“DS/AI 프로그래밍”

12주차 강화학습에 관하여

J.-K.- Seo
데이터과학원

What is “Reinforce Learning”?

강화 학습(Reinforcement learning)은 기계 학습의 한 영역이다. 행동심리학에서 영감을 받았으며, 어떤 환경 안에서 정의된 에이전트가 현재의 상태를 인식하여, 선택 가능한 행동들 중 보상을 최대화하는 행동 혹은 행동 순서를 선택하는 방법이다.

From Google…

KUIDS
고려대학교 데이터과학원

What is “Reinforce Learning”?

강화학습은 머신러닝의 한 부류입니다(그림 1). 비지도 및 지도 머신러닝과 다르게 강화학습은 정적 데이터셋에 의존하는 것이 아니라 역동적인 환경에서 동작하며 수집된 경험으로부터 학습합니다. 데이터 점 또는 경험은 훈련하는 동안 환경과 소프트웨어 에이전트 간의 시행착오 상호작용을 통해 수집됩니다. 강화학습의 이런 점은 지도 및 비지도 머신러닝에서는 필요한 훈련 전 데이터 수집, 전처리 및 레이블 지정에 대한 필요성을 해소하기 때문에 중요합니다. 이는 실질적으로 적절한 인센티브가 주어지면 강화학습 모델은 인간의 개입 없이 학습 행동을 자체적으로 시작할 수 있다는 것을 의미합니다.

From Matlab website…

딥러닝은 3가지 머신러닝 모두를 포함합니다. 강화학습과 딥러닝은 상호 배타적이지 않습니다. 복잡한 강화학습 문제는 주로 심층 강화학습이라고 알려진 분야인 심층 신경망에 의존합니다.

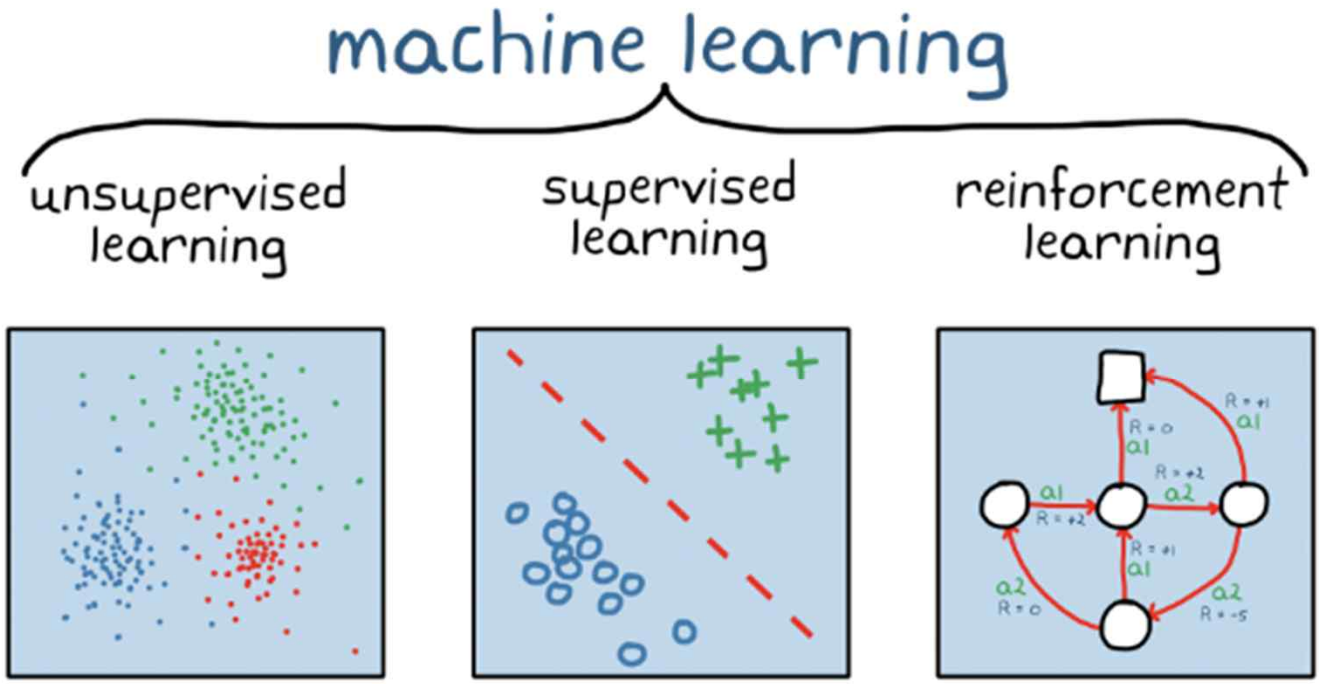
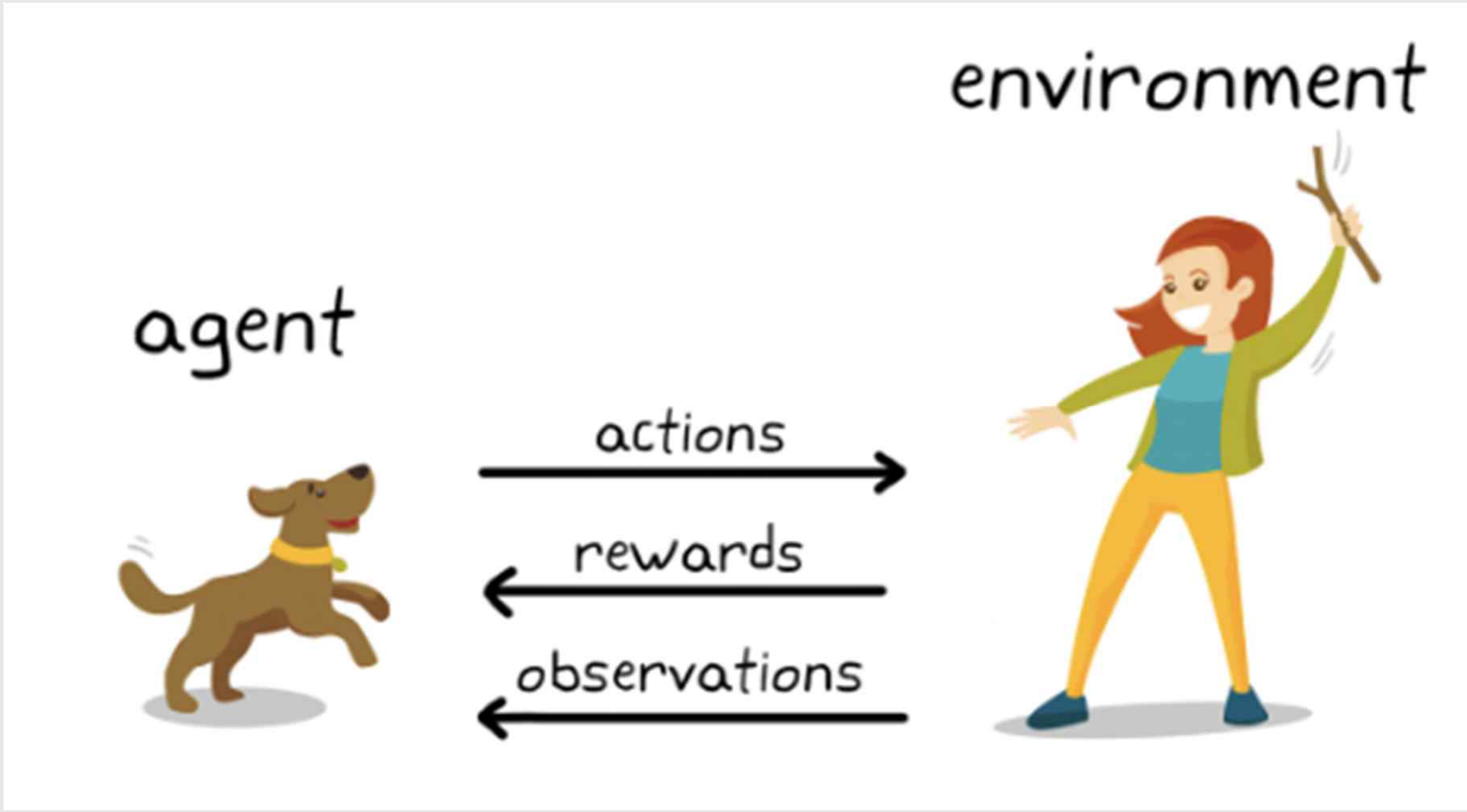


