

COM 3240

REINFORCEMENT LEARNING

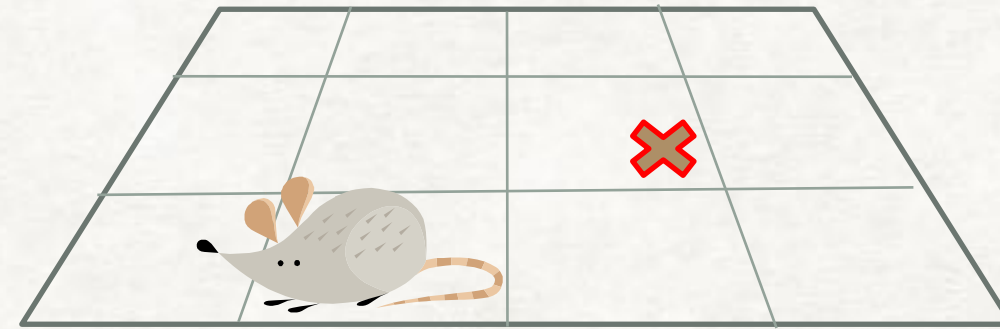


Create me a painting in the style of Dali with the title "waiting for the discounted expected returns"

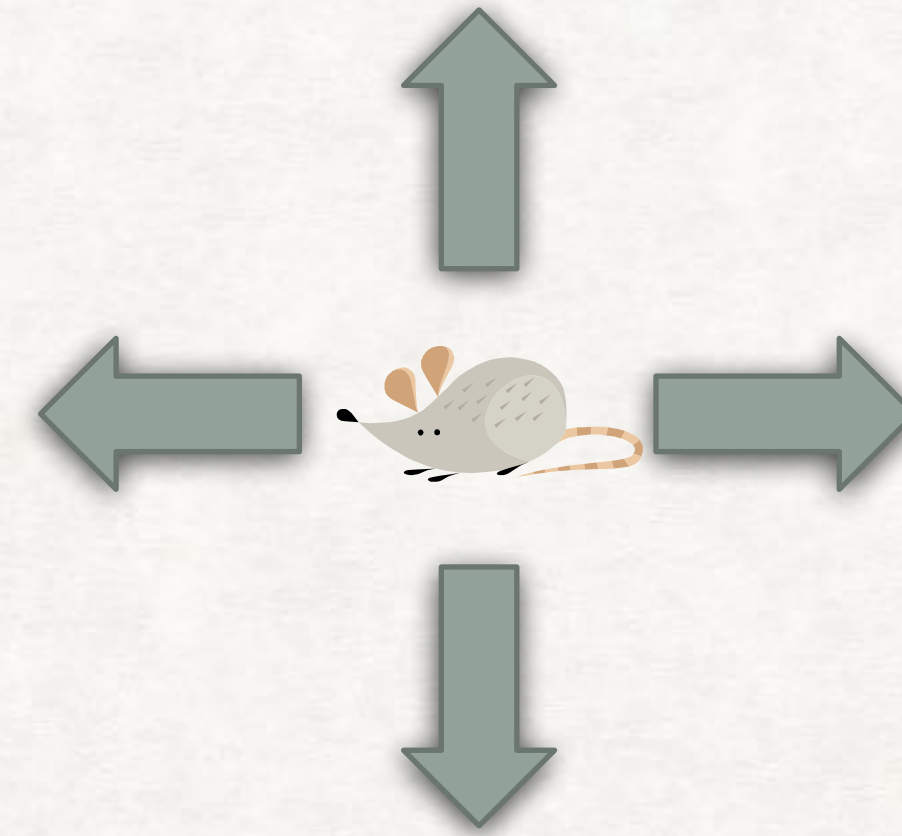
DALL·E

FUTURE REWARDS

MARCOVIAN PROPERTY



Squares = State (in this case)



Actions

The future state depends ONLY on the current state and not on the sequence of events that preceded it.

