

# CartPole-v1

---

WITH Q-LEARNING

# Action Space

---

## ❖ 종류

- (1) 0 : cart를 왼쪽으로 미는 행위
- (2) 1 : cart를 오른쪽으로 미는 행위

## ❖ 주의사항

: 차를 미는 행위에 의해 state의 4가지 변수가 모두 다 변함

# Action Space

---

1.  $\text{힘} = (\text{action} == 1) ? 10 : -10$
2.  $\text{Temp} = ( \text{힘} + ( \text{막대무게} * \text{막대길이} ) * \text{각속도} * \sin(\text{theta}) ) / \text{총 무게}$
3.  $\text{Thetacc} = ( \text{중력} * \sin(\text{theta}) - \cos(\text{theta}) * \text{temp} ) / \text{막대 길이} * ( 4/3 - \text{막대무게} * \cos(\text{theta})^2 / \text{총 무게} )$
4.  $\text{Xacc} = \text{temp} - ( \text{막대무게} * \text{막대길이} ) * \text{thetacc} * \cos(\text{theta}) / \text{총 무게}$

=> 상태 변수에 모두 영향을 미침

1.  $\text{카트 위치} = \text{카트 위치} + \text{time} * \text{카트 속도}$
2.  $\text{카트 속도} = \text{카트 속도} + \text{time} * \text{xacc}$
3.  $\text{theta} = \text{theta} + \text{time} * \text{각속도}$
4.  $\text{각 속도} = \text{각속도} + \text{time} * \text{thetacc}$

※ time : 다음 상태로 업데이트 되는 시간, 0.02로 지정

# Observation space

Observation	Min	Max
카트 위치	-4.8	+4.8
차 속도	-inf (-.5)	+inf (+.5)
Theta(막대 각도)	-.418rad(-24도)	-.418rad(24도)
각속도	-inf ( - .872 rad, -50도)	+inf ( + .872rad, 50도))

※ Starting state (reset시)

4가지 상태 변수가 모두 uniformly random하게 (- .05, + .05 ) 구간에서 배정

# Terminate case & Reward

---

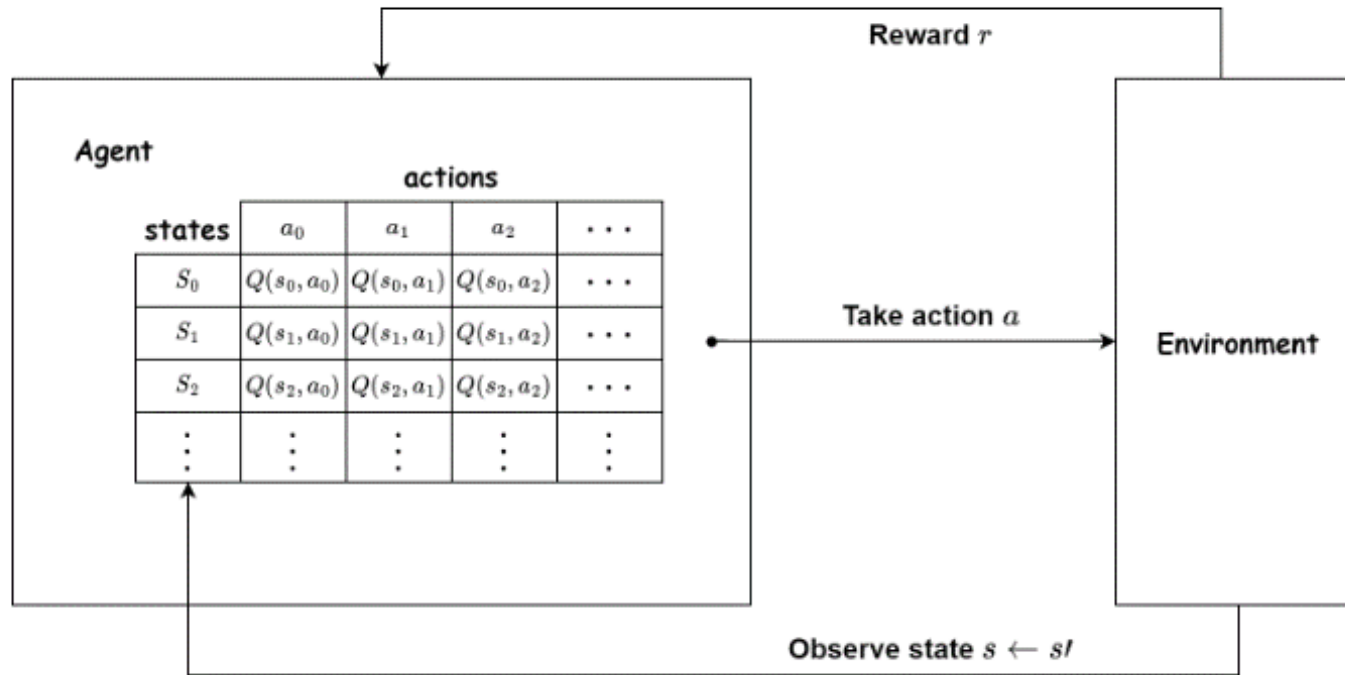
## ❖ Terminate case

1. 카트 위치가 -2.4, +2.4가 되면 episode 종료
2. Theta가 - .295, + .295가 넘어서면 episode 종료
3. step이 500을 넘어서면 episode 종료(truncated)

