Q-Learning

'StarCraft II로 배우는 강화학습' 웨비나

Aug 25, 2020

박석

Agenda

1. Monte Carlo Methods
- **MC Prediction**
- **MC Control**

2. Temporal Difference Methods
- **SARSA**
- **Q-Learning**
- **Expected SARSA**

RL Goal : Maximize expected cumulative reward

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

MDP

Definition

A (finite) Markov Decision Process (MDP) is defined by:

- a (finite) set of states $\mathcal{S}$
- a (finite) set of actions $\mathcal{A}$
- a (finite) set of rewards $\mathcal{R}$
- the one-step dynamics of the environment

$$p(s', r \mid s, a) = \mathbb{P}(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

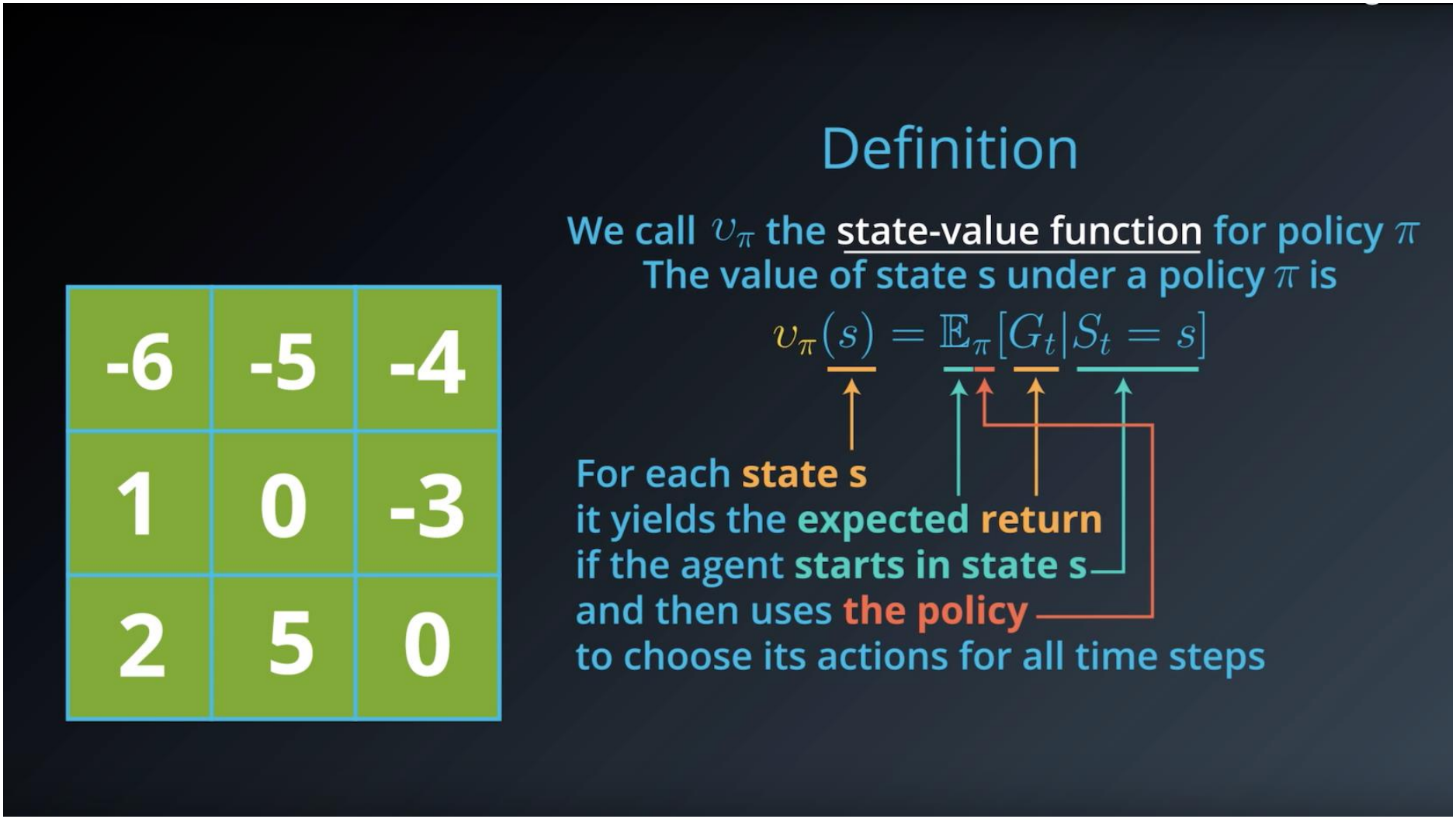
for all s, s', a and r

- a discount rate $\gamma \in [0, 1]$

- Agent knows - Agent does not know

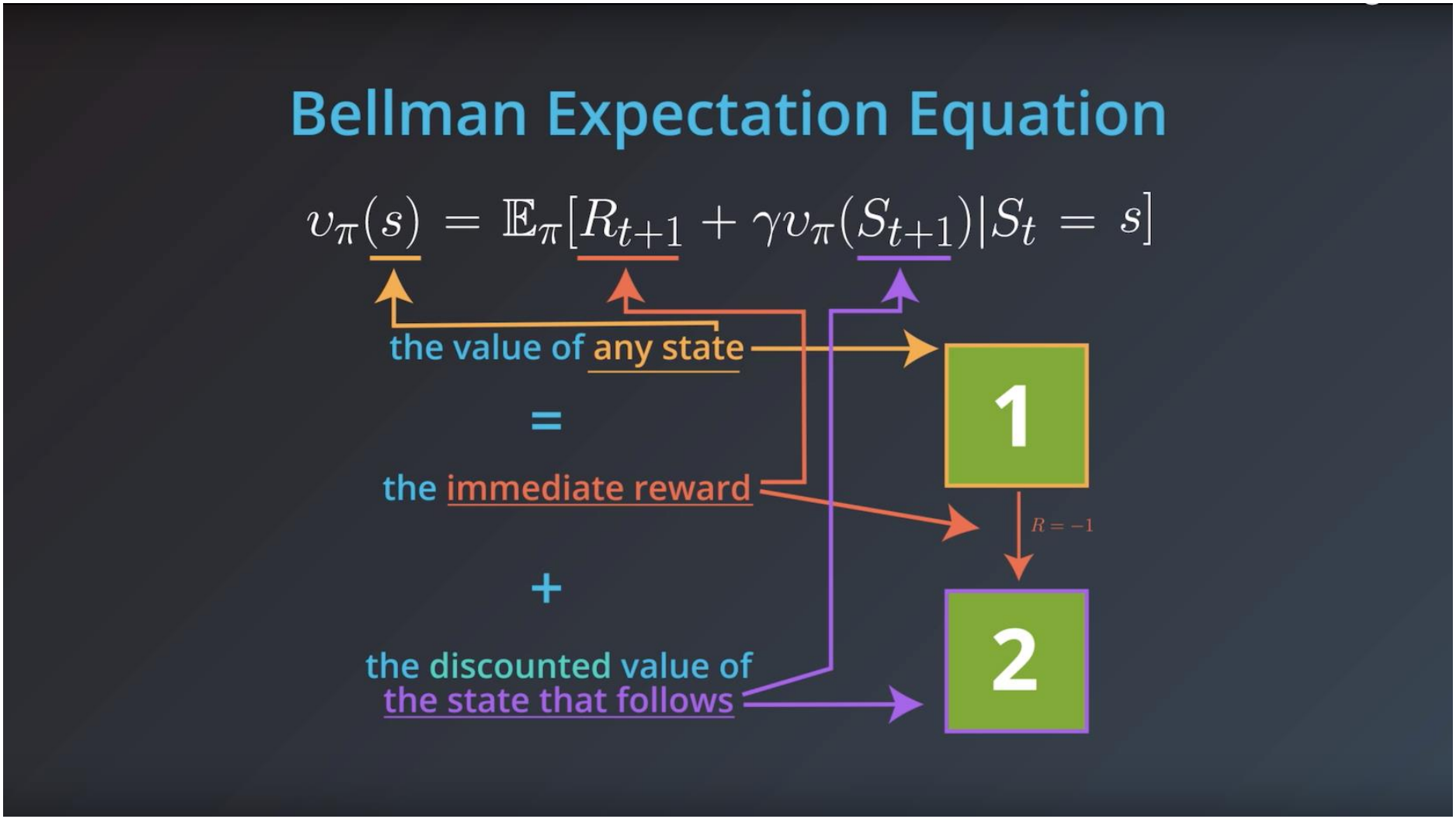
Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Definition of State-Value function



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Bellman Expectation Equation explanation



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Optimal Policy Definition and Notation pi star

Definition

$\pi' \geq \pi$ if and only if $v_{\pi'}(s) \geq v_{\pi}(s)$ for all $s \in \mathcal{S}$

(Note: It is often possible to find two policies that cannot be compared.)