FUTURE REWARDS

EXPECTED RETURN

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots + R_T$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

FUTURE REWARDS

EXPECTED RETURN

$$0 \leq \gamma < 1$$



$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

$$G_t = R_{t+1} + \gamma G_{t+1}$$

FUTURE REWARDS

OPTIMALITY & BELLMAN EQUATION FOR ACTION-VALUES

$$q^{\pi}(s, a) = \mathbb{E}_{\pi}[R_{t+1} + \gamma q^{\pi}(s', a') \mid S_t = s, A_t = a]$$

Simplify notation: $q^{\pi}(s, a) = \mathbb{E}_{\pi}[r + \gamma q^{\pi}(s', a') \mid s, a]$



Current state-action

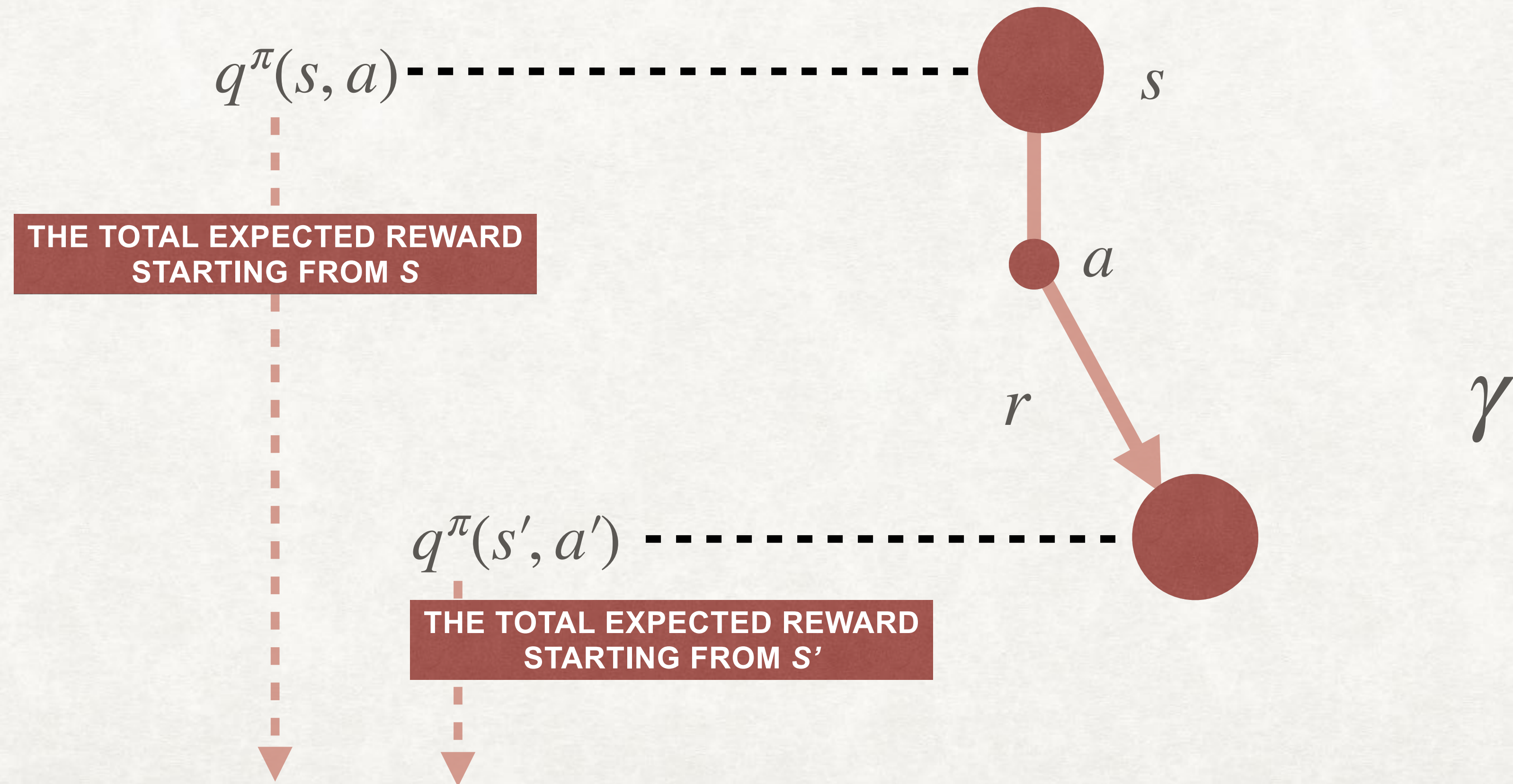
Next state-action

(Stochastic) immediate reward due to action a from state s

FUTURE REWARDS

ACTION-VALUE FUNCTIONS

$$q^{\pi}(s, a) = \mathbb{E}_{\pi}[r + \gamma q^{\pi}(s', a') \mid s, a]$$



FUTURE REWARDS

OPTIMALITY & BELLMAN EQUATION FOR ACTION-VALUES

$$q^*(s, a) = \max_{\pi} q^{\pi}(s, a)$$

One policy better or equal than any other

Greedy policy *

$$q^{\pi}(s, a) = \mathbb{E}_{\pi}[r + \gamma q^{\pi}(s', a') \mid s, a] \quad \pi = *$$

$$q^*(s, a) = \mathbb{E}[r + \gamma \max_{a'} q^*(s', a') \mid s, a]$$

IMMEDIATE REWARDS - A REMINDER

BANDITS

$$q^*(a) \doteq E[R \mid A_t = a]$$

Q-value : estimate of the expected return

$$Q(a) \approx q^*(a)$$

In RL we obtain an estimate via sampling

$$R(A_t = a) \sim P(R \mid A_t = a)$$

Desirable: $E[(Q(a) - R(A_t = a))^2] \approx 0$



IMMEDIATE REWARDS - A REMINDER

BANDITS

Minimise: $\frac{E[(Q(a) - R(A_t = a))^2]}{2}$