그림 1. 세 가지 머신러닝: 비지도 학습, 지도 학습 및 강화학습.

KUIDS
고려대학교 데이터과학원

What is “Reinforce Learning”?



Let’s go to the site!

https://kr.mathworks.com/discovery/reinforcement-learning.html

KUIDS
고려대학교 데이터과학원

Q-learning

Classic Q-learning

State s_t

Action a_t

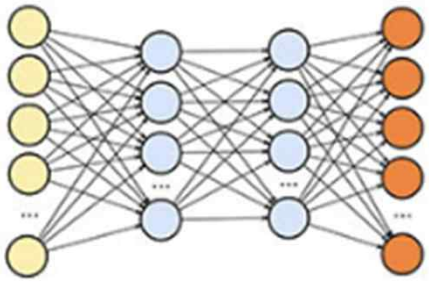
	a_1	a_2	a_3
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$Q(s_1, a_3)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$Q(s_2, a_3)$
s_3	$Q(s_3, a_1)$	$Q(s_3, a_2)$	$Q(s_3, a_3)$

→ Q-value $Q(s_t, a_t)$

Deep Q-learning

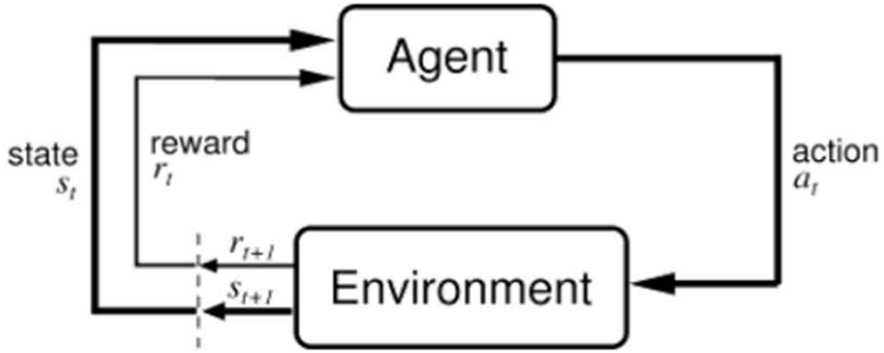
State s_t

Action a_t



→ Q-value $Q(s_t, a_t)$

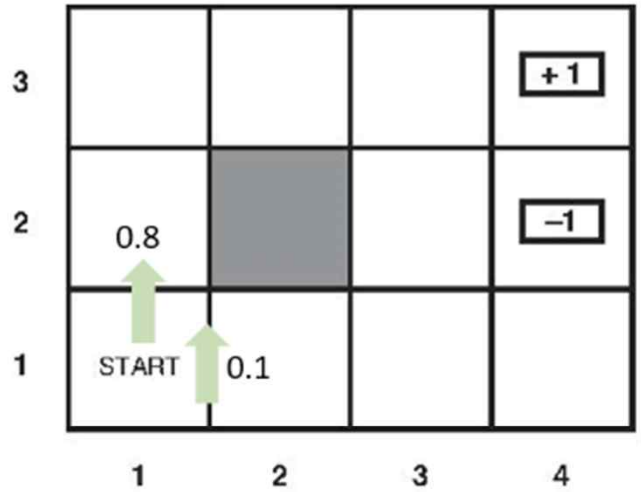
MDP(Markov Decision Process)



State : s_t
Action : a_t
Stochastic transition model : $P(s_{t+1}|s_t,a_t)$
Reward : $R(s_t,a_t,s_{t+1}) = R(s_{t+1}) = R_{t+1}$

- 상태(State) : 정적인 요소 + 동적인 요소를 의미합니다.
- 행동(Action) : 어떠한 상태에서 취할 수 있는 행동을 의미합니다.
- Stochastic transition model : 어떤 상태에서 특정 행동을 하여 다음 상태에 도달할 확률
- 보상(Reward) : Agent가 학습할 수 있는 유일한 정보를 의미합니다. 어떤 상태에서 행동을 하여 다음 상태가 되고, 이때 받는 보상값은 다음 상태가 되는 것에 대한 보상입니다.
- 정책(Policy) : 순차적 행동 결정 문제(MDP)에서 구해야할 답을 의미합니다. 모든 상태에 대해 Agent가 어떠한 Action을 해야 하는지 정해놓은 것을 의미합니다.
- 목표는 최적의 정책(Optimal Policy)을 찾는 것입니다.

