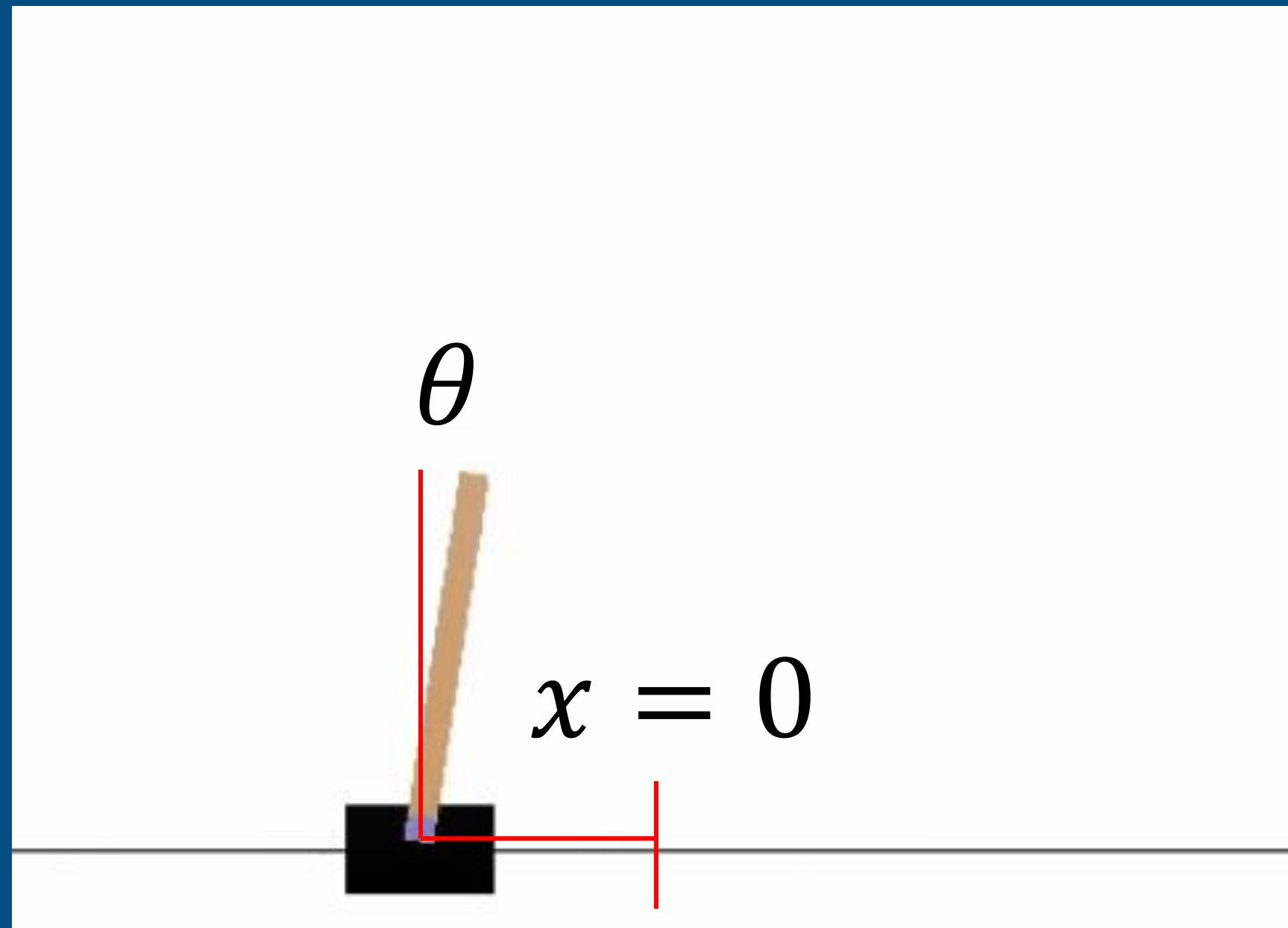
---

UNIST HeXA  
Week 7



# 카트폴 (Cartpole)

UNIST HeXA  
Week 7

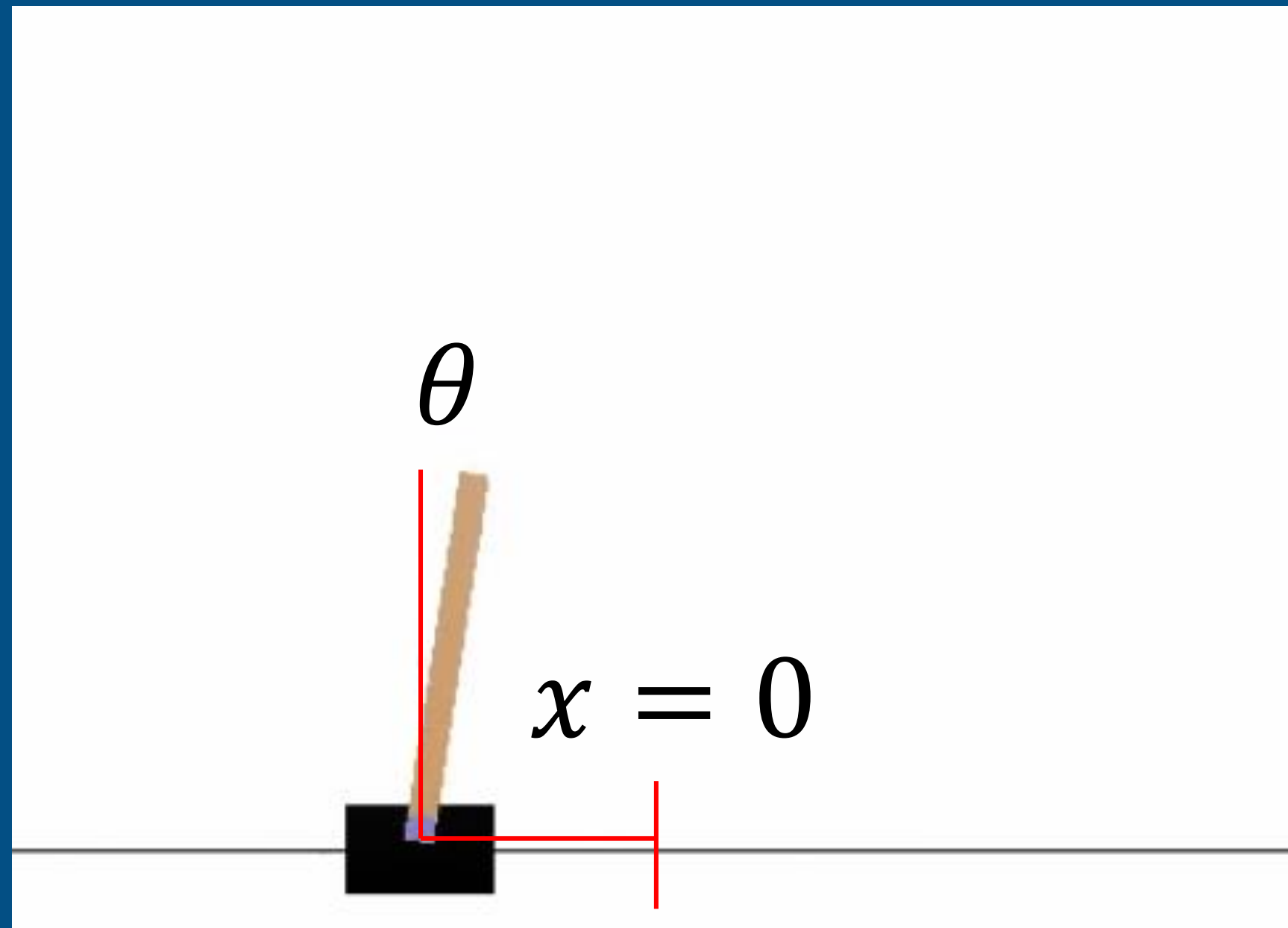


검은색 상자(카트)를  $+1, -1$ 의 힘으로 밀어  
갈색 막대(폴)이 넘어지지 않게 하는 문제

카트의 위치  $x$ 가  $\pm 2.5$  밖으로 넘어가거나  
 $\theta$ 가  $\pm 15^\circ$  밖으로 넘어가면 에피소드가 끝난다.

# 카트폴 (Cartpole)

UNIST HeXA  
Week 7



