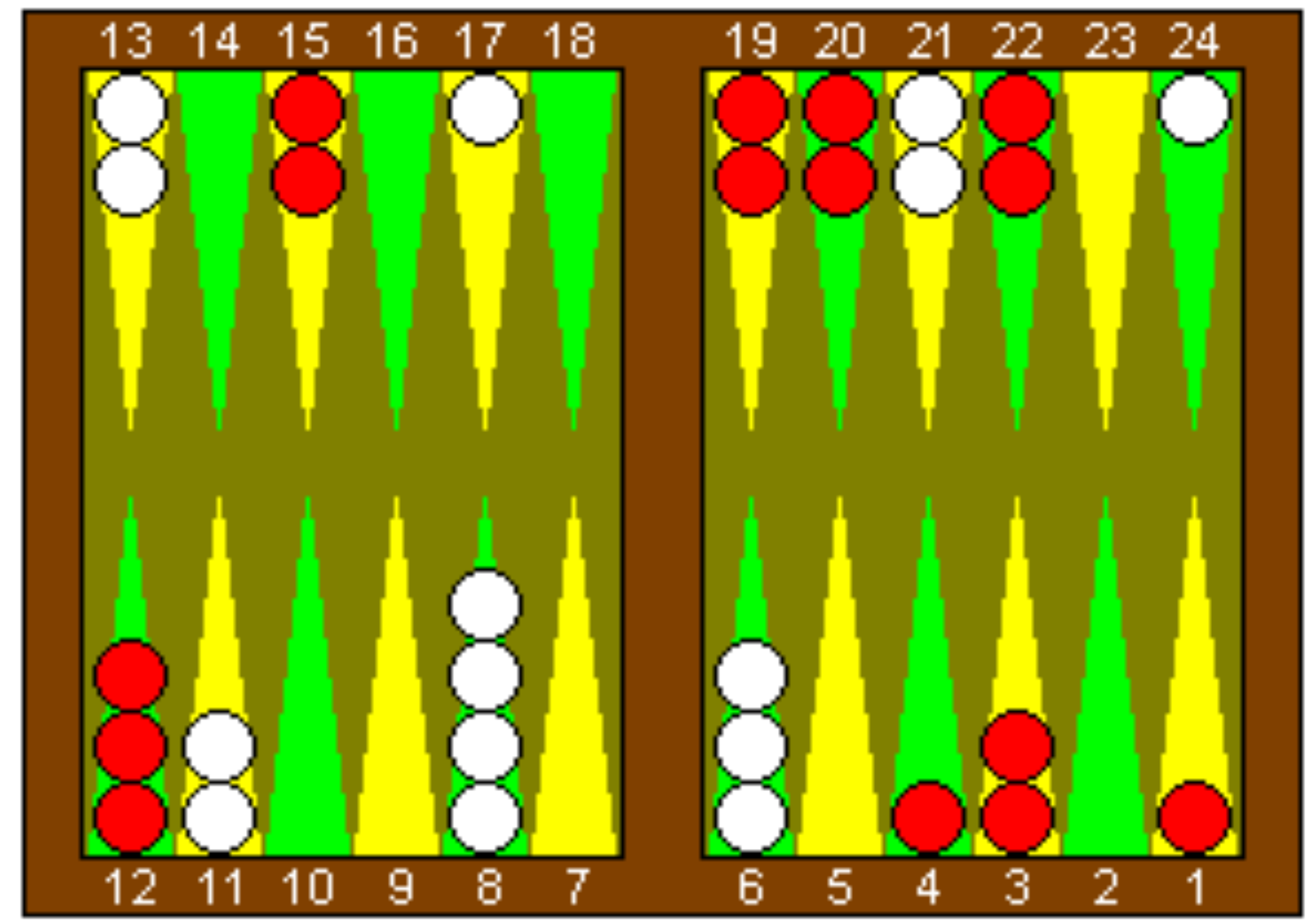
HSLU Lucerne University of Applied Sciences and Arts

Temporal Difference Learning



Information Technology
October 27, 2022

FH Zentralschweiz

Learning Objectives

- Understand how value functions are updated in Temporal Difference (TD) learning
- Understand SARSA and Q-learning for the control problem
- Understand the differences between those algorithms
- Understand how Q-learning is an off-policy algorithms without a model or importance sampling
- Understand the benefits (and drawbacks) of TD methods

Introduction

- Dynamic Programming (DP) methods required a model of the environment
- DP updated the value function after one time step
- Monte Carlo (MC) methods updated the value function only after the end of the episodes

- Can we combine an update after one step without using a model?

