Part 5: Q-learning and SARSA algorithms

EL 2805 – Reinforcement Learning

Alexandre Proutiere

KTH, The Royal Institute of Technology

Objectives of this part

First RL algorithms for optimal control

- Randomized policies
- Monte Carlo control algorithm
- Q-learning algorithm: off-policy optimal control
- SARSA algorithm: on-policy optimal control

References

- Barto-Sutton's book: chapters 5 and 6
- Q-learning: Watkins-Dayan, Q-leanring, Machine Learning (1992)
- SARSA: Rummery-Niranjan, On-line Q-learning using connectionist systems (1994)

Outline

0. Randomized policies
1. Algorithm summary (Q-learning and SARSA)
2. Monte Carlo control algorithm
3. Q-learning algorithm
4. SARSA algorithm

0. Randomized policies

Randomized policies are necessary in RL to ensure exploration.

Under π, in state s, the action a is selected w.p. $\pi(s,a)$.
The value function V^π of π satisfies:

$$\forall s \in \mathcal{S},\ V^\pi(s) = \mathbb{E}[\sum_{t\geq 1} \lambda^{t-1} r(s_t^\pi, a_t^\pi) | s_1^\pi = s]$$

$$= \sum_a \pi(s,a) \left(r(s,a) + \lambda \sum_j p(j|s,a) V^\pi(j) \right)$$

Its (state,action) value function Q^π satisfies:

$$\forall s \in \mathcal{S}, a \in A_s,\ Q^\pi(s,a) = r(s,a) + \lambda \sum_j p(j|s,a) V^\pi(j)$$

1. Algorithm summary

MC control with ϵ-soft policies (on-policy algorithm)

Monte Carlo for ϵ-soft policies:

1. **Initialization:** π ϵ-soft, $\forall s, a$, $Q(s,a) = 0$
   Returns$(s,a) \leftarrow$ empty list, $\forall s, a$
2. **Iterations:** for episode $i = 1, \ldots, n$
   generate $\tau_i = (s_{1,i}, a_{1,i}, r_{1,i}, \ldots, s_{T_i,i}, a_{T_i,i}, r_{T_i,i})$ under π
   $G = 0$
   for $t = T_i, T_i - 1, \ldots, 1$:
   a. $G = r_{t,i} + \lambda G$
   b. Unless $(s_{t,i}, a_{t,i})$ appears before in the episode:
   append G to Returns$(s_{t,i}, a_{t,i})$
   $Q(s_{t,i}, a_{t,i}) \leftarrow$ average(Returns$(s_{t,i}, a_{t,i})$)
   update π: $a \sim \pi(s_{t,i}, \cdot)$ such that

$$a = \begin{cases} \arg\max_b Q(s_{t,i}, b) & \text{w.p. } 1 - \epsilon, \\ \text{uniform}(\mathcal{A}_{s_{t,i}}) & \text{w.p. } \epsilon \end{cases}$$

Q-learning algorithm

Parameter. Step sizes (α_t)

1. **Initialization.** Select a Q-function $Q^{(0)} \in \mathbb{R}^{S \times A}$
2. **Observations.** (s_t, a_t, r_t, s_{t+1}) under the behavior policy π_b
3. Q**-function improvement.** For $t \geq 0$. Update the estimated Q-function as follows: $\forall s, a$,

$$Q^{(t+1)}(s,a) = Q^{(t)}(s,a)$$
$$+ 1_{(s_t,a_t)=(s,a)} \alpha_{n^{(t)}(s_t,a_t)} \left[r_t + \lambda \max_{b \in \mathcal{A}} Q^{(t)}(s_{t+1}, b) - Q^{(t)}(s_t, a_t) \right]$$

where $n^{(t)}(s,a) := \sum_{m=1}^{t} 1[(s,a) = (s_m, a_m)]$.

SARSA with ϵ-greedy

Parameter. Step sizes (α_t)

1. Initialization. Select a Q-function $Q^{(0)} \in \mathbb{R}^{S \times A}$

2. Observations. $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ under π_t ϵ-greedy w.r.t. $Q^{(t)}$

3. Q-function improvement. For $t \geq 0$. Update the estimated Q-function as follows: $\forall s, a$,

$$Q^{(t+1)}(s,a) = Q^{(t)}(s,a) + 1_{(s_t,a_t)=(s,a)} \alpha_{n^{(t)}(s_t,a_t)} \left[r_t + \lambda Q^{(t)}(s_{t+1}, a_{t+1}) - Q^{(t)}(s_t, a_t) \right]$$

where $n^{(t)}(s,a) := \sum_{m=1}^{t} 1[(s,a) = (s_m, a_m)]$.