상태  $s$

$$s = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

위치  
속도  
각도  
각속도

행동  $a$

$$a = \begin{bmatrix} +1 \\ -1 \end{bmatrix}$$

# 카트폴 (Cartpole)

---

UNIST HeXA  
Week 7

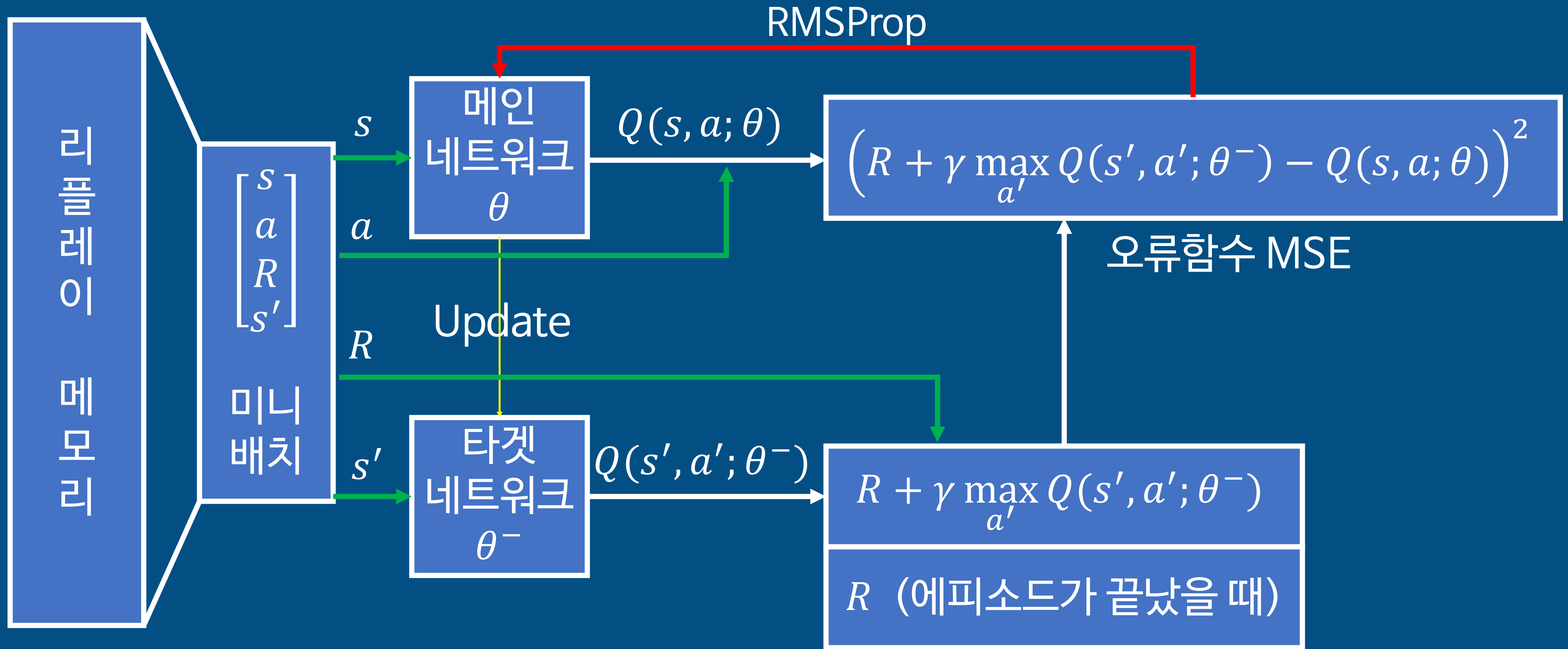
상태의 위치, 속도, 각도, 각속도 모두 연속적인 실수다.  
그러므로 카트폴의 경우에는 테이블을 만들어 풀 수 없고,  
큐함수를 근사하는 함수를 만들어야 한다. → 인공신경망을 이용