Definition

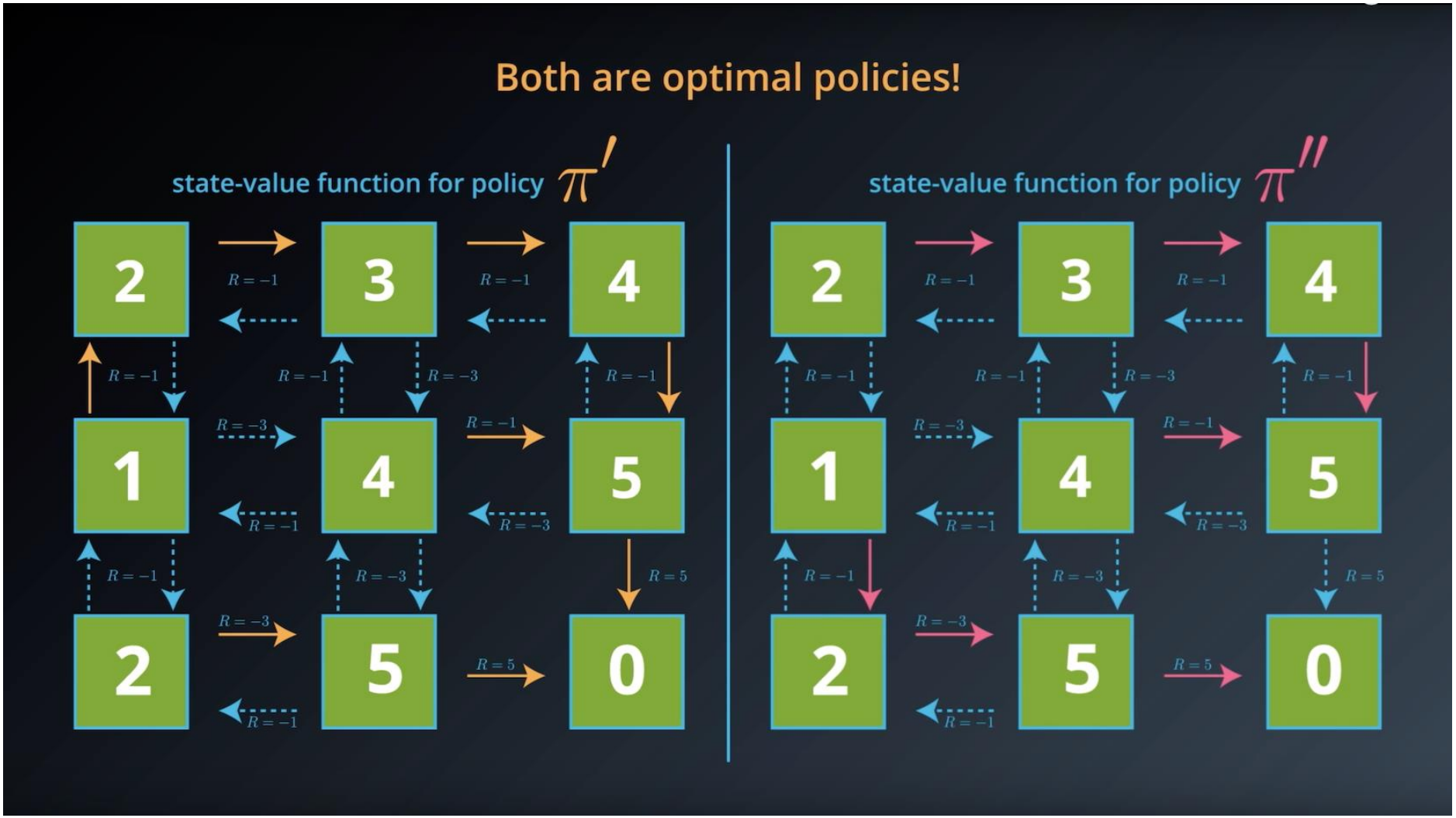
An optimal policy π_* satisfies $\pi_* \geq \pi$ for all π

(Note: An optimal policy is guaranteed to exist, but may not be unique.)

The optimal state-value function is denoted v_*

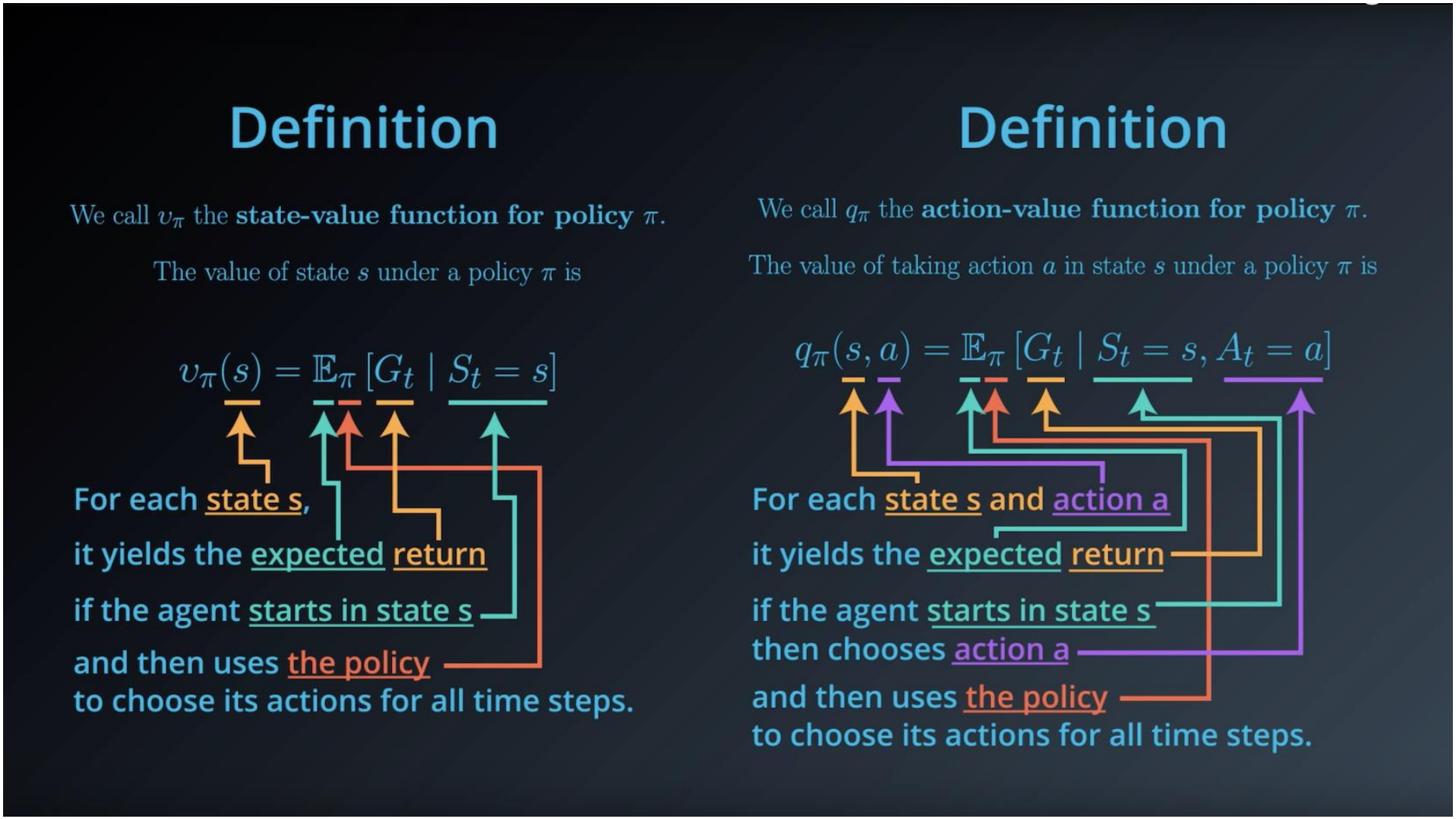
Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Optimal Policy : There could be many Optimal Policies



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Action-Value Function



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

State-Value Function vs Action-Value Function

For each state, the state-value function yields the expected return, if the agent starts in that state, and then follows the policy for all time steps.

For each state and action, the action-value function yields the expected return, if the agent starts in that state, takes the action, and then follows the policy for all future time steps.

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Optimal Action-Value Function Definition

Definition

$\pi' \geq \pi$ if and only if $v_{\pi'}(s) \geq v_{\pi}(s)$ for all $s \in \mathcal{S}$

(Note: It is often possible to find two policies that cannot be compared.)

Definition

An optimal policy π_* satisfies $\pi_* \geq \pi$ for all π

(Note: An optimal policy is guaranteed to exist, but may not be unique.)

The optimal state-value function is denoted v_*

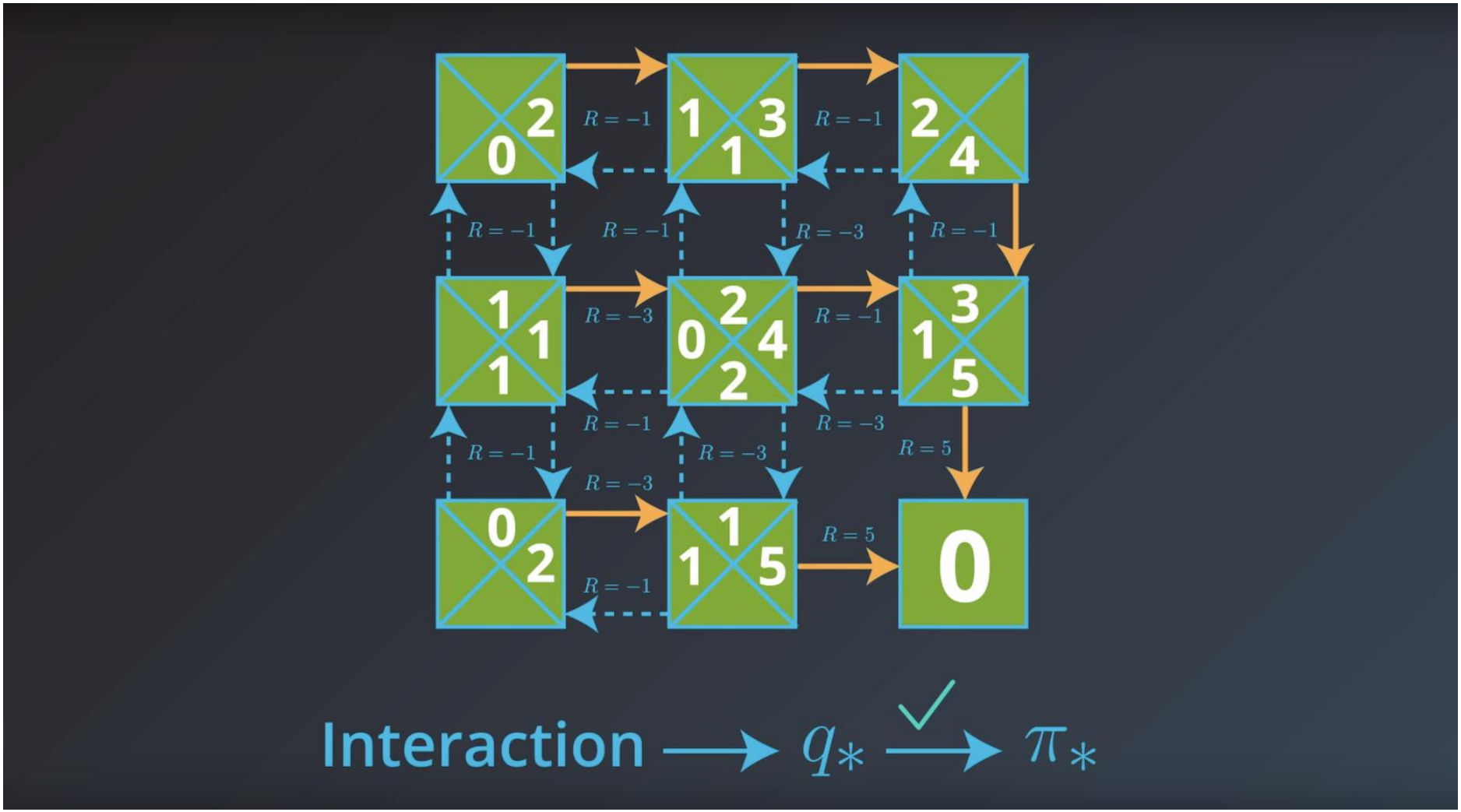
The optimal action-value function is denoted q_*

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

If we would have Optimal Action-Value Function then could we have Optimal Policy?