## ❖ Reward

1. Step 마다 +1
2. 막대가 넘어질 경우 -100

# Q-Table



각 상태에서 행동에 대한 최대로 예상되는 미래 보상을 계산해놓는 lookup 테이블의 일종

State 값이 연속적일 경우 그 값이 무한한 종류를 가질 수 있다

-> 연속적인 값을 이산적인 값으로 변환시켜주어야 한다.

# Q-Table

---

## Step 1

: State-Action 쌍으로 구성된 행렬을 모두 0으로 초기화

```
q_table = np.zeros(state_space_bins + [env.action_space.n])  
print(np.sum(q_table))
```

0.0

## Step 2

: 초기 Q-Table의 값은 모두 0 이므로, epsilon greedy strategy 사용

-> 초기에는 입실론을 크게 (1로) 설정하여 agent가 무작위로 행동

-> episode 가 진행됨에 따라 Q-Table에 각 행동에 대한 가치(value)값으로 업데이트

-> 입실론 값을 점차 줄여가며 Agent가 Q-Table을 활용하도록 조작







# Q-Table

## Step 3

: Q-Table은 행동가치함수 (Action – Value Function)으로 업데이트 됨

->  $Q(s, a)$ 는 현재 상태  $s$ 에서 행동  $a$ 를 취할 때 기대할 수 있는 return (보상 총합) 값

$$\text{New } Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \max_{a'} Q'(s', a') - Q(s, a)]$$

-  New Q Value for that state and the action
-  Learning Rate
-  Reward for taking that action at that state
-  Current Q Values
-  Maximum expected future reward given the new state ( $s'$ ) and all possible actions at that new state.
-  Discount Rate



# Q-Table

---

한 step을 진행 -> state 및 action 에 따라 next\_state, reward가 계산되어 나옴

즉, 현 state 에서 액션을 취했을 때 기대되는 reward 의 총 합  $Q(s, a)$ 는,

= 본래 저장된 값 + 학습률 (  $\text{reward} + (\text{할인율}) * \text{다음 상태에서 기대되는 최대 return 값} - \text{본래 저장된 값}$  )

# Q-Table

ex) Q-테이블의 직관성을 위해 각 구간을 2로 통일

0,0,0,0  
0,0,0,1  
0,0,1,0  
0,0,1,1  
  
0,1,0,0  
0,1,0,1  
0,1,1,0  
0,1,1,1

$$\begin{bmatrix} [[[[[0. & 0. & ] \\ [0. & 0.1]] \\ [[0. & 0. & ] \\ [0. & 0. & ]]]] \end{bmatrix}$$

$$\begin{bmatrix} [[[[[0. & 0. & ] \\ [0. & 0.1]] \\ [[0. & 0. & ] \\ [0. & 0. & ]]]] \end{bmatrix}$$

$$\begin{bmatrix} [[[[[0.1099 & 0. & ] \\ [0. & 0.1 & ]]] \\ [[0. & 0. & ] \\ [0. & 0. & ]]]] \end{bmatrix}$$

$$\begin{bmatrix} [[[[[0.1099 & 0. & ] \\ [0.1099 & 0.1 & ]]] \\ [[0. & 0. & ] \\ [0. & 0. & ]]]] \end{bmatrix}$$

$$\begin{bmatrix} [[[[[0.1099 & 0. & ] \\ [0.1099 & 0.2008801]] \\ [[0. & 0. & ] \\ [0. & 0. & ]]]] \end{bmatrix}$$

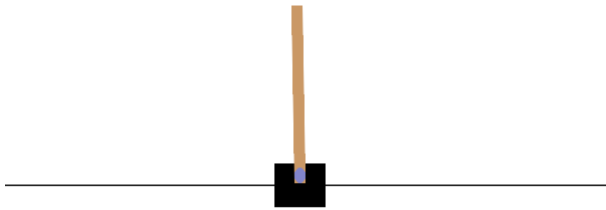
0, 1

(0,0,0,1), 1 → (0,1,0,0), 0 → (0,0,0,0), 0 → (0,0,0,1), 0 → (0,0,0,1), 1 → (0,0,0,0)

