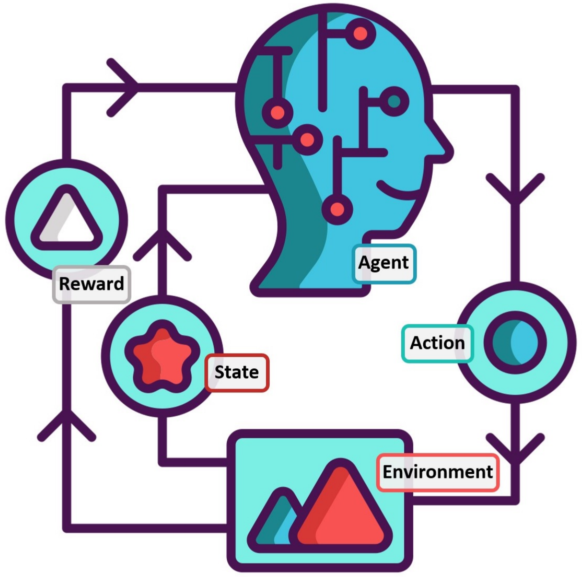
Reinforcement Learning 2025/2026

Lecture #10
Temporal Difference Learning for Control

Gian Antonio Susto



Announcements before starting

- 1st partial exam list now open (Google form - no uniweb enrollment)

- Content for the 1st partial exam will end this week:

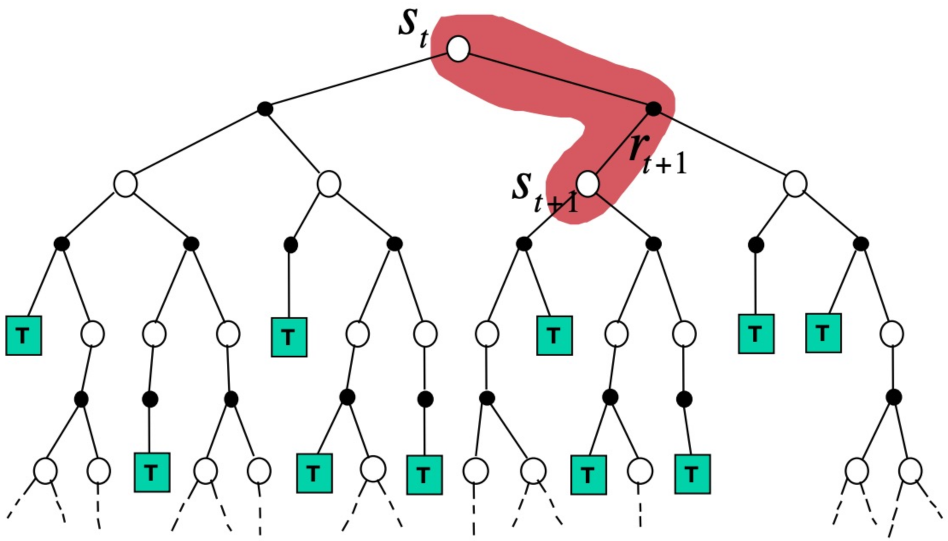
1. Lectures/slides: from lecture 1 to lecture 10
2. Book: from chapter 1 to chapter 6

- Next week:

1. Lecture on Wed. 5th of November: recap lecture! I will start preparing some materials based on your input! Send input, be prepared to ask questions!
2. Lecture on Thu. 6th of November: n-step bootstrapping + TD-lambda (content for 2nd partial)

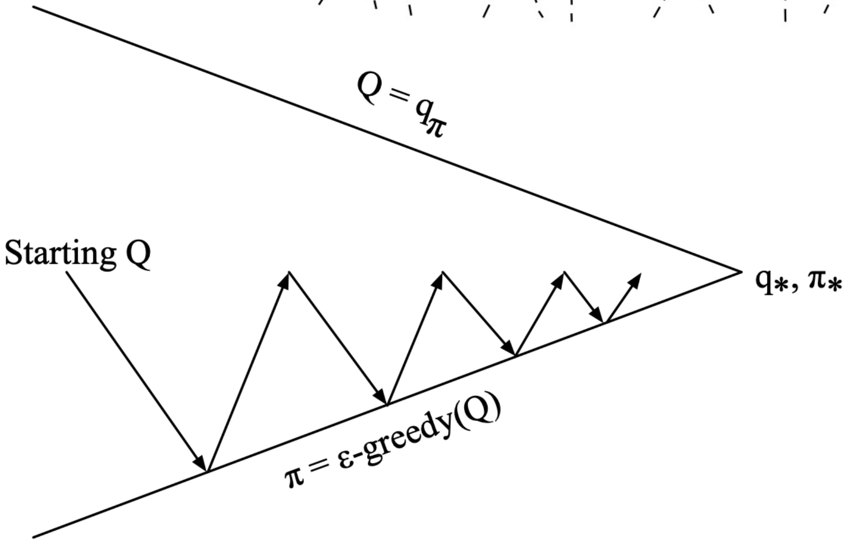
Recap – TD-Learning Control

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



- We have seen how Generalized Policy Iteration could be applied also to the TD-Learning framework
- We will see 3 approaches

1. SARSA (on-policy)
2. Q-learning
3. Expected SARSA



Control: Q-learning – (Off-policy) TD learning for Control

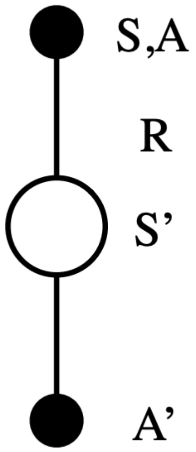
- Q-learning is the most popular approach to RL control
- We'll see how Q-learning (for TD(0)):

1. Can be derived as a slight modification from SARSA
2. Is associated with the Bellman Optimality Equation
3. (Can be considered off-policy)

Control: Q-learning vs. SARSA

SARSA

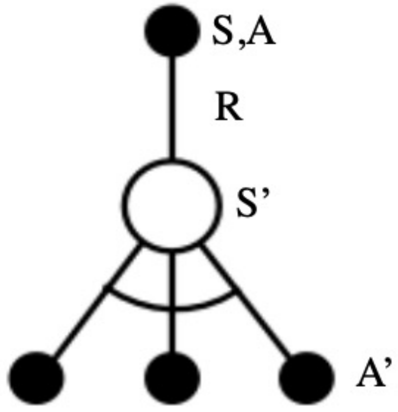
Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \boxed{\gamma Q(S',A')} - Q(S,A)\right]$
$S \leftarrow S'$; $A \leftarrow A'$;



The order of taking action and choosing action is inverted between sarsa and q lenrng, this changes the behavior

Q-learning

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \boxed{\gamma \max_a Q(S',a)} - Q(S,A)\right]$
$S \leftarrow S'$

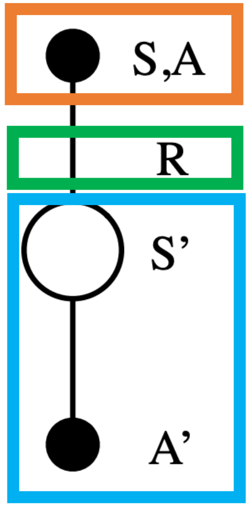


in Q learning we dont do A', we take the max in the next step

Control: Q-learning vs. SARSA

SARSA

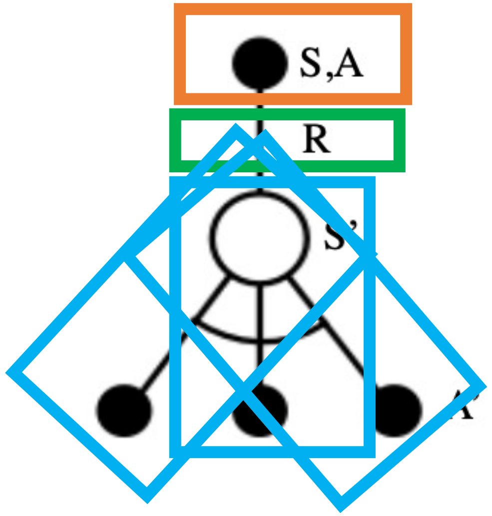
Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
$S \leftarrow S'$; $A \leftarrow A'$;



The order of taking action and choosing action is inverted between sarsa and q lenrng, this changes the behavior

Q-learning

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
$S \leftarrow S'$



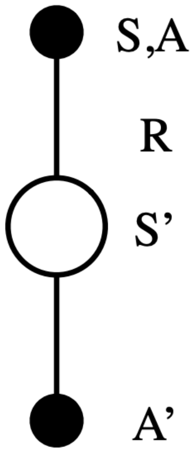
in Q learning we dont do A’, we take the max in the next step

Control: Q-learning vs. SARSA

The order of taking action and choosing action is inverted between sarsa and q lenrng, this changes the behavior