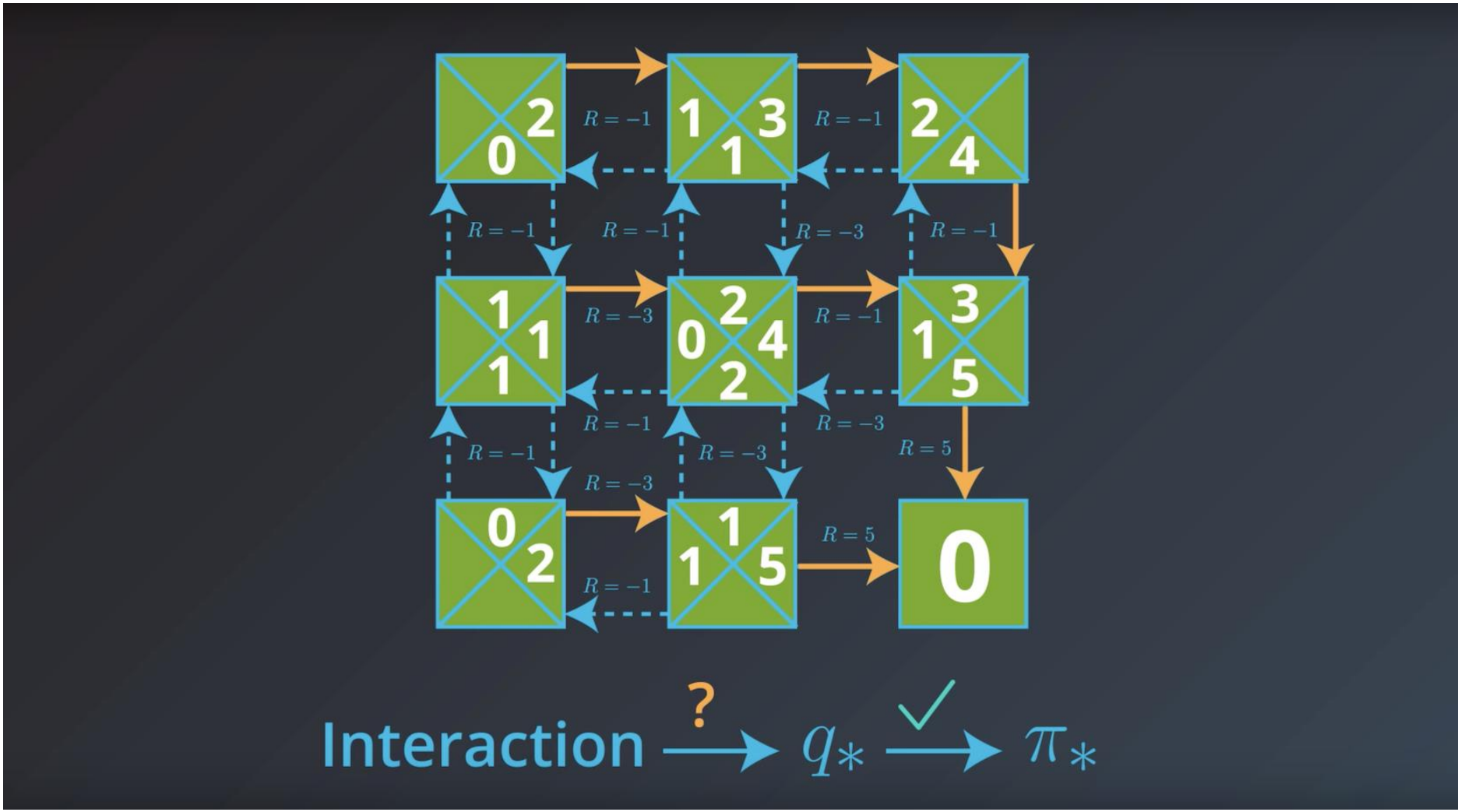
Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

If we would have Q star then we could get Pi star



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

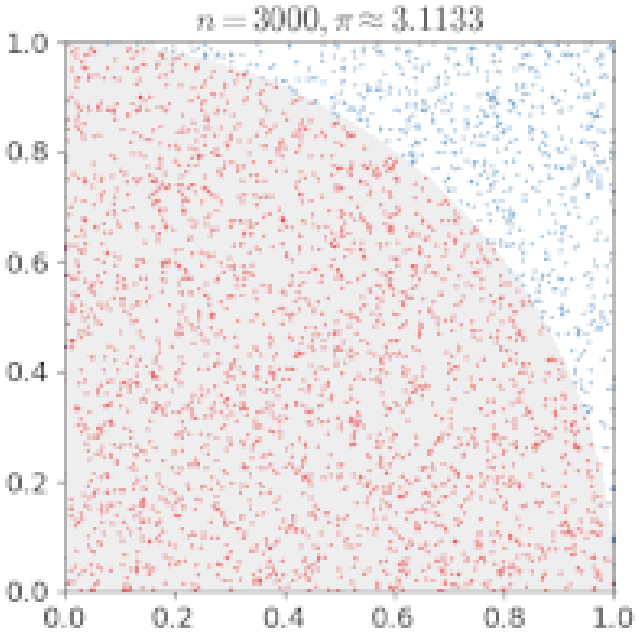
Getting Q star is important in RL.



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TRL – Monte Carlo Methods

계산하기 어려운 값을
다수의 확률 시행을 거쳐 **Estimation** 하는 기법



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Monte Carlo Methods

MC로 주어진 정책 π 에 대해서 $V_\pi(s)$ 를 Estimation 하는 것이 목적임.

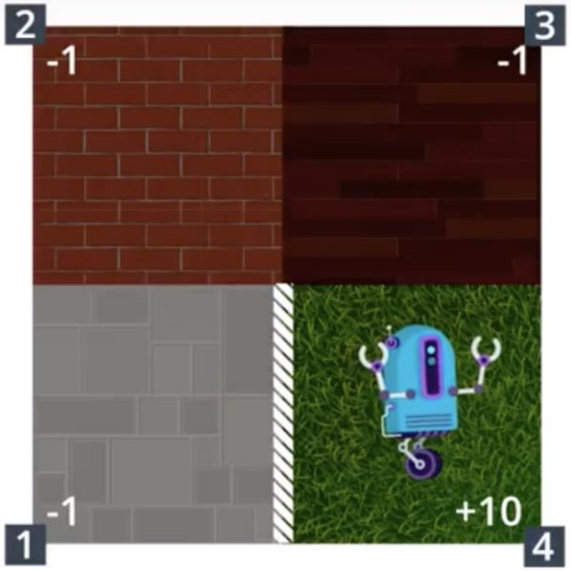
$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

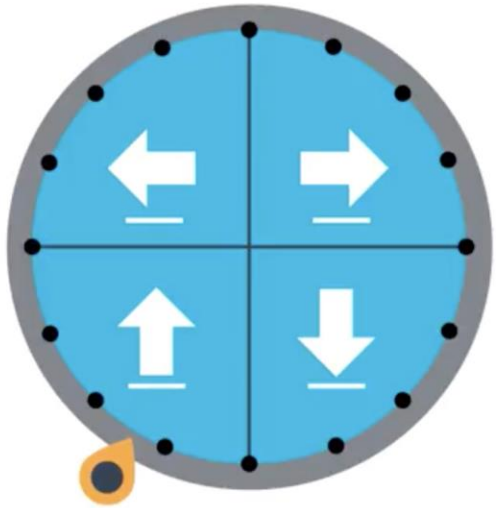
$$G_t \stackrel{\text{def}}{=} R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \ldots + \gamma^{T-1} R_T$$

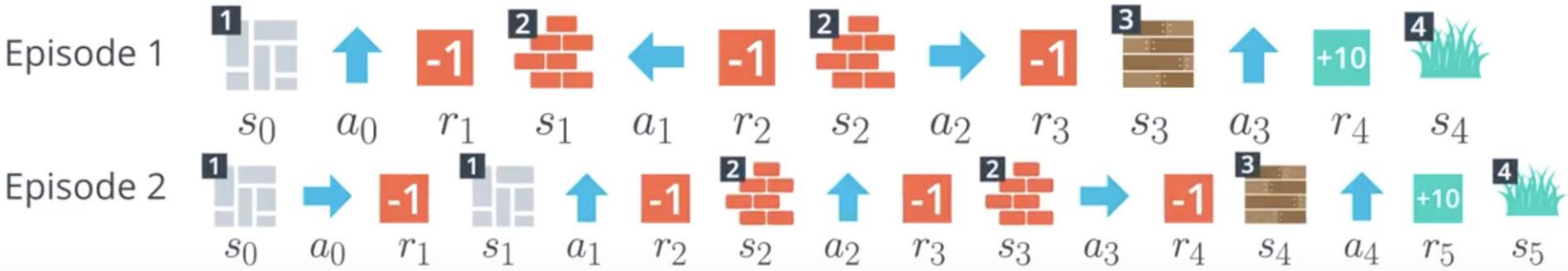
G_t 를 여러 번 샘플링해서
그 값들의 평균을 Calculation하면
$V_\pi(s)$의 값과 비슷해 진다.

