

Reinforcement Learning (AI644)

Prepared by Dr. Ashraf Elsayed

Lecture 8





مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY

Reinforcement Learning

Lecture 8

Prepared by
Dr. Ashraf Elsayed

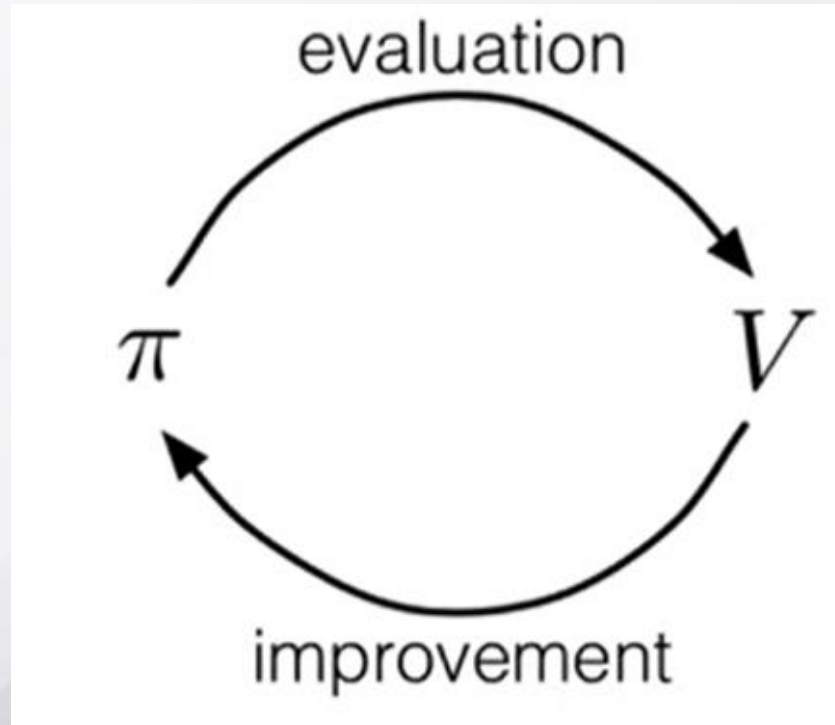
www.midocean.ae

Temporal Difference Learning Methods for Control

The objectives of this lecture is to:

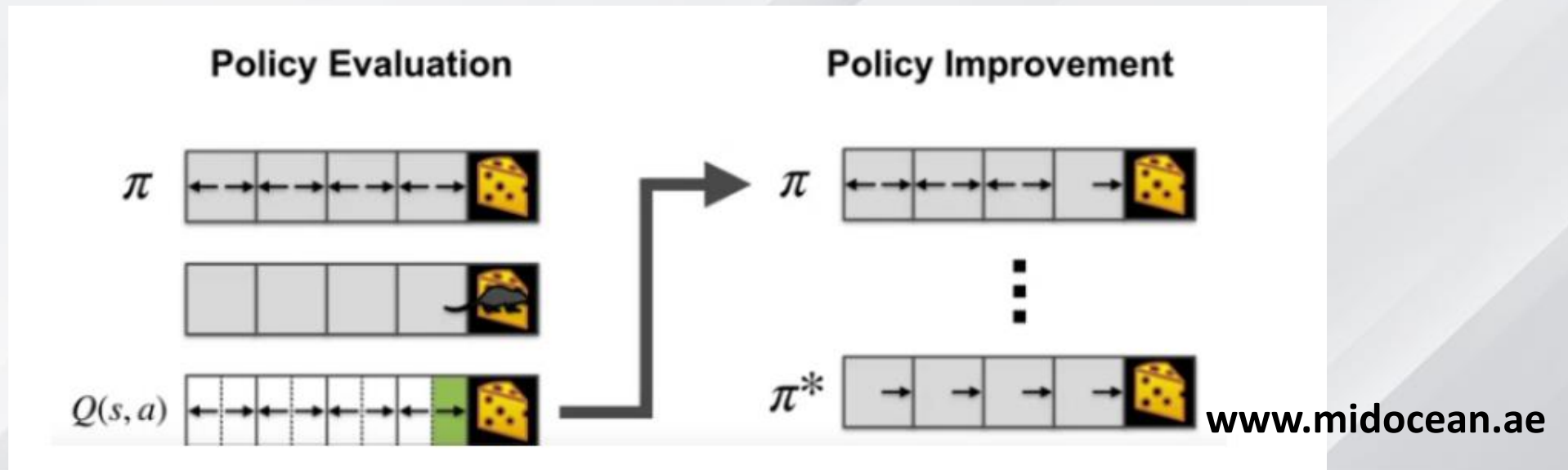
- Explain how generalized policy iteration can be used with TD to find improved policies
- Describe the Sarsa control algorithm
- Understand how the Sarsa control algorithm operates in an example MDP
- Describe the Q-learning algorithm
- Explain the relationship between Q-learning and the Bellman optimality equations.

Recall: Generalized Policy Iteration (GPI)



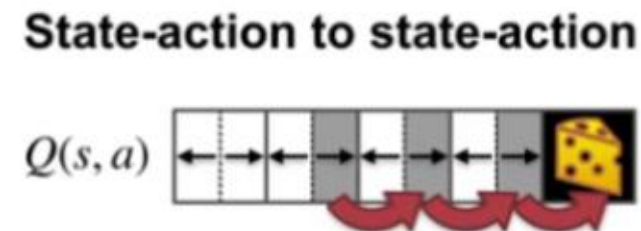
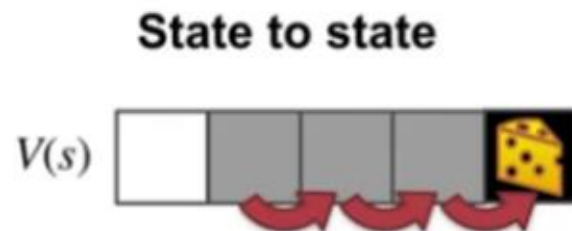
Recall: GPI with MC

- To better remember GPI with Monte Carlo, imagine a mouse in a four-state corridor with cheese at the end.
- The mouse starts out knowing nothing and follows a random policy. Eventually, the mouse will stumble into the cheese just by moving randomly.
- At that point, the mouse updates its action values. Then it improves its policy by greedy with respect to its action values.



Sarsa: GPI with TD

- To use TD within (Generalized policy iteration) GPI, we need to **learn an action value function**.
- So we'll need to look at slightly different version of TD than we've seen in the past.
- Instead of looking at transitions **from state to state** and learn the value of each state, let's look at transitions from **state action pair to state action pair** and learn the value of each pair → **SARSA Prediction**

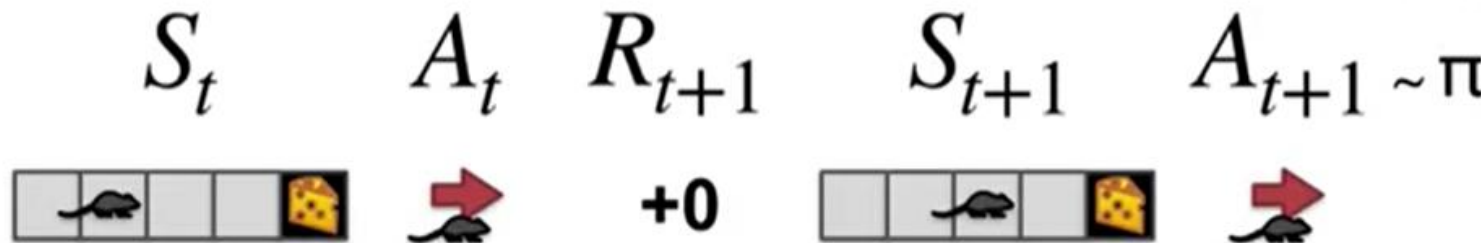


SARSA Algorithm

The Sarsa algorithm

$S_t \ A_t \ R_{t+1} \ S_{t+1} \ A_{t+1}$

$\sim \epsilon$ -greedy



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

Recall: $V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$

On-Policy vs Off-Policy

- On-Policy: improve and evaluate the policy being used to select actions.
 - In on-policy learning, the $Q(s,a)$ function is learned from actions that we took using our current policy $\pi(a|s)$
 - It updates the policy being used to take actions as the agent interacts with the environment.
 - The agent learns by following the current policy and then updates the policy based on the rewards received from those actions.
- Off-Policy: improve and evaluate a *different* policy from the one used to select actions
 - In off-policy learning, the $Q(s,a)$ function is learned from taking different actions.
 - It updates a different policy than the one being used to take actions.
 - The main advantage of off-policy learning is that it is more likely to find optimal policy

SARSA Algorithm

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Loop for each step of episode:

 Take action A , observe R, S'

 Choose A' from S' using policy derived from Q (e.g., ε -greedy)

