RL Tutorial 3

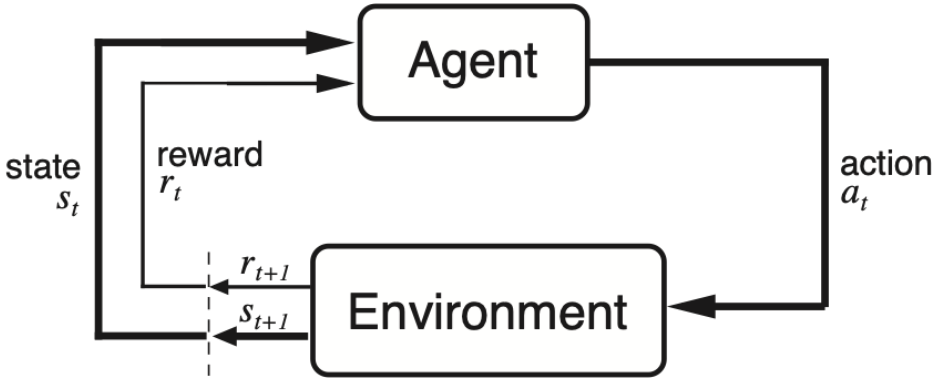
Marcelo Mattar

How do reinforcement learning agents act?

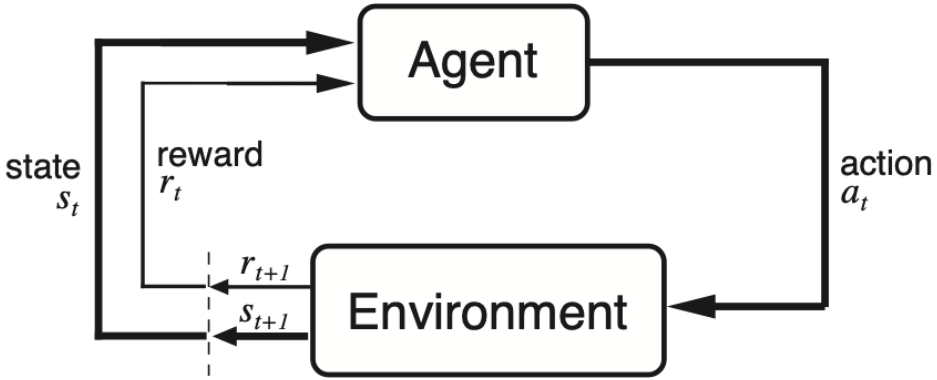
In the previous tutorial, we saw how agents learn to act in the bandit setting. In this tutorial, we will see how agents learn to act in the more general setting of Markov Decision Processes (MDPs).

- In bandits, the agent has no control over the state of the world.
- In MDPs, the agent's actions may influence the future states of the world

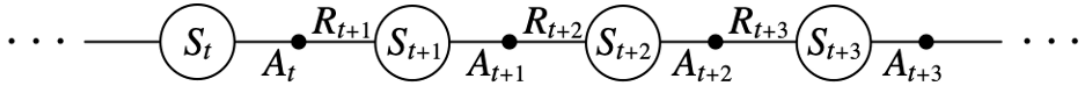
Why is reinforcement learning interesting?



Why is reinforcement learning interesting?



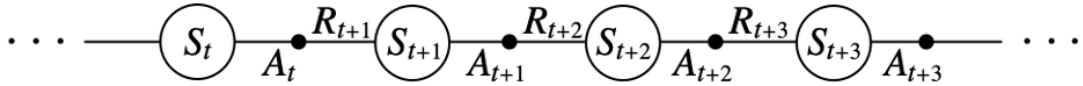
Choosing actions in MDPs



REMINDER: How are action values computed in bandit (one state) setting?

$$q(a) = \mathbb{E}\left[R_t \mid A_t = a\right]$$

Choosing actions in MDPs



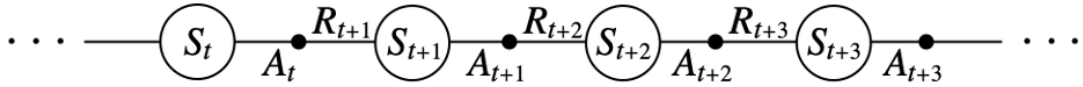
REMINDER: How are action values computed in bandit (one state) setting?

$$q(a) = \mathbb{E}\left[R_t \mid A_t = a\right]$$

NOW: How are action values computed in MDPs?

$$q(s, a) = \mathbb{E}\left[\sum_{k=1}^{\infty} R_{t+k} \mid S_t = s, A_t = a\right]$$

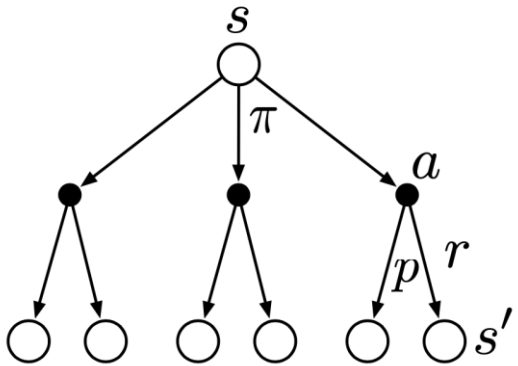
Computing action values



PROBLEM: Computing future reward can be difficult!