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

MC : Equiprobable Random Policy





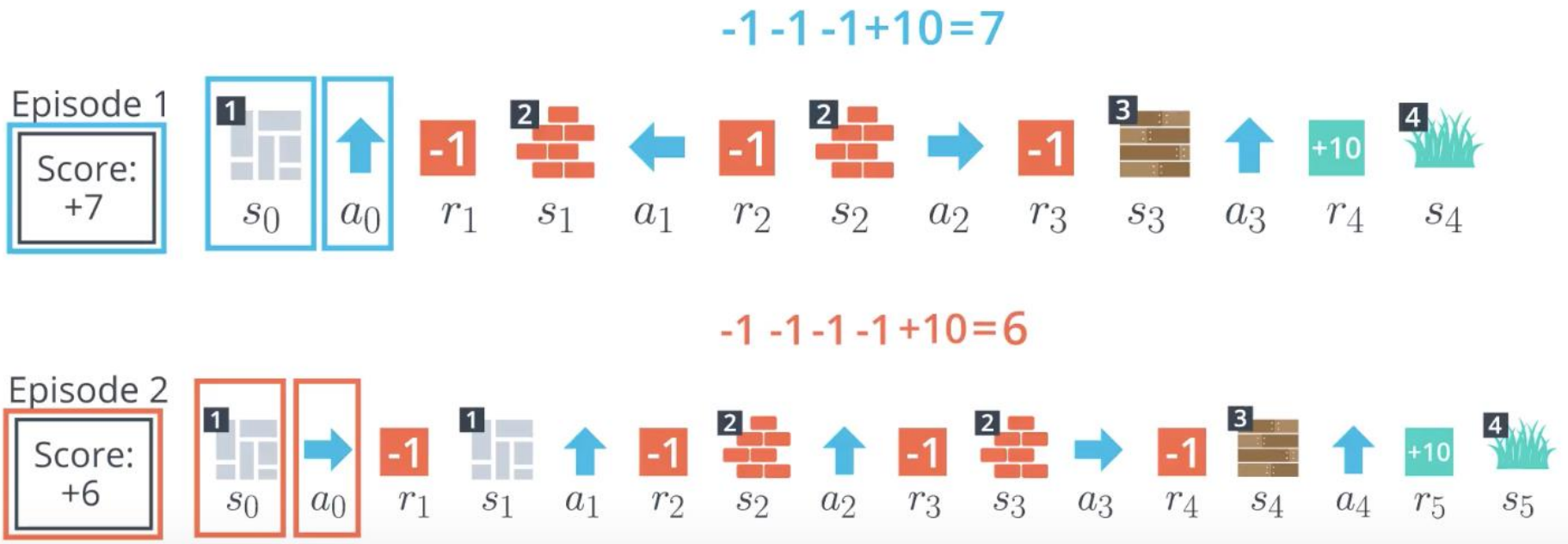


Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

MC : Each episode tells us which action is best for each state

Remember: the agent is looking for the optimal policy π_*

It tells us - for each state - which action is best.

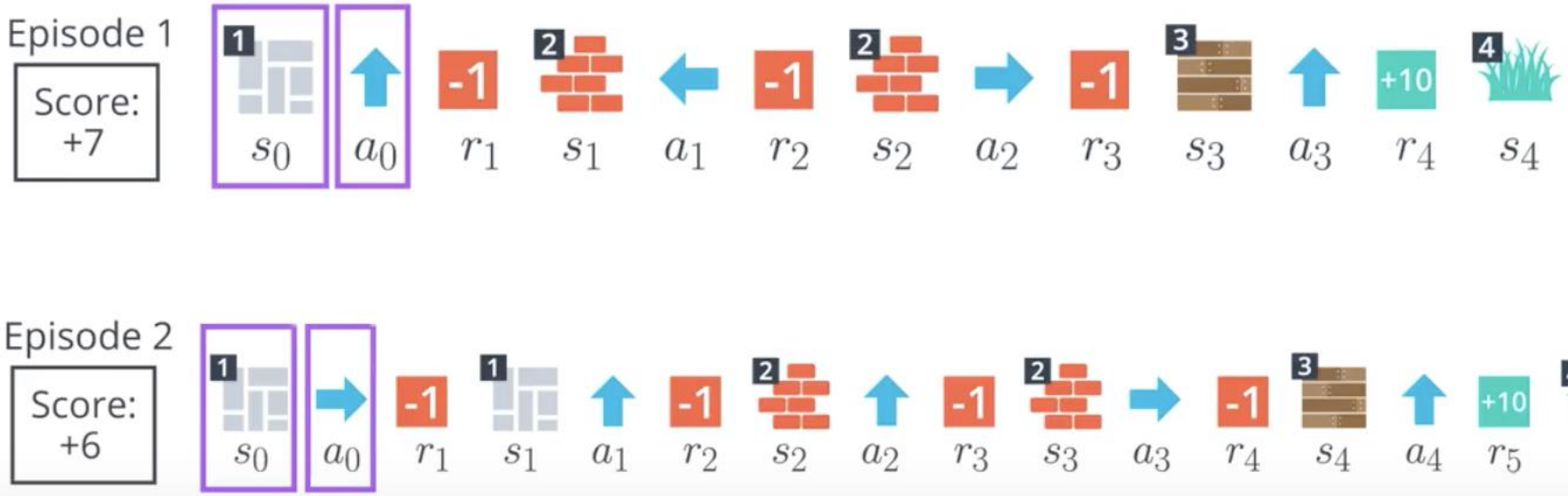


Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Monte Carlo Methods

To truly understand the environment, the agent needs more episodes!

Reason 1: The agent hasn't attempted each action from each state.

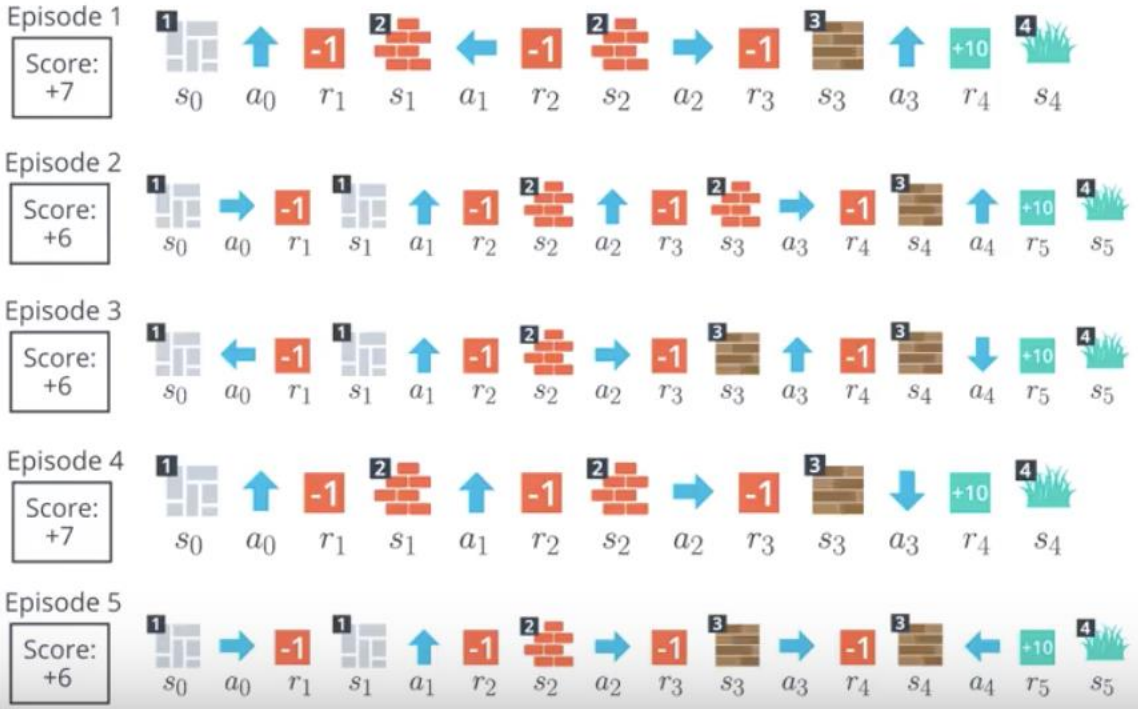


Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Monte Carlo Methods

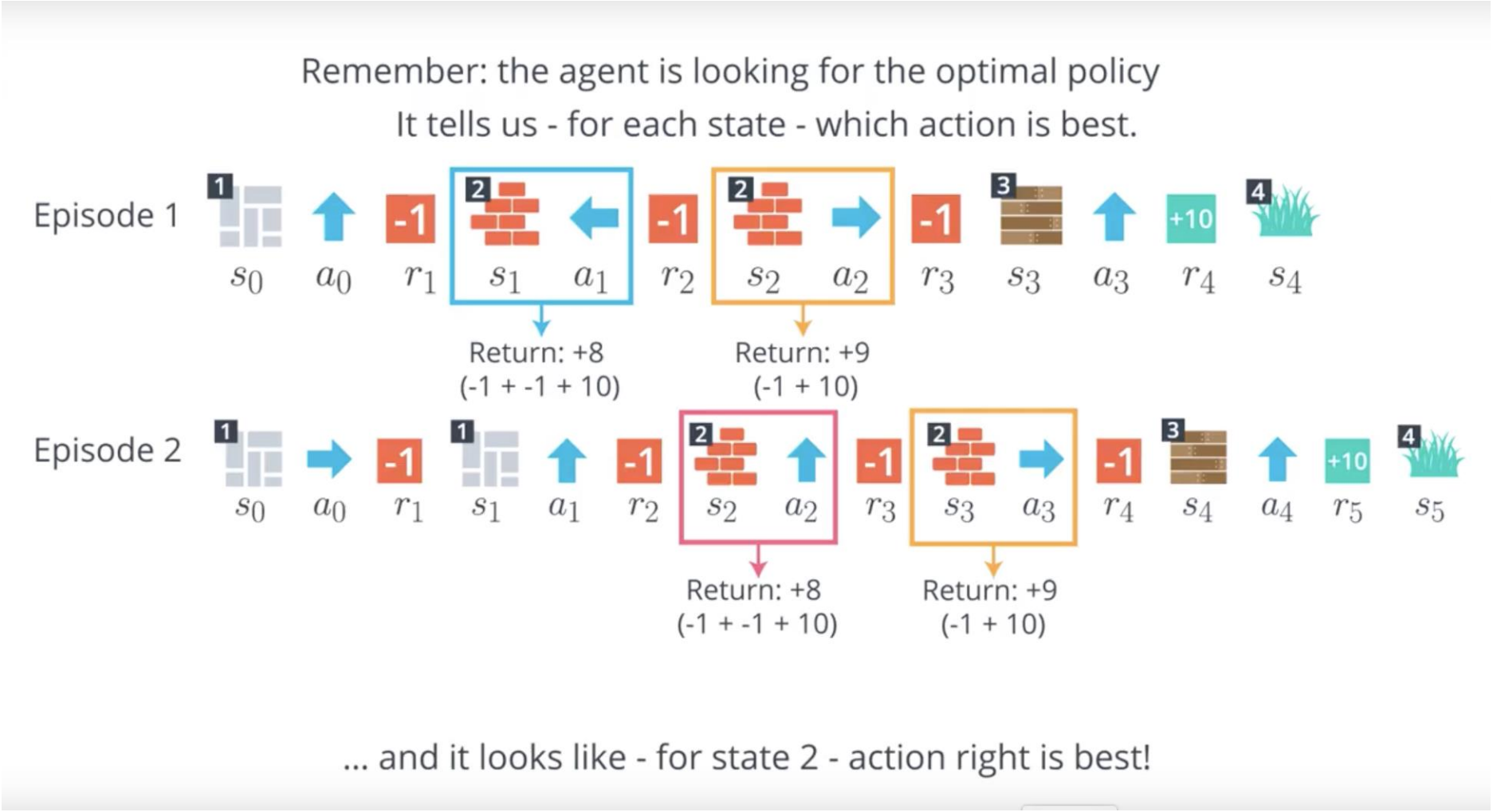
To truly understand the environment, the agent needs more episodes!