가치함수(Value Function)



Harim Kang - Davinci AI
https://davinci-ai.tistory.com/

S → action → S′ → action → S′′ → …

States : (1,1), (1,2), (1,3) …

Actions : Up, Down, Left, Right

Transitions : P((1,2) | (1,1), Up) = 0.8,
P((2,1) | (1,1), Up) = P((1,1) | (1,1), Up) = 0.1

Rewards : +1 at (4,3), -1 at (4,2), -0.04 at other states

Value Function : reaches (4,3) after 10 move → +1 – 10*(-0.04) = 0.6

Value Function of a state sequence

$v(s0, s1, s2, \dots) = R(s0) + \gamma R(s1) + \gamma^2 R(s2) + \dots = \sum_{k=0} \gamma^k R(Sk)$

γ : discount factor (감가율; 0~1) – preference of current reward

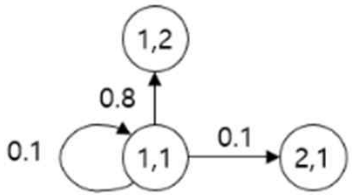
$\gamma^k R(Sk)$: 감가율을 고려한 미래 보상의 현재 가치

벨만 방정식(The Bellman Equation)

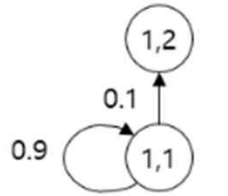
3	0.812	0.868	0.918	+1
2	0.762		0.660	−1
1	0.705	0.655	0.611	0.388
	1	2	3	4

The Bellman equation

$$v(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)v(s')$$



< Up >



< Left >

v(1,1) = -0.04 + γ max[0.8v(1,2)+0.1v(2,1)+0.1v(1,1), (Up)
0.9v(1,1)+0.1v(1,2), (Left)
0.9v(1,1)+0.1v(2,1), (Down)
0.8v(2,1)+0.1v(1,2)+0.1v(1,1)] (Right)

KUIDS
고려대학교 데이터과학원

정책(Policy)

Policy : π(a|s) = P[A_t=a|S_t=s]

Optimal policy π*

$$\pi^*(s) = \underset{a \in A(s)}{argmax} \sum_{s'} P(s'|s,a)v(s')$$

모든 상태에서 에이전트가 할 행동을 의미합니다. 최적의 정책은 부분 수열 상태의 기대값이 최대가 되는 정책입니다.

Classic Q-learning

State s_t

Action a_t

	a_1	a_2	a_3
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$Q(s_1, a_3)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$Q(s_2, a_3)$
s_3	$Q(s_3, a_1)$	$Q(s_3, a_2)$	$Q(s_3, a_3)$

→ Q-value $Q(s_t, a_t)$

KUIDS
고려대학교 데이터과학원

[기존의 Deep Q-learning algorithm]

1) 파라미터를 초기화하고, 매 스텝마다 2~5를 반복한다.

2) Action a_t 를 ϵ-greedy 방식에 따라 선택한다.

3) Action a_t 를 수행하여 transition $e_t = (s_t, a_t, r_t, s_{t+1})$ 를 얻는다.

4) Target value $y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta)$ 를 계산한다.

5) Loss function $(y_t - Q(s_t, a_t; \theta))^2$ 를 최소화하는 방향으로 θ 를 업데이트한다.

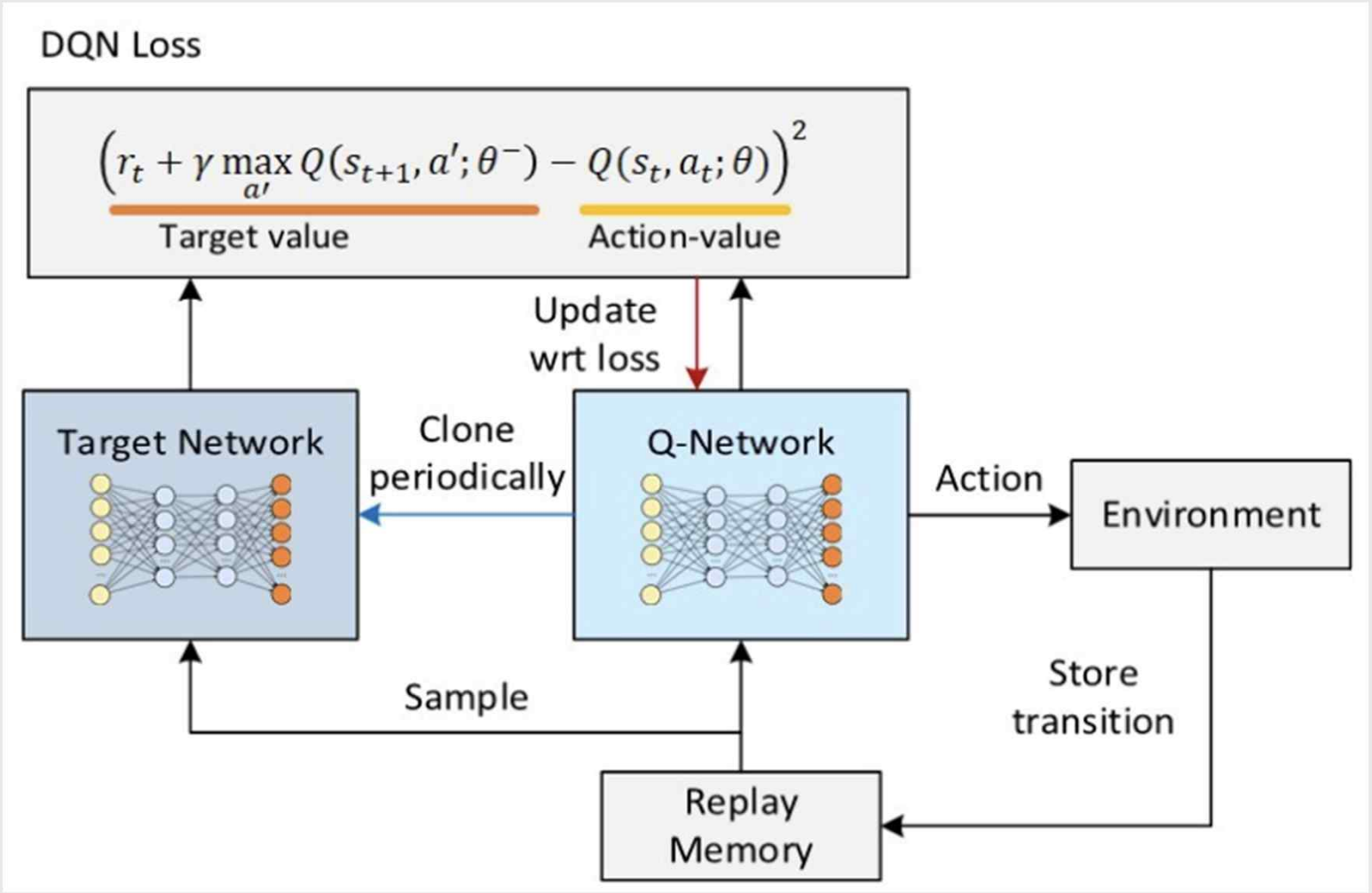
[Target network]