Temporal Difference Learning

- Temporal difference (TD) learning combines elements from Dynamic Programming (DP) and Monte Carlo (MC)
- TD learning is model free
- TD learning updates estimates without waiting for the final outcome (bootstrapping)

Prediction

- recall MC prediction with constant step size

$$V(S_t) \leftarrow V(S_t) + \alpha[\underbrace{G_t}_{\text{Target}} - V(S_t)]$$

- TD methods make the update immediately based on the estimates for the next state

$$V(S_t) \leftarrow V(S_t) + \alpha[\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{Target}} - V(S_t)]$$

General formulation

- More generally, the update can be written as

$$V(S_t) \leftarrow V(S_t) + \alpha\delta_t$$

- With the *error term*, defined as

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

- This is the error in the estimate made at the time. As it depends on the next state, it is only available at t+1

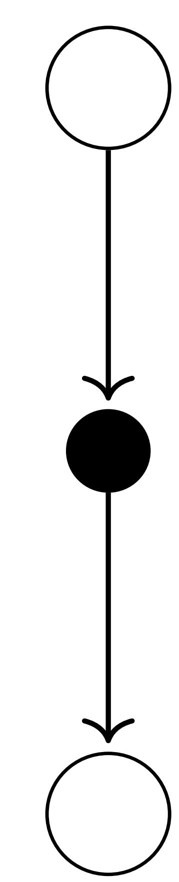
- This TD is known as TD(0), as the update takes place after 1 step

TD(0) Prediction

TD(0) for estimating v_π

Input:
  the policy π to be evaluated
  step size $\alpha \in (0, 1]$
Initialize:
  $V(S)$ arbitrarily (except $V(\text{terminal} = 0)$)

Loop for each episode:
  Initialize S
  Loop for each step of the episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S_t) \leftarrow V(S_t) + \alpha[R + \gamma V(S') - V(S)]$
    $S \leftarrow S'$
  until S is terminal



TD(0)

Example

How long does it take you to go home:

- Leave school
- Arrive at bus station
- Take bus
- Exit bus
- Go to bicycle
- Arrive home

Example

State	Elapsed time	Predicted time (from state)	Predicted time (total)
Leave school	0	45	45
Arrive bus station	5	50	55
Take bus	10	45	55
Exit bus	40	20	60
Go to bicycle	45	15	60
Arrive home	58	0	58

Rewards are the time elapsed; the value is the expected time to go home

Example

State	Elapsed time	Predicted time (from state)	Predicted time (total)	MC Error (alpha = 1.0)	TD Error
Leave school	0	45	45	58 - 45 = 13	55 - 45 = 10
Arrive bus station	5	50	55	58 - 55 = 3	55 - 55 = 0
Take bus	10	45	55	58 - 55 = 3	60 - 55 = 5
Exit bus	40	20	60	58 - 60 = -2	60 - 60 = 0
Go to bicycle	45	15	60	58 - 60 = -2	58 - 60 = -2
Arrive home	58	0	58	-	-

Advantages of TD

- Updates are based partly on existing estimates, this is called *bootstrapping*
- No model is required
- Updates are fully incremental and online. There is no need to wait for the end of the episode.
- Does it work? Yes, TD(0) has been shown to converge to the true v

Batch Updating

- If only a finite amount of experience is available, this is presented to the algorithm repeatedly until convergence.
- This is called batch updating.

SARSA: On-policy TD Control

- We use generalized policy iteration (GPI) for the control problem.
- As in MC methods, we must balance exploration and exploitation
- TD control methods generally learn an action-value function instead of a state-value function
- So we look at the transation from (**S**,**A**) with reward (**R**) to the next (**S**,**A**) (SARSA)

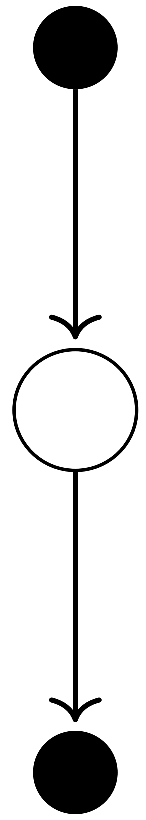
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

SARSA

Sarsa for estimating $Q \approx q_*$

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

Loop for each episode:
  Initialize S
  Choose A from S using a policy derived from Q (e.g., ϵ-greedy)
  Loop for each step of the episode:
    Take action A, observe R, S'
    Choose A' from S' using a 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