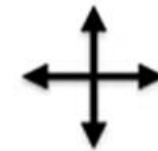
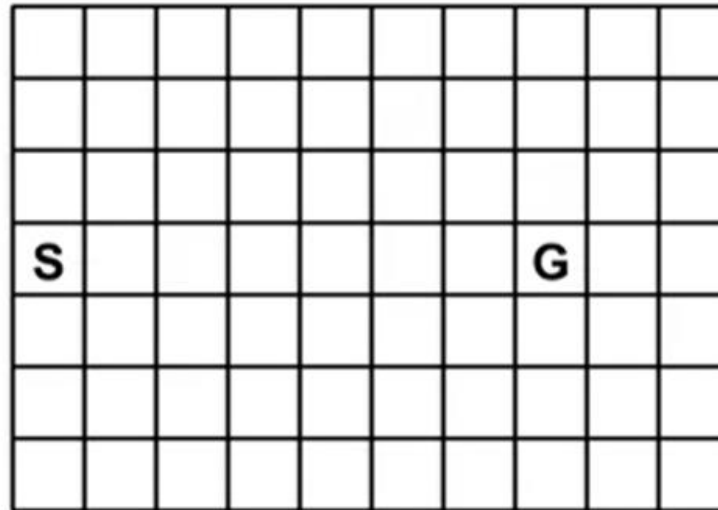
$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'; A \leftarrow A';$

