기말고사/프로젝트

- 기말고사
  - 일시: 12월 14일 (화) 13:00~14:15
  - 장소: 추후 공지 (offline)
- 기말 프로젝트 (recommender systems)
  - 주제, template 등 자세한 내용 이번주 주말에 녹화강의로 공지 예정
  - 제출기한: 12월 17일 (금) 23:59

Reinforcement Learning 1

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING

인공지능학과
Department of
Artificial Intelligence

정 우 환 (whjung@hanyang.ac.kr)

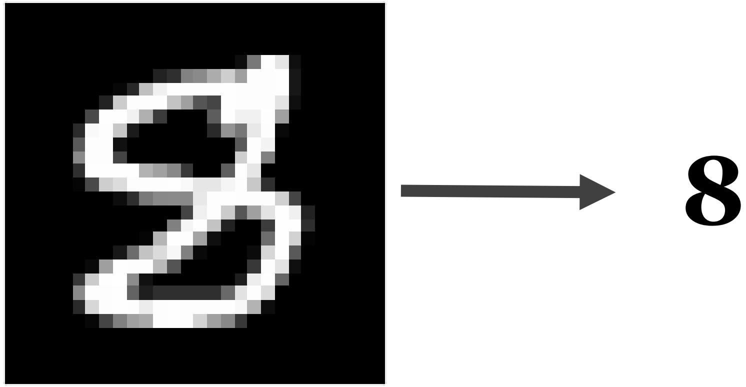
Fall 2021

Supervised Learning

Data: (x, y)
X is data, y is label

Goal: Learn a function to map
$x \rightarrow y$

Examples:
Classification, regression, …

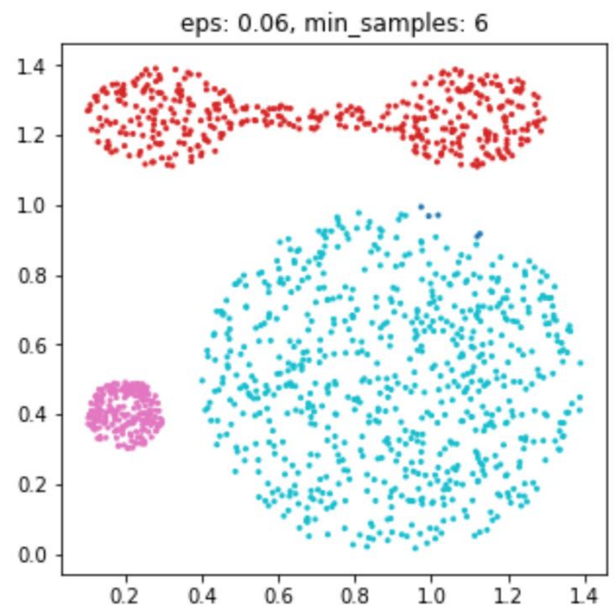


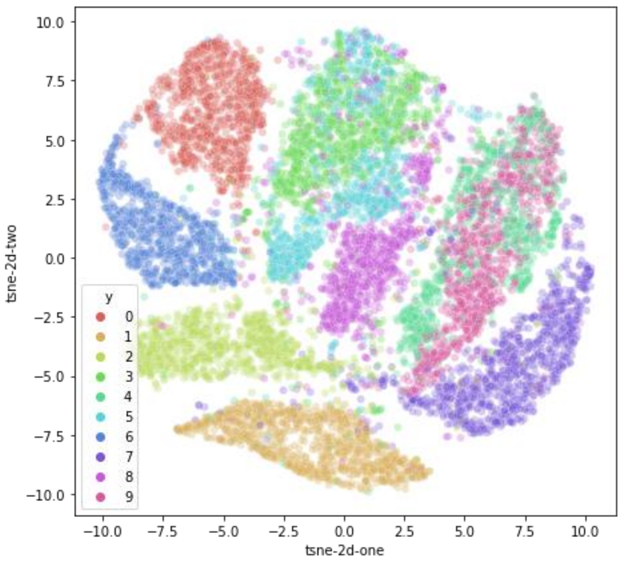
Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples
Clustering,
Dimensionality reduction,
…

