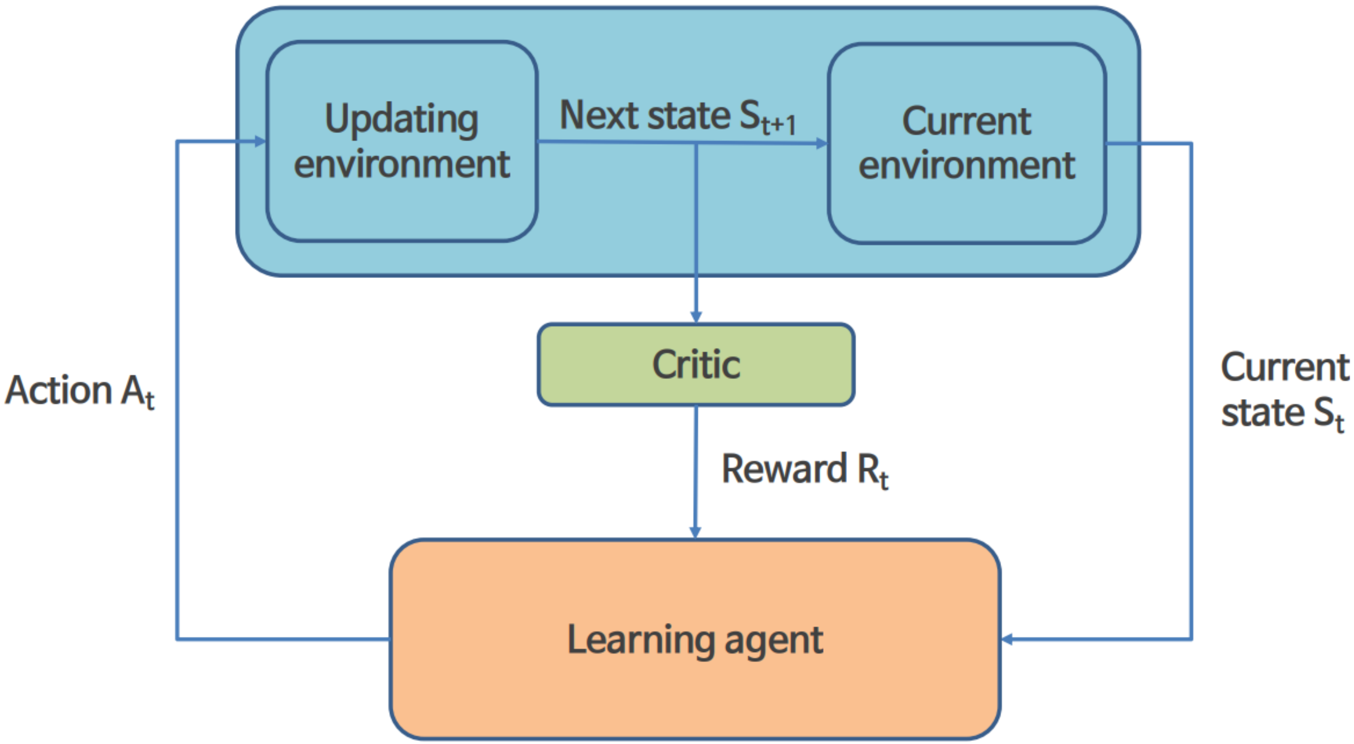
Introduction to Reinforcement Learning 2