SARSA

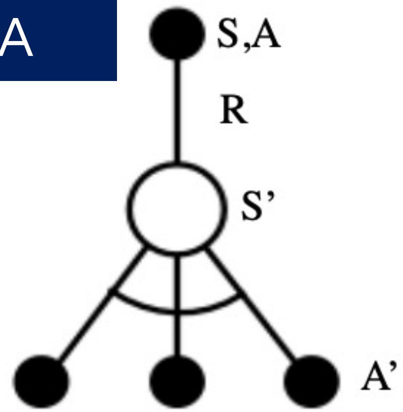
Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \boxed{\gamma Q(S',A')} - Q(S,A)\right]$$
$S \leftarrow S'$; $A \leftarrow A'$;



Q-learning is usually faster than SARSA

Q-learning

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \boxed{\gamma \max_a Q(S',a)} - Q(S,A)\right]$$
$S \leftarrow S'$



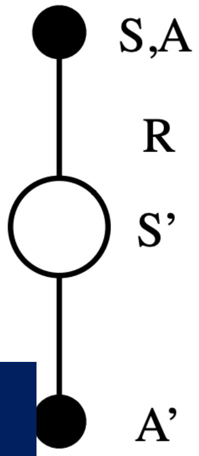
in Q learning we dont do A', we take the max in the next step

Control: Q-learning vs. SARSA

SARSA

Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
$S \leftarrow S'$; $A \leftarrow A'$;

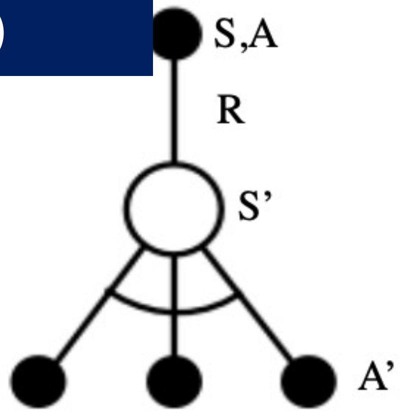
You actual perform next action, according to the policy and then update Q(s,a)



You look ahead and imagine greedy next action to update Q(s,a) (but you then perform the actual next action based on your current policy)

Q-learning

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
$S \leftarrow S'$



in Q learning we dont do A', we take the max in the next step

The order of taking action and choosing action is inverted between sarsa and q lenrng, this changes the behavior

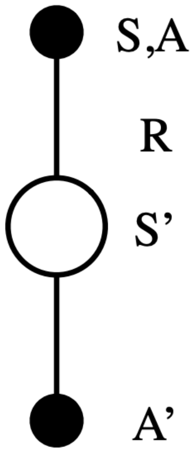
TD-Learning Control: Q-learning vs. SARSA

The order of taking action and choosing action is inverted between sarsa and q lenrng, this changes the behavior

SARSA

ON POLICY: We only have one (target) policy here

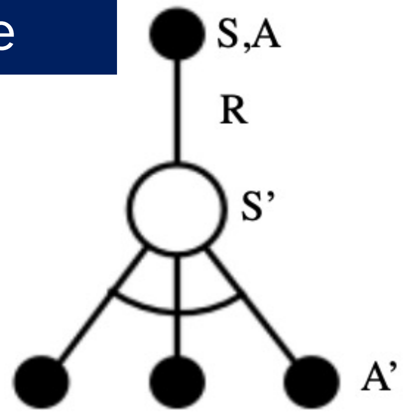
Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
$S \leftarrow S'$; $A \leftarrow A'$;



Q-learning

OFF-POLICY: We have a behaviour (epsilon-greedy) and a target policy (greedy!) here

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
$S \leftarrow S'$



in Q learning we dont do A', we take the max in the next step

TD-Learning Control: Q-learning vs. SARSA

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

Bellman Expectation Equation

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

SARSA is a sample-based version of Policy Iteration

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

Bellman Optimality Equation

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

Q-learning is a sample-based version of Value Iteration

A. White, M. White 'Sample-based Learning Methods'

Control: Q-learning – (Off-policy) TD learning for Control

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 $Q(\mathit{terminal}, \cdot) = 0$

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'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

Control: Q-learning – (Off-policy) TD learning for Control

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

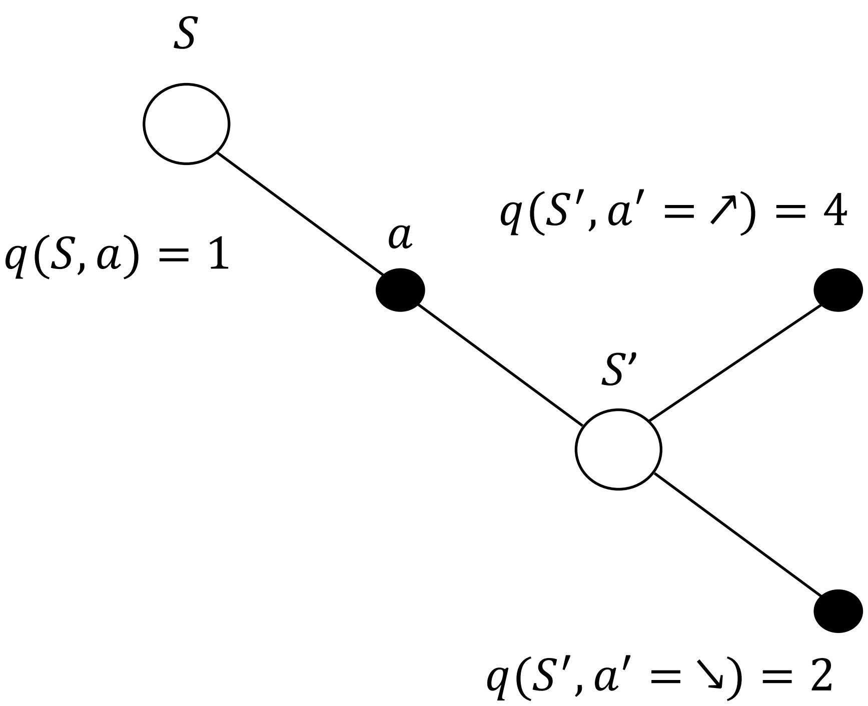
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

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'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

Control: Q-learning vs. SARSA – example

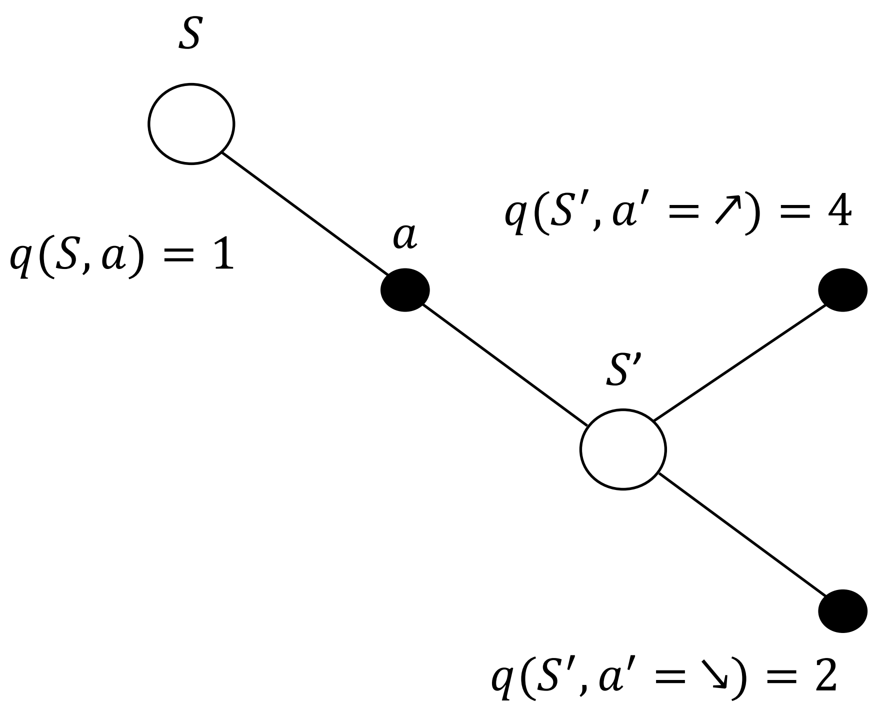
$\gamma = 0.5$
$\alpha = 0.1$

First episode we transition from S to S' by taking action a and we get a reward of +1



Control: Q-learning vs. SARSA – example

$\gamma = 0.5$
$\alpha = 0.1$



First episode we transition from S to $S'$by taking action a and we get a reward of +1

SARSA:

- Target:

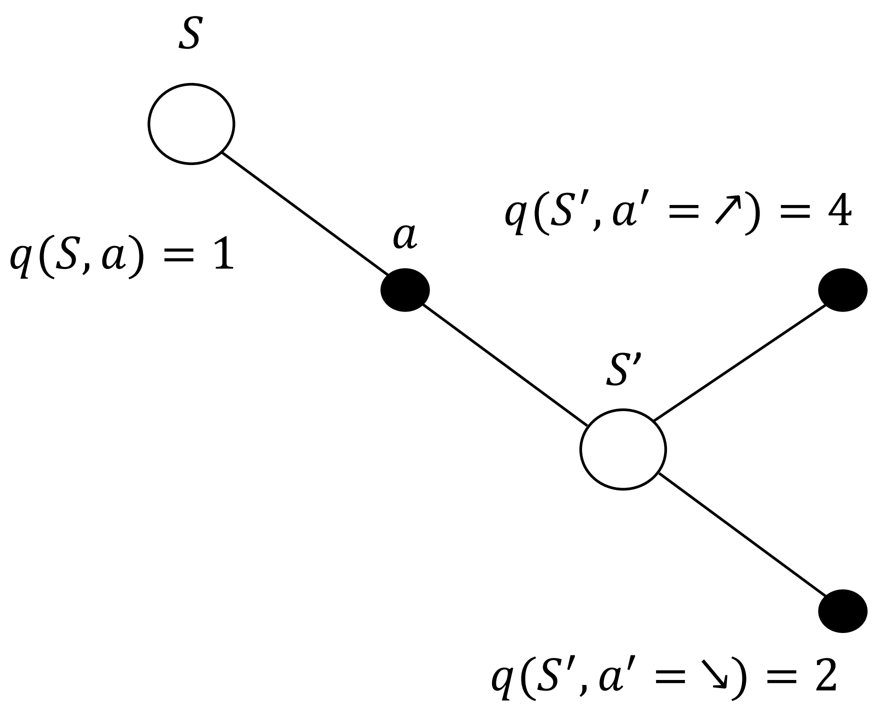
$\mathrm{r} + \gamma q(s', \nearrow)$ = +1+0.5(+4) = +3 if by policy π we have $a' = \nearrow$ in s'
$\mathrm{r} + \gamma q(s', \searrow)$ = +1+0.5(+2) = +2 if by policy π we have $a' = \searrow$ in s'

- Update

$q(S, a) = 1 + 0.1 * (3 - 1) = 1.2$ if by policy π we have $a' = \nearrow$ in s'
$q(S, a) = 1 + 0.1 * (2 - 1) = 1.1$ if by policy π we have $a' = \searrow$ in s'

Control: Q-learning vs. SARSA – example

$\gamma = 0.5$
$\alpha = 0.1$



First episode we transition from S to S' by taking action a and we get a reward of +1

SARSA:

- Target:

$\mathrm{r} + \gamma q(s', \nearrow)$ = +1+0.5(+4) = +3 if by policy π we have $a' = \nearrow$ in s'
$\mathrm{r} + \gamma q(s', \searrow)$ = +1+0.5(+2) = +2 if by policy π we have $a' = \searrow$ in s'

- Update

$q(S, a) = 1 + 0.1 * (3 - 1) = 1.2$ if by policy π we have $a' = \nearrow$ in s'
$q(S, a) = 1 + 0.1 * (2 - 1) = 1.1$ if by policy π we have $a' = \searrow$ in s'

Q-learning

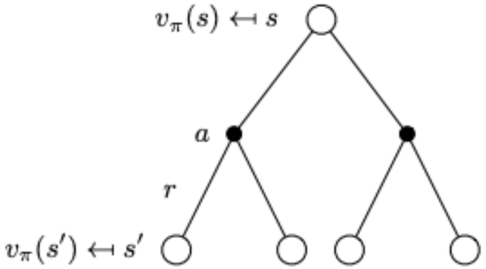
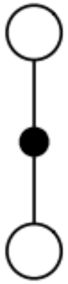
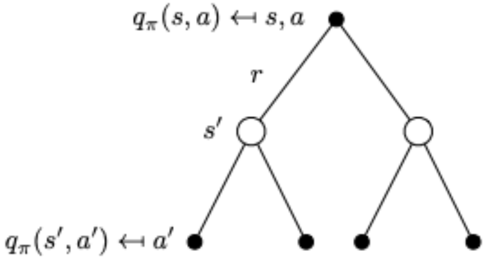
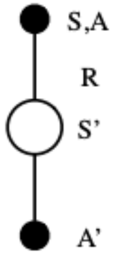
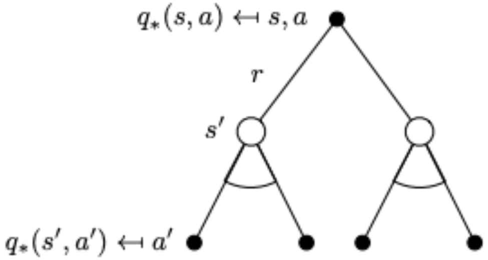
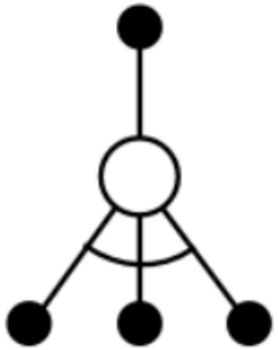
- Target:

$\mathrm{r} + \gamma \max_{a'} q(s', a')$ = +1+0.5(+4) = +3 indipendently from the current policy π (for this reason it is off-policy!)

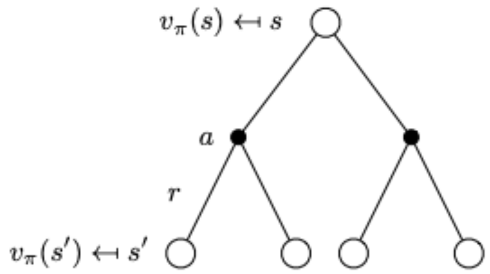
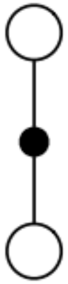
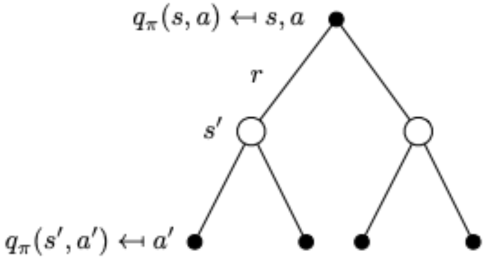
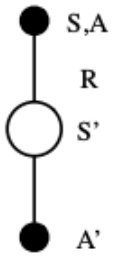
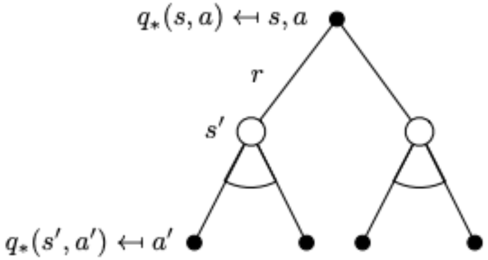
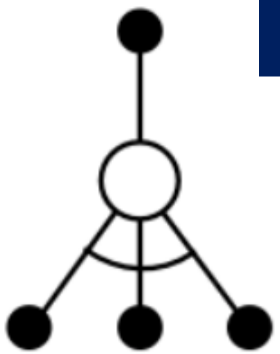
- Update:

$q(S, a) = 1 + 0.1 * (3 - 1) = 1.2$

H. Van Hasselt 'Reinforcement Learning' @ UCL 2021

	Full Backup (DP)	*Sample Backup (TD)*
Bellman Expectation Equation for $\boxed{v_\pi(s)}$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $\boxed{q_\pi(s, a)}$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $\boxed{q_*(s, a)}$	 Q-Value Iteration	 Q-Learning