 until S is terminal

Example: Windy Gridworld

The Windy Gridworld

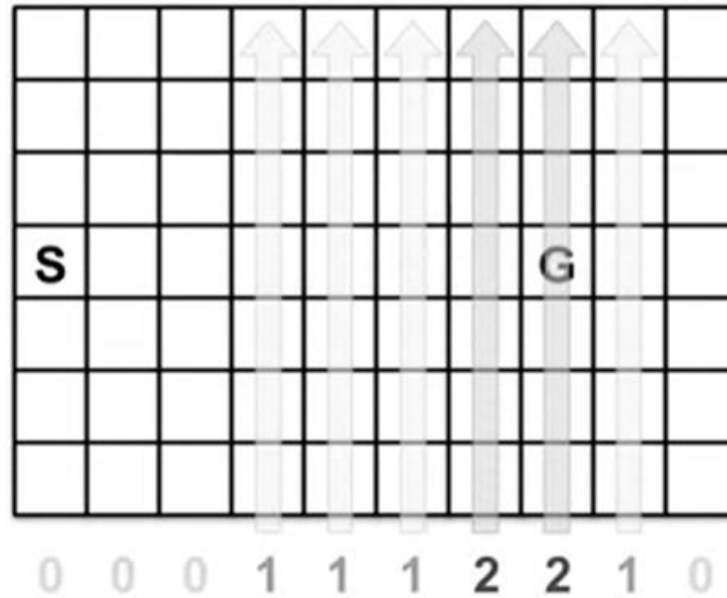


Actions

$$R_{step} = -1$$
$$\gamma = 1$$

Example: Windy Gridworld

The Windy Gridworld



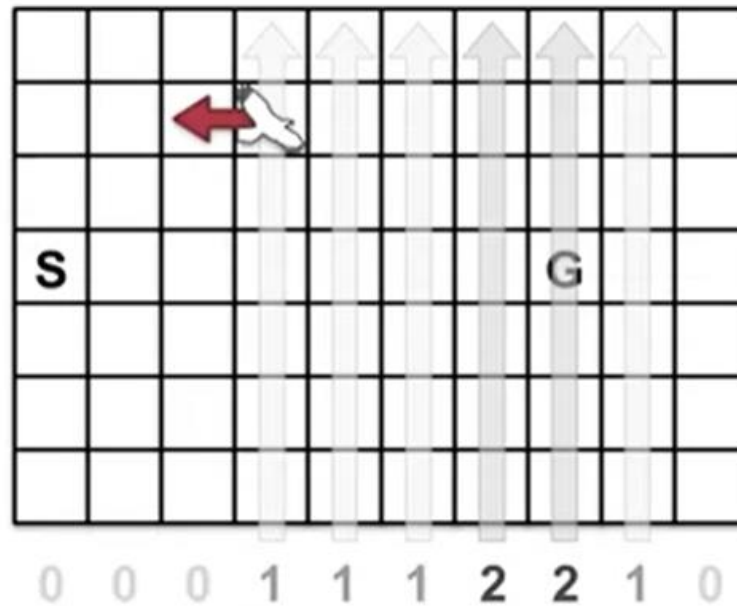
Actions

$$R_{step} = -1$$

$$\gamma = 1$$

Example: Windy Gridworld

The Windy Gridworld

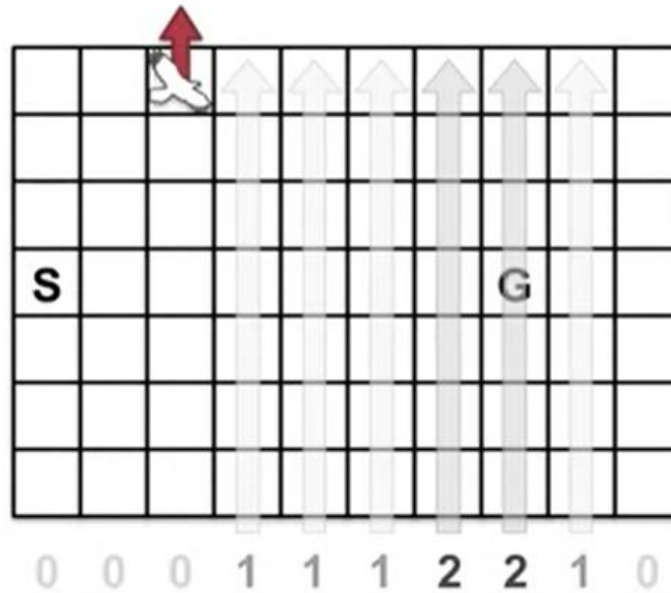


$$R_{step} = -1$$

$$\gamma = 1$$

Example: Windy Gridworld

The Windy Gridworld

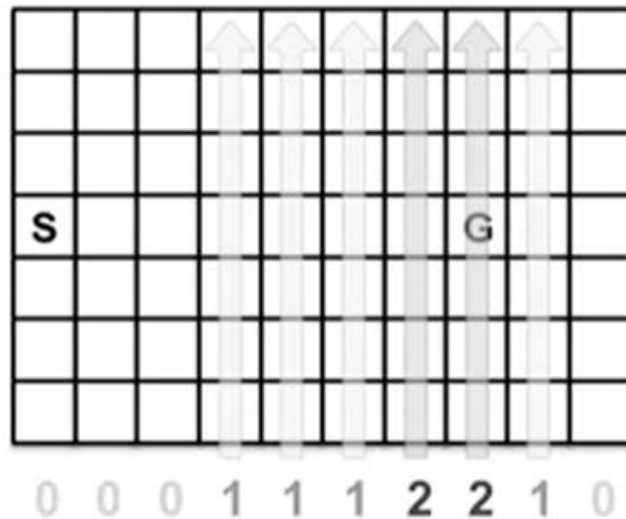


$$R_{step} = -1$$

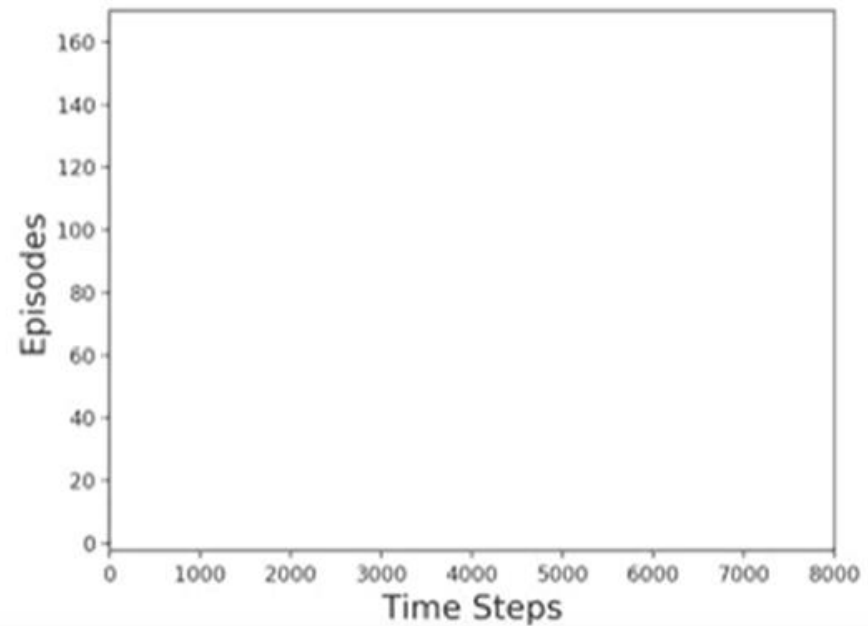
$$\gamma = 1$$

Example: Windy Gridworld

Sarsa on the Windy Gridworld $\sim \epsilon$ -greedy



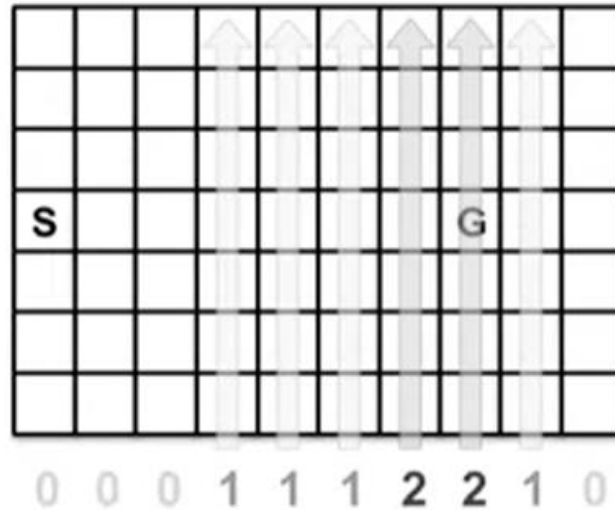
Sarsa: $\epsilon = 0.1$
 $\alpha = 0.5$



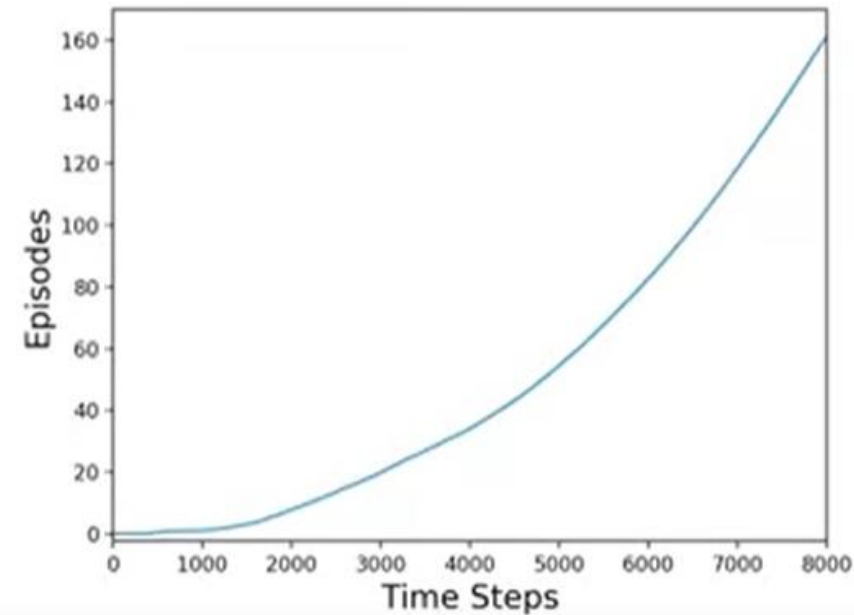
Take a random action
every 10 steps

Example: Windy Gridworld

Sarsa on the Windy Gridworld



Sarsa: $\epsilon = 0.1$
 $\alpha = 0.5$

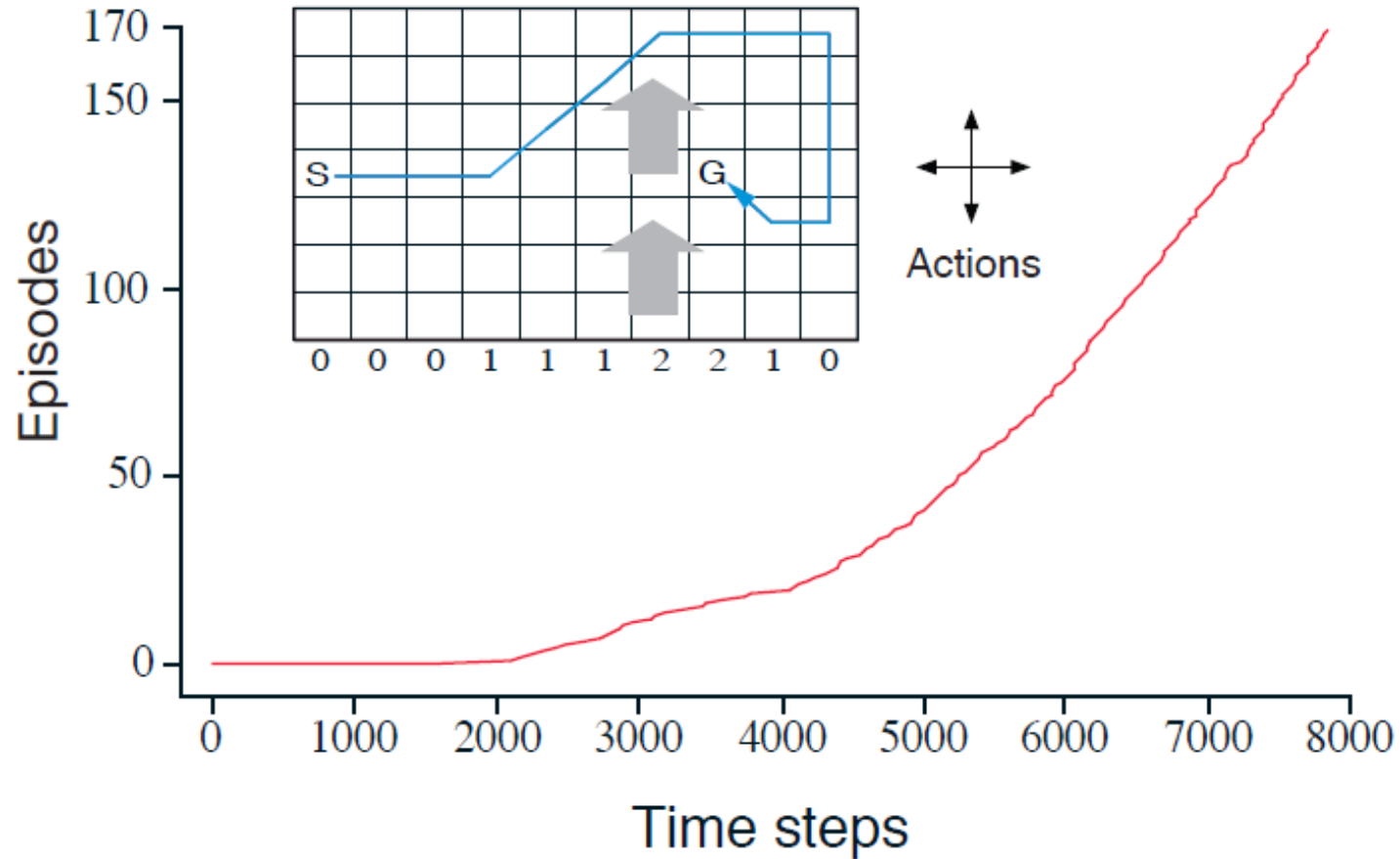


Can Monte Carlo methods be used on this task? !

No, since termination is not guaranteed for all policies.!

Example: Windy Gridworld

Sarsa on the Windy Gridworld



Unlike Monte Carlo Sarsa's Step-by-step learning methods can quickly learn during the episode that such policies are poor, and switch to something else.!!

What is Q-learning?

- Q-learning is a model-free, value-based, off-policy TD control algorithm that will find the best series of actions based on the agent's current state.
- The “Q” stands for quality. Quality represents how valuable the action is in maximizing future rewards.
- It is defined by:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

- Here the learned action-value function, Q , directly approximates q^* , the optimal action-value function, independent of the policy being followed.
- This dramatically simplifies the analysis of the algorithm and enabled early convergence proofs.

Q-learning Algorithm

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

 → $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

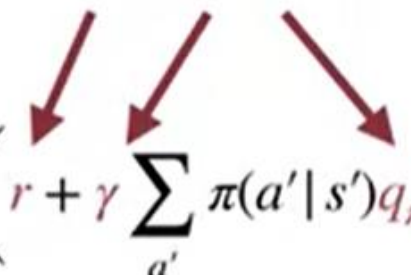
$S \leftarrow S'$

 until S is terminal

SARSA and Bellman Equation

Revisiting Bellman equations

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

$$q_{\pi}(s, a) = \sum_{s', r} p(s', r | s, a) \left(r + \gamma \sum_{a'} \pi(a' | s') q_{\pi}(s', a') \right)$$


Q-learning and Bellman Equation

Revisiting Bellman equations

Q-learning: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha (R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$

$$q_*(s, a) = \sum_{s', r} p(s', r | s, a) (r + \gamma \max_{a'} q_\pi(s', a'))$$