Reason 2: The environment's dynamics are stochastic!



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Monte Carlo Prediction



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Monte Carlo Prediction

For each state - which action is best?

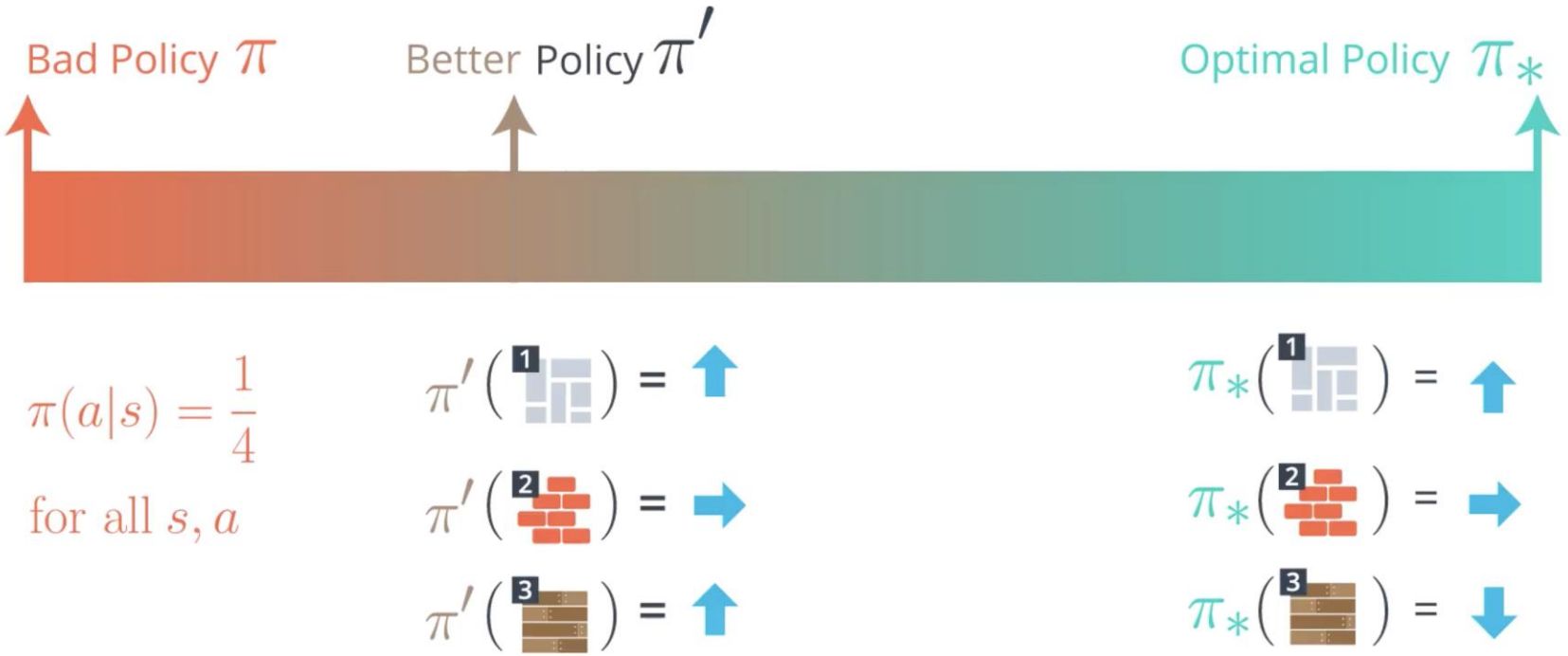
$\pi'($ 1 $) =$ ↑

$\pi'($ 2 $) =$ →

$\pi'($ 3 $) =$ ↑

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

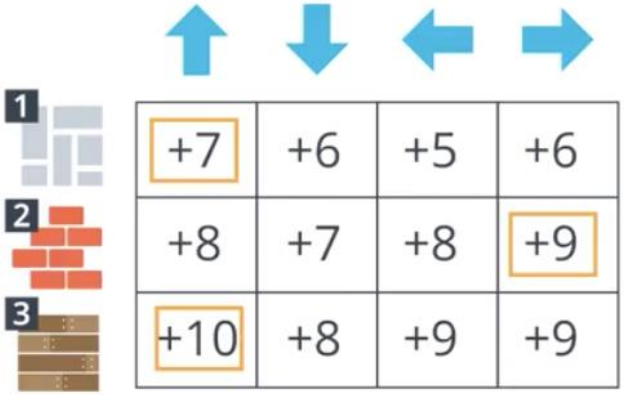
Monte Carlo Prediction



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Greedy Policies

Use the Q-table



To find a better policy π'

$\pi'($  $) =$ ↑

$\pi'($  $) =$ →

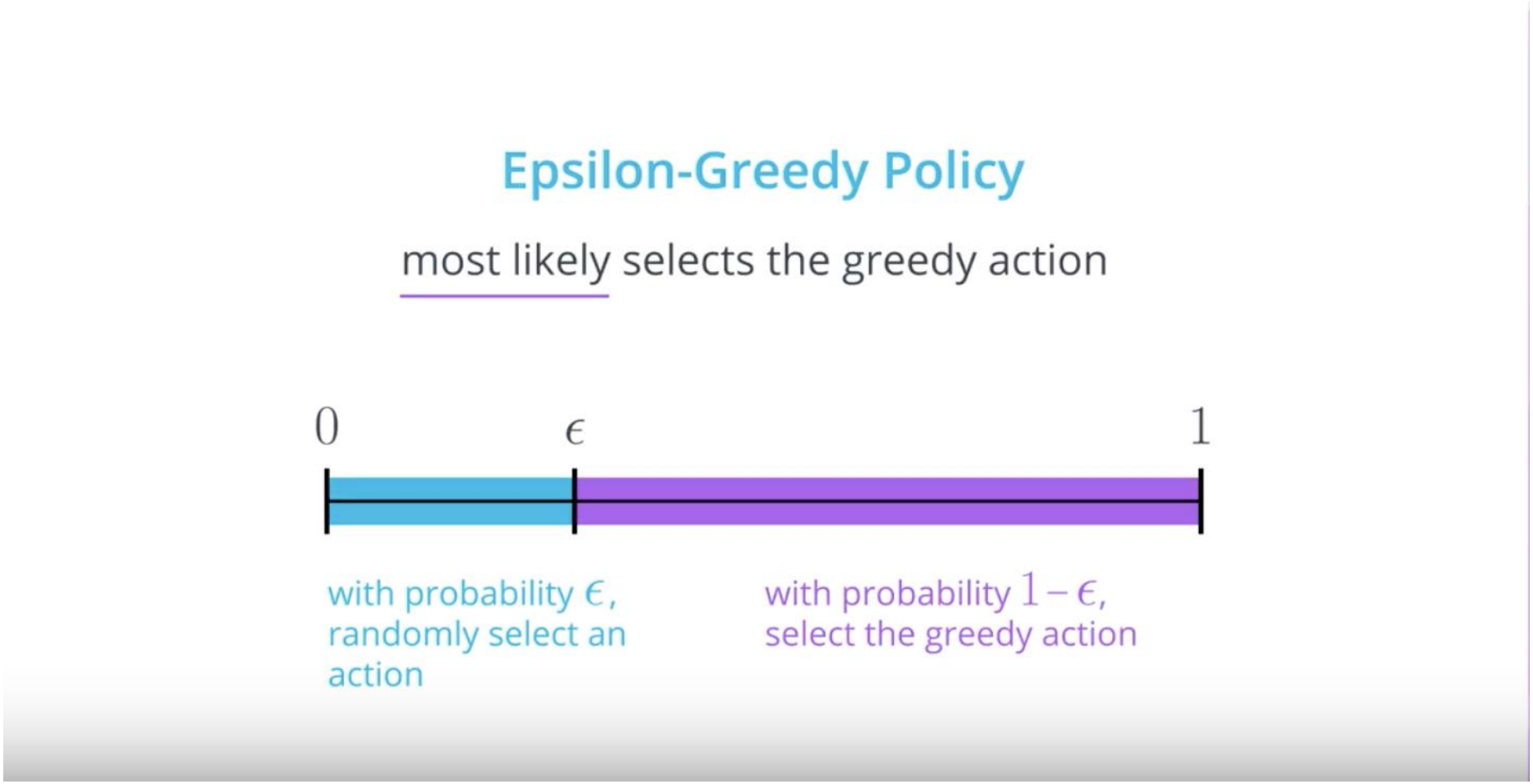
$\pi'($  $) =$ ↑

... in other words: construct the policy that is greedy with respect to the Q-table