D. Silver 'Reinforcement Learning' @ UCL

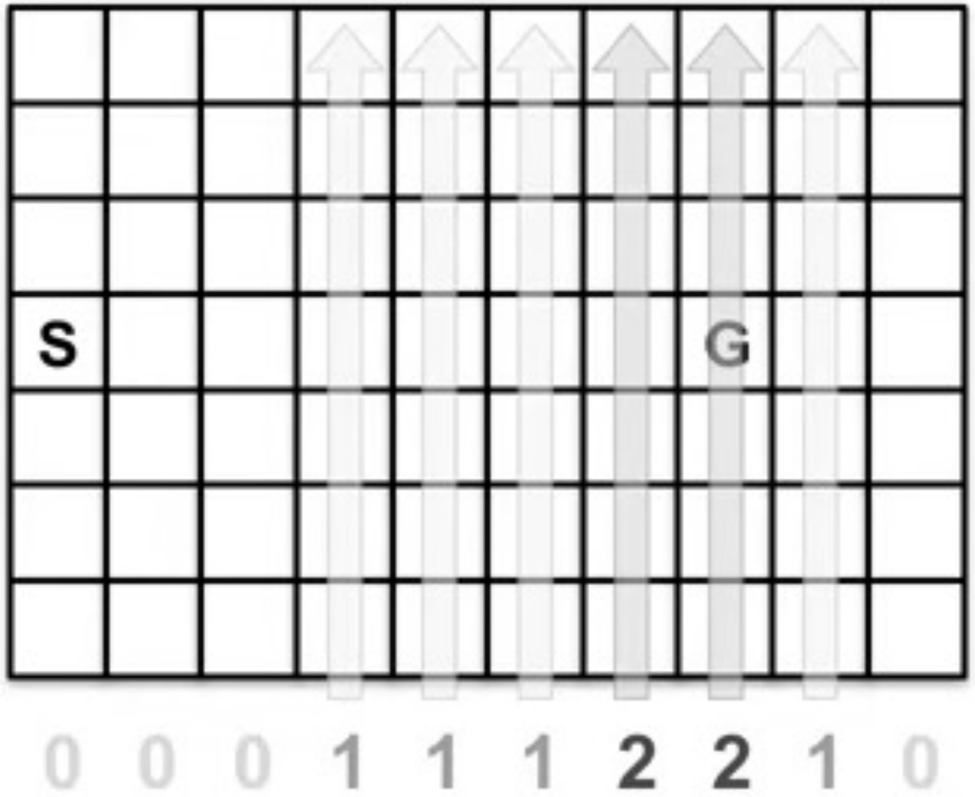
	Full Backup (DP)	*Sample Backup (TD)*
Bellman Expectation Equation for $\boxed{v_\pi(s)}$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $\boxed{q_\pi(s, a)}$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $\boxed{q_*(s, a)}$	 Q-Value Iteration	 Q-Learning

Q-learning is usually faster than SARSA

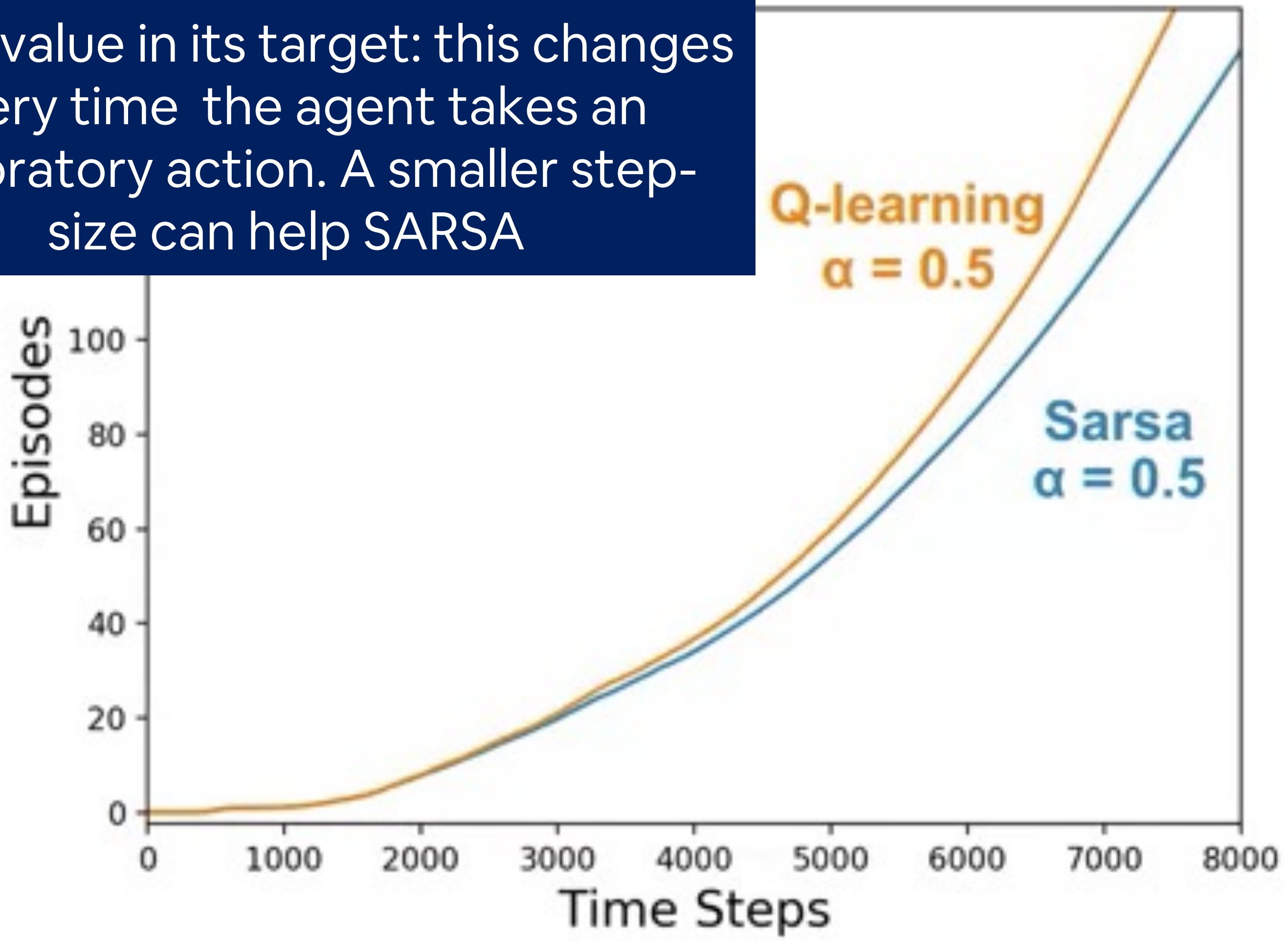
D. Silver 'Reinforcement Learning' @ UCL

Control: SARSA vs Q-learning – 'Windy Grid World' Example #01

SARSA uses the estimate of the next action value in its target: this changes every time the agent takes an exploratory action. A smaller step-size can help SARSA