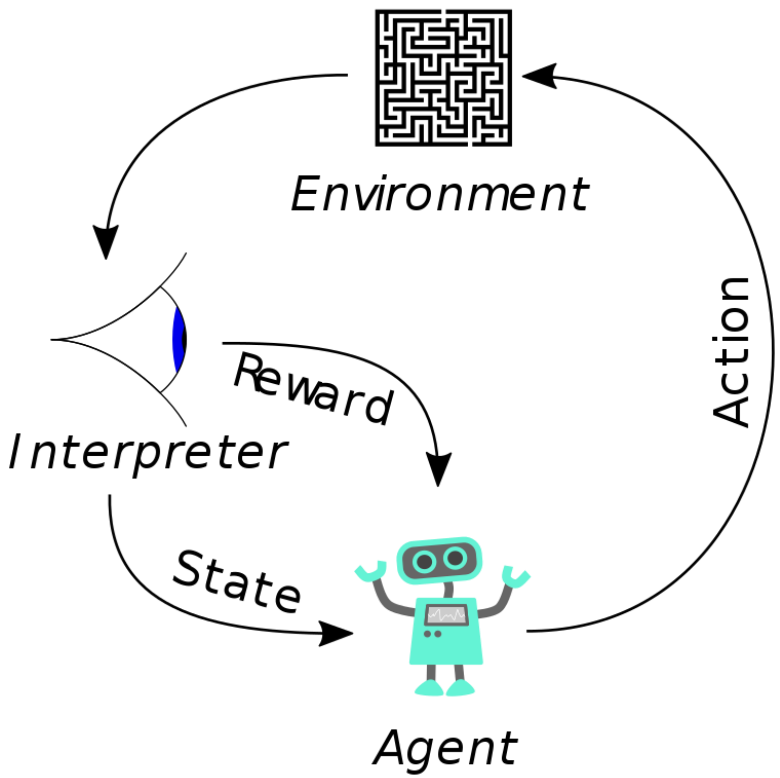




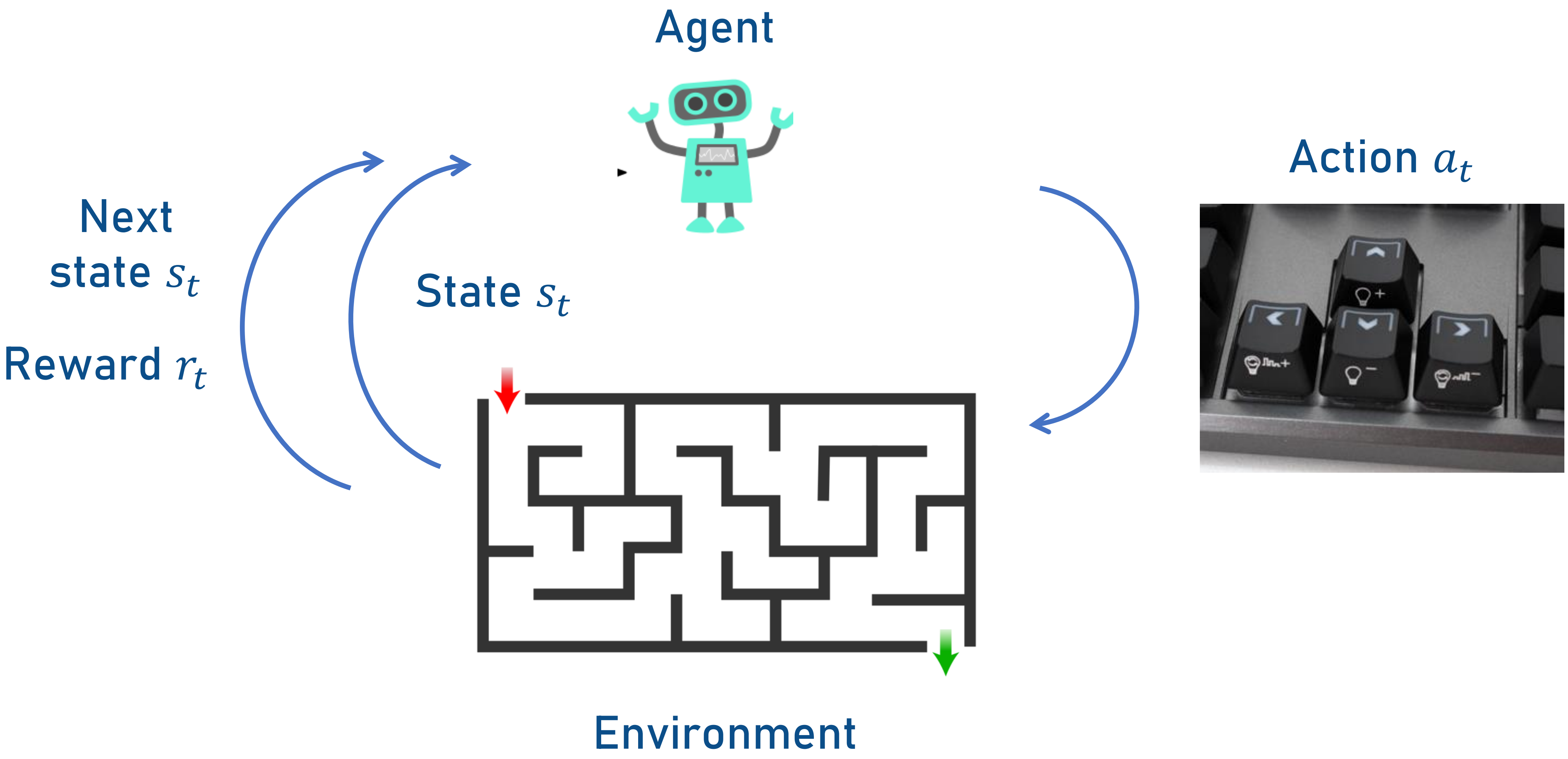
Reinforcement Learning

Problems involving an **agent** interacting with an **environment**, which provides numeric **reward** signals