1) Target network θ^- 를 이용하여 target value $y_j = r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-)$ 를 계산한다.

2) Main Q-network θ 를 이용하여 action-value $Q(s_j, a_j; \theta)$ 를 계산한다.

3) Loss function $(y_j - Q(s_j, a_j; \theta))^2$ 이 최소화되도록 main Q-network θ 를 업데이트한다.

4) 매 C 스텝마다 target network θ^- 를 main Q-network θ 로 업데이트한다.


DQN Loss
$\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta)\right)^2$
Target value
Action-value
Update
wrt loss
Target Network
Clone
periodically
Q-Network
Action
Environment
Sample
Store
transition
Replay
Memory

Q-learning is so complicated to code the algorithm for beginner's level.
Then, we apply Q-learning in another way in simple way, but, similar concept.

How?

We apply ANN in another concept, not in DQN's way.

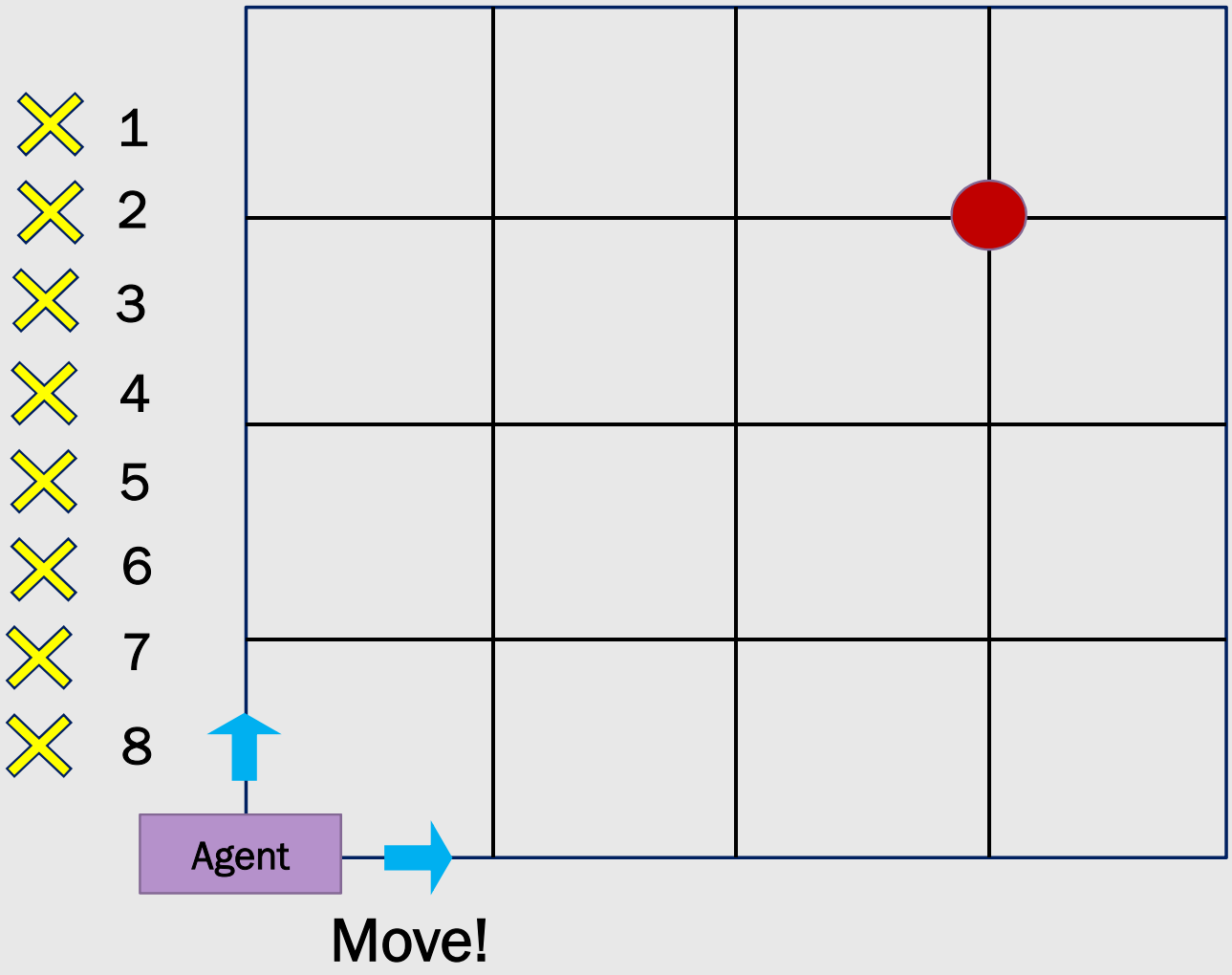
KUIDS
고려대학교 데이터과학원

Mission:

Let's go to

1
2
3
4
5
6
7
8

Move!