딥살사 : 온폴리시(On-Policy)인 살사 기반

오프폴리시(Off-Policy)인 큐러닝 + 인공신경망? → 딥-큐러닝(DQN)



# DQN 구조 (맞보기)



## DQN의 특징

1. 오프폴리시 (Off-Policy)
2. 리플레이 메모리 + 미니배치
3. 타겟 신경망

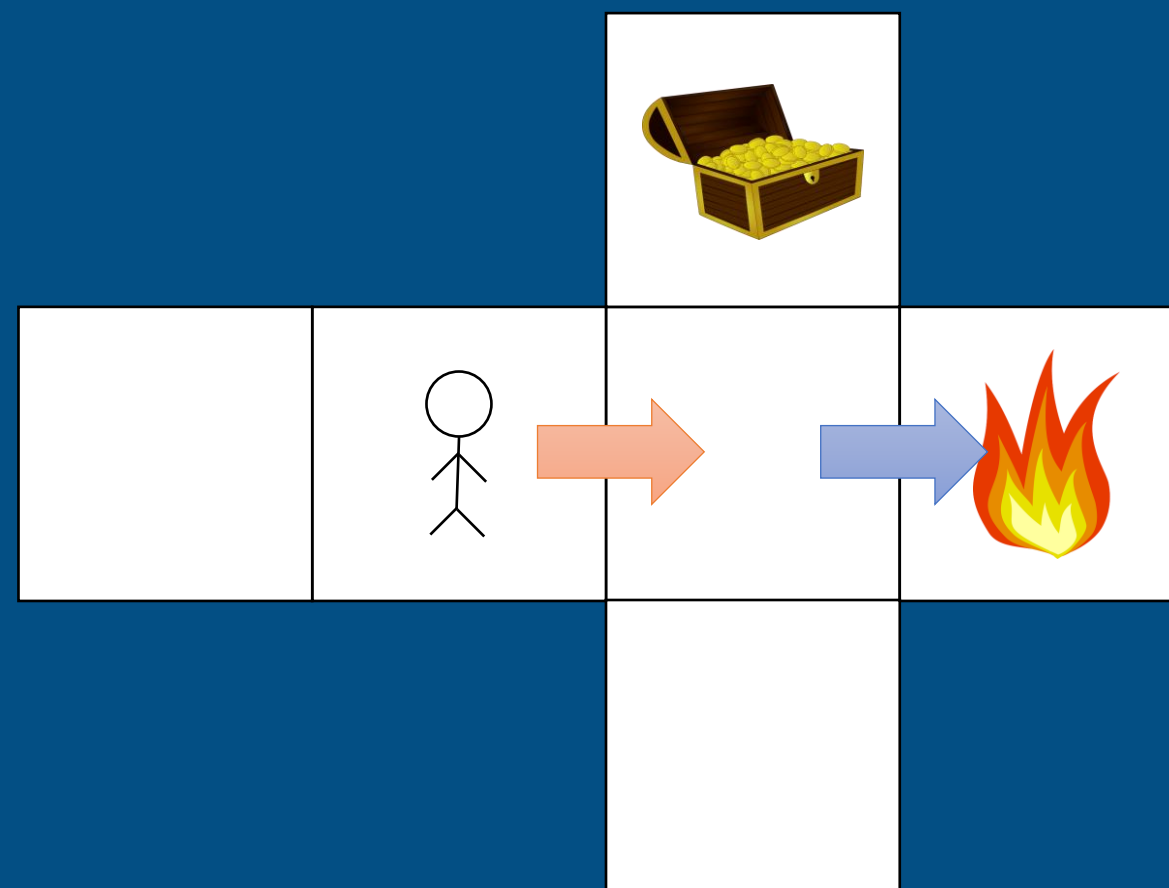


온폴리시 (예 : 살사)

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

학습하는 정책과 행동을 고르는 정책이 항상 같아야 한다.

- 다음 상태에서 한 안 좋은 행동이 현재에 큰 영향을 미친다.
- 정책을 업데이트하면 과거의 경험들을 학습에 이용 불가능해 비효율적이다.