Goal: Learn how to take actions in order to **maximize reward**



Reinforcement Learning



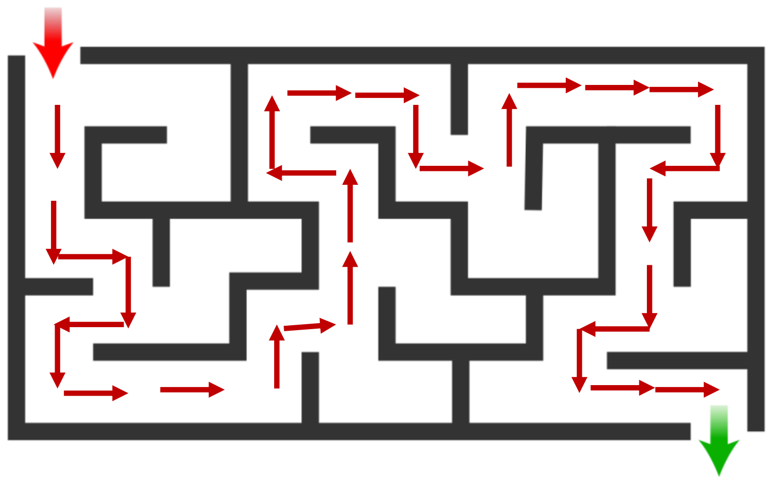
Properties of RL

https://youtu.be/V1eYniJ0Rnk

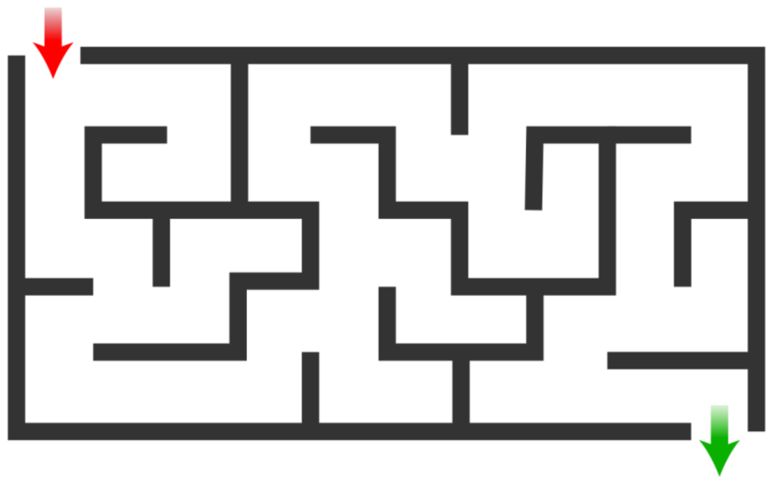
- Trial and Error (Agent chooses training data)
  - 행동 -> 보상
  - 보상 -> 행동의 수정
  - 행동 -> 보상
  - 보상 -> 행동의 수정
  - …
- Sparse/delayed reward signal

Sparse signal

Labels in supervised learning



Rewards in reinforcement learning



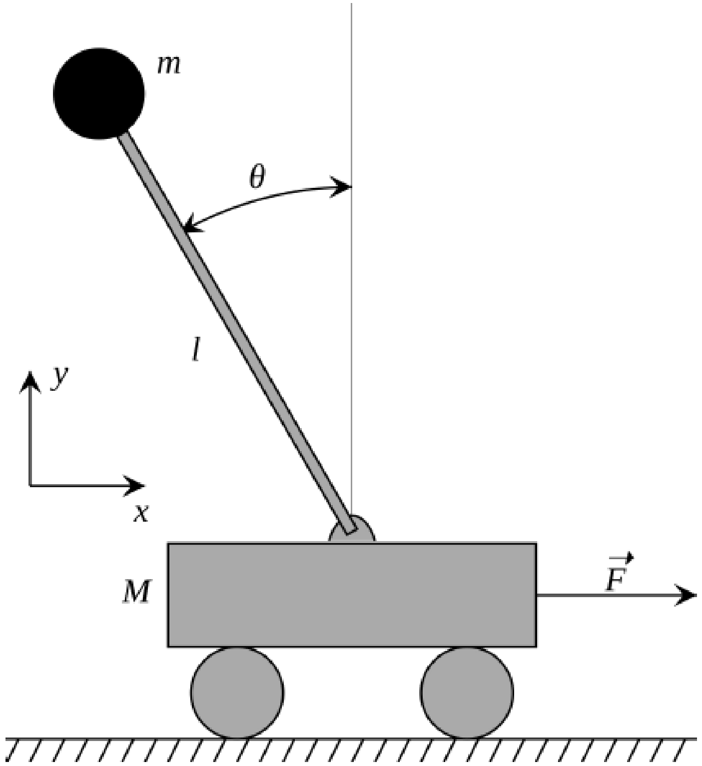
$-T$

Reward design tip

- Example) (Reward) = -(걸린 시간)

a_1	$r_1 = 0$	a_1	$r_1 = -1$
a_2	$r_2 = 0$	a_2	$r_2 = -1$
a_3	$r_3 = 0$	a_3	$r_3 = -1$
	$\vdots$		$\vdots$
a_T	$r_T = -T$	a_T	$r_T =$-1