Neuro-Machine Augmented Intelligence Lab

Review

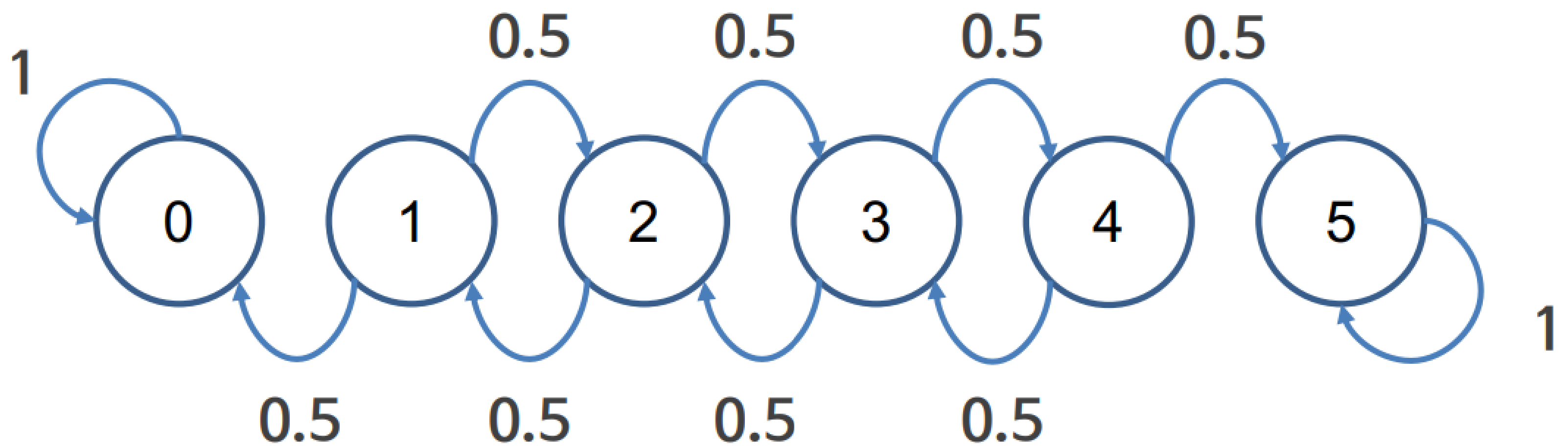
In the previous lecture, you learned:

- Basic concept of reinforcement learning



Previous Reinforcement Learning

- Markov Chain / Markov Decision Process



- State-Value function

$$V^{\pi}(s) = \mathbb{E}_{\tau}[R^{\pi}(\tau)|s_0 = s] = r\big(s, \pi(s)\big) + \gamma \sum_{s'} P_{ss'}^{\pi(s)} V^{\pi}(s')$$

Previous Reinforcement Learning

- Value iteration

1. Let V_0 be *any* vector in R^N
2. At each iteration $k = 1, 2, \ldots, K$
   - Compute $V_{k+1} = \mathcal{T} V_k$

$$\mathcal{T}W(x) = \max_{a \in A} \left[r(x,a) + \gamma \sum_y p(y|x,a) W(y) \right].$$

3. Return the *greedy* policy

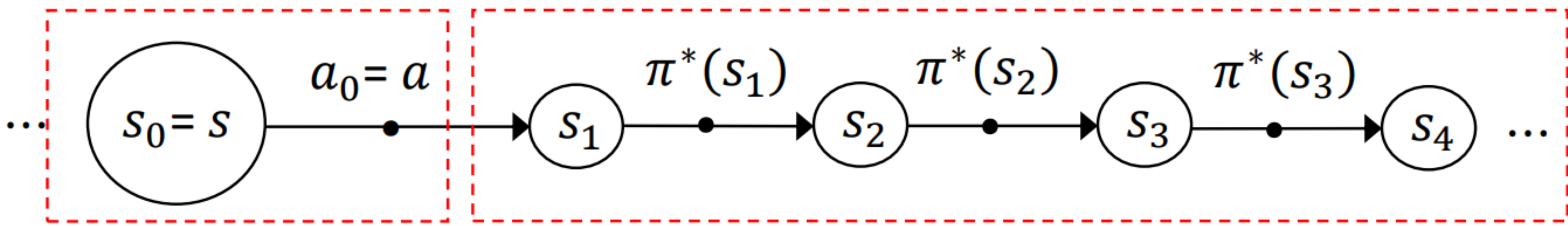
$$\pi_K(x) \in \arg\max_{a \in A} \left[r(x,a) + \gamma \sum_y p(y|x,a) V_K(y) \right].$$

- Policy iteration
- Curses of modeling, dimensionality

State-action Value Function

- Also known as **Q-function**, which measures the value of action a for given state s

$$Q^{\pi^*}(s,a) \stackrel{\text{def}}{=} r(s,a) + \mathbb{E}_{\tau}^{\pi^*}\left[\sum_{t=1}^{\infty} \gamma^t r_t\right]$$