Q-learning and Bellman Equation

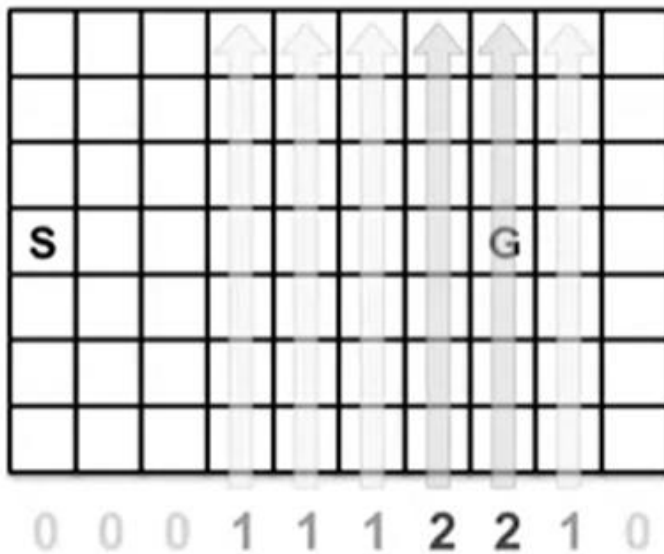
Revisiting Bellman equations

Q-learning: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha (R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$

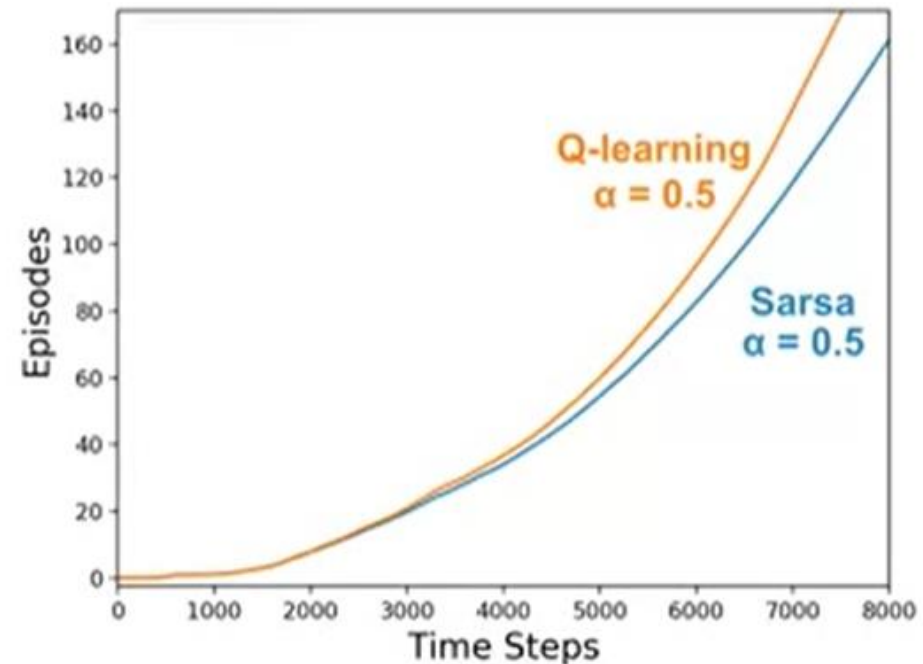
$$q_*(s, a) = \sum_{s', r} p(s', r | s, a) (r + \gamma \max_{a'} q_\pi(s', a'))$$


Example: Windy Gridworld (Q-learning vs SARSA)

The Windy Gridworld

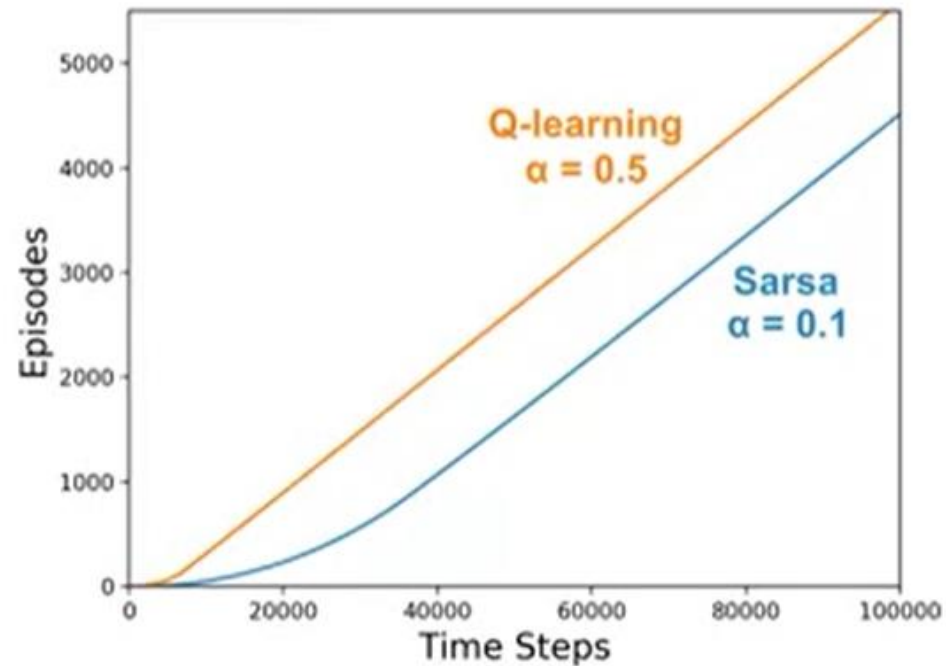
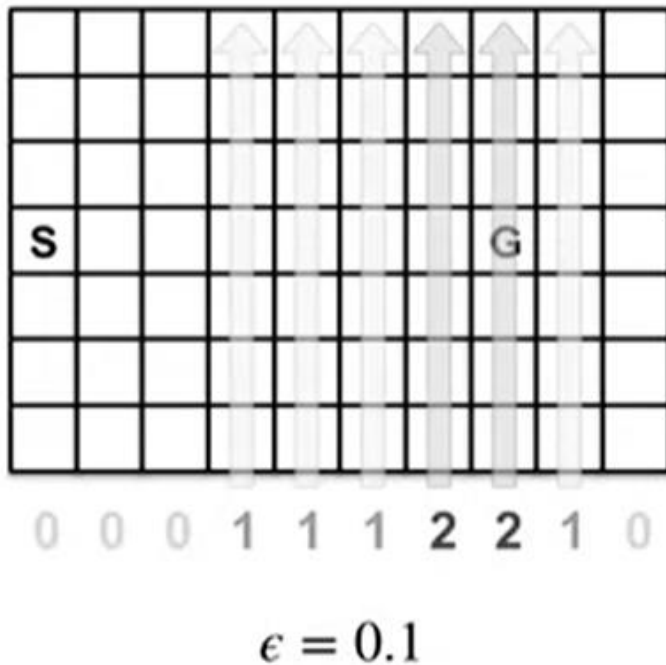


$\epsilon = 0.1$



Example: Windy Gridworld (Q-learning vs SARSA)