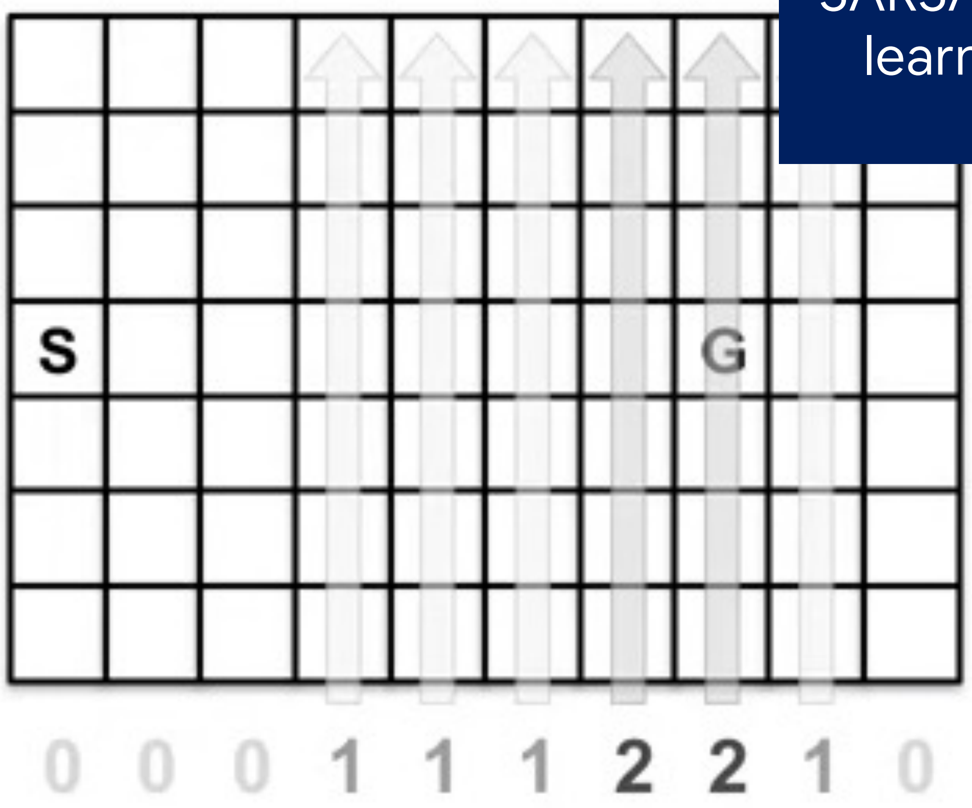
$\epsilon = 0.1$



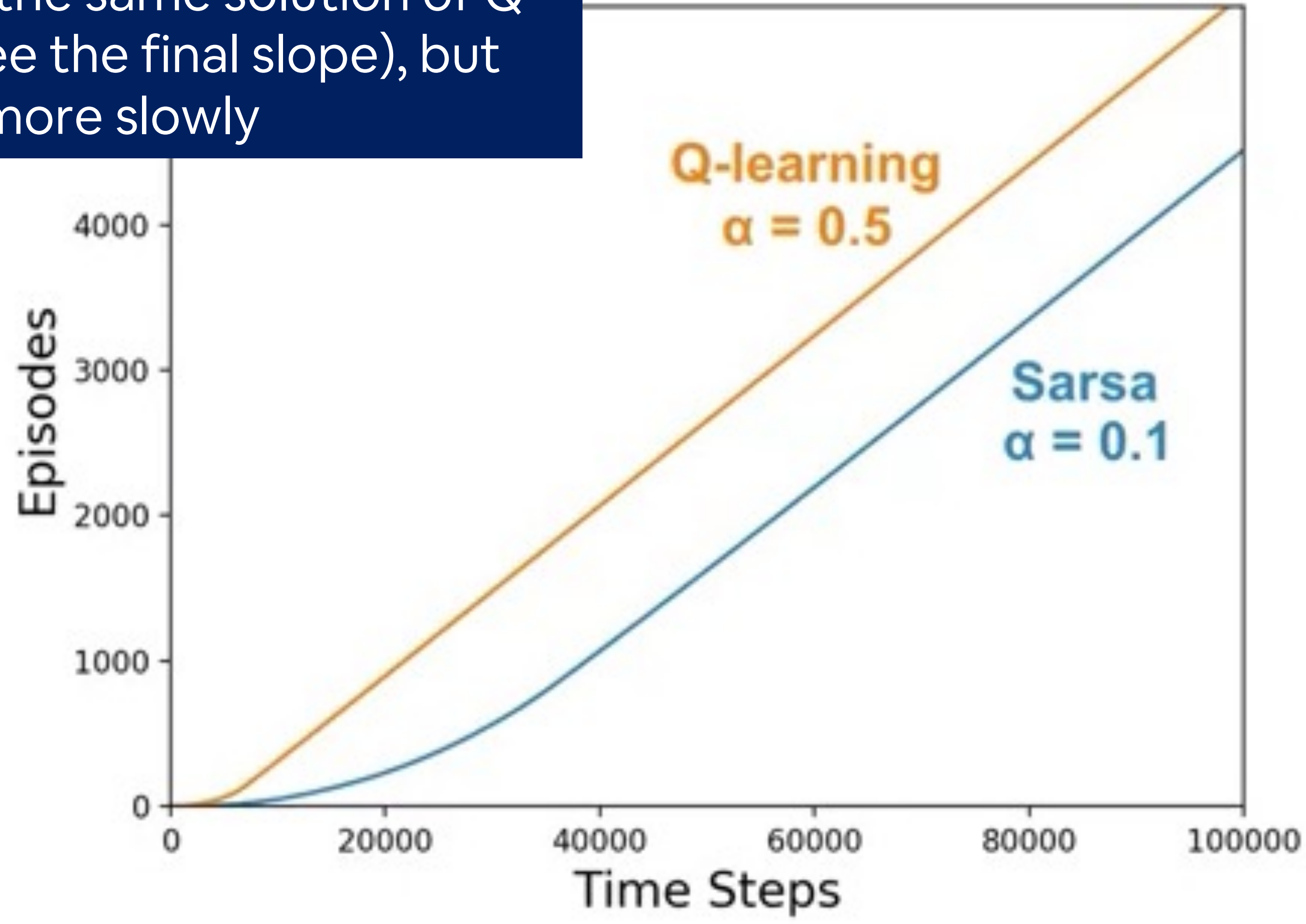
A. White, M. White 'Sample-based Learning Methods'

Control: SARSA vs Q-learning – 'Windy Grid World' Example #01

SARSA finds the same solution of Q-learning (see the final slope), but more slowly



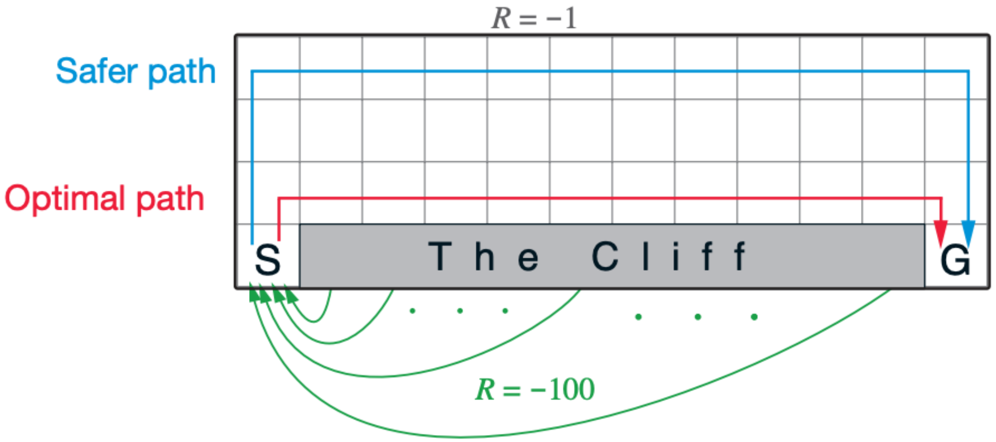
$\epsilon = 0.1$



A. White, M. White 'Sample-based Learning Methods'

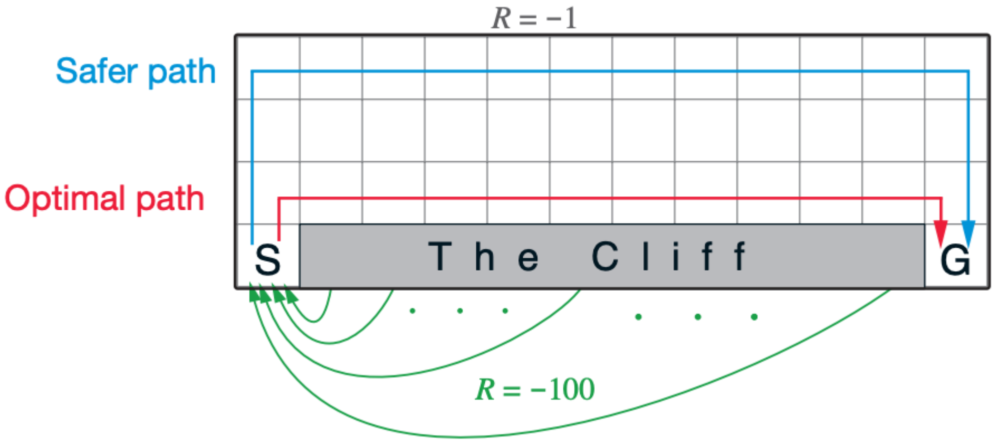
Control: SARSA vs Q-learning – ’The Cliff Gridworld’ Example #02

- Q-learning doesn't iterate between policy evaluation and policy improvement, but rather learns the optimal values directly. Not always ideal!



Control: SARSA vs Q-learning – 'The Cliff Gridworld' Example #02

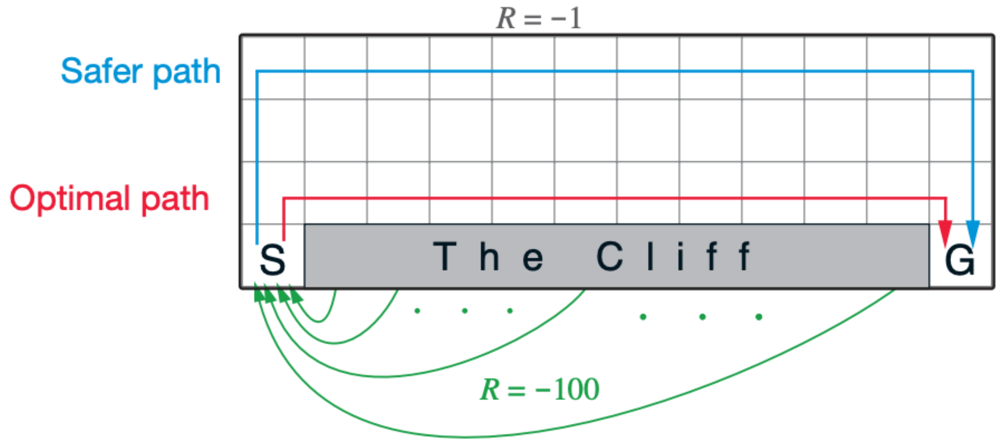
- Q-learning doesn't iterate between policy evaluation and policy improvement, but rather learns the optimal values directly. Not always ideal!
- Since Q-learning learns the optimal value function, it quickly learns that an optimal policy travels right alongside the cliff.
- However, since his actions or epsilon greedy, traveling alongside the cliff occasionally results and falling off the cliff.