- Optimal policy $\pi*$ is the action that maximizes the sum of current and future rewards

$$\boxed{\pi^*(s) = \underset{a \in A}{\operatorname{argmax}}\, Q^{\pi^*}(s,a)}$$

Action−Value Function

On policy vs. off policy

$$V^{\pi}(s) = r\big(s, \pi(s)\big) + \gamma \sum_{s'} P_{ss'}^{\pi(s)} V^{\pi}(s')$$

$$Q^{\pi}(s,a) = r(s,a) + \gamma \sum_{s'} P_{ss'}^{a} Q^{\pi}(s', \pi(s'))$$

Value iteration

$$V^{\pi}(s) = r(s,a) + \gamma \sum_{s'} P_{ss'}^{a} V^{\pi}(s')$$

$$Q^{\pi^*}(s,a) = r(s,a) + \gamma \sum_{s'} P_{ss'}^{a} \max_{a' \in A} Q^{\pi^*}(s', a')$$

Q-learning

How do we solve the curse of modeling?

With Q-learning (value iteration, but 2nd step is approximated with sampling)!

With the (k+1)th sample of (s, a, r, s'):

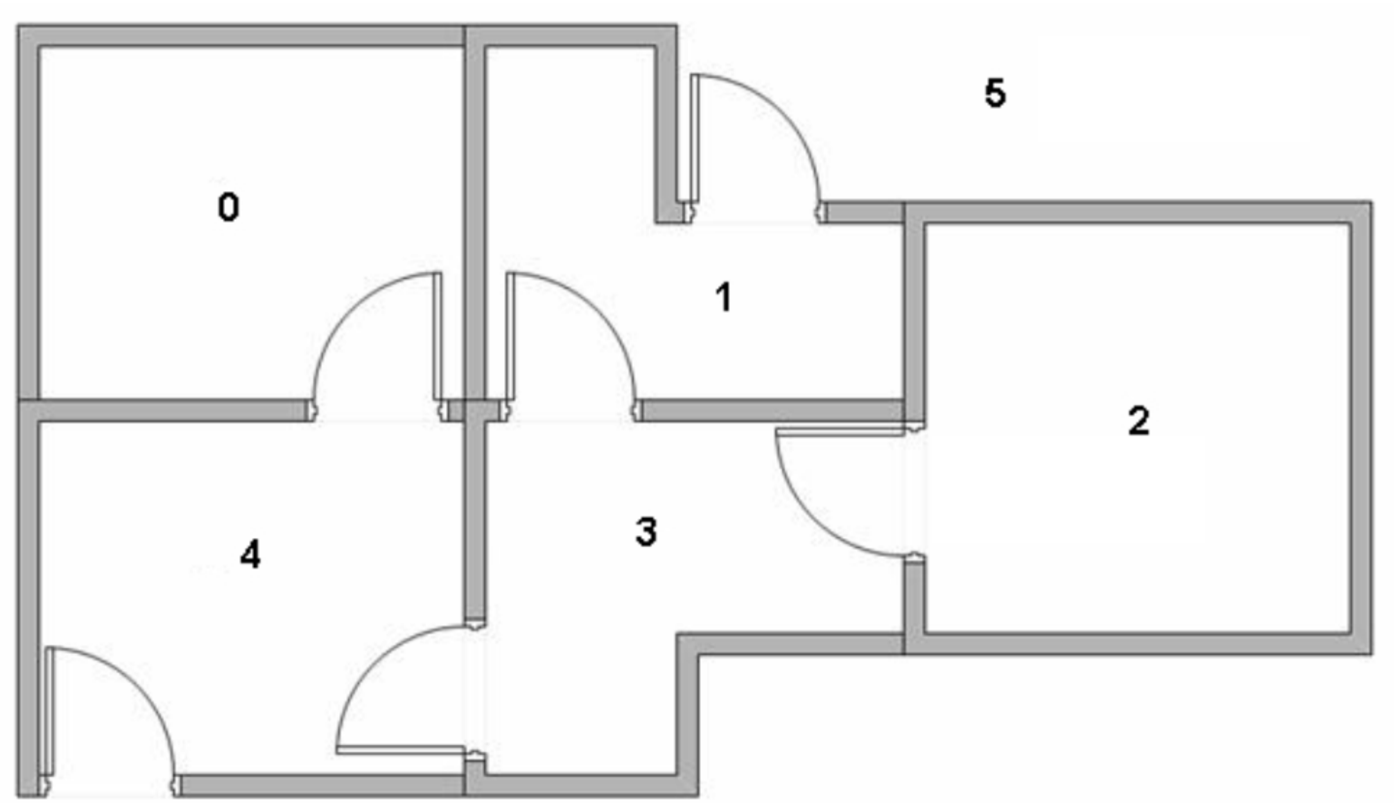
$$Q^*_{k+1}(s,a) \approx Q^*_k(s,a) + \alpha_k \underbrace{(\underbrace{r + \gamma \max_{a' \in A} Q^*_k(s',a')}_{\text{TD target}} - Q^*_k(s,a))}_{\text{Temporal-difference (TD) error}}$$