$$L(a) = \frac{1}{2T_a} \sum_{t=1}^T (Q(a) - R)^2 \mathbf{1}_{A_t=a}$$

$\mathbf{1}_{A_t=a} = \begin{cases} 1 & \text{if } A_t = a \\ 0 & \text{otherwise.} \end{cases}$ Indicator function

$$T(a) = \sum_{t=1}^T \mathbf{1}_{A_t=a} \quad \mathbf{R}=\mathbf{R}(t) \text{ is the reward at trial } t$$



IMMEDIATE REWARDS

BATCH AND ONLINE UPDATES

$$\Delta Q(a) = -\eta \frac{dL(a)}{dQ} = -\eta \sum_{t=1}^T (Q(a) - R) \mathbf{1}_{A_t=a}$$

**Note a: action,
 α : learning rate!**

We can suppress the sum. This suggests we now update our estimate after each trial (sample) known as online learning. We then write:

$$\Delta Q(a) = -\eta (Q(a) - R) \mathbf{1}_{A_t=a}$$

i.e. for trial 1 to T, if action a is taken, we update its Q value by:

$$Q(a) = Q(a) - \eta (Q(a) - R)$$

FUTURE REWARDS

SARSA

$$q^{\pi}(s, a) = E_{\pi}[r + \gamma q^{\pi}(s', a') \mid s, a]$$

$$E_{\pi}[r + \gamma q^{\pi}(s', a') - q^{\pi}(s, a) \mid s, a] = 0$$

$$\mathcal{L}(s, a) = \frac{1}{2N} \sum_{i=1}^N \left(Q(s, a) - [r^{(i)} + \gamma Q(s'^{(i)}, a'^{(i)})] \right)^2$$

FUTURE REWARDS

SARSA

$$\mathcal{L}(s, a) = \frac{1}{2N} \sum_{i=1}^N \left(Q(s, a) - [r^{(i)} + \gamma Q(s'^{(i)}, a'^{(i)})] \right)^2$$

$$\frac{\partial \mathcal{L}(Q(s, a))}{\partial Q(s, a)} = \frac{1}{N} \sum_{i=1}^N \left(Q(s, a) - [r + \gamma Q(s', a')] \right)$$

$$\Delta Q(s, a) = -\eta \frac{1}{N} \sum_{i=1}^N \left(Q(s, a) - [r + \gamma Q(s', a')] \right)$$

FUTURE REWARDS

SARSA

$$\Delta Q(s, a) = -\eta \frac{1}{N} \sum_{i=1}^N (Q(s, a) - [r + \gamma Q(s', a')])$$