The Windy Gridworld



On-Policy vs Off-Policy

Comparison with Sarsa

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

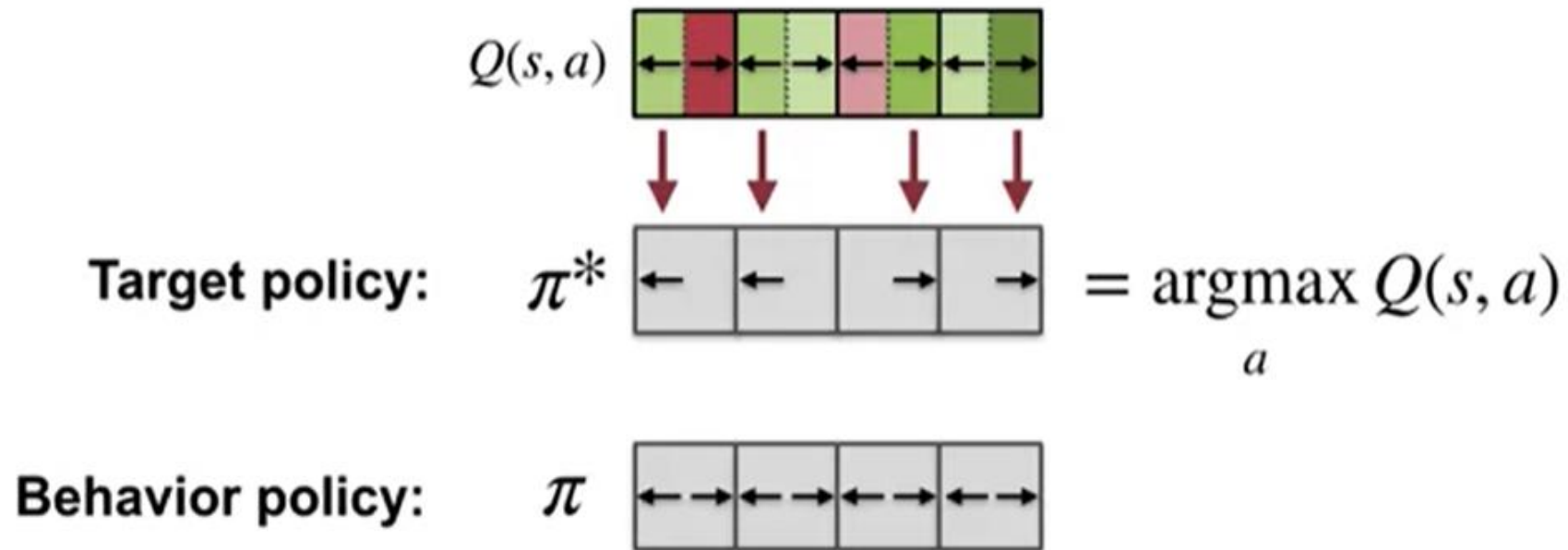
On-Policy

Q-learning: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$

$\sim \pi_* \neq \pi$
Off-Policy

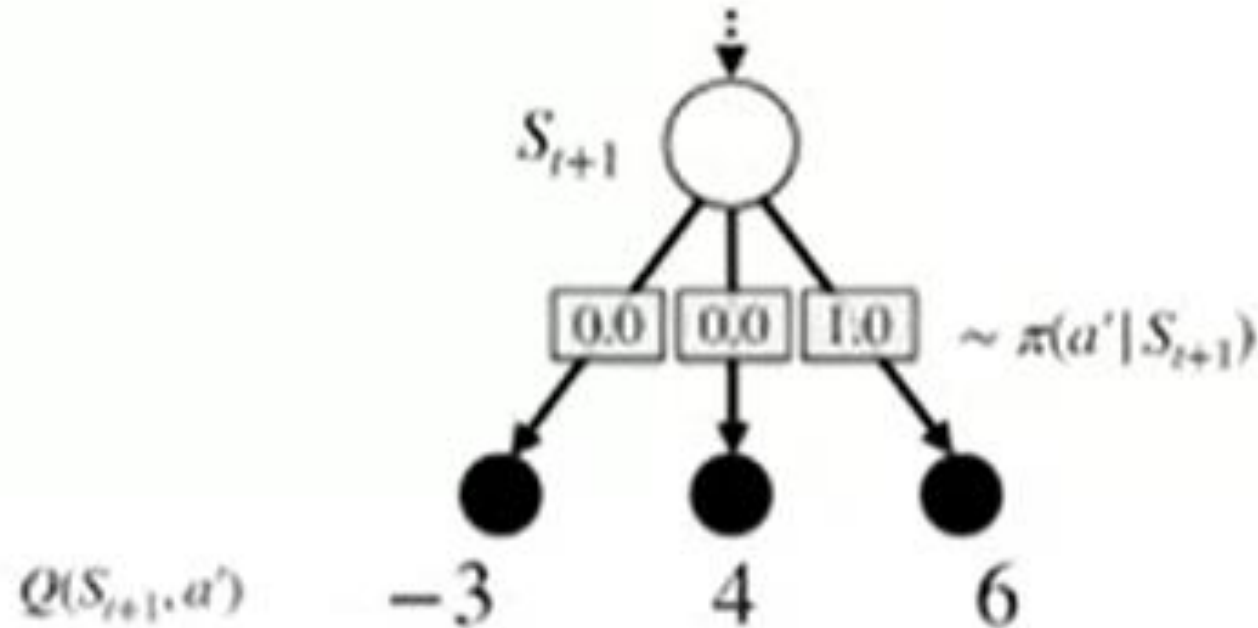
Behavior vs Target Policies

Behavior and target policies



Can be ϵ -greedy

Q-learning Off-Policy Learning Technique

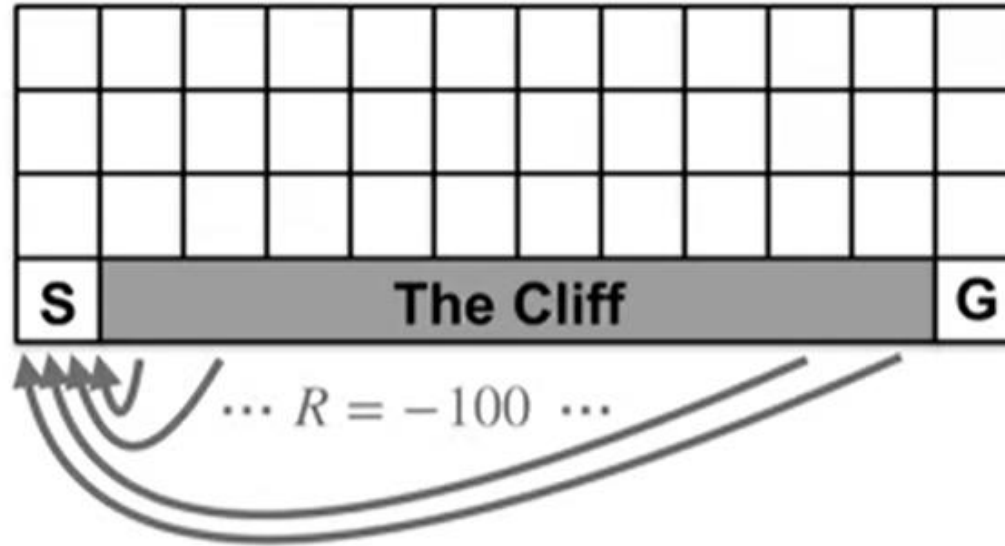


$$\sum_{a'} \pi(a' | S_{t+1}) Q(S_{t+1}, a') = E_{\pi}[G_{t+1} | S_{t+1}] = \max_{a'} Q(S_{t+1}, a') = 6$$

Since the agents target policies greedy, with respect to its action values, all non-maximum actions have probability 0.

Example: Cliff Walking

The cliff walking environment

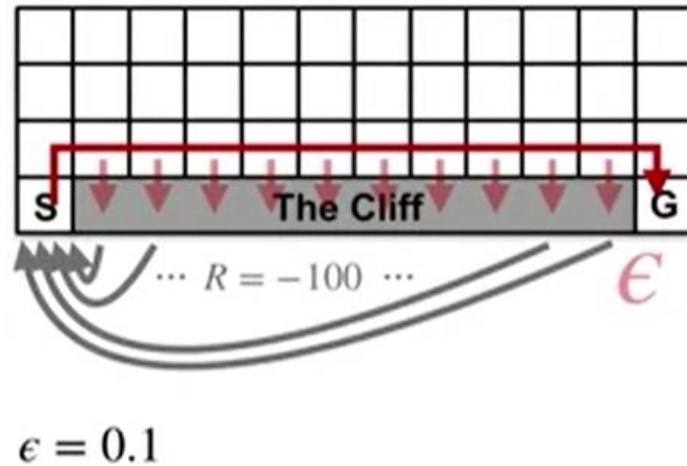


$$\gamma = 1$$

$$R_{step} = -1$$

Example: Cliff Walking

The cliff walking environment



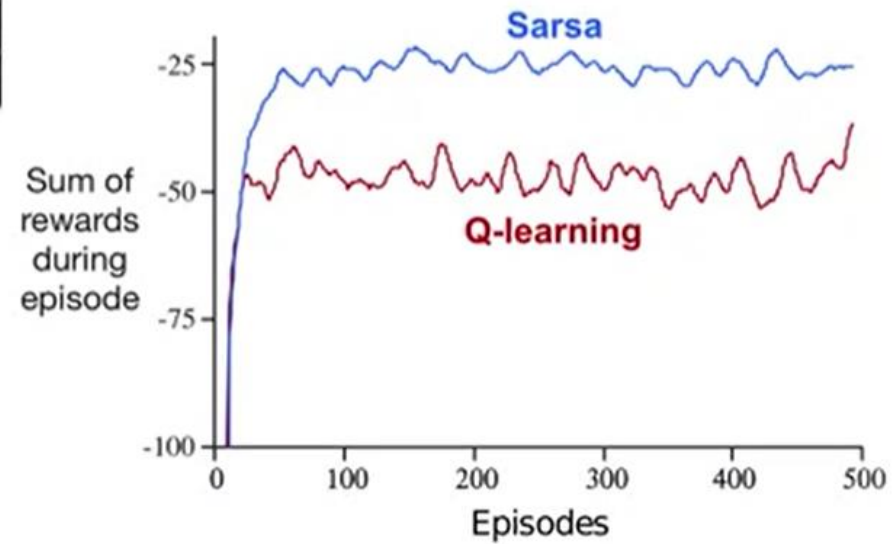
Q-Learning is less desirable in specific situations