Q^*_{k+1} converges to the optimal Q-function after many iterations (which should contain all state-action pairs).

As you can see, Q-learning is model-free (no need to know transition probabilities)

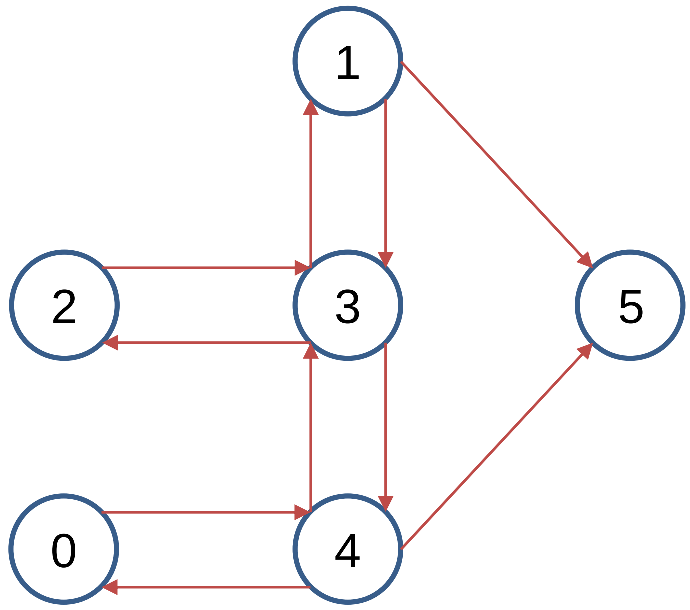
Q- Learning Example

- In this example, we will show how to find the optimal way to get to room 5 from inside the room with the Q-learning algorithm.



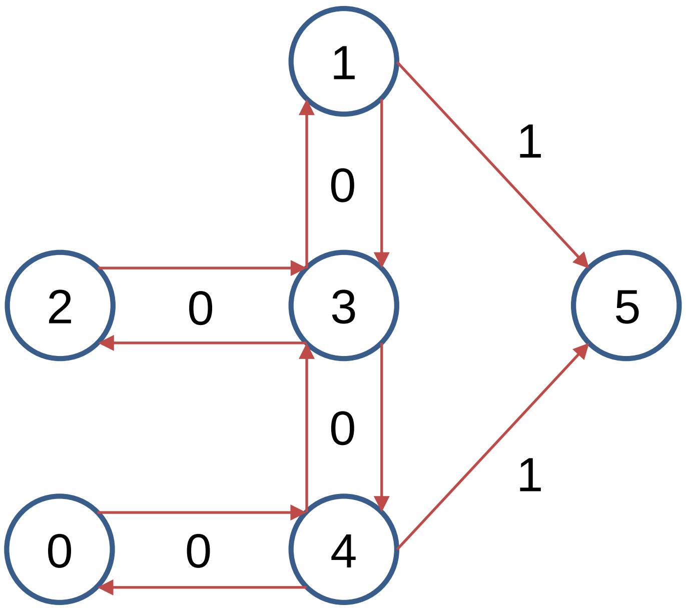
Q- Learning Example

- We can represent this problem with a graph in which each room is a node, and each door is a link



Q- Learning Example

- Now we define the reward for each link. The doors that lead immediately to the goal (node 5) have a reward of 1.
- Other doors not directly connected to the target have zero reward like below.