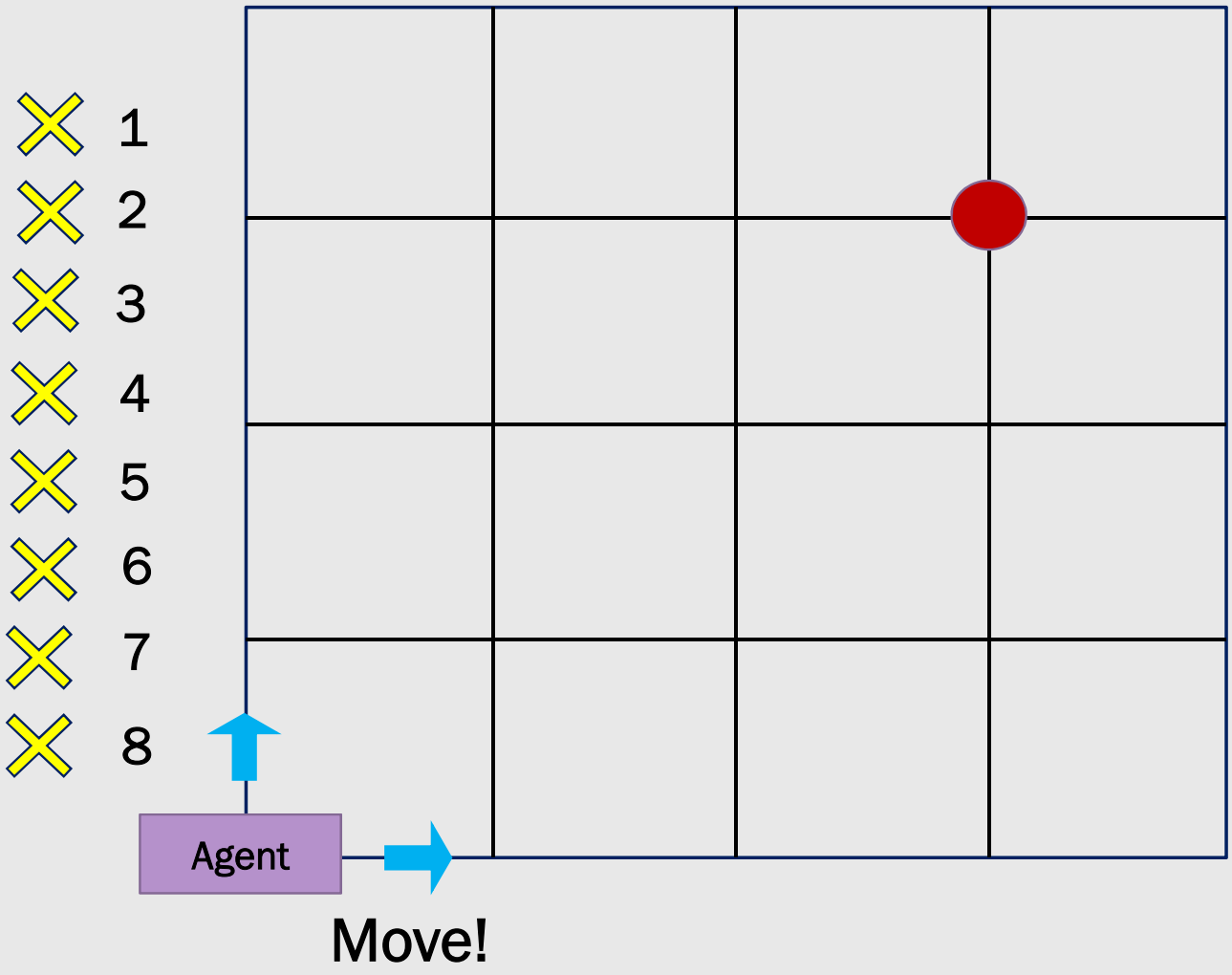
KUIDS
고려대학교 데이터과학원

Mission:

Let's go to

Who is agent?

1. Does not know the condition where it reach (Does not know how to calculate the distance).
2. Only it can do is moving somewhere and memo the candy points.



KUIDS
고려대학교 데이터과학원

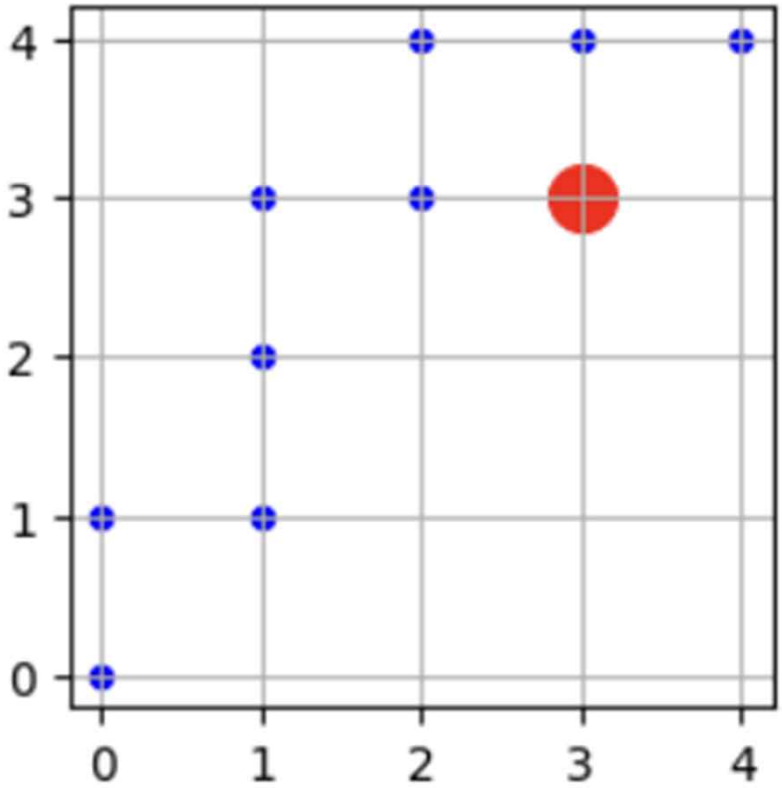
Mission:

Let's go to

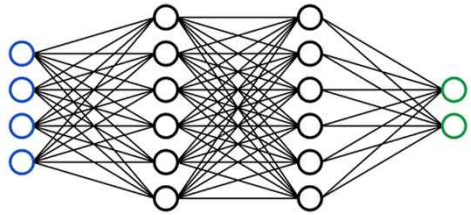
What we have to do?