Example: Cliff Walking

The cliff walking environment



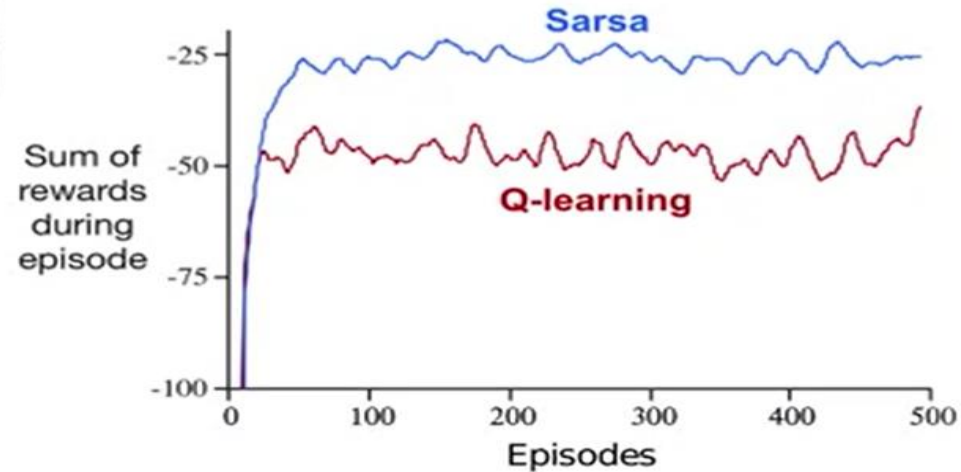
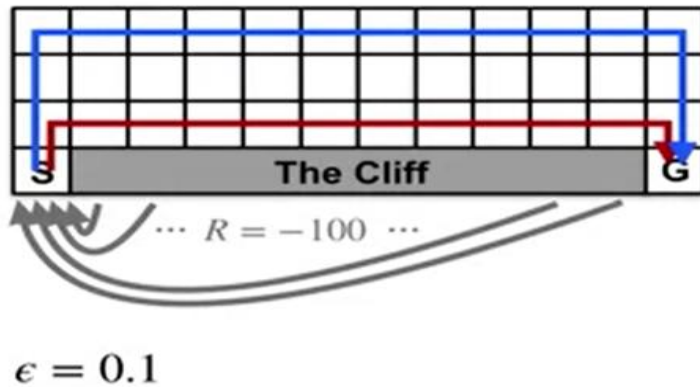
$\epsilon = 0.1$



Q-Learning is less desirable in specific situations

Example: Cliff Walking

The cliff walking environment



Q-learning learns quickly **values for the optimal policy**, that which travels right along the edge of the cliff. Unfortunately, this results in its occasionally falling off the cliff because of the ϵ -greedy action selection.

Sarsa takes the **action selection into account** and learns the longer but safer path through the upper part of the grid.

Q-learning Solved Example 1

d	e	f
a	b	c

$$\alpha = 0.9 \quad \gamma = 1$$

Initialize

$$Q(s, a) = 0$$

d	e	f
a	b	c

$$c \rightarrow f$$

$$Q(c, \uparrow) \leftarrow 0 + 0.9[100 + 0 - 0] = 90$$

d	e	f
a	b	c

$$e \rightarrow f$$

$$Q(e, \rightarrow) \leftarrow 0 + 0.9[100 + 0 - 0] = 90$$

Q-learning Solved Example 1

d	e	f
a	b	c



$b \rightarrow c$

$$Q(b, \rightarrow) \leftarrow 0 + 0.9[0 + 90 - 0] = 81$$

d	e	f
a	b	c



$b \rightarrow e$

$$Q(b, \uparrow) \leftarrow 0 + 0.9[0 + 90 - 0] = 81$$

d	e	f
a	b	c



$a \rightarrow b$

$$Q(a, \rightarrow) \leftarrow 0 + 0.9[0 + 81 - 0] \approx 73$$

Q-learning Solved Example 1

d	e	f
a	b	c

$b \rightarrow a$

$$Q(b, \leftarrow) \leftarrow 0 + 0.9[0 + 73 - 0] \approx 66$$

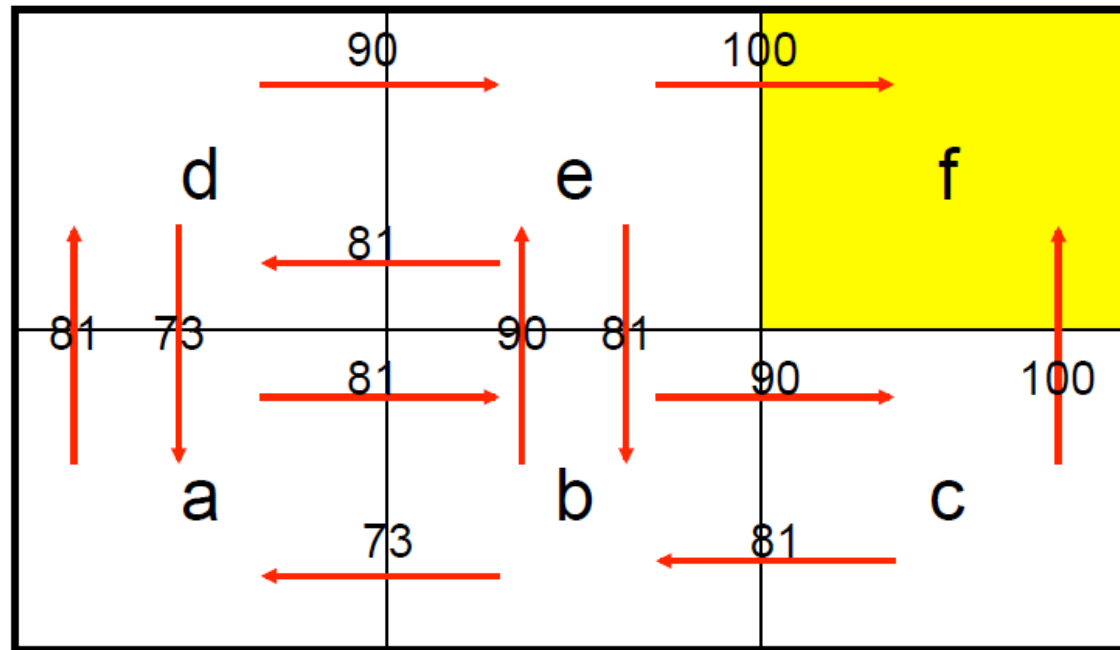
d	e	f
a	b	c

81

66

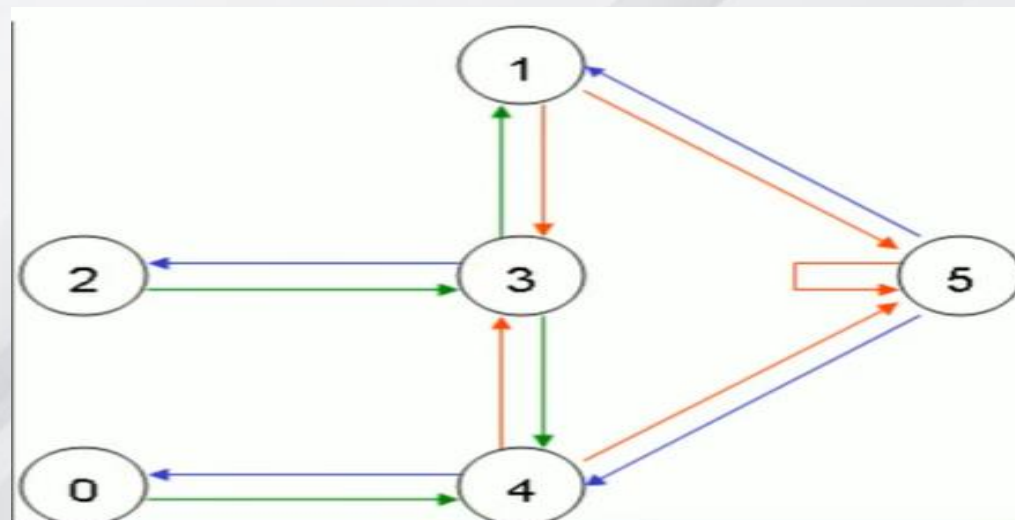
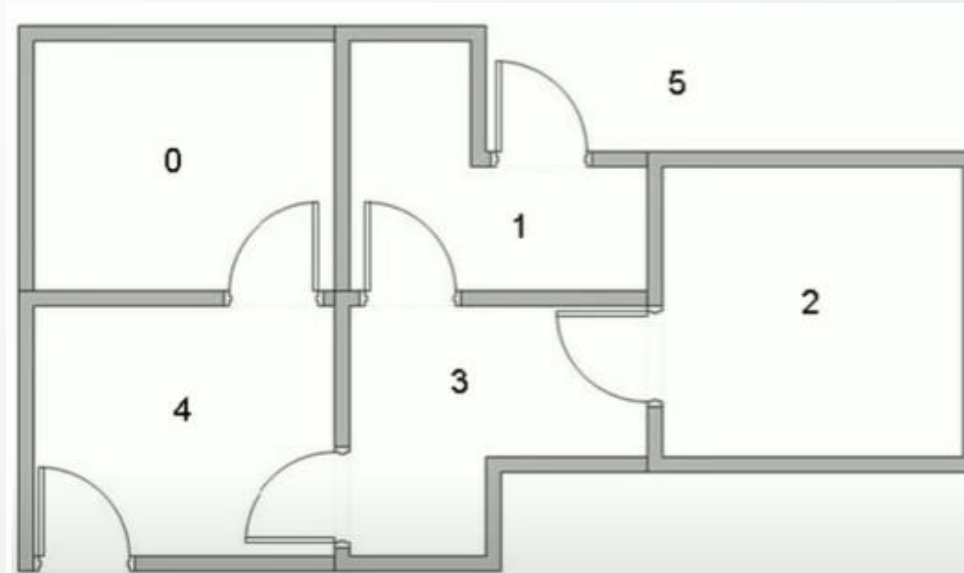
81