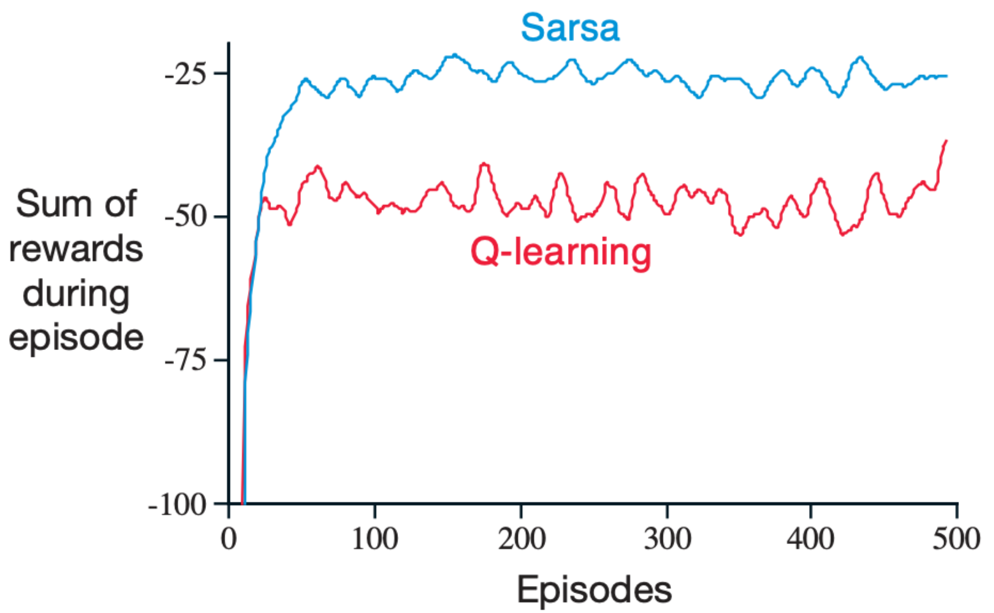


Q-learning trained agent falls down the cliff sometimes!

Control: SARSA vs Q-learning – 'The Cliff Gridworld' Example #02

- Q-learning doesn't iterate between policy evaluation and policy improvement, but rather learns the optimal values directly. Not always ideal!
- Since Q-learning learns the optimal value function, it quickly learns that an optimal policy travels right alongside the cliff.
- However, since his actions or epsilon greedy, traveling alongside the cliff occasionally results and falling off the cliff.
- Sarsa learns about his current policy, considering the effect of epsilon greedy action selection.



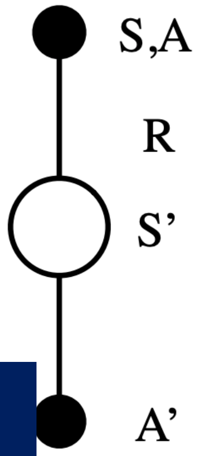


Control: why is Q-learning off-policy?

SARSA

Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
$S \leftarrow S'$; $A \leftarrow A'$;

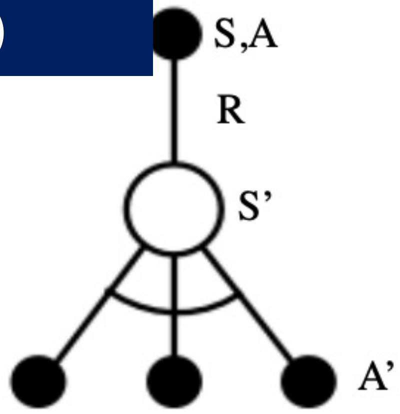
You actual perform next action, according to the policy and then update Q(s,a)



Q-learning

You look ahead and imagine greedy next action to update Q(s,a) (but you then perform the actual next action based on your current policy)

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
$S \leftarrow S'$

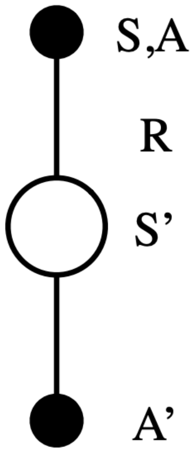


Control: why is Q-learning off-policy?

SARSA

Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
$S \leftarrow S'$; $A \leftarrow A'$;

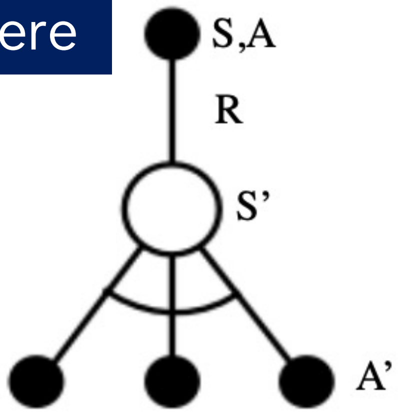
We only have one (target) policy here



Q-learning

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
$S \leftarrow S'$

We have a behaviour (epsilon-greedy) and a target policy (greedy!) here



Control: why is Q-learning off-policy?

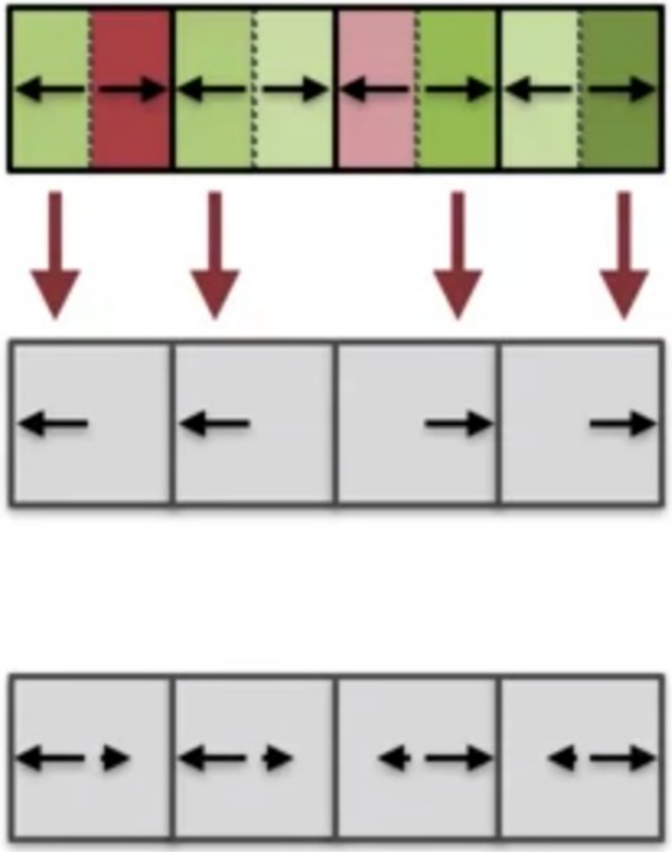
Sarsa: $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big(R_{t+1} + \gamma Q(S_{t+1}, \underbrace{A_{t+1}}_{\sim \pi}) - Q(S_t, A_t)\big)$$

Q-learning: $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, \underbrace{a'}_{\sim \pi_* \neq \pi}) - Q(S_t, A_t)\big)$$

A. White, M. White 'Sample-based Learning Methods'

Control: why is Q-learning off-policy?

$Q(s, a)$



Target policy: π^* $= \underset{a}{\operatorname{argmax}} \, Q(s, a)$

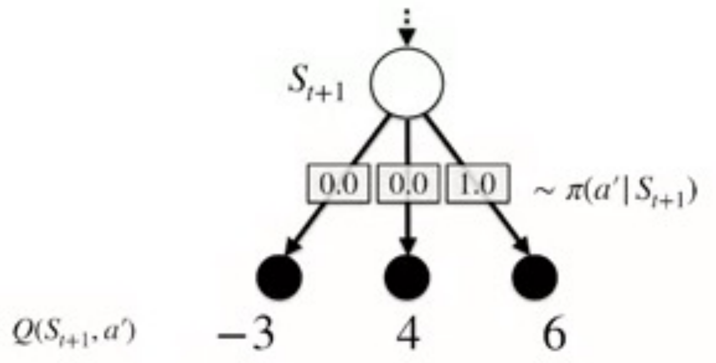
Behavior policy: π

A behaviour policy can be for example ε-greedy

A. White, M. White 'Sample-based Learning Methods'

Control: why is Q-learning off-policy?

- No importance sampling is required: it is because the agent is estimating action values with unknown policy and it does not need important sampling ratios to correct for the difference in action selection.
- The action value function represents the returns following each action in a given state: the agents target policy represents the probability of taking each action in a given state.
- Putting these two elements together, the agent can calculate the expected return under its target policy from any given state,
- Q-learning uses exactly this technique to learn off-policy.
- Since the agents target policies greedy, with respect to its action values, all non-maximum actions have probability 0.
- As a result, the expected return from that state is equal to a maximal action value from that state.