1. Calculate the distance from the agent's position.
2. Give candy when it reach the point!

[[0, 0], [0, 1], [1, 1], [1, 2], [1, 3], [2, 3], [2, 4], [3, 4], [4, 4]]

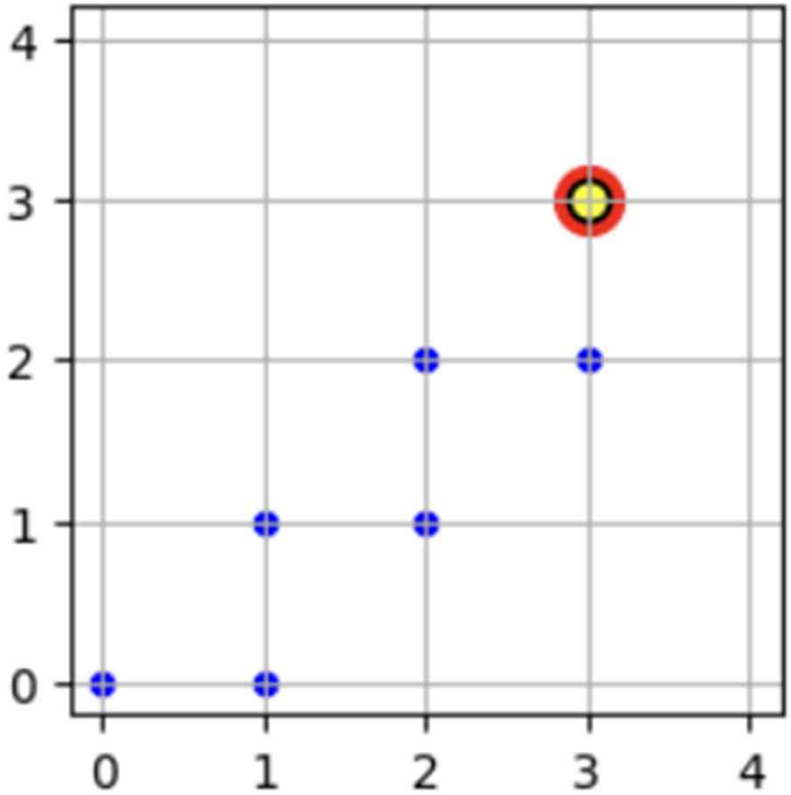


Reach

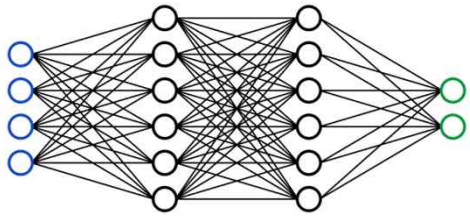


Not correct
Then, memo
And do it again!

[[0, 0], [1, 0], [1, 1], [2, 1], [2, 2], [3, 2], [3, 3]]

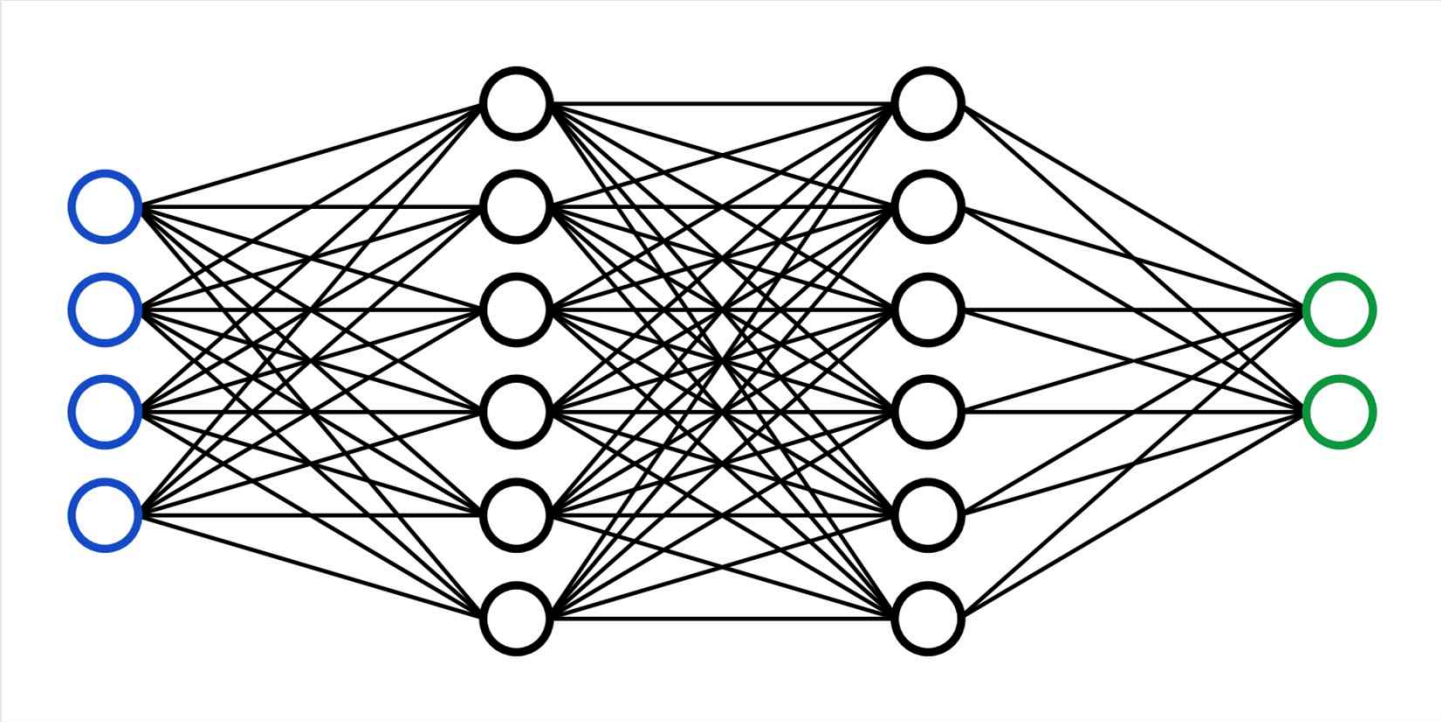


Reached the target
Then,
memo
And do it again!