N=1 (online learning):

$$\Delta Q(s, a) = -\eta (Q(s, a) - [r + \gamma Q(s', a')])$$

$$\Delta Q(s, a) = \eta (r + \gamma Q(s', a') - Q(s, a))$$

FUTURE REWARDS

SARSA: REWARD - “ANTICIPATED REWARD”

$$\Delta Q = \eta \left(\underbrace{r + \gamma Q(s', a')}_{\text{What I “actually” get}} - \underbrace{Q(s, a)}_{\text{Anticipated reward}} \right)$$

What I “actually” get

Anticipated reward

FUTURE REWARDS

POLICIES

Greedy $a = \operatorname{argmax}_a Q(s, a)$

Optimistic Greedy: initialise Q-values unrealistically high

Epsilon-Greedy : explore with probability epsilon, greedy otherwise

Softmax: $P(a) = \frac{e^{Q(s,a)/\tau}}{\sum_b e^{Q(s,b)/\tau}}$

FUTURE REWARDS

THE SARSA ALGORITHM

1. Initialise $Q(s, a)$ arbitrarily for all $s \in \mathbf{S}$ and $a \in \mathbf{A}(s)$.
2. Repeat (for each episode):
 - a. Initialise s .
 - b. Choose an action a from s using a policy derived from Q (e.g., ϵ -greedy).
 - c. Repeat (for each step of episode):
 - i. Take action a , observe reward r and next state s' .
 - ii. Choose a' from s' using policy derived from Q (e.g., ϵ -greedy).
 - iii. $Q(s, a) \leftarrow Q(s, a) + \eta * [r + \gamma * Q(s', a') - Q(s, a)]$.
 - iv. $s \leftarrow s'$; $a \leftarrow a'$.
 - d. until s is terminal.

On-policy

FUTURE REWARDS

THE SARSA ALGORITHM

In 1994, Gavin Rummery and Mahesan Niranjan published a paper titled “Online Q-Learning using Connectionist Systems,” in which they introduced an algorithm they called at the time “Modified Connectionist Q-Learning.” In 1996, Singh and Sutton dubbed this algorithm Sarsa because of the quintuple of events that the algorithm uses: $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$. People often like knowing where these names come from as you will soon see, RL researchers can get pretty creative with these names.

<https://livebook.manning.com/concept/reinforcement-learning/this-algorithm>

FUTURE REWARDS

THE SARSA ALGORITHM

Right after obtaining his Ph.D. in 1995, Gavin became a programmer and later a lead programmer for the company responsible for the series of the Tomb Raider games. Gavin has had a very successful career as a game developer.

Maheesan, who became Gavin's Ph.D. supervisor after the unexpected death of Gavin's original supervisor, followed a more traditional academic career holding lecturer and professor roles ever since his Ph.D. graduation in 1990.

<https://livebook.manning.com/concept/reinforcement-learning/this-algorithm>

FUTURE REWARDS

Q-LEARNING

$$q^*(s, a) = \max_{\pi} q^{\pi}(s, a)$$

$$q^*(s, a) = E[R_{t+1} + \gamma \max_{a'} q^*(s', a') \mid S_t = s, A_t = a]$$

SARSA

$$\Delta Q(s, a) = \eta (r + \gamma Q(s', a') - Q(s, a))$$

Q-Learning

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

FUTURE REWARDS

Q-LEARNING

$$\Delta Q = \eta \left(\underbrace{r + \gamma \max_{a'} Q(s', a')}_{\text{What I "actually" get}} - \underbrace{Q(s, a)}_{\text{Anticipated reward}} \right)$$