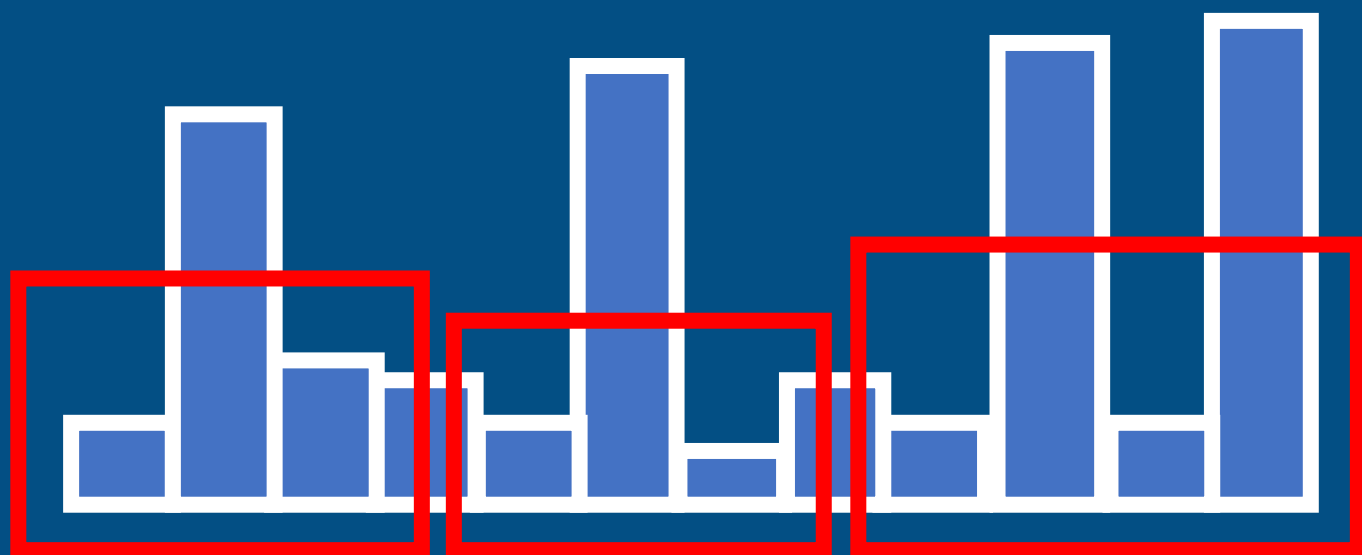
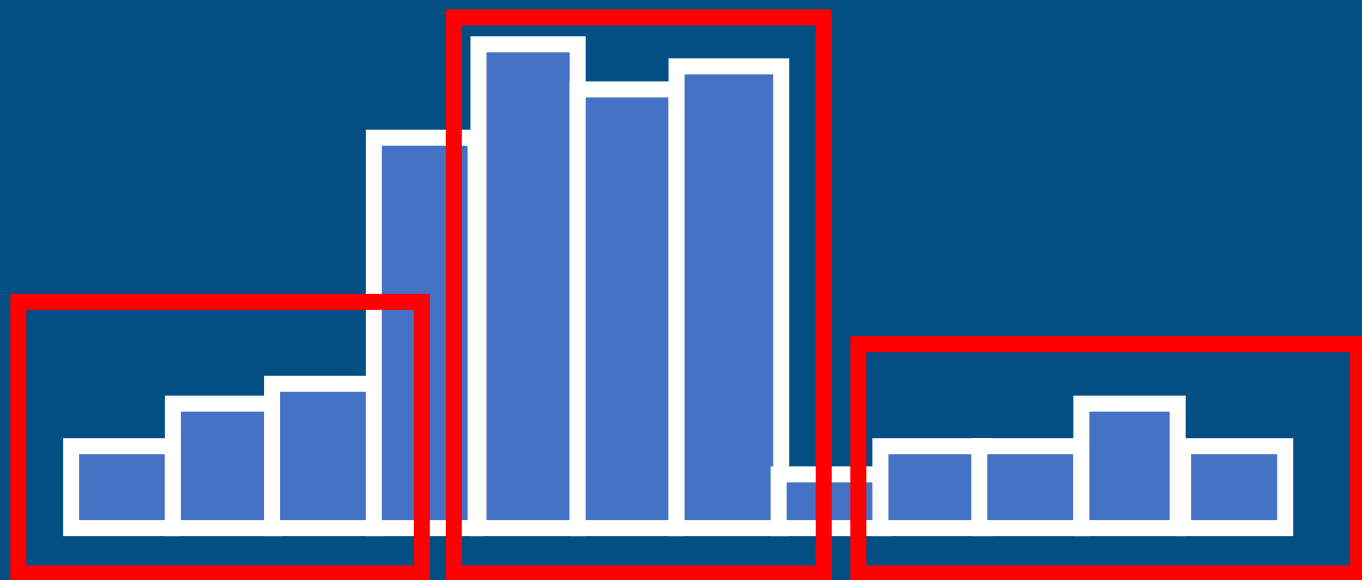


오프폴리시 (예 : 큐러닝)

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

학습하는 정책과 행동을 고르는 정책이 달라도 된다.