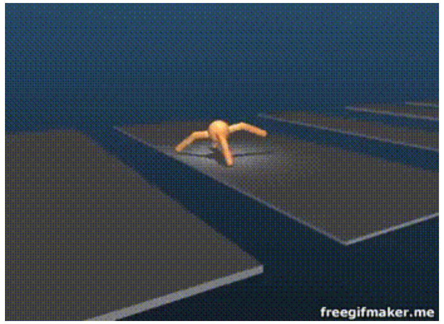
Cart-Pole Problem



Objective: Balance a pole on top of a movable cart

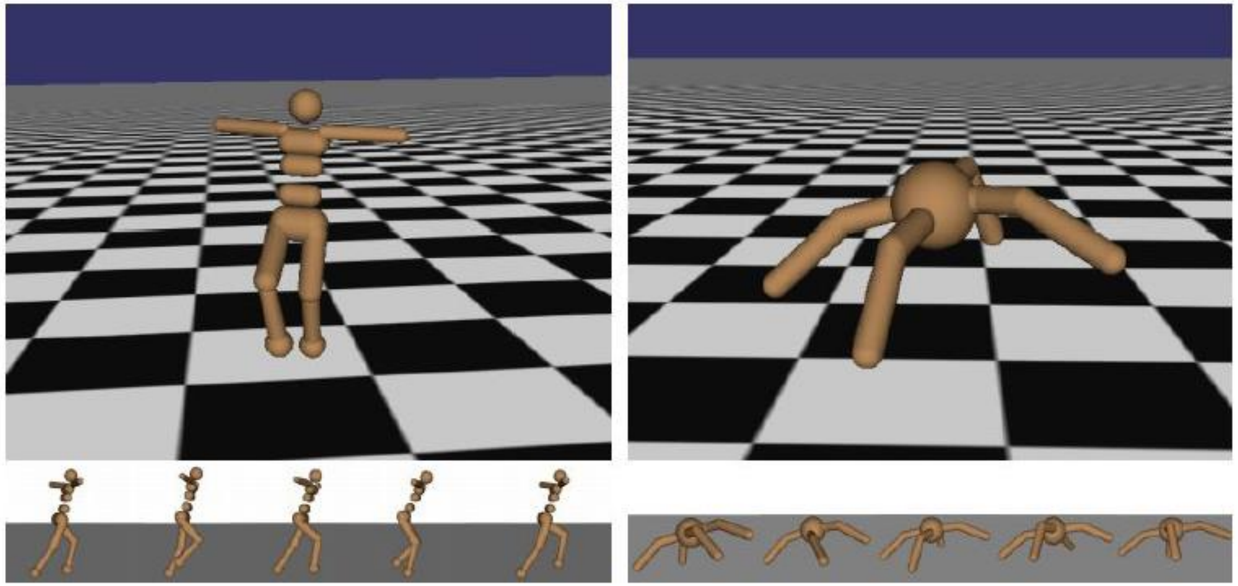
State: angle, angular speed, position, horizontal velocity
Action: horizontal force applied on the cart
Reward: 1 at each time step if the pole is upright

Robot Locomotion



https://youtu.be/hx_bgoTF7bs

2017 paper https://arxiv.org/pdf/1707.02286.pdf



Objective: Make the robot move forward

State: Angle and position of the joints
Action: Torques applied on joints
Reward: 1 at each time step upright + forward movement