$$\pi' \leftarrow \text{greedy}(Q)$$

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Epsilon-Greedy Policies



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Greedy vs E-Greedy Policies

Greedy Policy

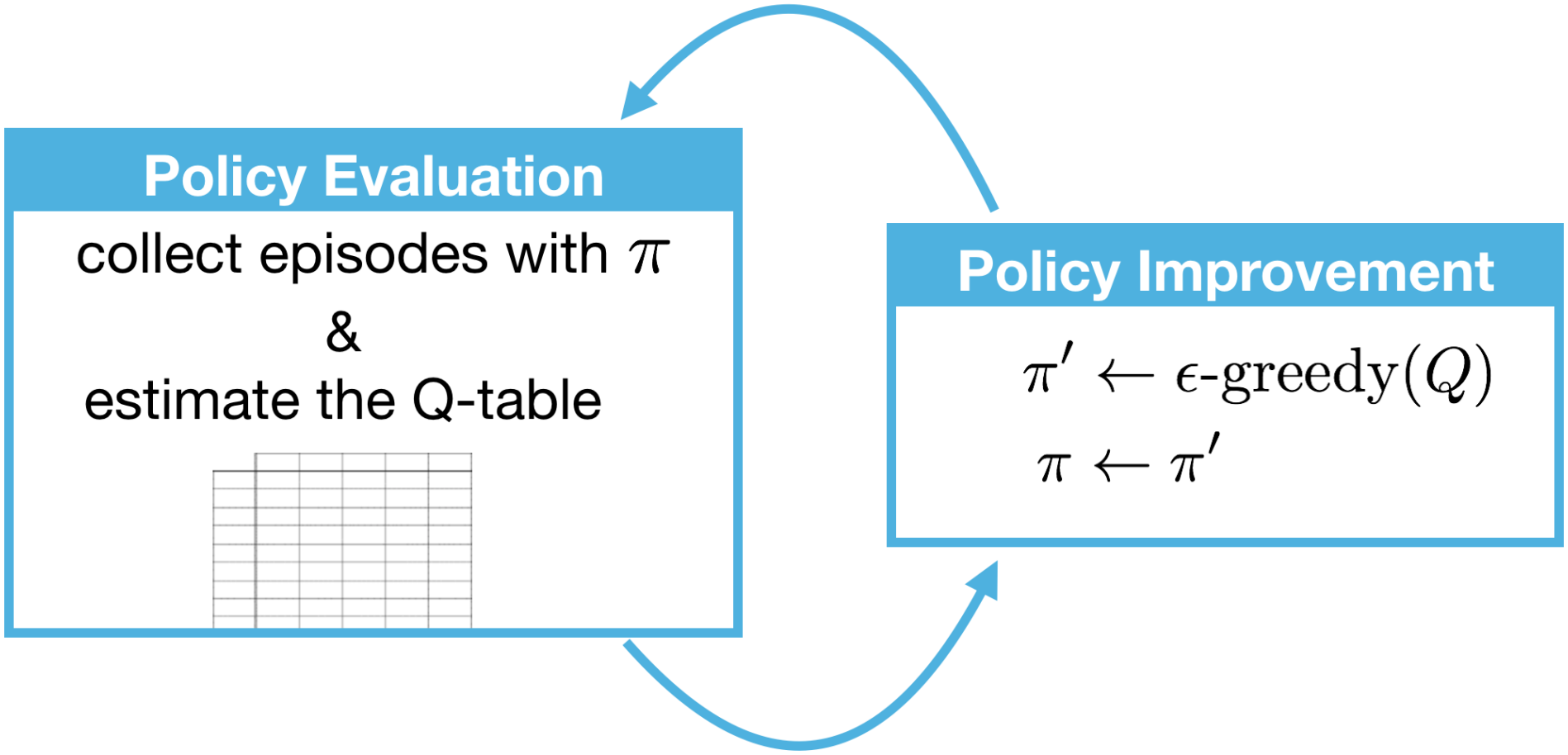
always selects the greedy action

Epsilon-Greedy Policy

most likely selects the greedy action

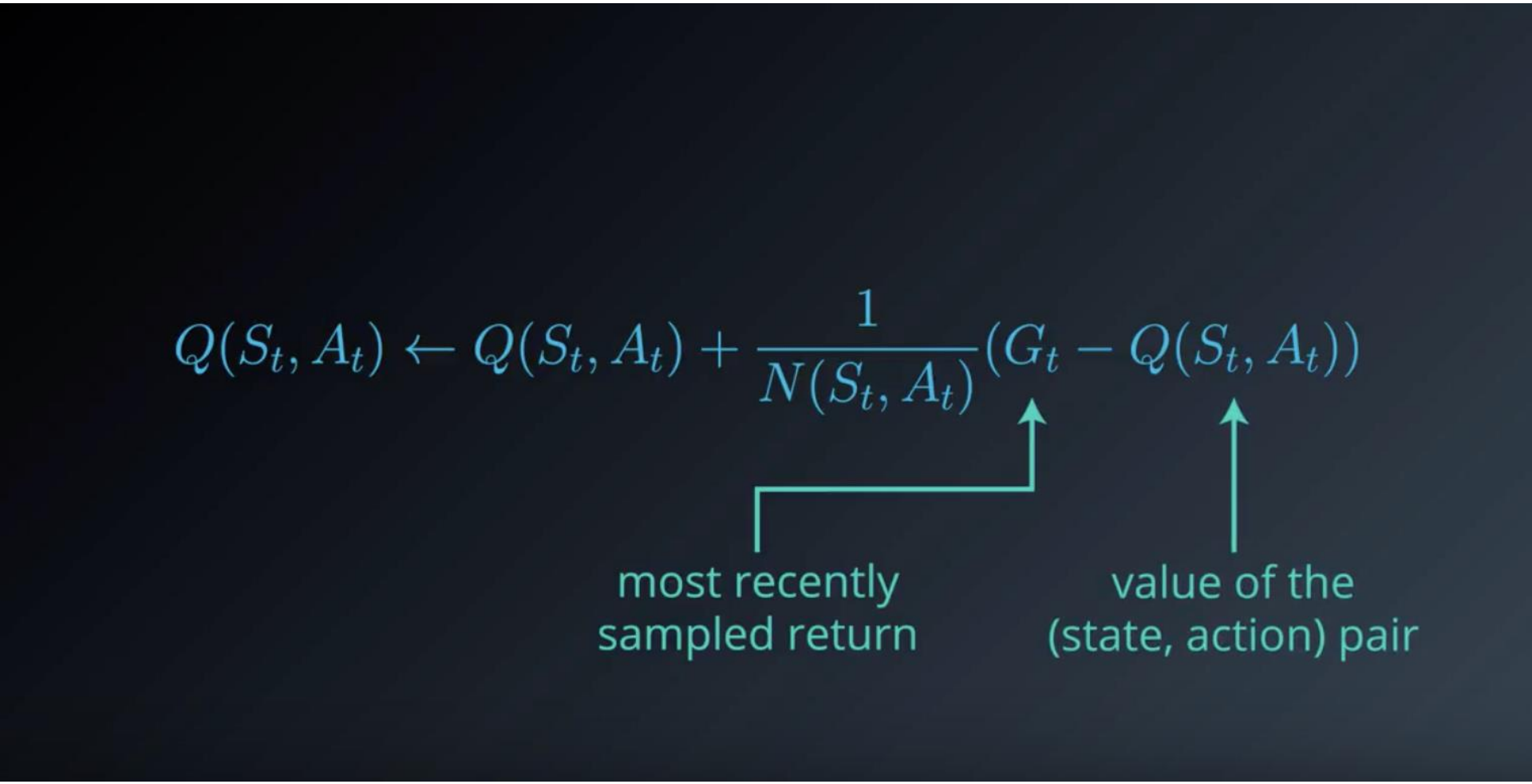
Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Monte Carlo Control



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Monte Carlo Control



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Monte Carlo Control

Initialize $Q(s,a) = 0$ for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$.
Begin with starting policy π.
Repeat:

Evaluation Generate an episode $S_0, A_0, R_1, ..., S_T$ using π.

For $t \leftarrow 0$ to $T-1$:

If (S_t, A_t) is a first visit (with return G_t):

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

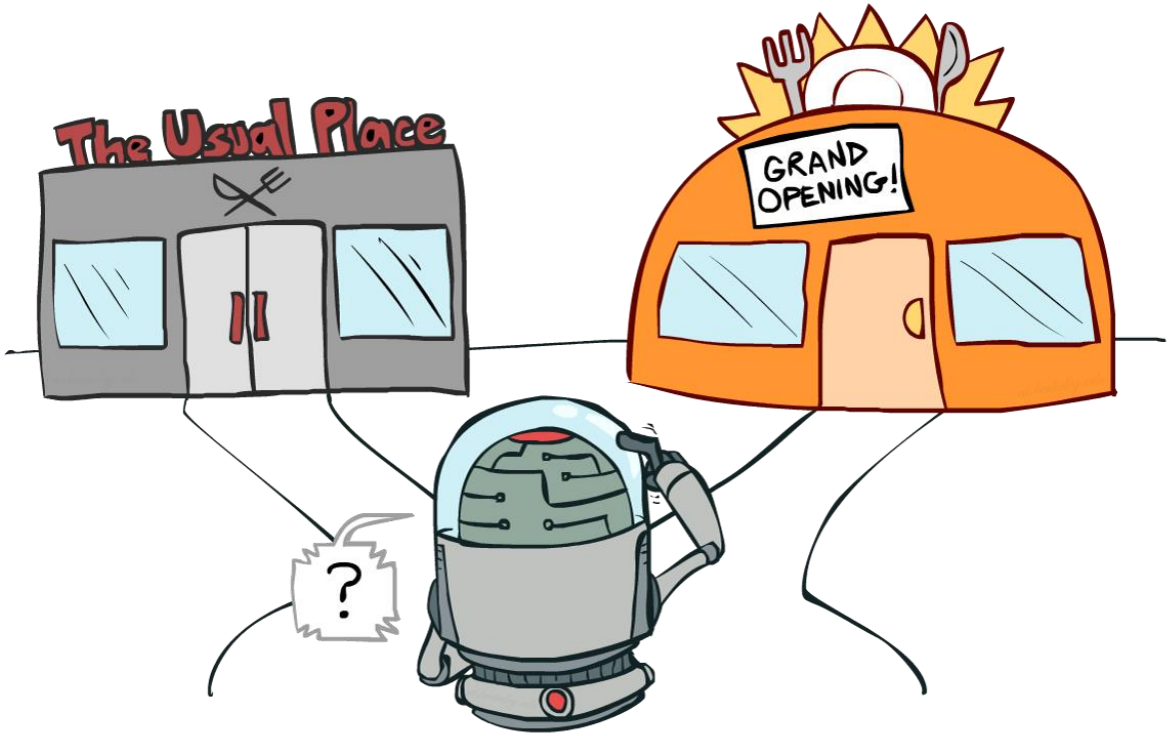
Improvement $\pi \leftarrow \epsilon\text{-Greedy}(Q)$

*** 적당히 작은 α에 대해서 수렴함이 수학적으로 증명됨

(Ref paper : "A stochastic approximation method" by Herbert Robbins and Sutton Monro)

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Exploration vs Exploitation



Exploration-Exploitation Dilemma.
One potential solution to this dilemma is implemented by gradually modifying the value of ϵ when constructing ϵ-greedy policies.