Q-learning Solved Example 1

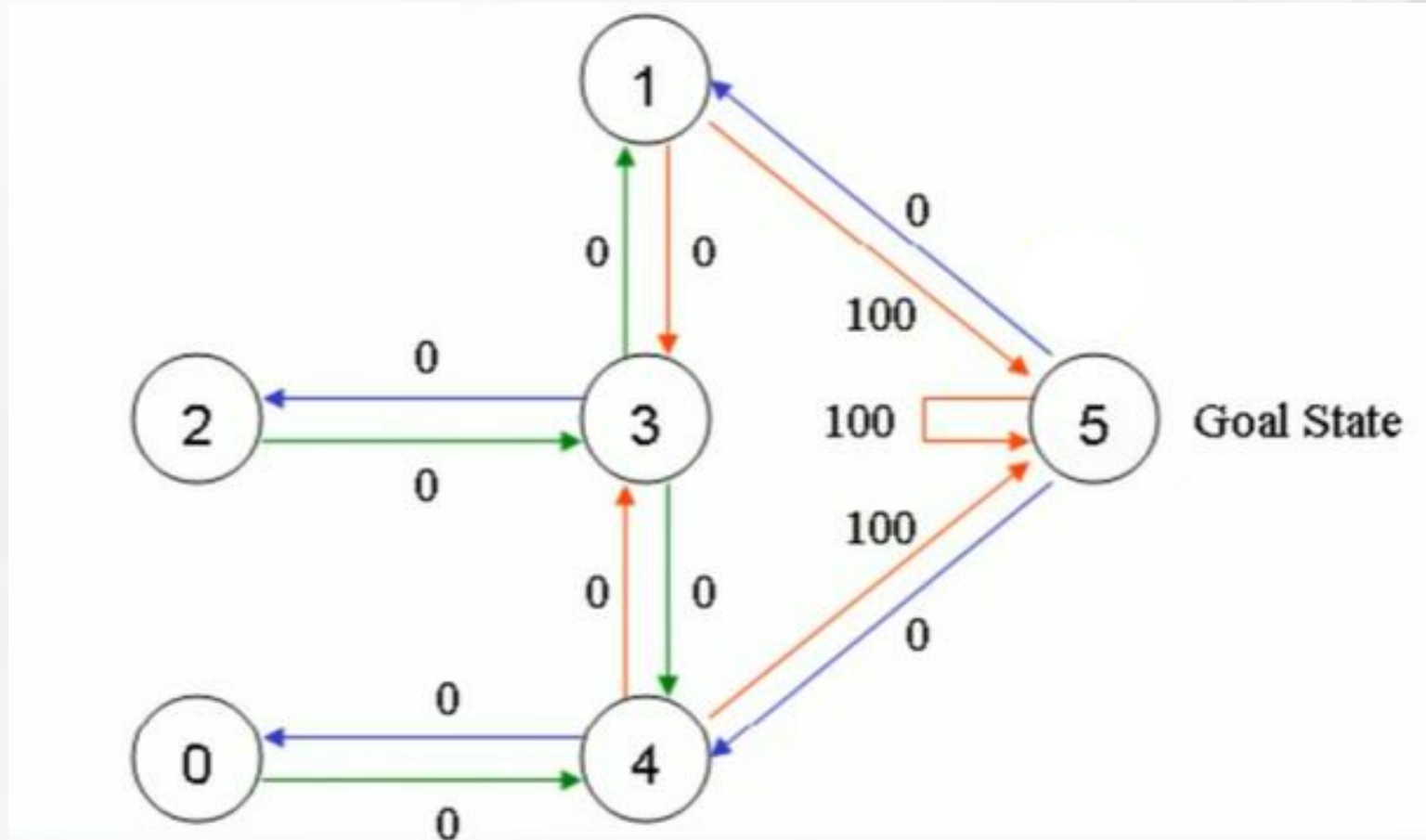


$$\gamma = 0.9$$

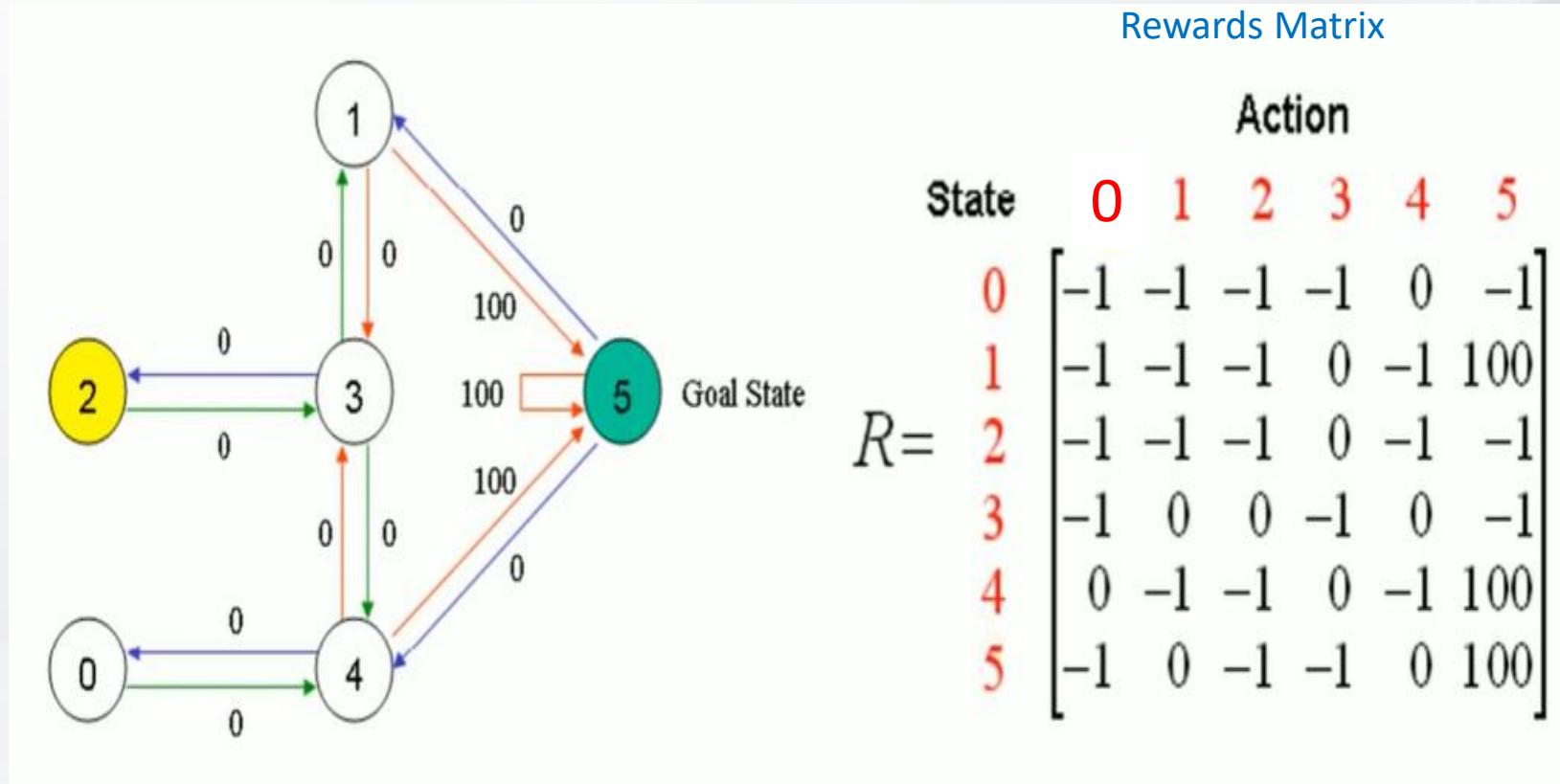
Q-learning Solved Example 2



Q-learning Solved Example 2



Q-learning Solved Example 2



The -1 in the table represent null values (where there isn't a link between nodes)

Q-learning Solved Example 2

- $\alpha = 0.8$ and $\gamma=1$
- Initial state at room 0
- Initial Q-Matrix is 0

Q-Matrix (Action-Value Function)

$$R = \begin{array}{c|cccccc} & \text{Action} \\ \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$

$$Q = \begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Q-learning Solved Example 2

- Look at the second row (Initial state 1) in matrix R
- There are two possible actions for the current state 1: go to state 3 or go to state 5
- By random selection, we select to 5 as our action
- What would happen if our agent were in state 5 (next state)? It can go to state 1, 4 or 5 (the new action)

Q-Matrix (Action-Value Function)

$$R = \begin{array}{c|cccccc} & \text{Action} \\ \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$

$$Q = \begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Q-learning Solved Example 2