과거에 경험한 에피소드들도 학습에 계속해서 이용 가능하다. → 효율적



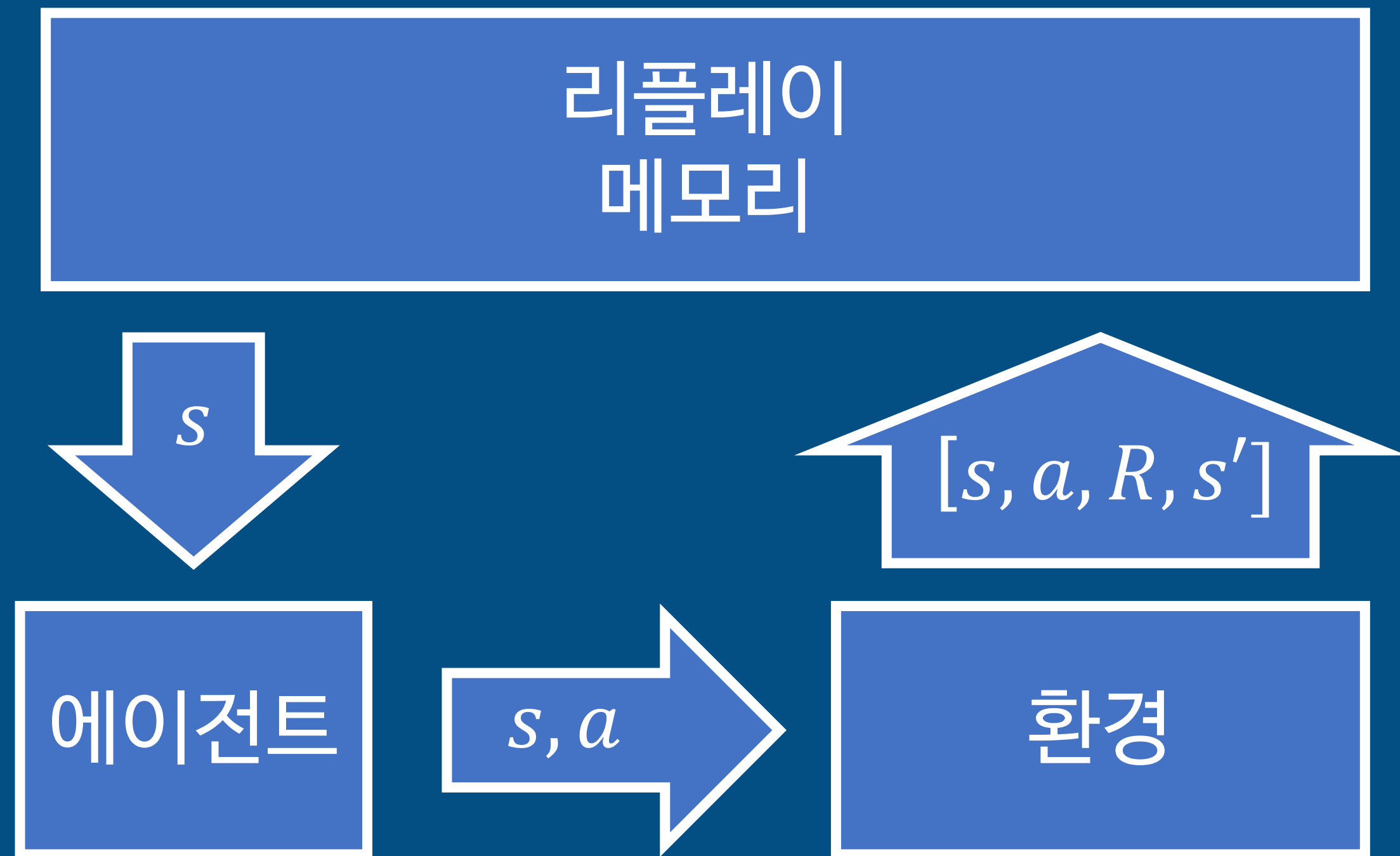
강화학습이 잘 일어나기 위해서는 에이전트가  
알 수 없는 데이터들 사이의 연관성이 없어야 한다.

하지만 게임과 같은 환경에서는 데이터들이 행동에  
대한 연속적인 결과이기 때문에 연관성이 심하다.

그래서 데이터들을 리플레이 메모리에 저장하고,  
이 중 샘플을 고르는 미니배치 방법을 사용한다.

# 리플레이 메모리 + 미니배치

UNIST HeXA  
Week 7



리플레이 메모리는 환경에서 받은  $[s, a, R, s']$ 을 저장한다.

받은  $s'$ 를 새로운  $s$ 로 에이전트에게 전달해 에피소드를 계속 진행시키킨다.