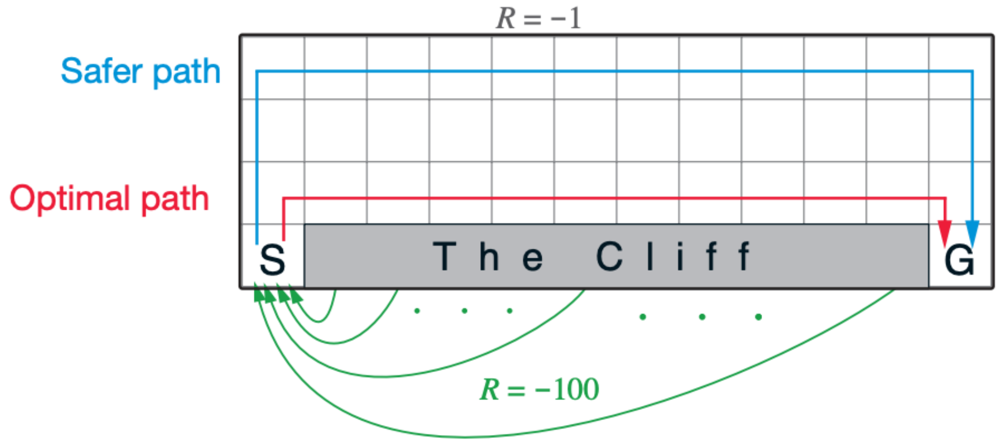


$$\sum_{a'} \pi(a' \mid S_{t+1}) Q(S_{t+1}, a') = \mathbb{E}_\pi[G_{t+1} \mid S_{t+1}] = \max_{a'} Q(S_{t+1}, a') = 6$$

A. White, M. White 'Sample-based Learning Methods'

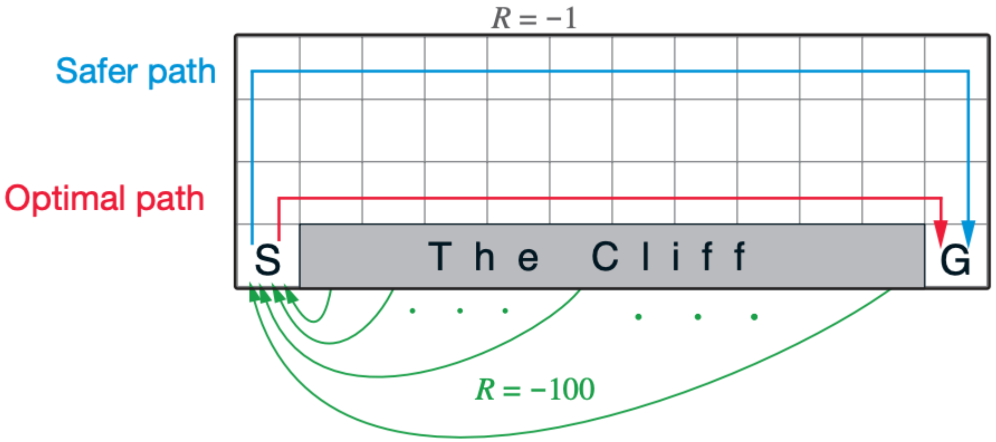
Control: why is Q-learning off-policy?

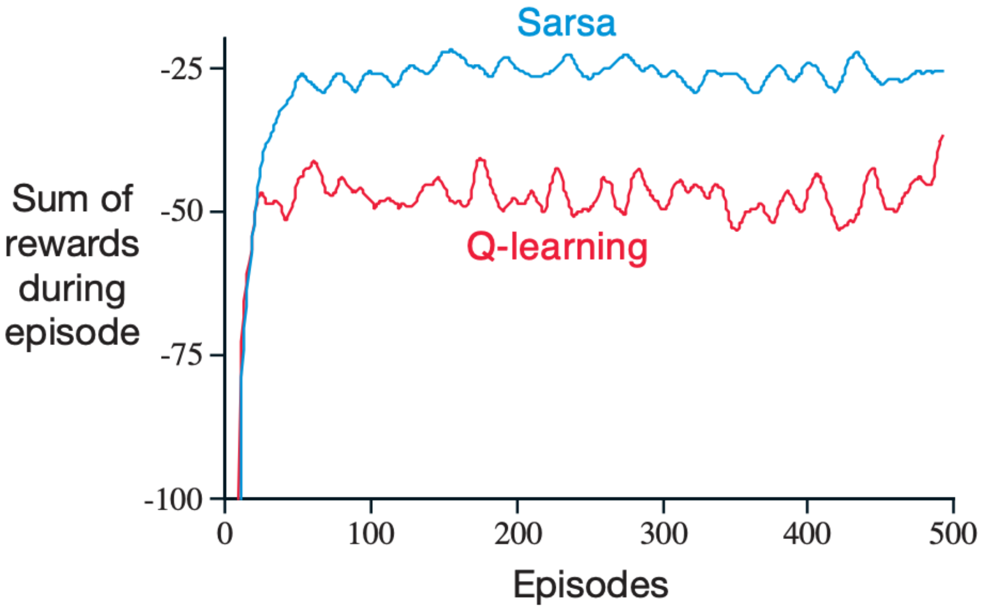
- Q-learning doesn't iterate between policy evaluation and policy improvement, but rather learns the optimal values directly. Not always ideal!



Control: why is Q-learning off-policy?

- Q-learning doesn't iterate between policy evaluation and policy improvement, but rather learns the optimal values directly. Not always ideal!
- Since Q-learning learns the optimal value function, it quickly learns that an optimal policy travels right alongside the cliff.
- However, since his actions are epsilon greedy, traveling alongside the cliff occasionally results and falling off of the cliff.
- Sarsa learns about his current policy, taking into account the effect of epsilon greedy action selection.





Control: Expected SARSA

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

$\sim p(s',r|s,a)$

$\sim \boxed{\pi(a'|s')}$

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

SARSA is a sample-based version of Policy Iteration that exploits the Bellman Expectation Equation: it approximates the expectation by sampling from the environment and from its policy

A. White, M. White 'Sample-based Learning Methods'

Control: Expected SARSA

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

$\sim p(s',r|s,a)$

$\sim \boxed{\pi(a'|s')}$

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

SARSA is a sample-based version of Policy Iteration that exploits the Bellman Expectation Equation: it approximates the expectation by sampling from the environment and from its policy

However, the policy is already known by the agent: why sampling its next action?

A. White, M. White 'Sample-based Learning Methods'

Control: Expected SARSA

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

$\sim p(s',r|s,a)$

$\sim \boxed{\pi(a'|s')}$

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

$$\sum_{a'} \pi(a'|S_{t+1})Q(S_{t+1},a')$$

The agent can compute an expectation: the weighted* sum of all possible next actions

*weights are the probability of next action given the current policy

A. White, M. White 'Sample-based Learning Methods'

Control: Expected SARSA

$$\begin{aligned} Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \mathbb{E}_\pi [Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t) \Big] \\ &= Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \Big], \end{aligned}$$

Control: Expected SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big]$$
$$= Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big],$$

At each step, the agent must average the next state's action values according to how likely they are under the policy.

$Q(S_{t+1}, a') =$	0.0	-1.0	2.0	1.0
$\pi(a' \mid S_{t+1}) =$	0.1	0.1	0.7	0.1

$$\sum_{a'} \pi(a' \mid S_{t+1}) Q(S_{t+1}, a') = 0.0 - 0.1 + 1.4 + 0.1 = 1.4$$

A. White, M. White 'Sample-based Learning Methods'

Control: Expected SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big]$$
$$= Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big],$$

Do you think SARSA or Expected SARSA is better?

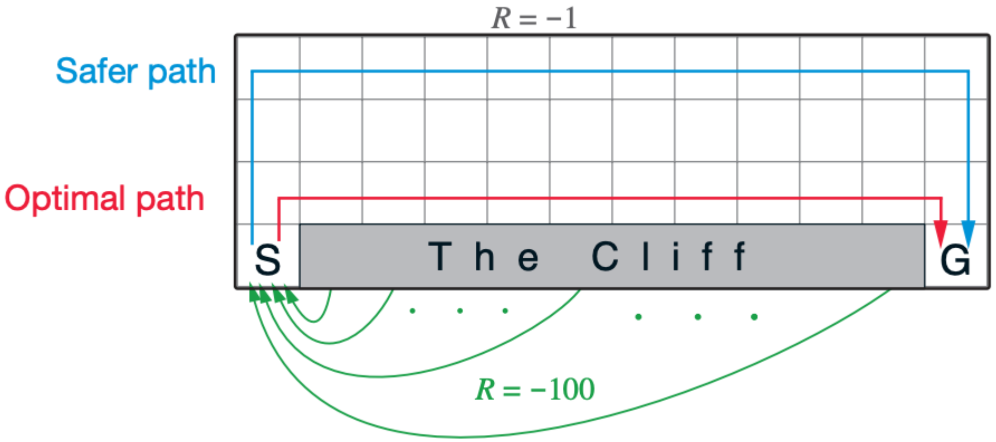
At each step, the agent must average the next state's action values according to how likely they are under the policy.