KUIDS
고려대학교 데이터과학원

How to memo

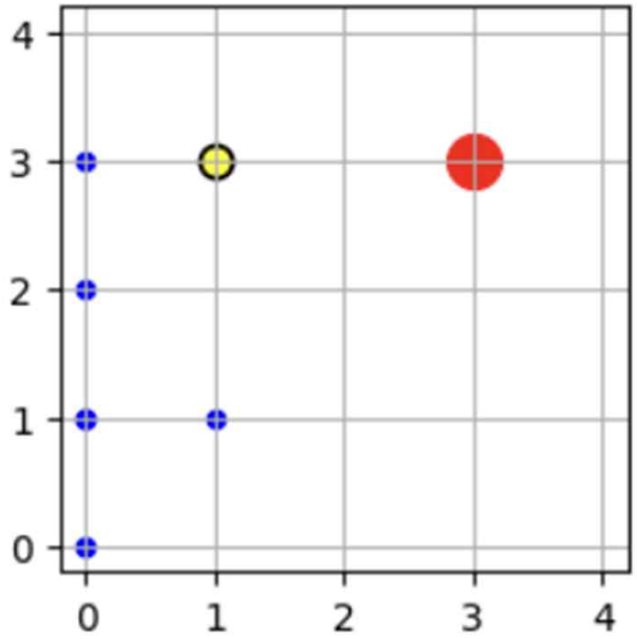


1 , when reach
or
0, when not reach

After full memory... What happen?

Test the memo!

```
[[0, 0], [0, 1], [0, 0], [0, 1], [1, 1], [0, 1], [0, 2], [0, 3], [1, 3]]
```

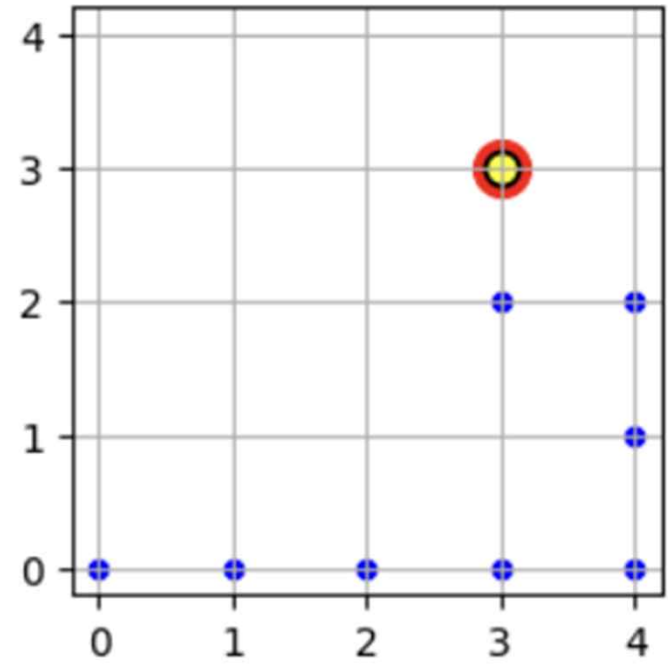


Output inference value: `[6.98395037e-05]`

Reached the point!
But, what's wrong?

KUIDS
고려대학교 데이터과학원

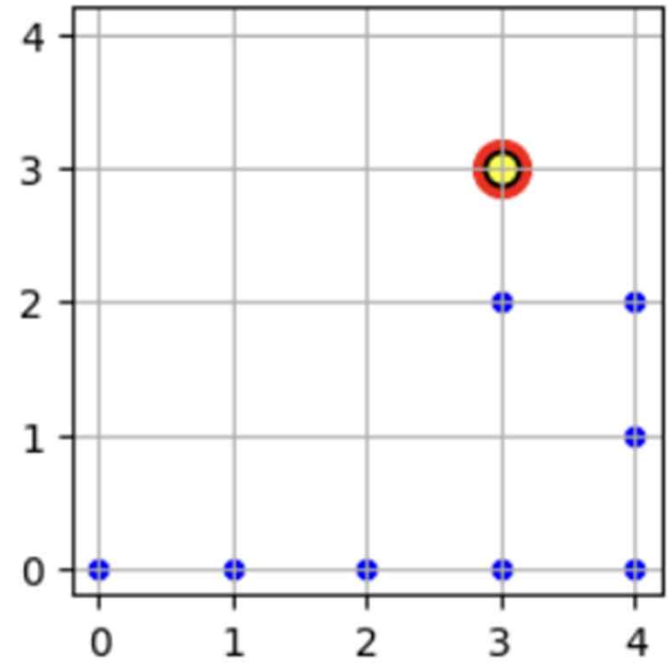
```
[[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [4, 1], [4, 2], [3, 2], [3, 3]]
```




```
[0.15493324]
```

Reached the point!
But, what's wrong?

[[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [4, 1], [4, 2], [3, 2], [3, 3]]



[0.15493324]

Small inference value!

Then how?