Example: Controlled random walk

A discounted RL problem. When reaching borders, restart in state C.

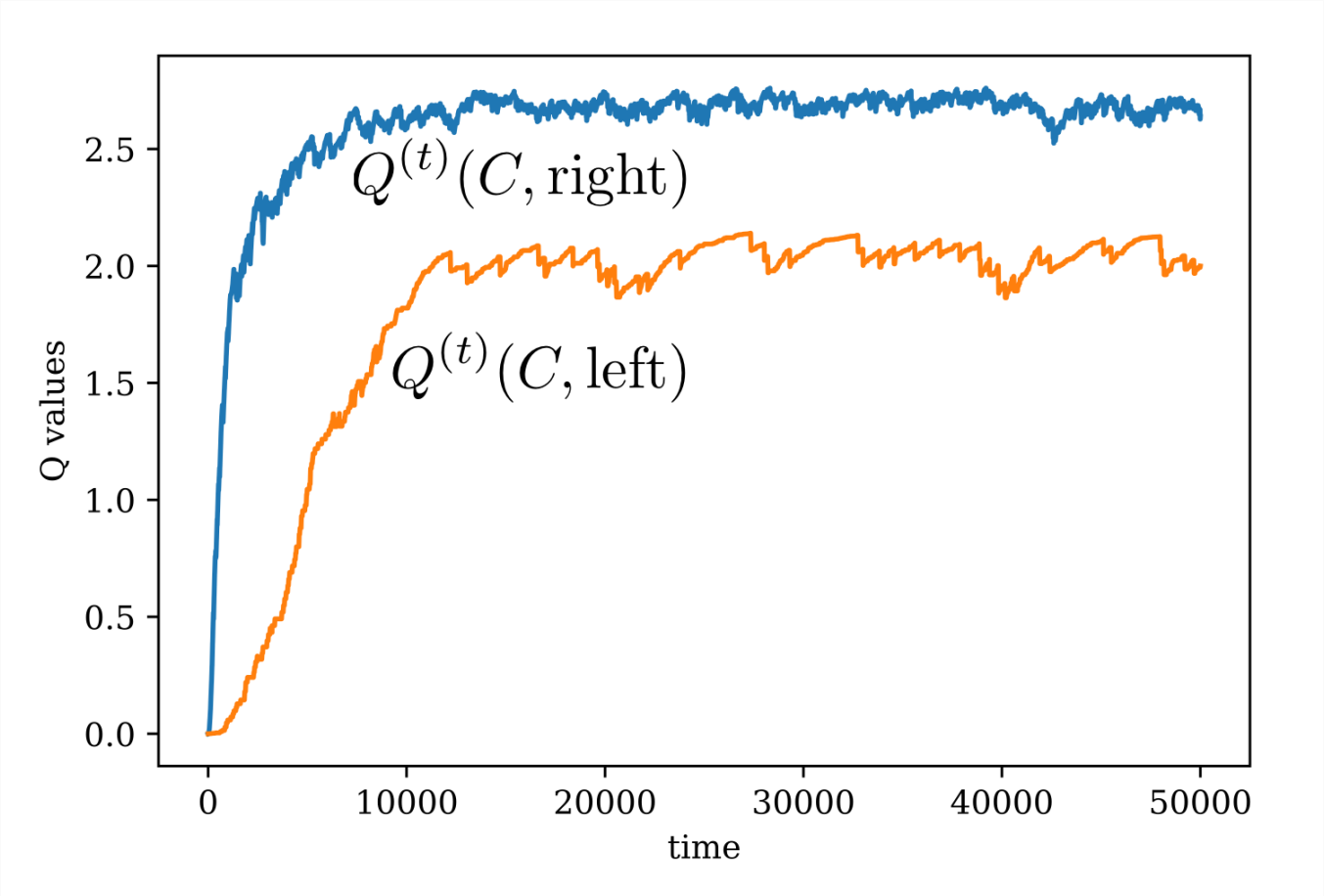
Discount factor $\lambda = 0.9$.

Reward collected only in the transition from E to C.

Initial random policy (go left w.p. 0.5)

Example: SARSA

Exploration rate $\epsilon = 0.1 + 0.5/\text{time}$ and constant step-sizes $= 0.04$.