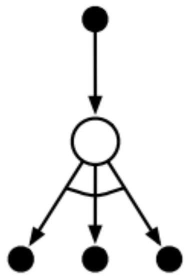
$Q(S_{t+1}, a') =$	0.0	-1.0	2.0	1.0	
$\pi(a' \mid S_{t+1}) =$	0.1	0.1	0.7	0.1	
$\sum_{a'} \pi(a' \mid S_{t+1}) Q(S_{t+1}, a') =$	0.0	$-$ 0.1	+ 1.4	+ 0.1	$= 1.4$

A. White, M. White 'Sample-based Learning Methods'

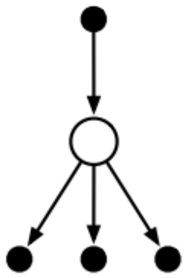
Control: Expected SARSA vs. SARSA

- Expected SARSA has slower updates, but it is more stable (lower variance than SARSA).





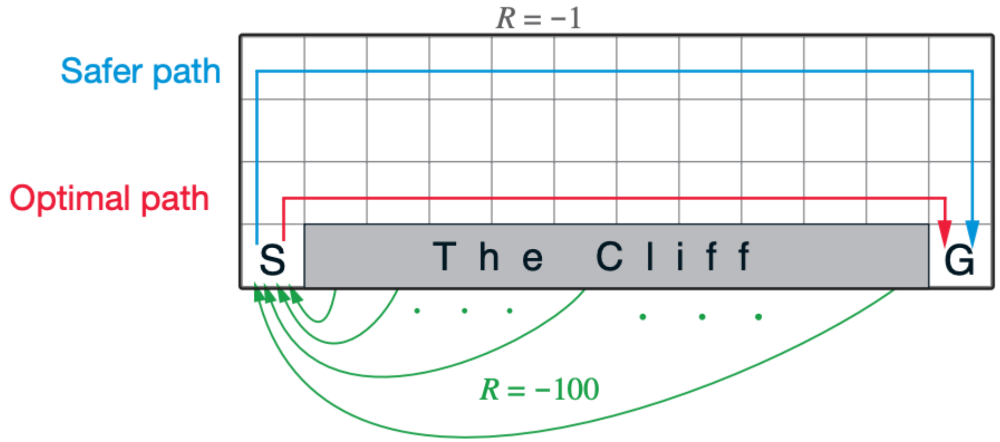
Q-learning

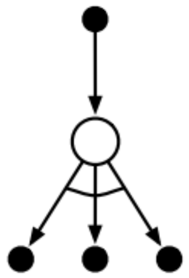


Expected Sarsa

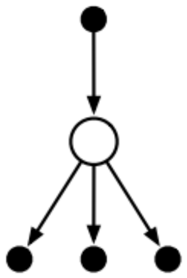
Control: Expected SARSA vs. SARSA

- Expected SARSA has slower updates, but it is more stable (lower variance than SARSA).
- Expected SARSA moves deterministically in the same direction as SARSA moves in expectation.





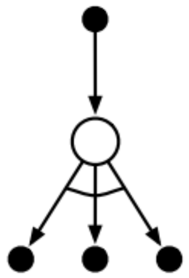
Q-learning



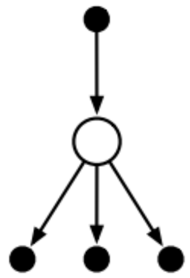
Expected Sarsa

Control: Expected SARSA vs. SARSA

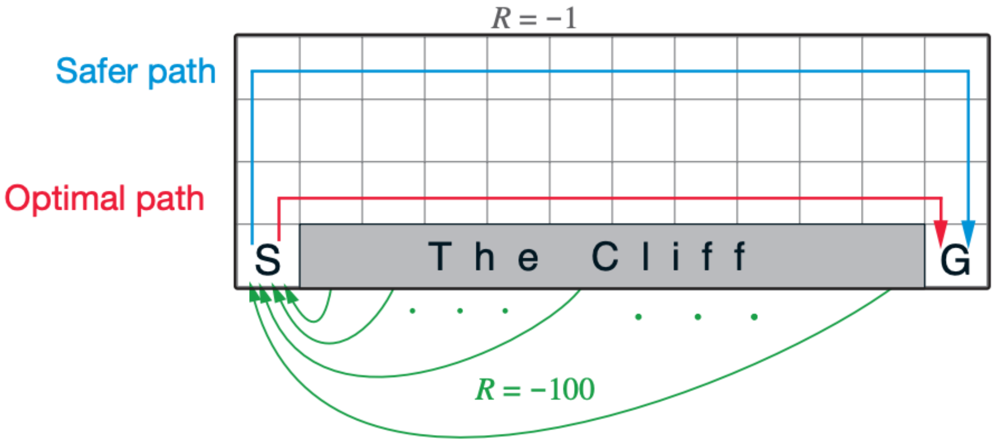
- Expected SARSA has slower updates, but it is more stable (lower variance than SARSA).
- Expected SARSA moves deterministically in the same direction as SARSA moves in expectation.
- Thanks to its more stable updates, Expected SARSA can handle higher values of α

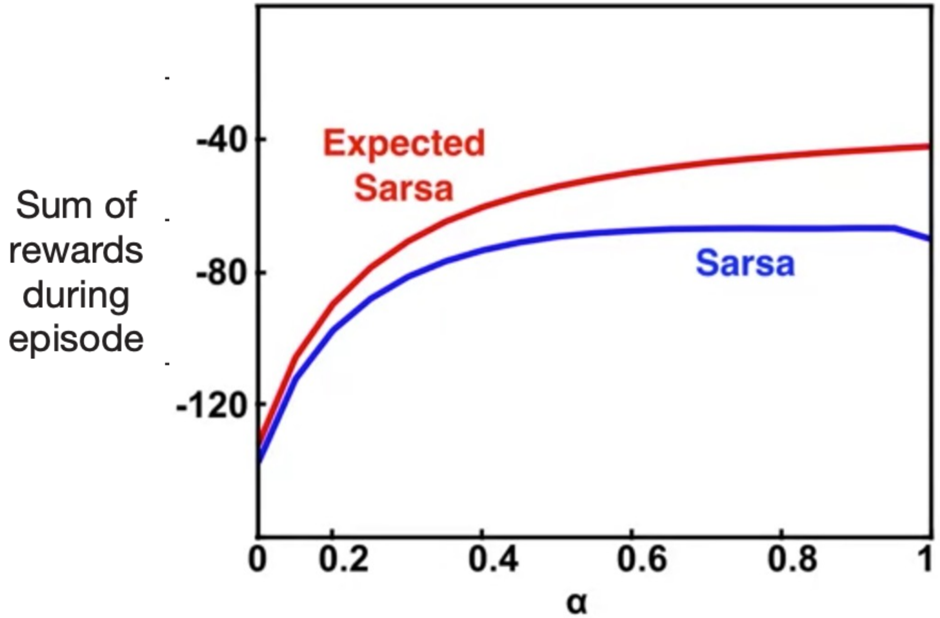


Q-learning



Expected Sarsa





Control: TD control and Bellman equations

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

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

$\downarrow$ $\downarrow$ $\downarrow$

Sarsa

$$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$$

Expected Sarsa

$$R_{t+1} + \gamma \sum_{a'} \pi(a'|S_{t+1}) Q(S_{t+1}, a')$$

Q-learning

$$R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$$

A. White, M. White 'Sample-based Learning Methods'

Control: TD control and Bellman equations

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

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

Sarsa

$$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$$

On-policy

Expected Sarsa

$$R_{t+1} + \gamma \sum_{a'} \pi(a'|S_{t+1}) Q(S_{t+1}, a')$$

?

Q-learning

$$R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$$

Off-policy

A. White, M. White 'Sample-based Learning Methods'

Control: TD control and Bellman equations

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

$$q_*(s,a) = \sum_{s',r} p(s',r \mid s,a)\left(r + \gamma \max_{a'} q_*(s',a')\right)$$

Sarsa

$$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$$

On-policy

Expected Sarsa

$$R_{t+1} + \gamma \sum_{a'} \pi(a' \mid S_{t+1}) Q(S_{t+1}, a')$$

Off-policy

Q-learning

$$R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$$

Off-policy

You look ahead and average over potential next actions and then you update Q(s,a)!

A. White, M. White 'Sample-based Learning Methods'

Algorithm 15: Expected Sarsa

Input: policy π, positive integer *num_episodes*, small positive fraction α

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**

  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

You look ahead and average over potential next actions and then you update Q(s,a)

TD-learning methods: Exam

- All the content of Chapter 6 are Exam material, beside:

i. Section 6.8 can be considered optional
ii. Sections 6.7 can be skipped

- (Again!) Pay particular attention to the algorithms!

Credits

- Image of the course is taken from C. Mahoney 'Reinforcement Learning' https://towardsdatascience.com/reinforcement-learning-fda8ff535bb6

Thank you!
Questions?

Lecture #10
Temporal Difference Learning for Control

Gian Antonio Susto