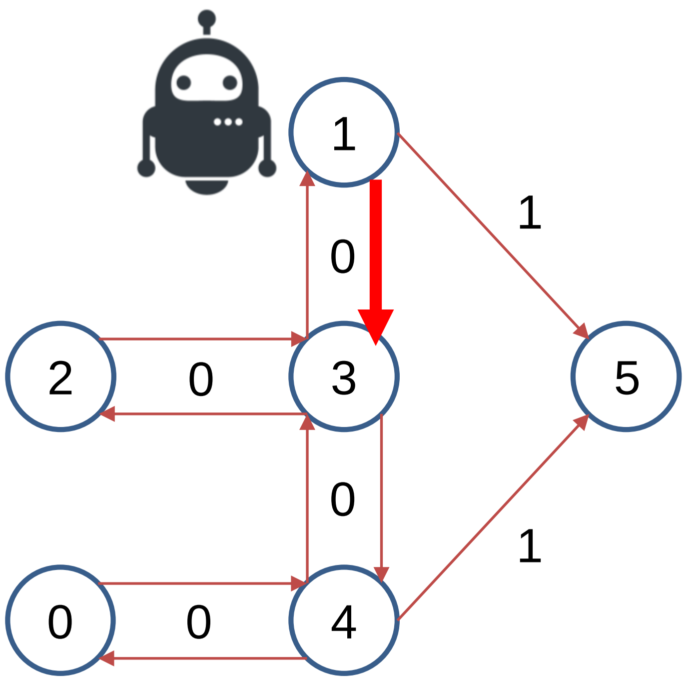
Q- Learning Example

- We can summarize the rewards for each state-action pair like the below table (R = reward table).
- The “state” means the current position of the agent and the “action” means moving to the room of that number.
- −1 represents an impossible state-action pair.

$$
R = \begin{array}{c c} & \text{Action} \\ \text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 1 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 1 \end{pmatrix} \end{array}
$$

Q- Learning Example

- For example (state :1 , action:3) means the agent in room 1 took an action to move to room 3, and the reward is 0



State	Action 0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	1
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	1

Q- Learning Example

- The Q-Learning algorithm (which is essentially value iteration) goes as follows:
  1. Set environment rewards and both the alpha and gamma parameters to a value between 0 and 1.
  2. Initialize matrix Q (or the Q-table).
  3. For each episode:
     - Set the initial state.
       Do while the current state is not a terminal state
       - Choose one among all possible actions for the current state.
       - Compute:

$$Q^*_{k+1}(s,a) \approx Q^*_k(s,a) + \alpha_k(r + \gamma \max_{a' \in A} Q^*_k(s',a') - Q^*_k(s,a))$$

       - Set the next state (determined by the action that was chosen) as the current state.

  4. The optimal policy is then the action that has the max value in the Q-table for each state.

Q- Learning Example

- As we start, we will set the parameters gamma as 0.8, alpha as 1, and initial state as Room 1
- Initialize the Q-table as a zero matrix like below (Don't confuse this with the R matrix!)

$$
Q = \begin{array}{c|c}
 & \text{Action} \\
\text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\end{array}
$$

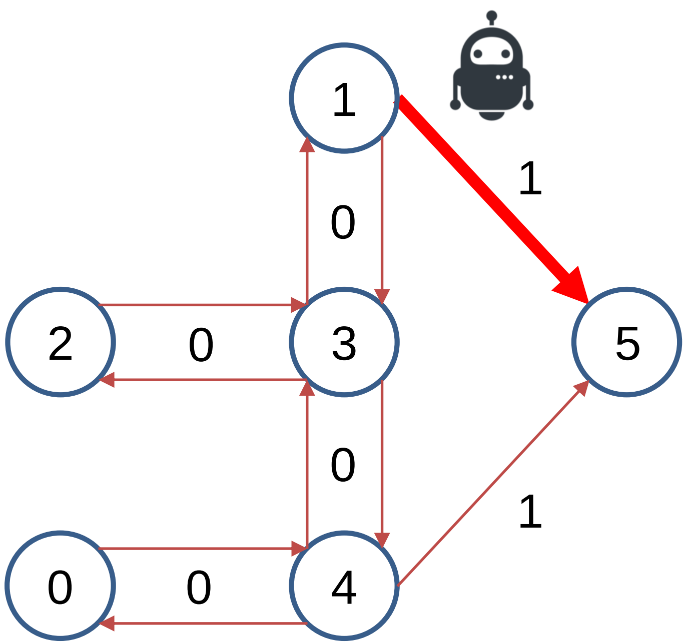
Q- Learning Example

- Look at the second row (state 1) of matrix R. There are two possible actions for the current state 1: go to state 3, or go to state 5. Let's say we select to go to 5 as our action.