FUTURE REWARDS

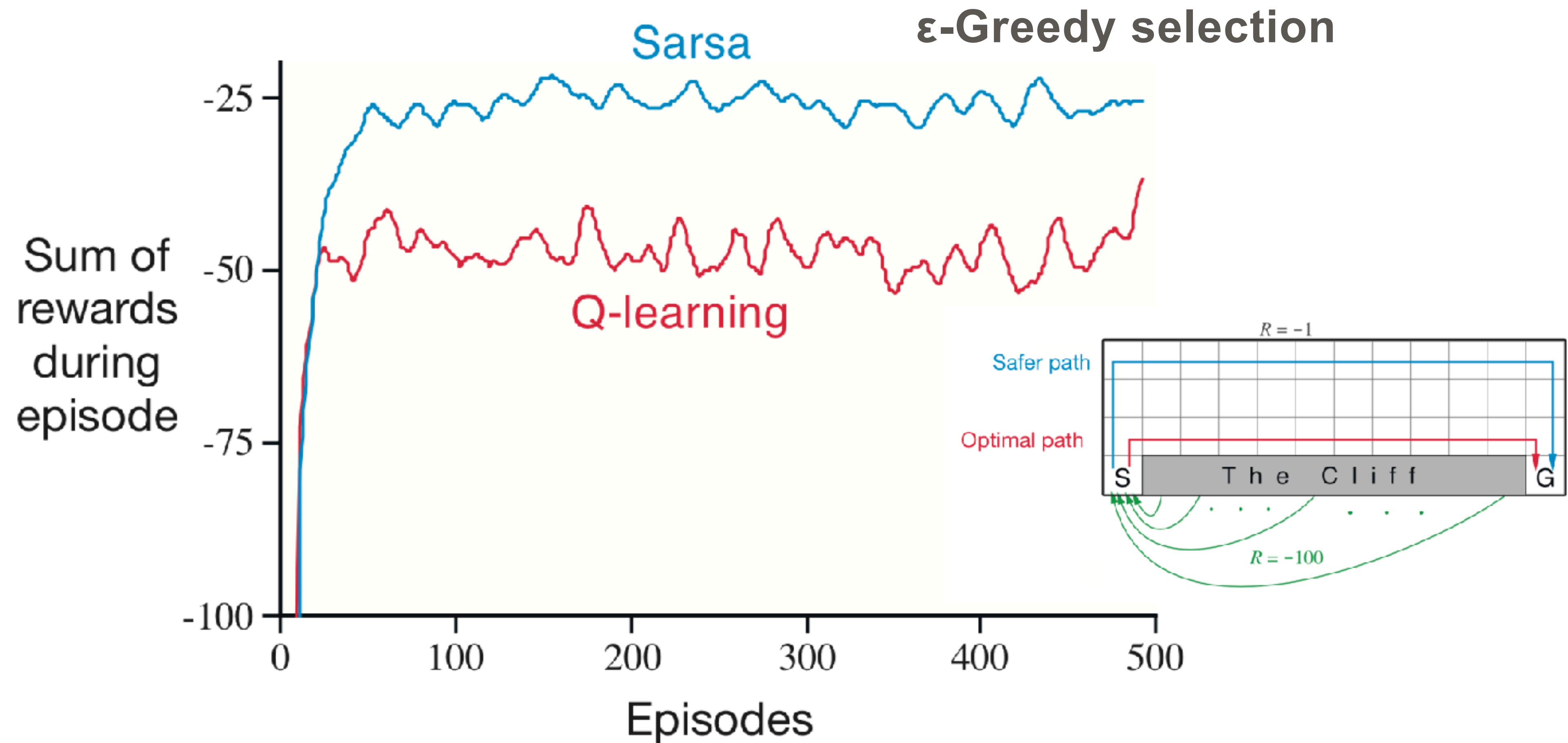
THE Q-LEARNING ALGORITHM

1. Initialise $Q(s, a)$ arbitrarily for all $s \in S$ and $a \in A(s)$.
2. Repeat (for each episode):
 - a. Initialise s .
 - b. Repeat (for each step of episode):
 - i. Choose an action a from s using a policy derived from Q (e.g., ϵ -greedy).
 - ii. Take action a , observe reward r and next state s' .
 - iii. $Q(s, a) \leftarrow Q(s, a) + \eta * [r + \gamma * \max_{a'} Q(s', a') - Q(s, a)]$.
 - iv. $s \leftarrow s'$.
 - c. until s is terminal.