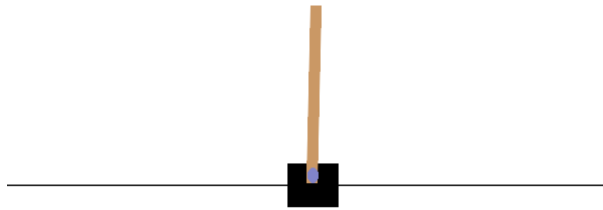
=> Step이 진행됨에 따라 Q-table의 값이 bellman의 행동 가치 함수로 인해 업데이트 되고 있음을 확인할 수 있음

# CartPole 시각화

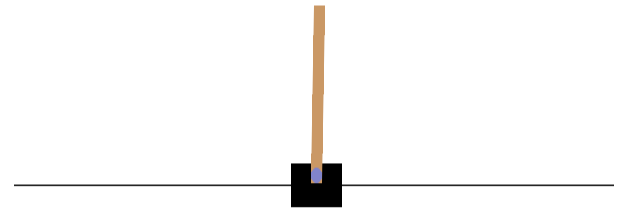
---



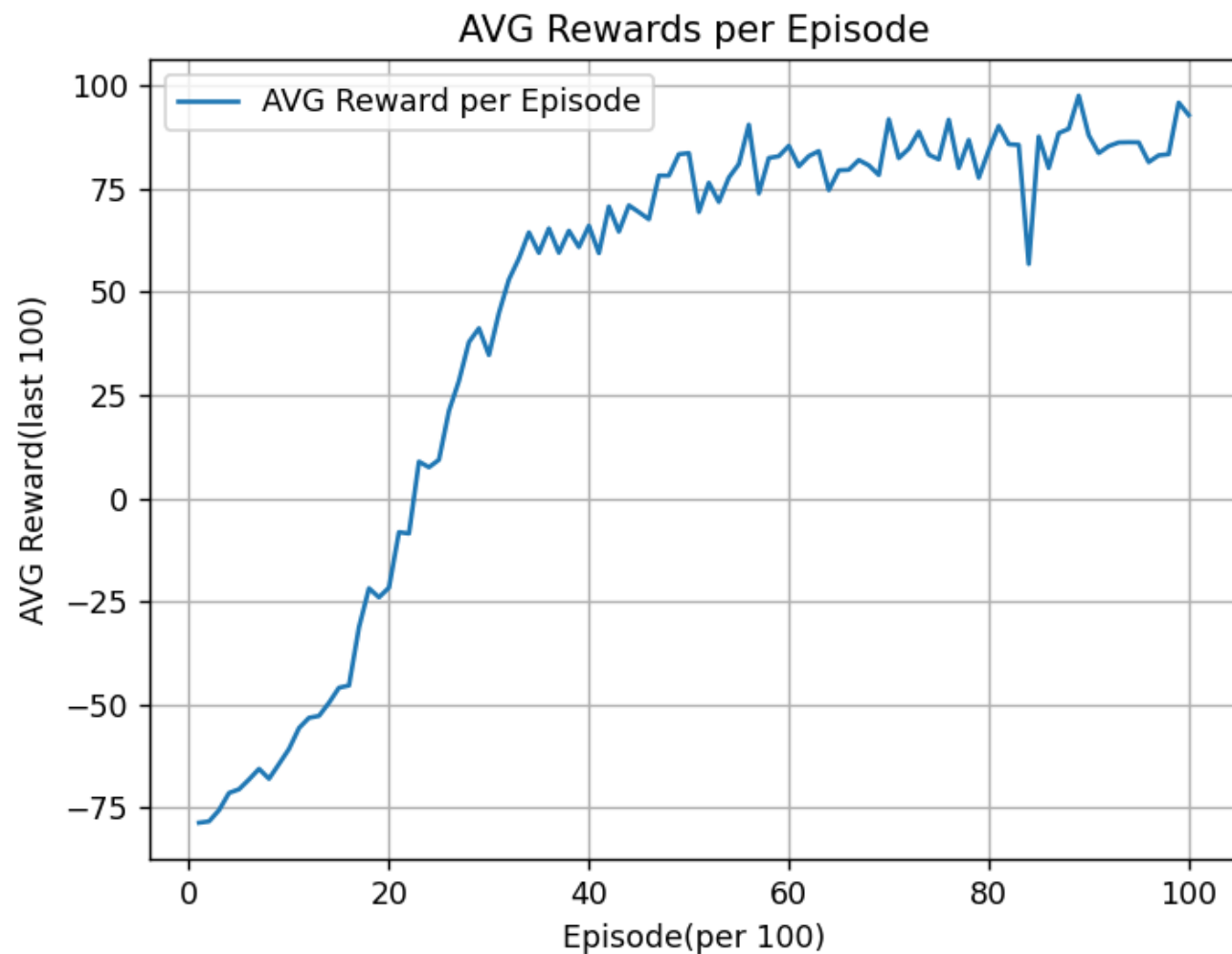
0번 훈련



3000번 훈련



10000번 훈련



# CartPole 시각화

---

# Thank you

---

2024.07.23