$$
\begin{array}{cc}
 & \text{Action} \\
\text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & \boxed{0} & -1 & \boxed{1} \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 1 \end{pmatrix}
\end{array}
$$

Q- Learning Example

- We update the Q-table.



State	Action 0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	1
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	1

$$Q(1, 5) = R(1, 5) + 0.8 * \mathrm{Max}\,[\underline{Q(5, *)}] = 1 + 0.8 * 0 = 1$$

Since room 5 is a terminal state, $Q(5, *) = 0$.

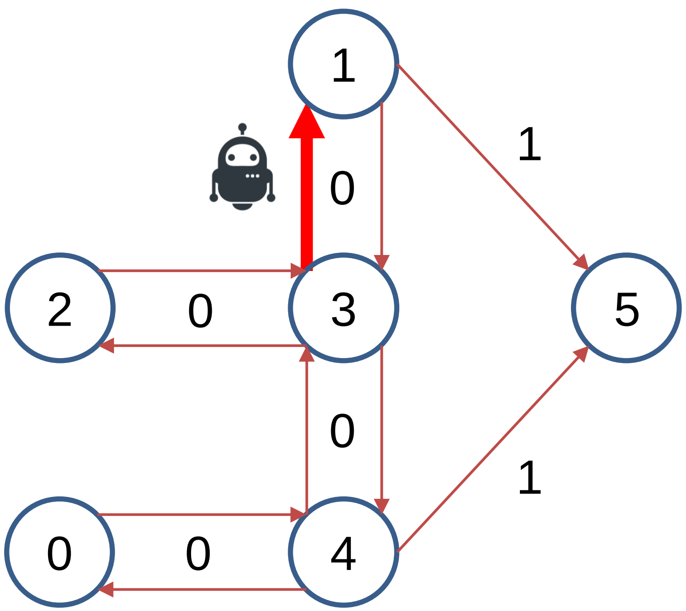
Q- Learning Example

- And then we can update the Q-table like below

$$
Q = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array}
\begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}
$$

Q- Learning Example

- For the next episode, let’s say state 3 is our initial state.
- Look at the fourth row of matrix R; it has 3 possible actions: go to state 1, 2 or 4. By random selection, we select to go to state 1 as our action.



$$
\begin{array}{c|c}
 & \text{Action} \\
\text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} &
\begin{pmatrix}
-1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & 0 & -1 & 1 \\
-1 & -1 & -1 & 0 & -1 & -1 \\
-1 & \boxed{0} & \boxed{0} & -1 & \boxed{0} & -1 \\
0 & -1 & -1 & 0 & -1 & 1
\end{pmatrix}
\end{array}
$$

$$Q(3, 1) = R(3, 1) + 0.8 * \text{Max}[Q(1, 3), Q(1, 5)] = 0 + 0.8 * 1 = 0.8$$

Q- Learning Example

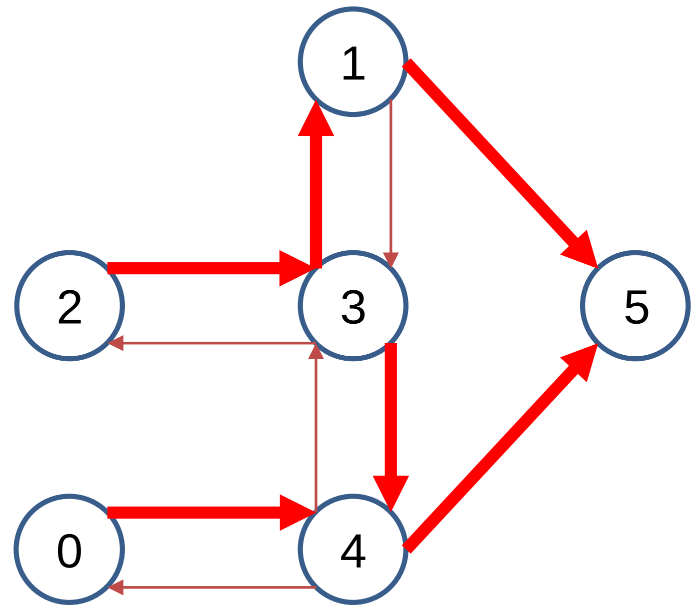
- And then we can update the value of Q(3,1) in Q Matrix like below