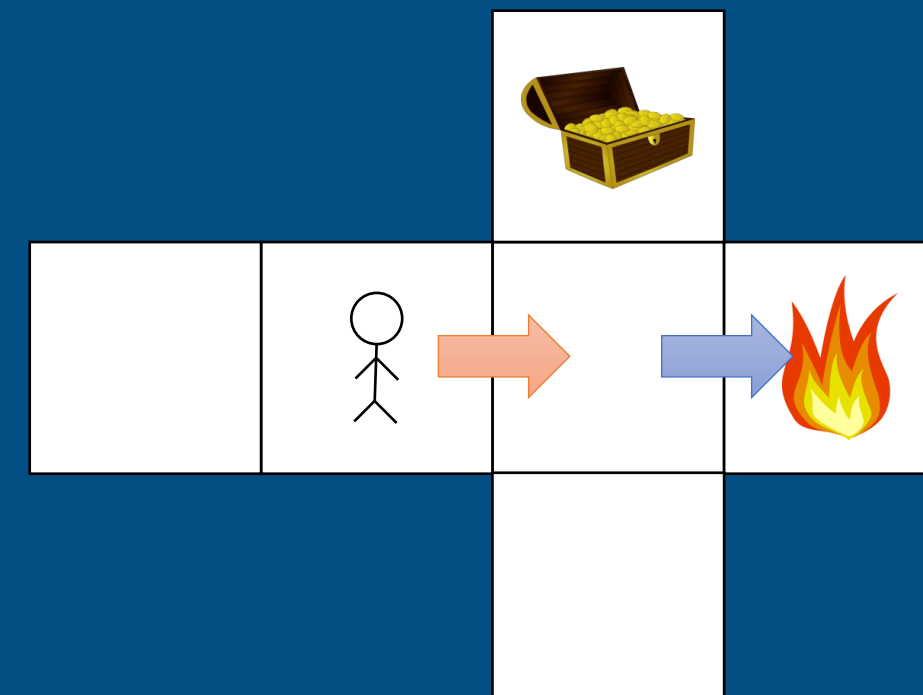
리플레이  
메모리

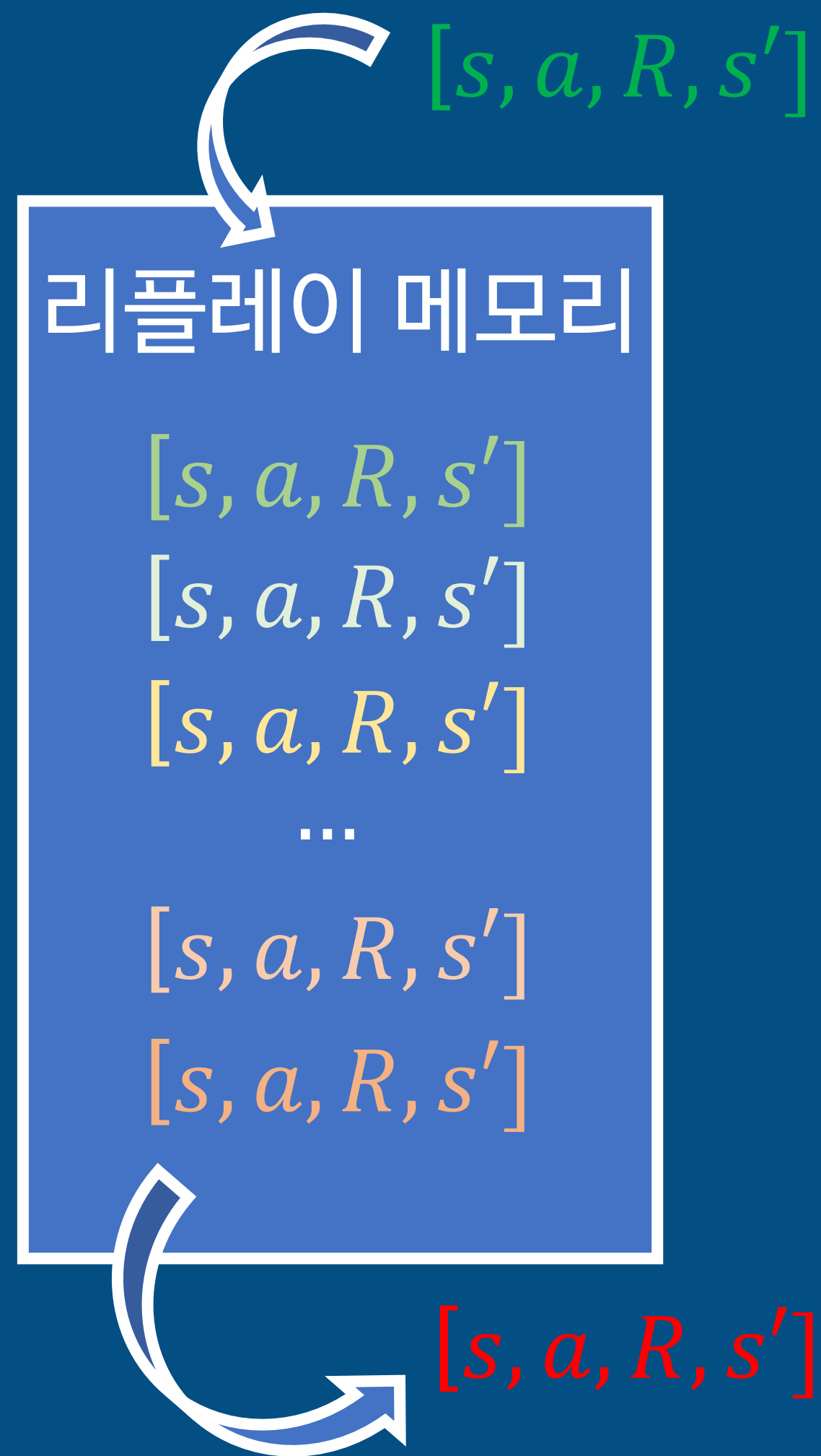


샘플  
 $[s, a, R, s']$   
 $[s, a, R, s']$   
 $[s, a, R, s']$   
...

리플레이 메모리에 저장되어 있는  $[s, a, R, s']$  집합에서 일부를 무작위로 샘플링하고, 이걸로 에이전트를 학습시킨다.

이렇게 하면 샘플 사이의 시간적 상관관계가 줄어들고, 실사에서 발생했던 문제를 줄일 수 있다.