Atari Games

Objective: Complete the game with the highest score

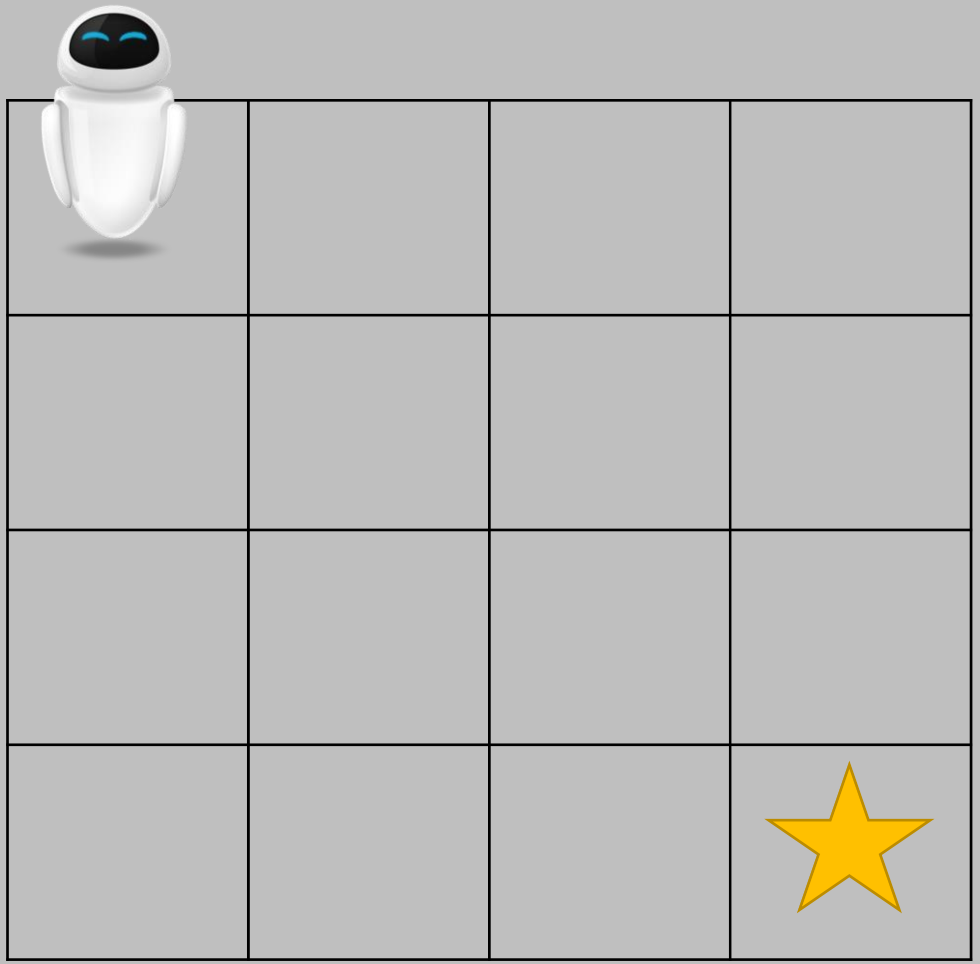
State: Raw pixel inputs of the game state
Action: Game controls e.g. Left, Right, Up, Down
Reward: Score increase/decrease at each time step

Q-learning

Value functions

- Total reward $R_t = r_{t+1} + r_{t+2} + \cdots$
- Value functions measure the expected total reward after t
  - Value of a **state** $V(s)$
    - $V(s) = E[r_{t+1} + r_{t+2} + \cdots | s_t = s]$
  - Value of **of taking an action in a state** $Q(s,a)$
    - $Q(s,a) = E[r_{t+1} + r_{t+2} + \cdots | a_t = a, s_t = s]$
- Policy π: mapping from **state** to **action**
  - Optimal policy $\pi^*(s) = \arg\max_a Q(s,a)$

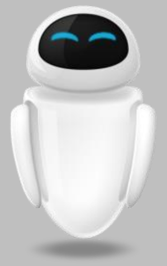
Q-Learning



Q-Function (State-action value) Q(state,action)

State (x,y)

Action



Future reward

$$Q(s_t, a_t) = E[r_{t+1} + r_{t+2} + \cdots | a_t, s_t]$$

$Q\big((1,1), LEFT\big) = 0.0$	$Q\big((3,4), LEFT\big) = 0.0$
$Q\big((1,1), RIGHT\big) = 0.5$	$Q\big((3,4), RIGHT\big) = 0.0$
$Q\big((1,1), UP\big) = 0.0$	$Q\big((3,4), UP\big) = 0.0$
$Q\big((1,1), DOWN\big) = 0.3$	$Q\big((3,4), DOWN\big) = 1.0$