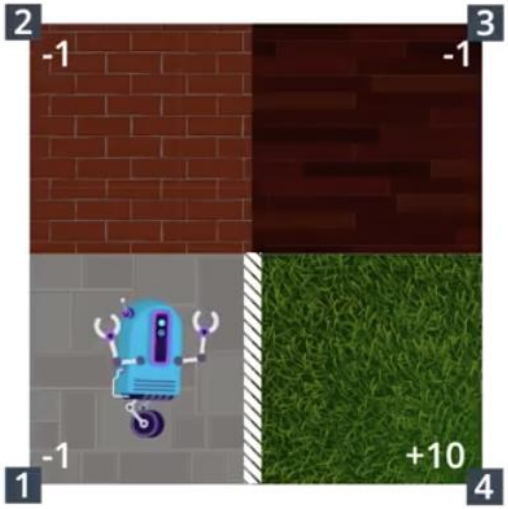
Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TRL – TD Methods

- Real life is far from an episodic task and it requires its agents, it requires us to constantly make decisions all day everyday. We get no break with our interaction with the world. Remember that Monte Carlo learning needed those breaks, it needed the episode to end so that the return could be calculated, and then used as an estimate for the action values. So, we'll need to come up with something else if we want to deal with more realistic learning in a real world setting.
- So, the main idea is this, if an agent is playing chess, instead of waiting until the end of an episode to see if it's won the game or not, it will at every move be able to estimate the probability that it's winning the game, or a self-driving car at every turn will be able to estimate if it's likely to crash, and if necessary, amend its strategy to avoid disaster. To emphasize, the Monte Carlo approach would have this car crash every time it wants to learn anything, and this is too expensive and also quite dangerous.
- TD learning will solve these problems. Instead of waiting to update values when the interaction ends, it will amend its predictions at every step, and you'll be able to use it to solve both continuous and episodic tasks.

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Sarsa



Q-Table

1	+7	+6	+5	+6
2	+8	+7	+9	+8
3	+10	+8	+9	+9

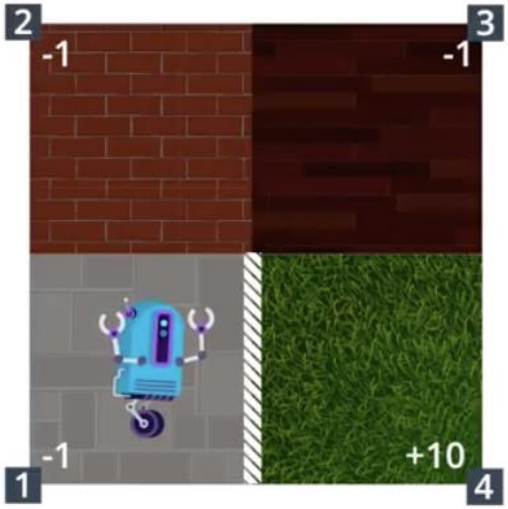
Episode s_0 a_0 r_1 s_1 a_1

Current estimate +6

Alternative estimate +7 = -1 + +8

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Sarsa



Q-Table

	↑	↓	←	→
1	+7	+6	+5	+6.2
2	+8	+7	+9	+8
3	+10	+8	+9	+9

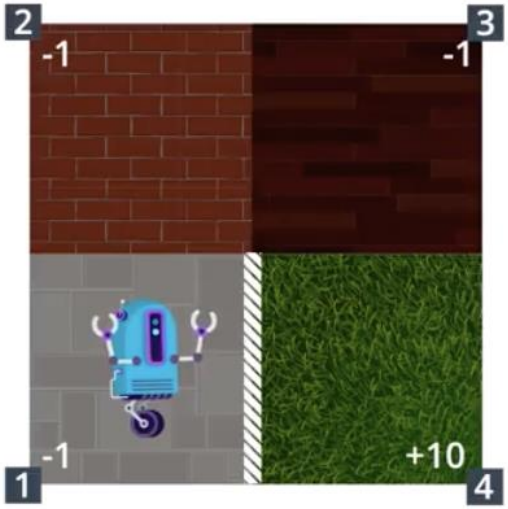
Episode 1 s_0 a_0 -1 r_1 2 s_1 a_1

Current estimate +6

Alternative estimate +7 = -1 + +8

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Sarsa



Q-Table

1	+7	+6	+5	+6.2
2	+8	+7	+9	+8.2
3	+10	+8	+9	+9

Episode s_0 a_0 r_1 s_1 a_1 r_2 s_2 a_2

Current estimate +8

Alternative estimate +9 = -1 + +10

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Sarsa

(From Monte Carlo Control)

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

alternative estimate — current estimate

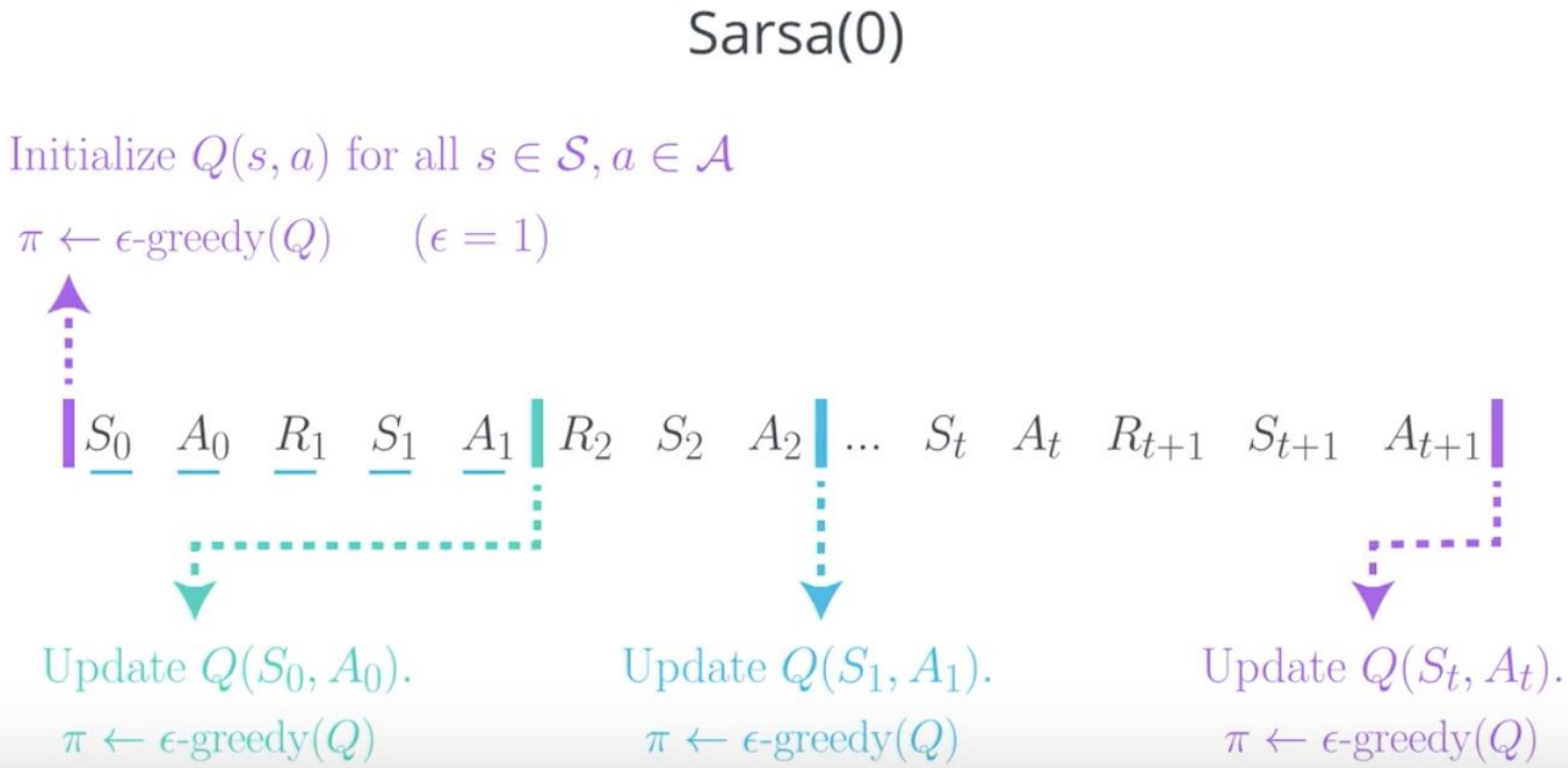
(From Temporal-Difference Control)