사용할 수 있는 메모리의 크기는 유한하므로,  
리플레이 메모리의 크기에 제한을 두어야 한다.

제한 이상으로 경험이 추가되면 오래된 순서대로  
리플레이 메모리에서 지워준다.

그러면 좋지 않은 과거의 경험이 줄어들면서,  
더 좋은 경험이 샘플로 뽑힐 확률이 증가한다.

DQN에서도 딥살사와 비슷하게 오류함수로 MSE를 사용한다.

DQN 에이전트가 학습할 때 사용하는 오류함수는 아래와 같았다. (2013년)

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

$$\text{Set } y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$$

하지만 위 식에는 문제점이 있다. 인공신경망이 스스로 목표를 만들어 내기 때문에 인공신경망이 업데이트될 때 목표가 되는 정답 부분이 계속 변하고, 그 결과 학습이 굉장히 불안하게 이루어진다.

그래서  $\theta^-$  를 매개변수로 갖는 타겟 신경망을 추가한다. (2015년)

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

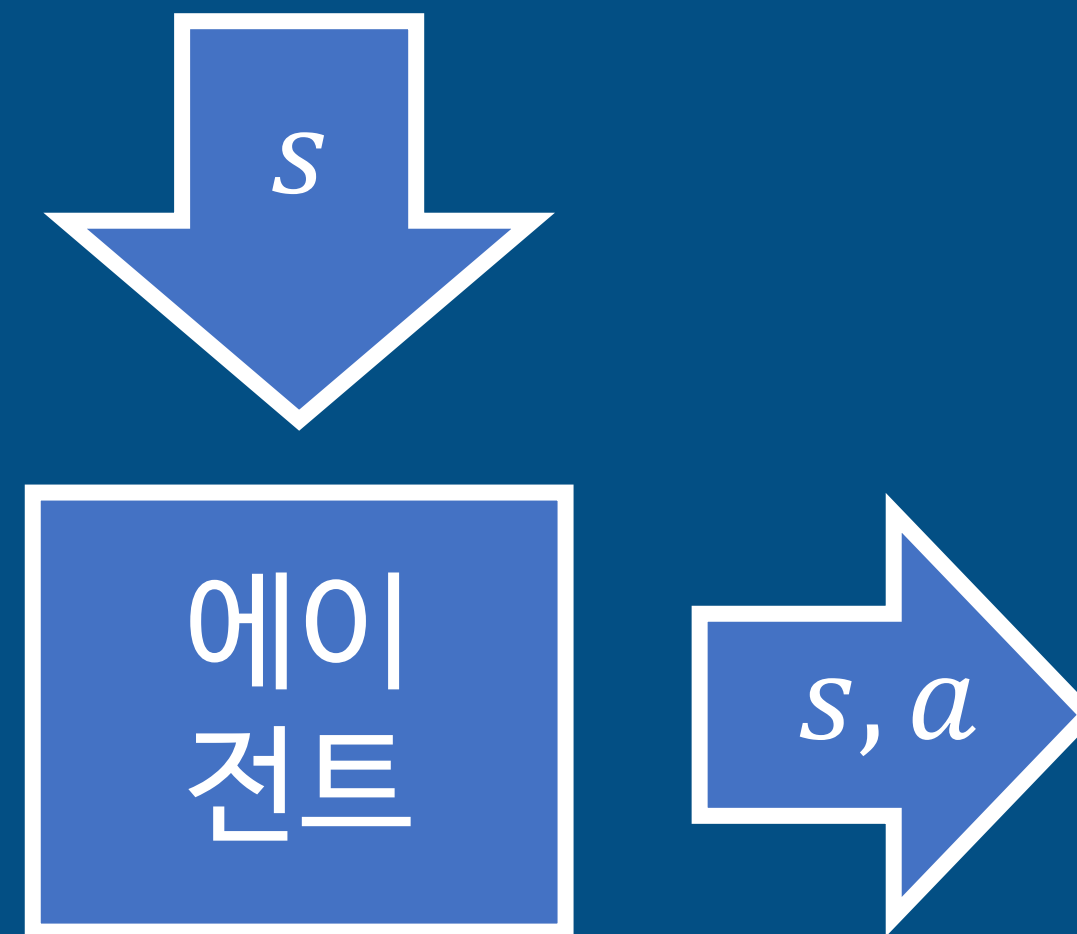
$$\text{Set } y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$$

타겟 신경망은 일정 시간동안 그대로 유지되며 정답을 만들어내다가,  
에피소드가 끝날 때마다 업데이트된다.