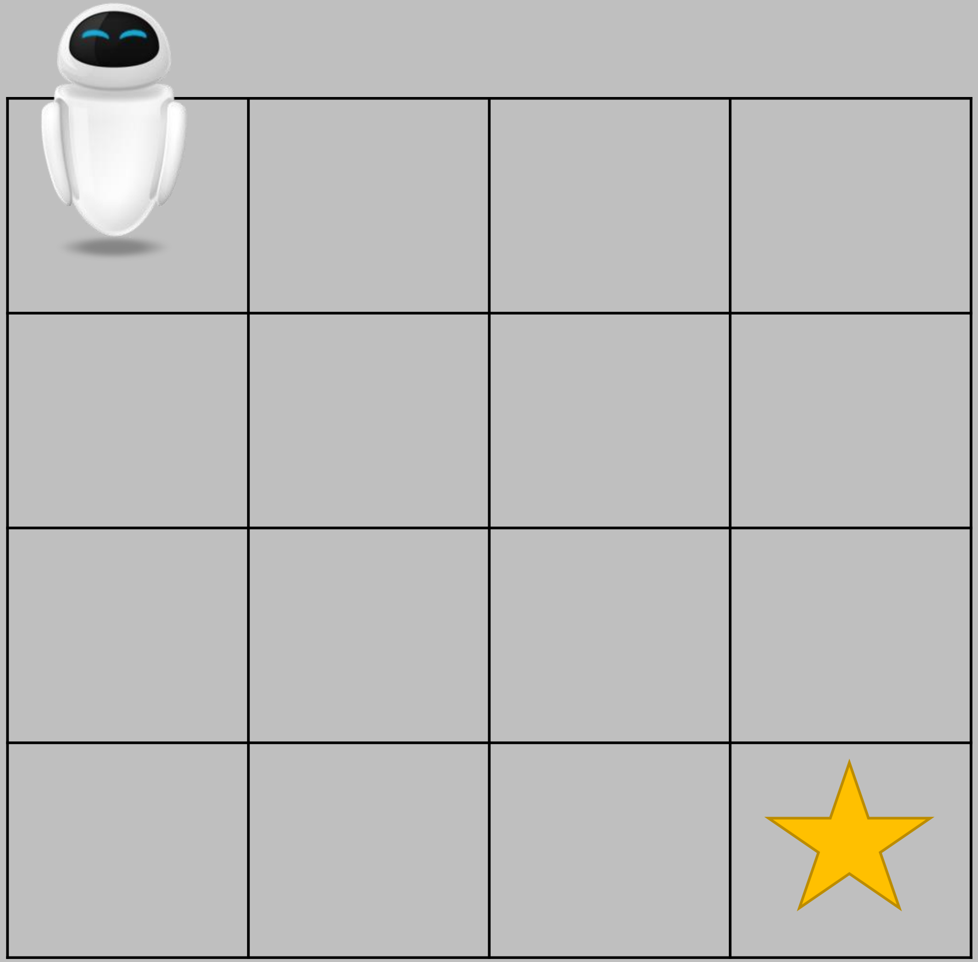
Optimal policy

$$\pi^*(s) = \arg\max_a Q(s, a)$$

$\pi * \big((1,1)\big) \rightarrow RIGHT$	$\pi * \big((3,4)\ \big) \rightarrow DOWN$

Joongheon Kim (https://youtu.be/m1FC3dMmY78)

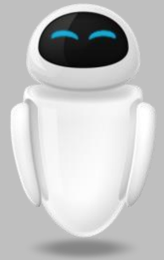
Q-Learning



Q-Function (State-action value)   Q(state,action)

State (x,y)