$$Q = \begin{array}{c c} & \text{Action} \\ \text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Q- Learning Example

- Eventually the Q matrix converges

$$
Q = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array}
\begin{array}{c}
\begin{array}{cccccc} & & \text{Action} & & & \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{pmatrix}
0 & 0 & 0 & 0 & 0.8 & 0 \\
0 & 0 & 0 & 0.64 & 0 & 1 \\
0 & 0 & 0 & 0.64 & 0 & 0 \\
0 & 0.8 & 0.51 & 0 & 0.8 & 0 \\
0.64 & 0 & 0 & 0.64 & 0 & 1
\end{pmatrix}
\end{array}
$$



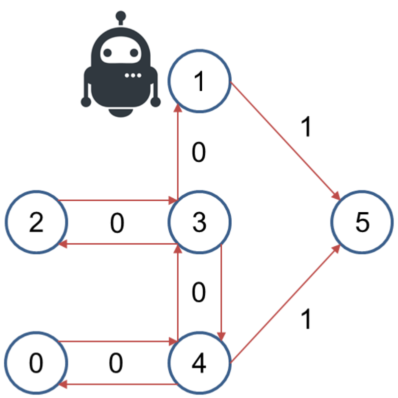
Random Choice of the Action

- Reinforcement learning needs to balance exploitation and exploration
  - Exploitation: going for the best path
  - Exploration: choosing a random path just for the sake of exploring new paths

Epsilon-greedy algorithm:
Choose an exploration probability ε
Choose to either exploit or explore according to the probability (where exploiting would be the best action by looking at the Q-table)

Conclusion

- We learned about State-action value function (Q-function)
- We discussed Q-Learning process
- We Applied Q-learning concept in house escape example



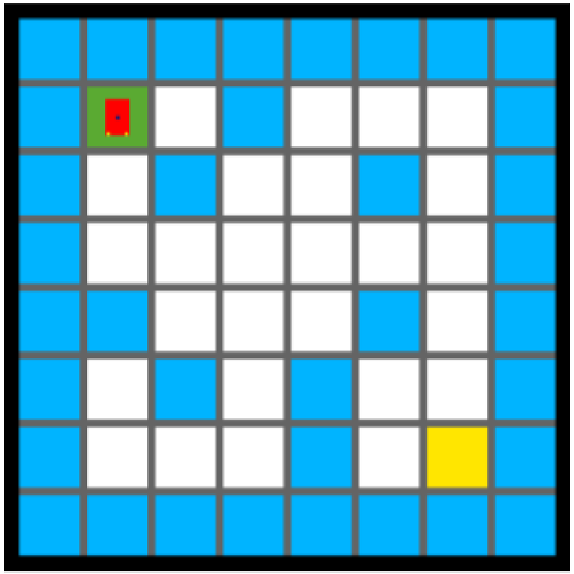
	Action					
State	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	1
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	1

Assignment 3

- The third assignment is open on KLMS
- You will be provided a Frozen Lake map like side picture
- Your goal is to train your robot (the red one) to reach the yellow destination point with Q-learning method

- You can refer to the uploaded documentation in KLMS to get more detailed information

- Due date : 5/25 23:59
- Submission Guidelines
  submit a zip file named "studentid_studentname.zip" containing the logs folder, the Q-table file named "q_table.npy", and student.py to KLMS



- If you have any questions, contact Email to TA with the following address
  - munjw777@kaist.ac.kr
  - shk0724@kaist.ac.kr
  - hochang.lee91@kaist.ac.kr

Thank you