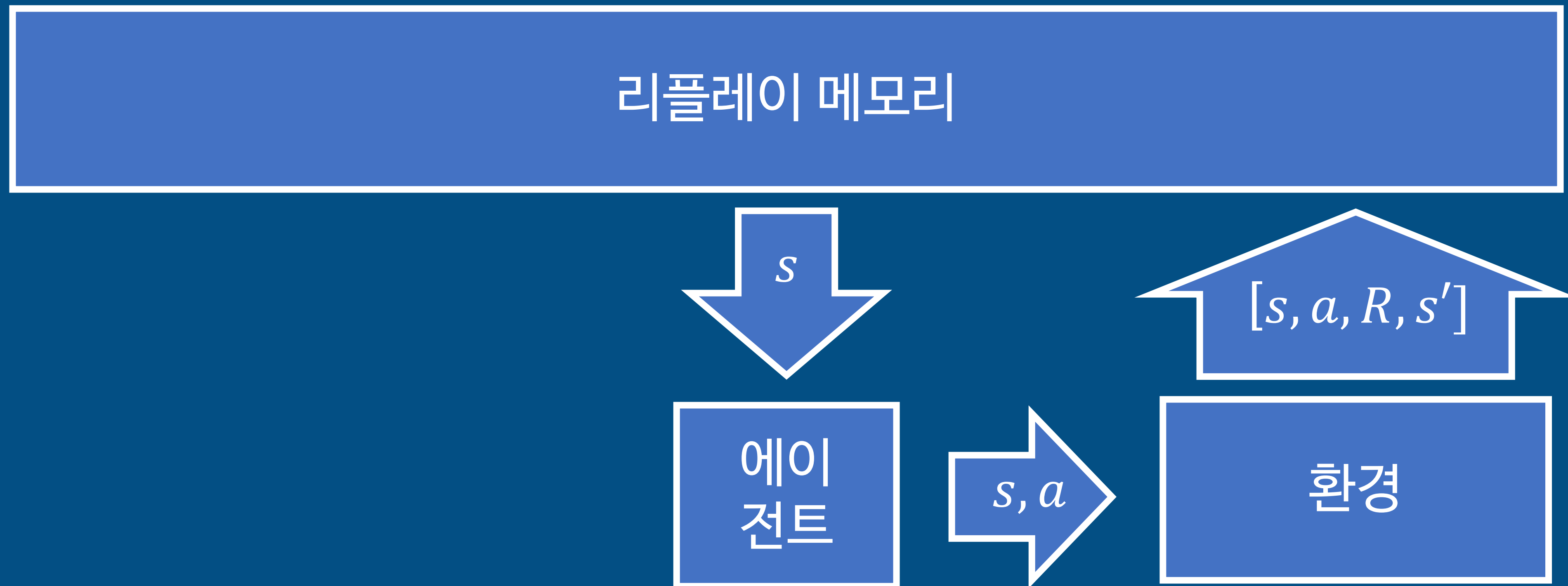
→ 이를 통해 학습의 불안정성을 줄일 수 있다.



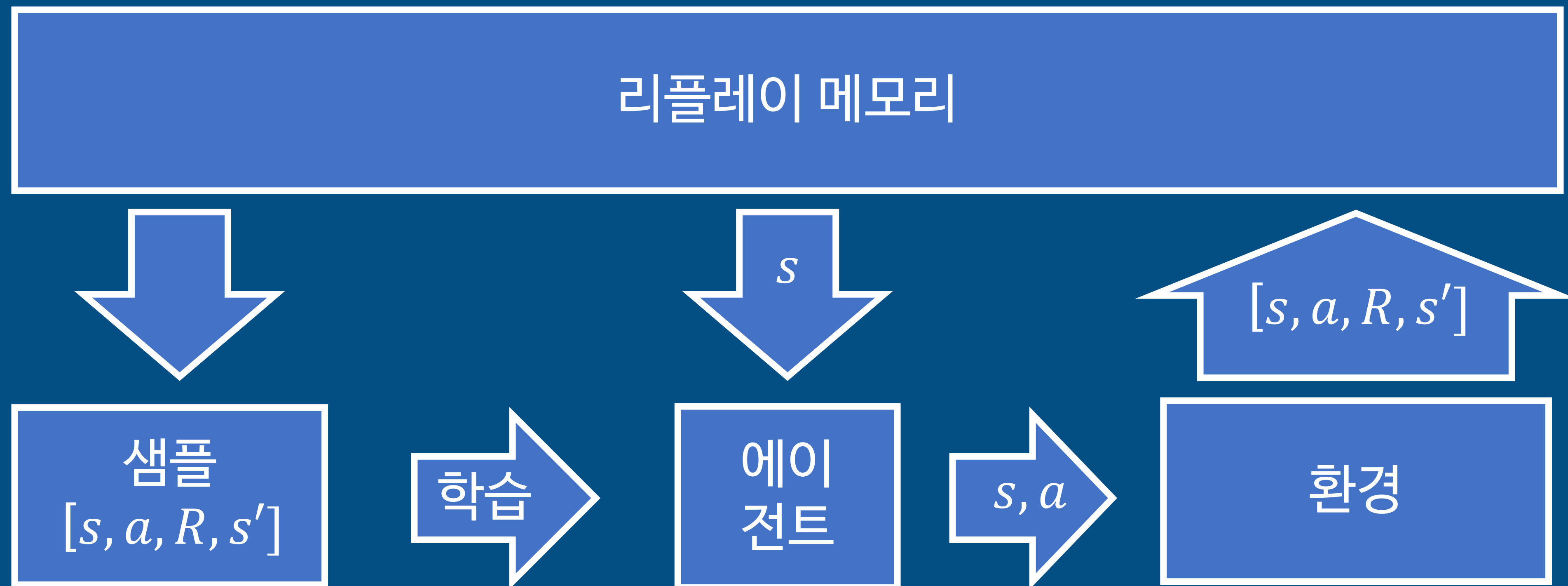
## 1. 상태에 따라 행동을 선택한다.



2. 선택한 행동으로 환경에서 한 스텝을 진행하고, 샘플을 리플레이 메모리에 저장한다.



3. 리플레이 메모리에서 추출한 샘플로 에이전트를 학습시킨다.
4. 에피소드마다 타겟 모델을 업데이트한다.



감사합니다!

스터디 듣느라 고생 많았습니다.