Action

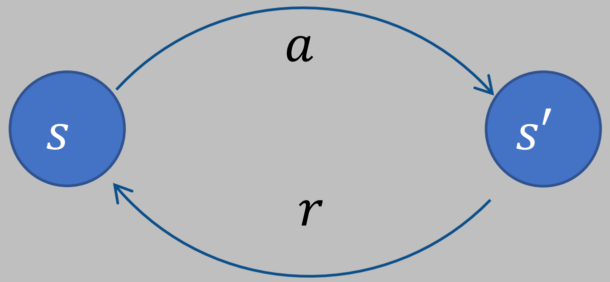


Future reward

$$Q(s_t, a_t) = E[r_{t+1} + r_{t+2} + \cdots | a_t, s_t]$$

Optimal policy $\pi^*(s) = \arg\max_a Q(s, a)$

Recurrence equation



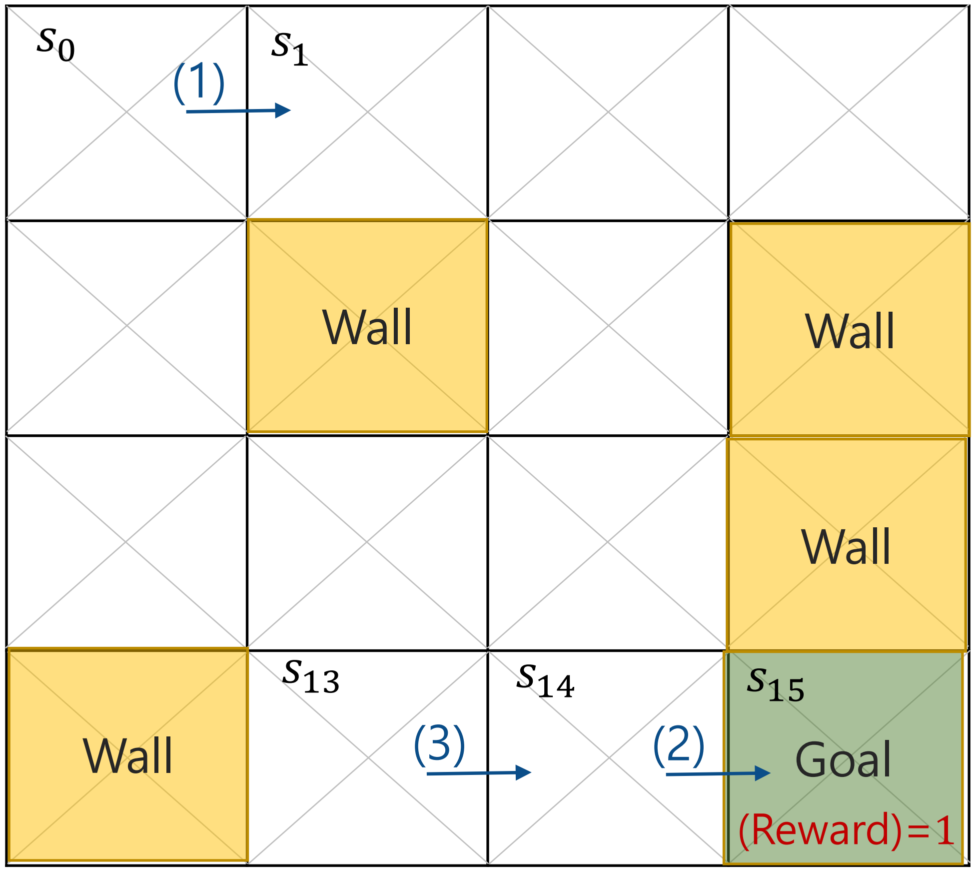
$$Q(s, a) = r + \max_{a'} Q(s', a')$$

Joongheon Kim (https://youtu.be/m1FC3dMmY78)

$$Q(s,a) = r + \max_{a'} Q(s',a')$$

Q-Learning

16 states and 4 actions (U, D, L, R)



- Initial status
  - $Q(s,a) = 0$ for all s, a
  - Reward are all zero except in s_{15}

Case (1) →

$$Q(s_0, R) = r + \max_{a'} Q(s_1, a') = 0 + \max_{a'}\{0,0,0,0\} = 0$$

Case (2) →

$$Q(s_{14}, R) = 1 + \max_{a'} Q(s_{15}, a') = 0 + \max_{a'}\{0,0,0,0\} = 1$$

Case (3) →

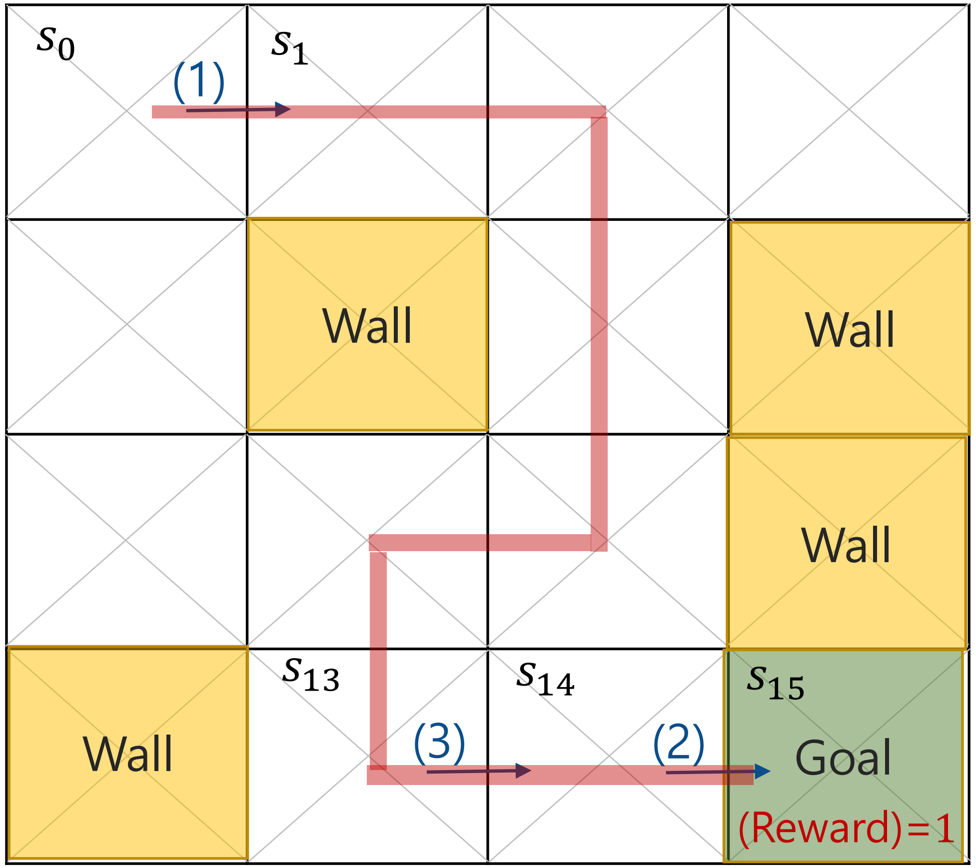
$$Q(s_{13}, R) = r + \max_{a'} Q(s_{14}, a') = 0 + \max_{a'}\{0,0,1,0\} = 1$$

Joongheon Kim (https://youtu.be/m1FC3dMmY78)

$$Q(s,a) = r + \max_{a'} Q(s',a')$$

Q-Learning

16 states and 4 actions (U, D, L, R)



- Initial status
  - $Q(s,a) = 0$ for all s, a
  - Reward are all zero except in s_{15}