SOLUTION: Bellman optimality equation:

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\middle|\; S_t = s, A_t = a\right]$$



(i.e., use the value of the next state/action as a stand-in for future rewards)

Learning (optimal) action values: Q-learning

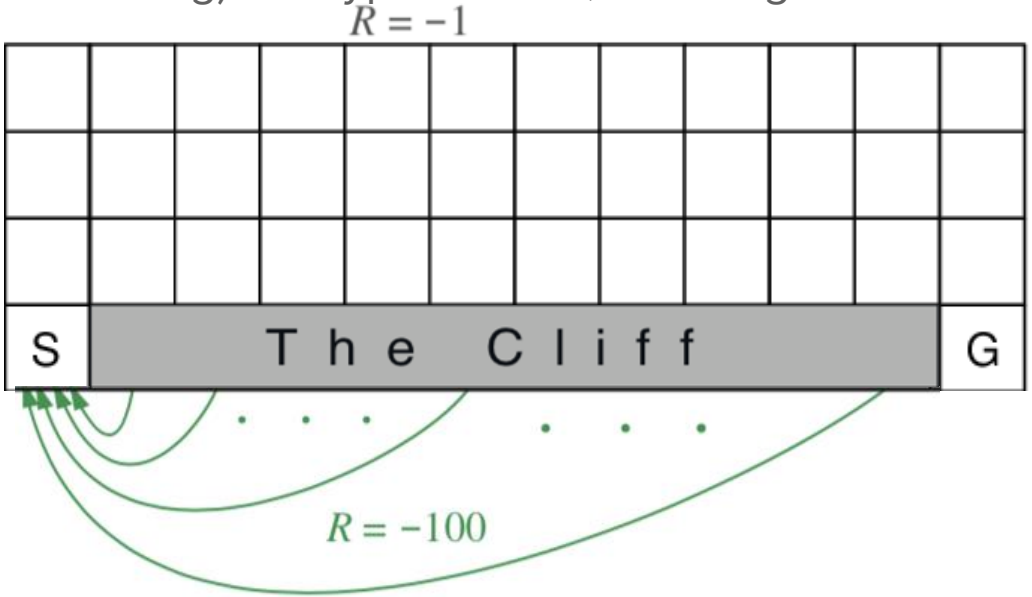
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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

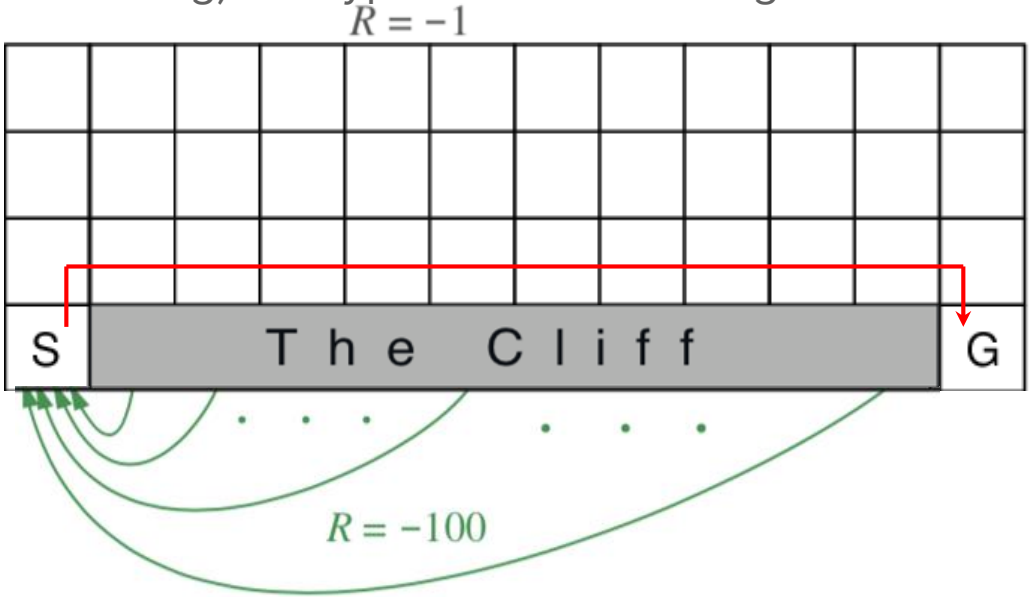
Exercise

In this exercise, you will implement an RL agent that learns how to act (using Q-learning) in a type of MDP, called "grid-world"



Exercise

In this exercise, you will implement an RL agent that learns how to act (using Q-learning) in a type of MDP called "grid-world"



On-policy vs. off-policy learning

When estimating the value of future states/actions:

- Off-policy methods estimate values that are ***not*** based on the agent's current policy. Most often, values are estimated under the assumption of optimal future behavior (i.e. optimal policy).
- Q-learning is an example of off-policy learning algorithm.

- On-policy methods estimate values under the agent's current policy, including possible randomness/stochasticity in future behavior (e.g. ε-greedy)
- SARSA is an example of on-policy learning algorithm

Learning (practical) action values: SARSA

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

Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Case study: accounting for future exploration