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

alternative estimate — current estimate

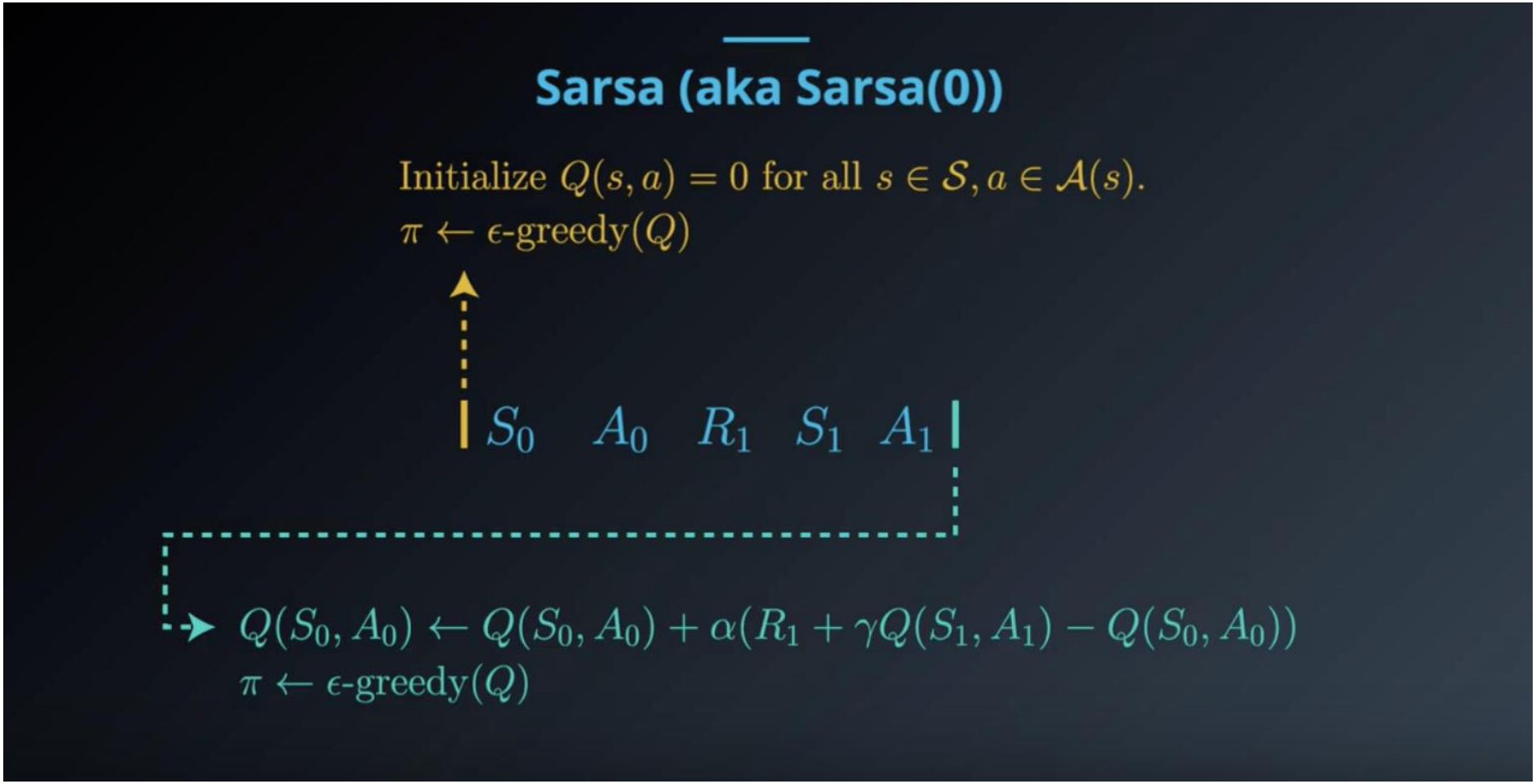
Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Sarsa



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Sarsa



Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Sarsa

Algorithm 13: Sarsa

Input: policy π, positive integer *num_episodes*, small positive fraction α, GLIE $\{\epsilon_i\}$
Output: value function Q ($\approx q_\pi$ if *num_episodes* is large enough)
Initialize Q arbitrarily (e.g., $Q(s,a) = 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, and $Q(\textit{terminal-state}, \cdot) = 0$)
for $i \leftarrow 1$ ***to*** *num_episodes* **do**
  $\epsilon \leftarrow \epsilon_i$
  Observe S_0
  Choose action A_0 using policy derived from Q (e.g., ϵ-greedy)
  $t \leftarrow 0$
  repeat
    Take action A_t and observe R_{t+1}, S_{t+1}
    Choose action A_{t+1} using policy derived from Q (e.g., ϵ-greedy)
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$
    $t \leftarrow t + 1$
  until S_t *is terminal*;
end
return Q

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Q-Learning

Sarsa (aka Sarsa(0))

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

Sarsamax (aka Q-Learning)

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t))$$

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Q-Learning

Algorithm 14: Sarsamax (Q-Learning)

Input: policy π, positive integer *num_episodes*, small positive fraction α, GLIE $\{\epsilon_i\}$
Output: value function Q ($\approx q_\pi$ if *num_episodes* is large enough)
Initialize Q arbitrarily (e.g., $Q(s,a) = 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, and $Q(\textit{terminal-state}, \cdot) = 0$)
for $i \leftarrow 1$ ***to*** *num_episodes* **do**
  $\epsilon \leftarrow \epsilon_i$
  Observe S_0
  $t \leftarrow 0$
  repeat
    Choose action A_t using policy derived from Q (e.g., ϵ-greedy)
    Take action A_t and observe R_{t+1}, S_{t+1}
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$
    $t \leftarrow t + 1$
  until S_t *is terminal*;
end
return Q

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Expected Sarsa

Sarsa

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

Sarsamax

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t))$$

Expected Sarsa

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \sum_{a \in \mathcal{A}} \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t))$$

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Expected Sarsa

Algorithm 15: Expected Sarsa

Input: policy π, positive integer *num_episodes*, small positive fraction α, GLIE $\{\epsilon_i\}$
Output: value function Q ($\approx q_\pi$ if *num_episodes* is large enough)
Initialize Q arbitrarily (e.g., $Q(s,a) = 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, and $Q(\textit{terminal-state}, \cdot) = 0$)
for $i \leftarrow 1$ ***to*** *num_episodes* **do**
  $\epsilon \leftarrow \epsilon_i$
  Observe S_0
  $t \leftarrow 0$
  repeat
    Choose action A_t using policy derived from Q (e.g., ϵ-greedy)
    Take action A_t and observe R_{t+1}, S_{t+1}
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t))$
    $t \leftarrow t + 1$
  until S_t *is terminal*;
end
return Q

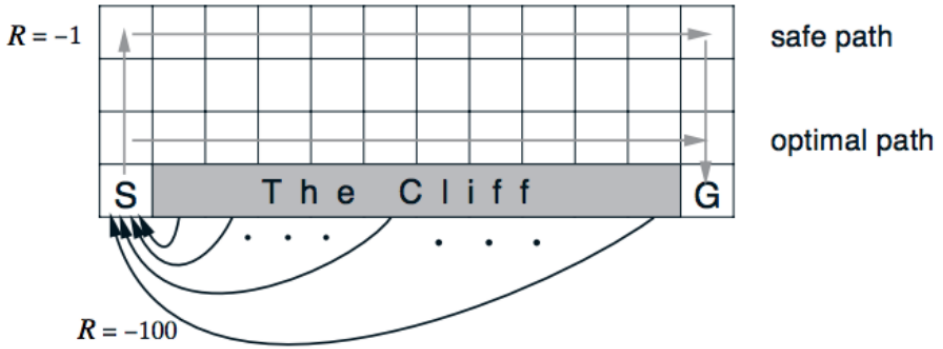
Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD Control: Differences

- The differences between these algorithms are summarized below:
- Sarsa and Expected Sarsa are both on-policy TD control algorithms. In this case, the same (ϵ-greedy) policy that is evaluated and improved is also used to select actions.
- Sarsamax is an off-policy method, where the (greedy) policy that is evaluated and improved is different from the (ϵ-greedy) policy that is used to select actions.
- On-policy TD control methods (like Expected Sarsa and Sarsa) have better online performance than off-policy TD control methods (like Sarsamax).
- Expected Sarsa generally achieves better performance than Sarsa.

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

TD : Cliff-walking task result



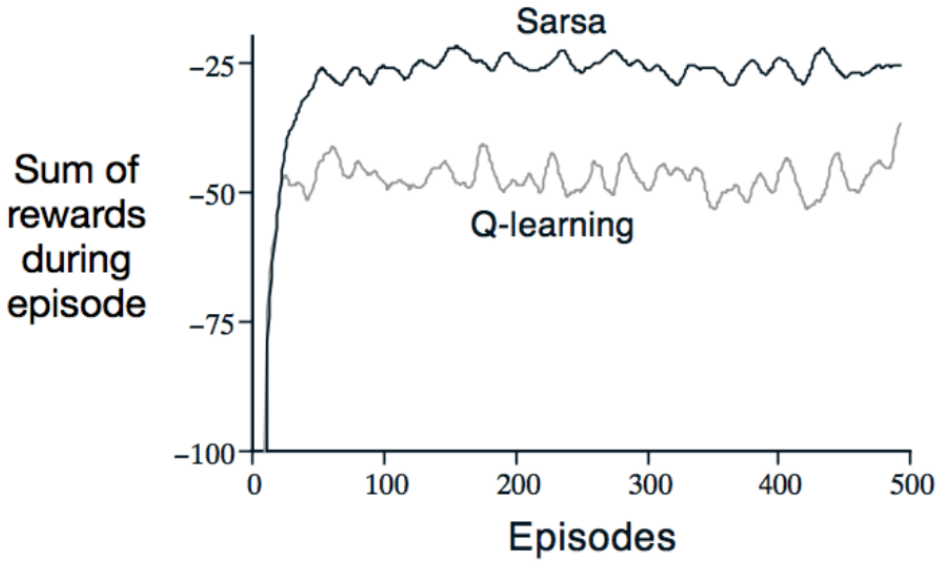


Figure 6.4: The cliff-walking task. The results are from a single run, but smoothed by averaging the reward sums from 10 successive episodes. ■

Main Reference – Udacity DRL nanodegree contents (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
[Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto - Second Edition] http://incompleteideas.net/book/the-book.html

Thank you