Case (1) →

$$Q(s_0,R) = r + \max_{a'} Q(s_1,a') = 0 + \max_{a'}\{0,0,0,0\} = 0$$

Case (2) →

$$Q(s_{14},R) = 1 + \max_{a'} Q(s_{15},a') = 0 + \max_{a'}\{0,0,0,0\} = 1$$

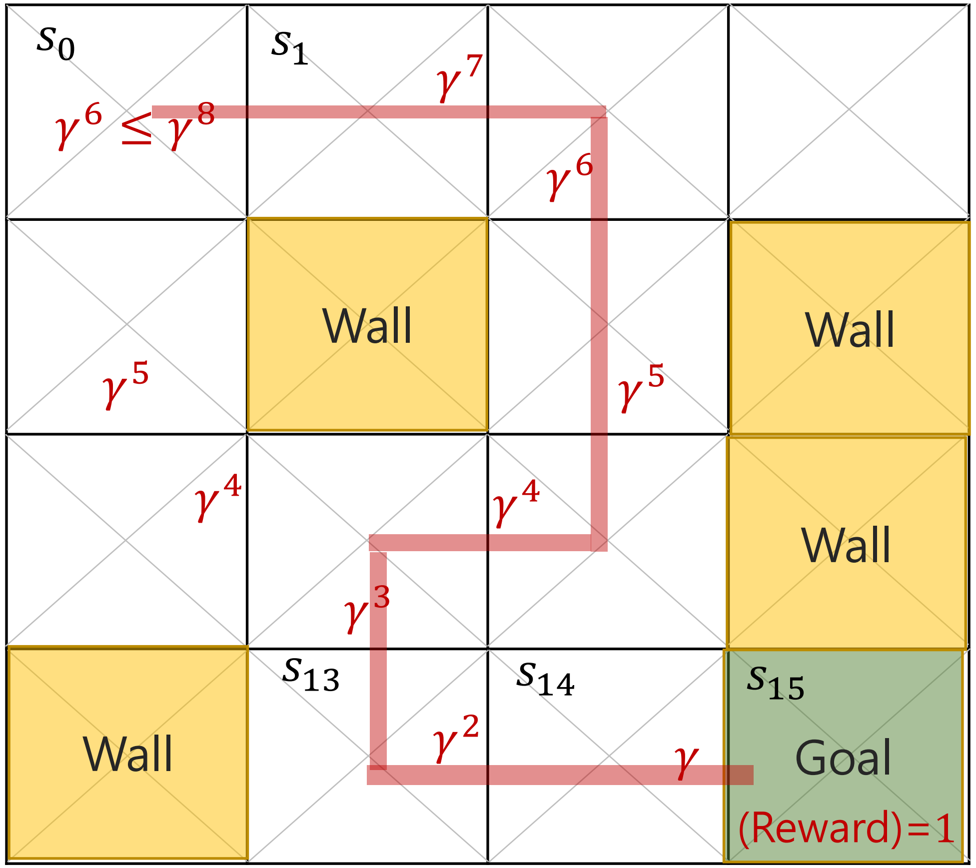
Case (3) →

$$Q(s_{13},R) = r + \max_{a'} Q(s_{14},a') = 0 + \max_{a'}\{0,0,1,0\} = 1$$

Joongheon Kim (https://youtu.be/m1FC3dMmY78)

Q-Learning: Discounted reward

16 states and 4 actions (U, D, L, R)



$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

Joongheon Kim (https://youtu.be/m1FC3dMmY78)

$$Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s',a')$$

Q-Learning: Temporal Difference

- Iterati

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \Bigg)}^{\text{temporal difference}}$$

Q-learning

- For each s, a, initialize table entry $Q(s,a) \leftarrow 0$
- Do until Q converges
  - Initialize s
  - Do until s *is terminal*
    - Select an action a using policy π derived from Q
    - Take action a
    - Receive immediate reward r
    - Observe the new state s'
    - $Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$
    - $s \leftarrow s'$

Policy

- Testing phase
  - Optimal policy $\pi^*(s) = \arg\max_a Q(s,a)$
- Training phase (ϵ-greedy)
  - Exploration-Exploitation
  - Exploration allows an agent to improve its current knowledge
  - Exploitation chooses the greedy action to get the most reward by exploiting the agent's current action-value estimates

$$\text{Action at time(t)} \quad \begin{cases} \max Q_t(a) & \text{with probability } 1-\epsilon \\ \text{any action (a)} & \text{with probability } \epsilon \end{cases}$$

https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/

Q-learning

- For each s, a, initialize table entry $Q(s,a) \leftarrow 0$
- Do until Q converges
  - Initialize s
  - Do until s is terminal
    - Draw a random value $v \sim Uniform(0,1)$
    - If $v < \varepsilon$
      - Randomly select a
    - Else:
      - $a = \underset{a'}{\operatorname{argmax}}\, Q(s,a')$
    - Take action a
    - Receive immediate reward r
    - Observe the new state s'
    - $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    - $s \leftarrow s'$

References

- Fei-Fei Li & Justin Johnson & Serena Yeung, CS231n Lecture 14: Reinforcement Learning, Stanford University
- [https://sumniya.tistory.com/ 숨니의 무작정 따라하기](https://sumniya.tistory.com/)
- https://youtu.be/m1FC3dMmY78 Joongheon Kim, Korea University