Off-policy

FUTURE REWARDS

SARSA VS Q-LEARNING

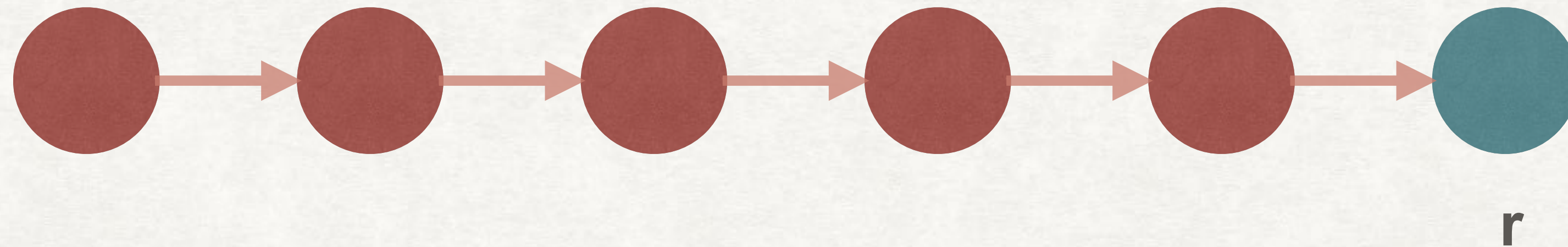


From Sutton & Barto, 2020 - with permission.

FUTURE REWARDS

SARSA

$$\Delta Q = \eta \left(r + \gamma Q(s', a') - Q(s, a) \right)$$



A robot walks this corridor and finds reward. How many Q-values will be updated?

FUTURE REWARDS

ELIGIBILITY TRACE



Design an ant leaving a trace of pheromone.

DALL·E

FUTURE REWARDS

ELIGIBILITY TRACE

We mark that action pair that were visited/used:

$e(s, a)$

Update eligibility trace for the most recent state action pair:

$e(s, a) \leftarrow e(s, a) + 1$

Decay eligibility trace for all other states:

$e(s, a) \leftarrow \gamma \lambda * e(s, a)$

FUTURE REWARDS

SARSA: TABULAR ELIGIBILITY TRACE

1. Initialise $Q(s, a)$ arbitrarily for all $s \in S$ and $a \in A(s)$.
 2. Initialise eligibility traces $e(s, a)$ for all s, a to zeros.
 3. Repeat (for each episode):
 - a. Initialise s , eligibility traces $e(s, a)$ for all s, a to zeros.
 - b. Choose an action a from s using policy derived from Q (e.g., ϵ -greedy).
 - c. Repeat (for each step of episode):
 - i. Take action a , observe reward r , and next state s' .
 - ii. Choose a' from s' using policy derived from Q (e.g., ϵ -greedy).
 - iii. For the visited s, a :
 - Update delta: $\delta \leftarrow r + \gamma * Q(s', a') - Q(s, a)$
 - Update eligibility trace: $e(s, a) \leftarrow e(s, a) + 1$
 - iv. For all states:
 - Update $Q(s, a) \leftarrow Q(s, a) + \eta * \delta * e(s, a)$
 - Decay eligibility trace: $e(s, a) \leftarrow \gamma\lambda * e(s, a)$
 - v. $s \leftarrow s'; a \leftarrow a'$.
- until s is terminal.

FUTURE REWARDS

TABULAR ELIGIBILITY TRACE

1. Initialise $Q(s, a)$ arbitrarily for all $s \in S$ and $a \in A(s)$.
 2. Repeat (for each episode):
 - a. Initialise s and eligibility traces $e(s, a)$ for all s, a to zeros.
 - b. Repeat (for each step of episode):
 - i. Choose an action a from s using policy derived from Q (e.g., ϵ -greedy).
 - ii. Take action a , observe reward r , and next state s' .
 - iii. For the selected s, a :
 - Update delta: $\delta \leftarrow r + \gamma * \max_{a'} Q(s', a') - Q(s, a)$
 - Update eligibility trace: $e(s, a) \leftarrow e(s, a) + 1$
 - iv. For all s, a
 - Update $Q(s, a) \leftarrow Q(s, a) + \eta * \delta * e(s, a)$
 - Decay eligibility trace: $e(s, a) \leftarrow \gamma\lambda * e(s, a)$
 - v. $s \leftarrow s'$.
- until s is terminal.

FUTURE REWARDS

TEMPORAL DIFFERENCE LEARNING

- **By approximating the action value functions by an estimate Q and by writing an error function based on a form of the Bellman equation we derived two Temporal difference algorithms**
 - **SARSA : State Action Reward State Action - “safe”**
 - **Q-Learning : optimal**
- **An eligibility trace can speed up the performance of the algorithms by propagating to the pathway taken information about the reward.**

THANK YOU!