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

$$\begin{aligned} Q(1, 5) &= 0 + 1 [100 + 0.8 (\max (Q(5, \text{goto } 1), Q(5, \text{goto } 4), Q(5, \text{goto } 5)) - 0)] \\ &= 0 + 1 [100 + 0.8 \times 0 - 0] = 100 \end{aligned}$$

End of the 1st episode

Q-Matrix (Action-Value Function)

$R =$

State \ Action	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

$Q =$

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Q-learning Solved Example 2

- Let the initial state be state 3 for the next episode
- There are three possible actions for the current state: go to state 1 or 2 or 4
- Assume using current policy we will go to state 1
- In this case we can either goto state 3 or state 5 from state 1

Q-Matrix (Action-Value Function)

$$R = \begin{matrix} & \text{Action} \\ \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{matrix}$$

$$Q = \begin{matrix} & 0 & 1 & 2 & 3 & 4 & 5 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

Q-learning Solved Example 2

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

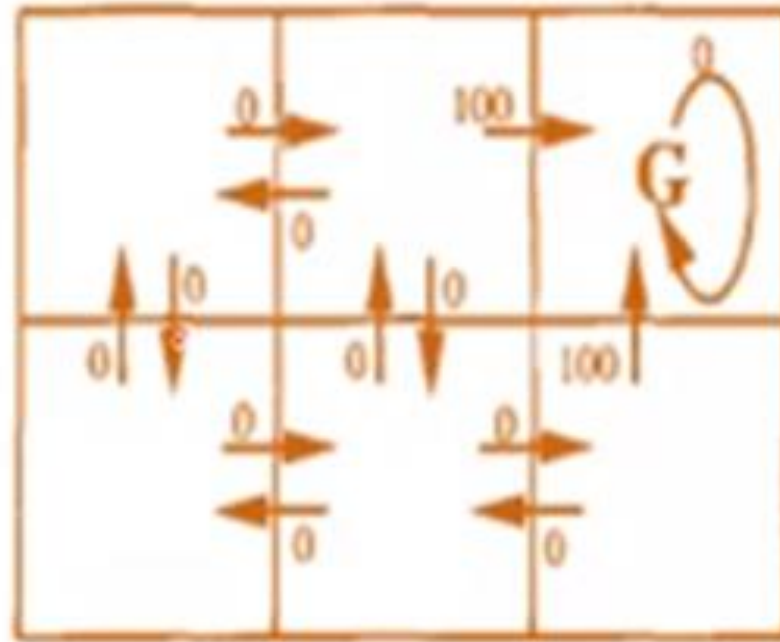
$$\begin{aligned} Q(3, 1) &= 0 + 1 [0 + 0.8 (\max (Q(1, \text{goto } 3), Q(1, \text{goto } 5)) - 0)] \\ &= 0 + 1 [0 + 0.8(\max(0, 100) - 0)] = 80 \end{aligned}$$

Q-Matrix (Action-Value Function)

$$R = \begin{array}{c|cccccc} & \text{Action} \\ \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$

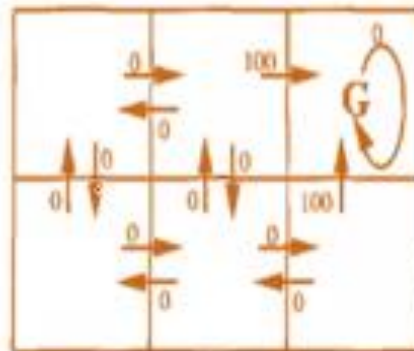
$$Q = \begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 100 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 80 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Q-learning Solved Example 3



$r(s, a)$ (immediate reward) values

Example



$r(s, a)$ (immediate reward) values

Reward Matrix

	1	2	3	4	5	6
1	-1	0	-1	0	-1	-1
2	0	-1	100	-1	0	-1
3	-1	-1	0	-1	-1	-1
4	0	-1	-1	-1	0	-1
5	-1	0	-1	0	-1	0
6	-1	-1	100	-1	0	-1

Q-Matrix

	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Q-learning Solved Example 3

$$\begin{aligned} Q(2, 3) &= 100 + 0.9 * \max[(Q(3,3))] \\ &= 100 + 0.9 * \text{MAX}[0] \\ &= 100 + 0 \\ &= 100 \end{aligned}$$

$$\begin{aligned} Q(6, 3) &= 100 + 0.9 * \max[(Q(3,3))] \\ &= 100 + 0.9 * \text{MAX}[0] \\ &= 100 + 0 \\ &= 100 \end{aligned}$$

$$\begin{aligned} Q(1, 2) &= 0 + 0.9 * \max[(Q(2,3), Q(2,5))] \\ &= 0 + 0.9 * \text{MAX}[100, 0] \\ &= 0.9 * 100 \\ &= 90 \end{aligned}$$

Reward Matrix

	1	2	3	4	5	6
1	-1	0	-1	0	-1	-1
2	0	-1	100	-1	0	-1
3	-1	-1	0	-1	-1	-1
4	0	-1	-1	-1	0	-1
5	-1	0	-1	0	-1	0
6	-1	-1	100	-1	0	-1

Q-Matrix

	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

$$\begin{aligned} Q(1,4) &= R(1,4) + 0.9 * \text{MAX} [Q(4,1), Q(4,5)] \\ &= 0 + 0.9 * \text{MAX}[81, 0] \\ &= 0.72 \end{aligned}$$

$$\begin{aligned} Q(5,2) &= R(5,2) + 0.9 * \text{MAX}[Q(2,1), Q(2,3), Q(2,5)] \\ &= 0 + 0.9 * \text{MAX} [81, 100, 0] \\ &= 90 \end{aligned}$$

$$\begin{aligned} Q(5,4) &= R(5,4) + 0.9 * \text{MAX}[Q(4,1), Q(4,5)] \\ &= 0 + 0.9 * \text{MAX} [81, 0] \\ &= 72 \end{aligned}$$

$$\begin{aligned} Q(2,5) &= R(2,5) + 0.9 * \text{MAX}[Q(5,2), Q(5,4), Q(5,6)] \\ &= 0 + 0.9 * \text{MAX} [90, 72, 0] \\ &= 81 \end{aligned}$$

$$\begin{aligned} Q(4,5) &= R(4,5) + 0.9 * \text{MAX}[Q(5,2), Q(5,4), Q(5,6)] \\ &= 0 + 0.9 * \text{MAX} [90, 72, 0] \\ &= 81 \end{aligned}$$

$$\begin{aligned} Q(6,5) &= R(4,5) + 0.9 * \text{MAX}[Q(5,2), Q(5,4), Q(5,6)] \\ &= 0 + 0.9 * \text{MAX} [90, 72, 0] \\ &= 81 \end{aligned}$$

$$\begin{aligned} Q(5,6) &= R(5,6) + 0.9 * \text{MAX}[Q(6,3), Q(6,5)] \\ &= 0 + 0.9 * \text{MAX} [100, 81] \\ &= 90 \end{aligned}$$

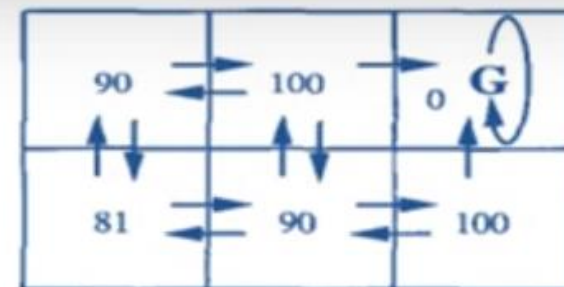
	1	2	3	4	5	6
1	0	90	0	0	0	0
2	81	0	100	0	0	0
3	0	0	0	0	0	0
4	81	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	100	0	0	0

	1	2	3	4	5	6
1	0	90	0	0.72	0	0
2	81	0	100	0	81	0
3	0	0	0	0	0	0
4	81	0	0	0	81	0
5	0	90	0	72	0	90
6	0	0	100	0	81	0

Q-learning Solved Example 3



$Q(s, a)$ values



$V^*(s)$ values



One optimal policy

Q-learning Solved Example 4

Example of Learned Q-Function



SARSA algorithm is a slight variation of the popular Q-Learning algorithm. For a learning agent in any Reinforcement Learning algorithm its policy can be of two types:-

1.On Policy: In this, the learning agent learns the value function according to the current action derived from the policy currently being used.

2.Off Policy: In this, the learning agent learns the value function according to the action derived from another policy.

Q-Learning technique is an **Off Policy** technique and uses the greedy approach to learn the Q-value. SARSA technique, on the other hand, is an **On Policy** and uses the action performed by the current policy to learn the Q-value.

This difference is visible in the difference of the update statements for each technique:-

1. Q-Learning:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

2. SARSA:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Here, the update equation for SARSA depends on the current state, current action, reward obtained, next state and next action. This observation lead to the naming of the learning technique as SARSA stands for **State Action Reward State Action** which symbolizes the tuple (s, a, r, s', a').

Summary

- Understand SARSA Algorithm
- Explain the difference between On-Policy vs Off-Policy
- Understand and explain Q-learning



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY



Thank You