Let’s test more!

```
[[0, 0], [1, 0], [2, 0], [2, 1], [2, 0], [3, 0], [4, 0], [3, 0], [4, 0]]
[[0, 0], [1, 0], [2, 0], [3, 0], [2, 0], [2, 1], [3, 1], [3, 2], [3, 3]]
[[0, 0], [0, 1], [0, 2], [1, 2], [2, 2], [2, 3], [3, 3]]
[[0, 0], [1, 0], [1, 1], [0, 1], [1, 1], [2, 1], [2, 0], [3, 0], [3, 1]]
[[0, 0], [0, 1], [0, 0], [0, 1], [1, 1], [2, 1], [3, 1], [4, 1], [3, 1]]
[[0, 0], [1, 0], [2, 0], [2, 1], [2, 2], [1, 2], [1, 1], [0, 1], [0, 2]]
[[0, 0], [0, 1], [0, 2], [0, 1], [0, 2], [1, 2], [1, 1], [0, 1], [0, 0]]
[[0, 0], [1, 0], [0, 0], [1, 0], [0, 0], [0, 1], [1, 1], [2, 1], [3, 1]]
[[0, 0], [1, 0], [1, 1], [2, 1], [3, 1], [4, 1], [3, 1], [4, 1], [4, 2]]
[[0, 0], [0, 1], [1, 1], [0, 1], [1, 1], [1, 2], [1, 3], [1, 2], [1, 1]]
[[0, 0], [0, 1], [1, 1], [2, 1], [1, 1], [1, 2], [0, 2], [0, 3], [1, 3]]
[[0, 0], [1, 0], [0, 0], [0, 1], [0, 0], [1, 0], [1, 1], [1, 0], [1, 1]]
[[0, 0], [1, 0], [1, 1], [0, 1], [0, 2], [1, 2], [1, 1], [0, 1], [1, 1]]
[[0, 0], [1, 0], [1, 1], [1, 2], [1, 1], [0, 1], [1, 1], [1, 0], [1, 1]]
[[0, 0], [1, 0], [1, 1], [1, 0], [0, 0], [1, 0], [0, 0], [1, 0], [0, 0]]
[[0, 0], [0, 1], [1, 1], [0, 1], [1, 1], [0, 1], [1, 1], [1, 2], [0, 2]]
```

▪▪▪

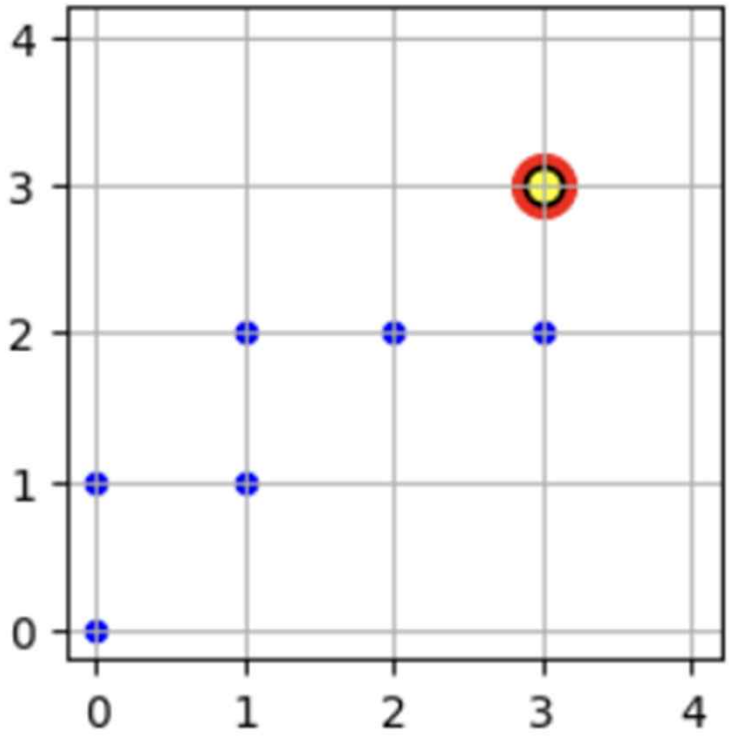
```
[[0, 0], [1, 0], [1, 1], [2, 1], [3, 1], [3, 0], [2, 0], [1, 0], [2, 0]]
[[0, 0], [0, 1], [0, 0], [0, 1], [0, 0], [1, 0], [1, 1], [1, 2], [1, 1]]
[[0, 0], [1, 0], [1, 1], [2, 1], [2, 2], [2, 1], [3, 1], [2, 1], [2, 0]]
[[0, 0], [1, 0], [0, 0], [0, 1], [1, 1], [2, 1], [2, 0], [1, 0], [1, 1]]
[[0, 0], [1, 0], [2, 0], [2, 1], [2, 2], [1, 2], [0, 2], [1, 2], [2, 2]]
[[0, 0], [1, 0], [0, 0], [1, 0], [0, 0], [1, 0], [0, 0], [0, 1], [1, 1]]
[[0, 0], [0, 1], [0, 2], [0, 3], [0, 2], [1, 2], [2, 2], [3, 2], [4, 2]]
[[0, 0], [1, 0], [1, 1], [2, 1], [2, 0], [2, 1], [2, 0], [3, 0], [4, 0]]
[[0, 0], [0, 1], [1, 1], [1, 0], [1, 1], [0, 1], [1, 1], [0, 1], [1, 1]]
[[0, 0], [0, 1], [1, 1], [2, 1], [2, 2], [2, 1], [2, 2], [3, 2], [4, 2]]
[[0, 0], [0, 1], [1, 1], [0, 1], [0, 2], [0, 3], [1, 3], [1, 4], [0, 4]]
[[0, 0], [0, 1], [0, 2], [1, 2], [2, 2], [2, 3], [2, 2], [1, 2], [1, 1]]
[[0, 0], [1, 0], [1, 1], [2, 1], [2, 2], [2, 1], [1, 1], [0, 1], [0, 2]]
[[0, 0], [1, 0], [1, 1], [2, 1], [1, 1], [1, 0], [2, 0], [1, 0], [0, 0]]
[[0, 0], [1, 0], [1, 1], [1, 2], [1, 3], [0, 3], [1, 3], [2, 3], [3, 3]]
[[0, 0], [0, 1], [1, 1], [1, 2], [2, 2], [3, 2], [3, 3]]
step: 63
```

Let's test more!
Until we get best answer...

Here, we go 63 trials,
and
We got it, at last!

But, what is the idea?

```
[[0, 0], [0, 1], [1, 1], [1, 2], [2, 2], [3, 2], [3, 3]]
step: 63
[0.34193333]
```



KUIDS
고려대학교 데이터과학원

The End

KUIDS
고려대학교 데이터과학원

Reference

https://kr.mathworks.com/discovery/reinforcement-learning.html