Q-learning: Off-policy TD Control

Q-learning is defined by

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

- The learned action-value function Q directly approximate q_* and is independent of the policy being followed.
- The policy determines which state-action pairs are visited and updated, so it must visit all pairs eventually

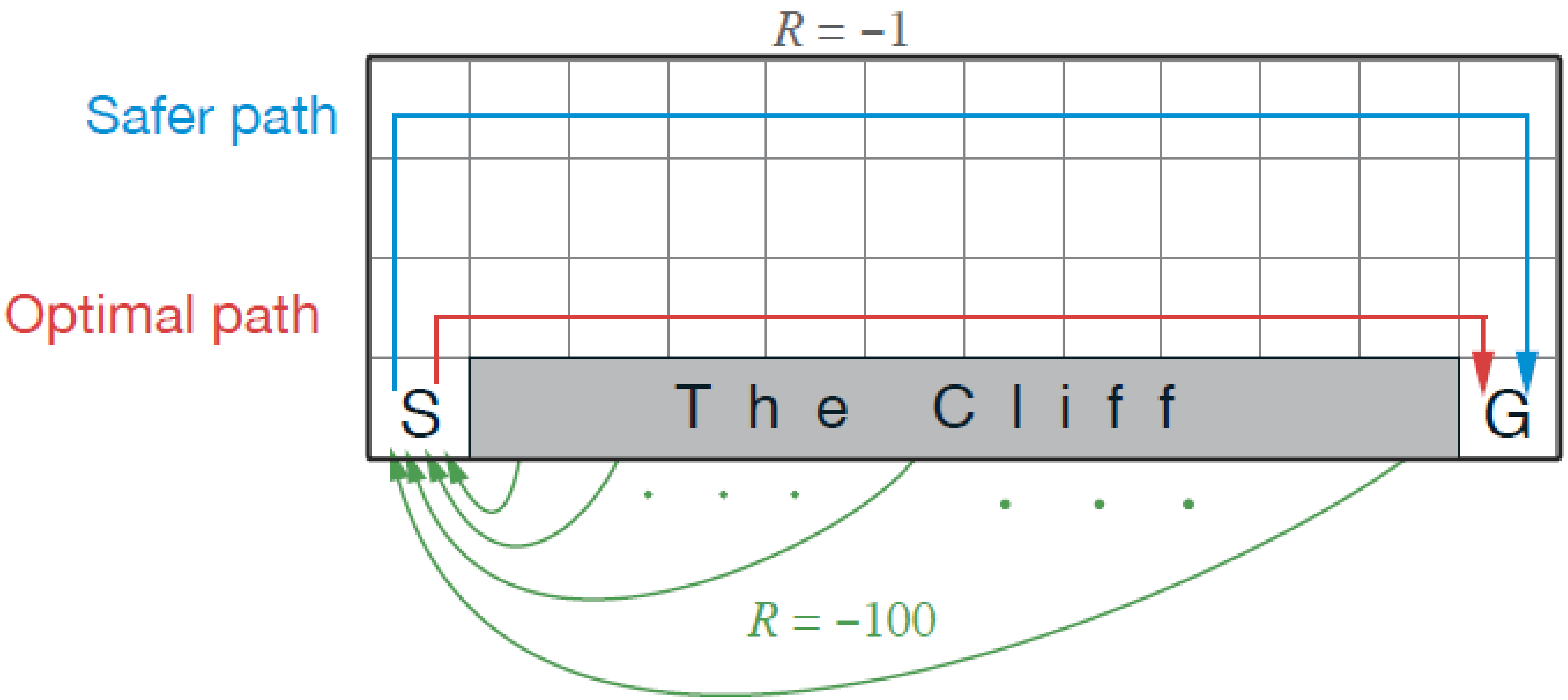
Q-Learning

Q-learning for estimating $Q \approx q_*$

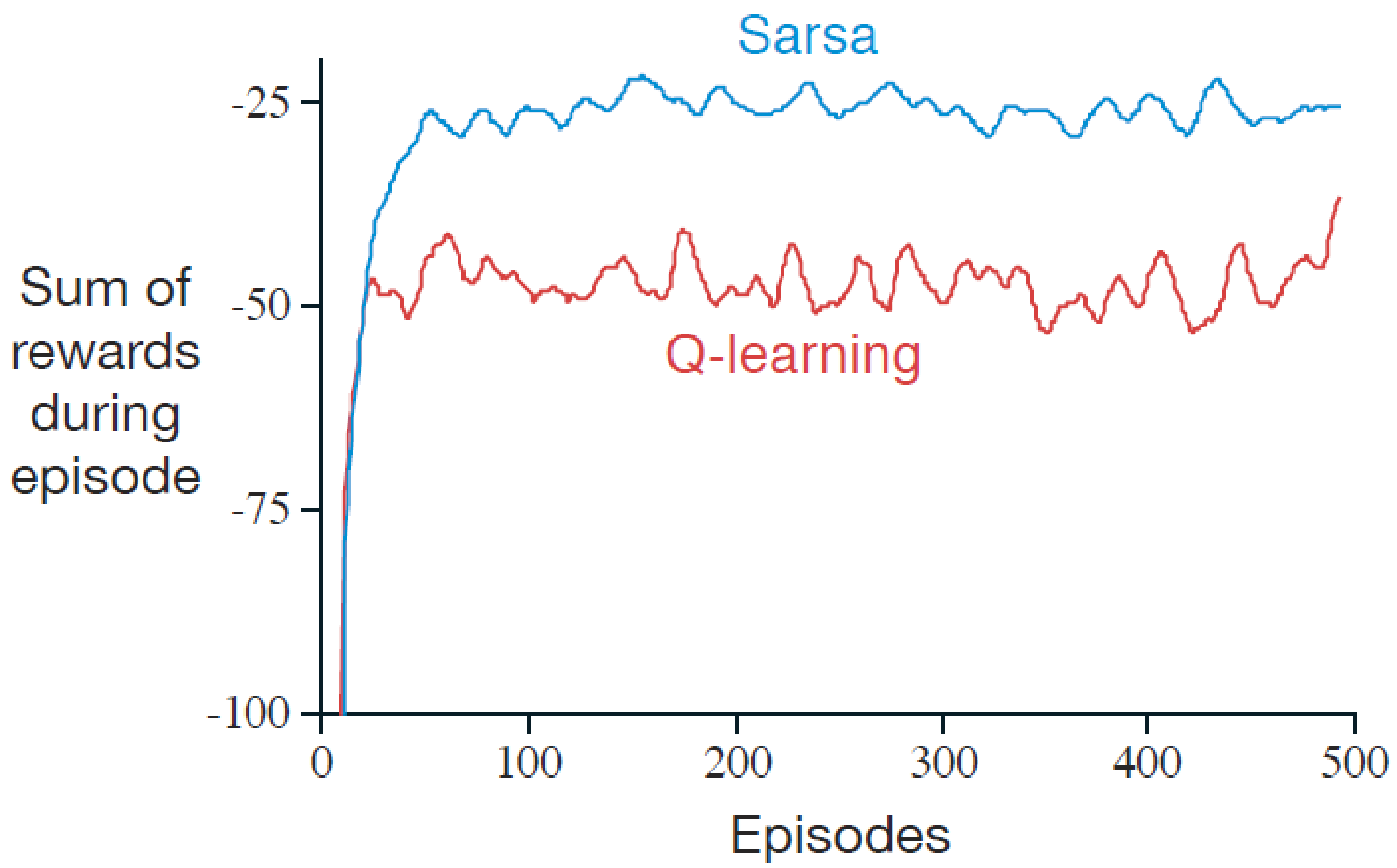
Input:
  step size $\alpha \in (0, 1]$
  small $\epsilon > 0$
Initialize:
  $Q(s, a)$ for all $s \in \mathcal{S}^+, a \in \mathcal{A}$ arbitrarily (except $Q(\text{terminal}, \cdot) = 0$)

Loop for each episode:
  Initialize S
  Loop for each step of the episode:
    Choose A from S using a 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

Example



Example



Sarsa
Q-learning
Sum of rewards during episode
-25
-50
-75
-100
0
100
200
300
400
500
Episodes

Expected SARSA

Bellman equation

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

Sarsa:

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

sampled from

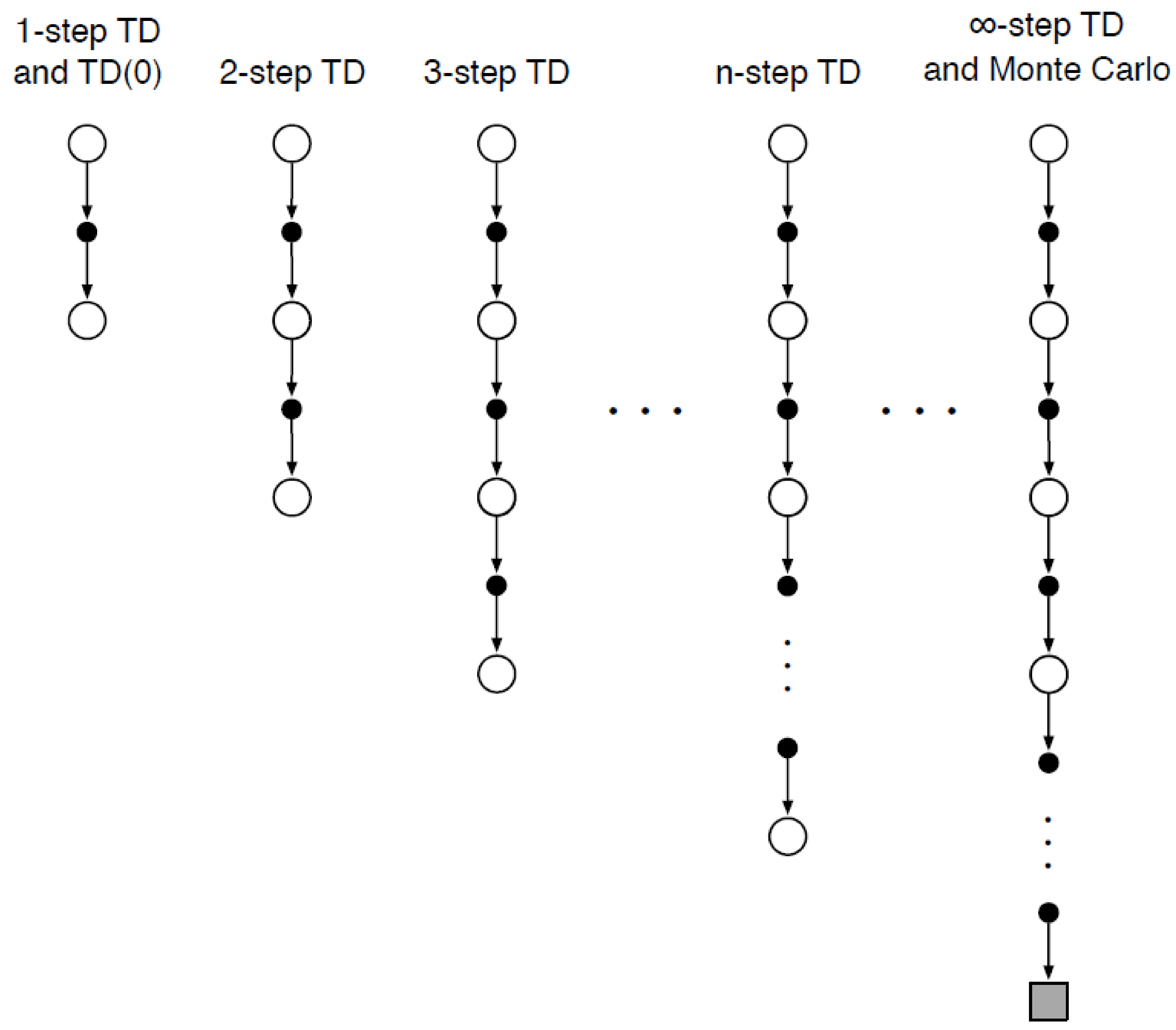
$$\pi(a'|s')$$

$$p(s', r \mid s, a)$$

Expected SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[\underbrace{R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, A_{t+1})}_{\text{Expected value of the return}} - Q(S_t, A_t)]$$

n-step TD Prediction



Targets of n-step returns

$$G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1})$$

$$G_{t:t+2} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2})$$

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} + \gamma^n V_{t+n-1}(S_{t+n})$$

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha[G_{t:t+n} - V_{t+n-1}(S_t)]$$

Summary

- TD(0) methods allow updating value estimates after only 1 step
- New estimates are therefor based on the estimates of the successor states, this is called bootstrapping
- SARSA is an on-policy control algorithm that uses TD(0) as prediction step with an ε-soft policy
- Q-Learning is an off-policy control algorithm that uses TD(0) as prediction step and learns a greedy policy while following an ε-soft policy
- Expected SARSA improves SARSA by summing over the possible actions from the policy