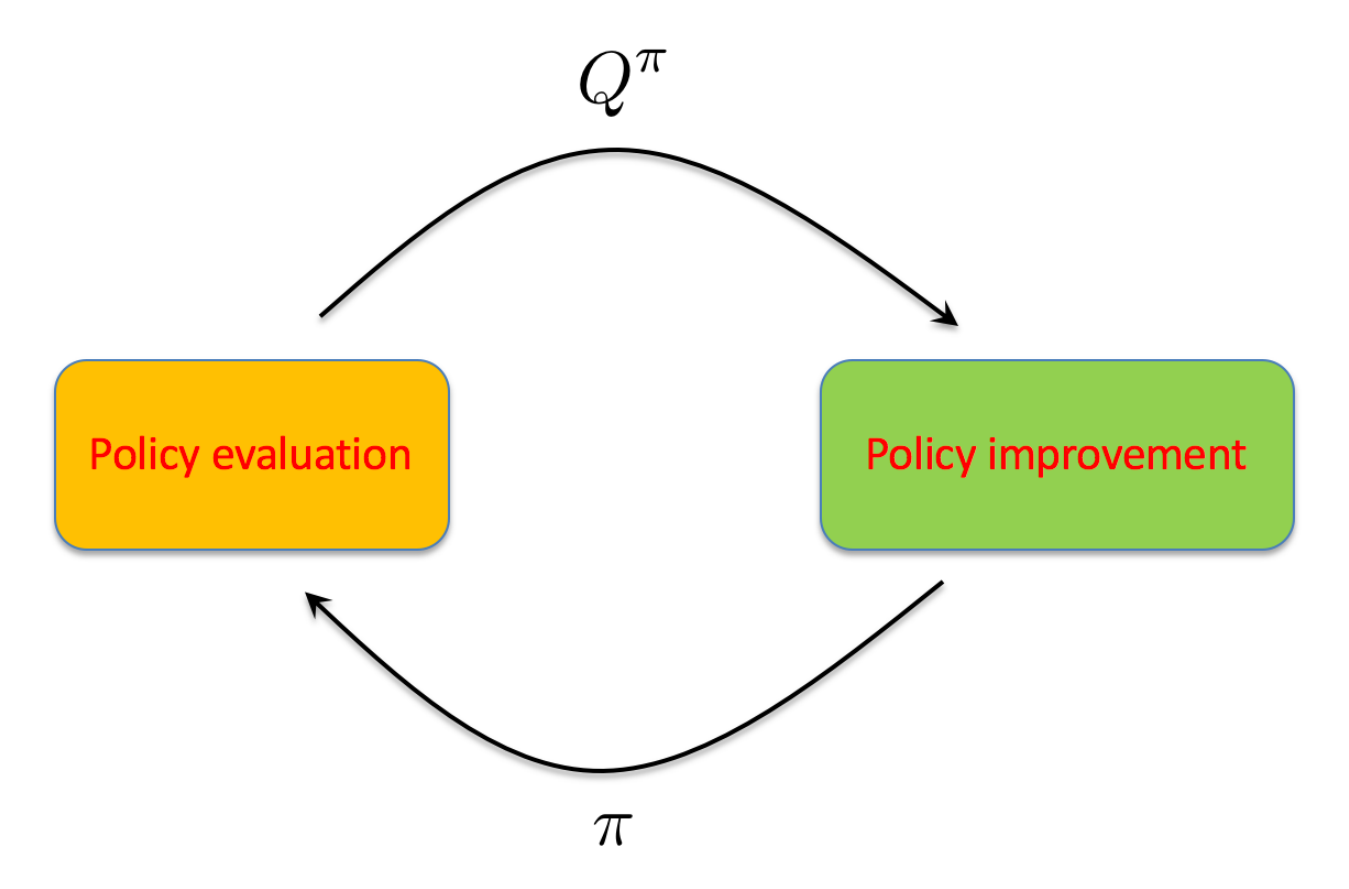


2. Monte Carlo control

Discounted MDP with terminal state

- MDP: $(\lambda, S, A_s, p(\cdot|s,a), r(s,a), a \in A_s, s \in S)$.
- **Terminal state:** There is a state s_{end} after which no reward is collected.
- **Episode:** It starts at time $t = 1$ in state s_1 and finishes after a random time when s_{end} is reached.
- Assumption: under any policy, episodes finish in finite time almost surely.

Monte Carlo control



Use the Policy Iteration algorithm, replacing the policy evaluation step by its Monte Carlo equivalent.

Monte Carlo control

PI algorithm.

1. Policy evaluation: $\pi^{(k)} \to Q^{\pi^{(k)}}$

where the (state, action) value function $Q^{\pi^{(k)}}$ is defined as

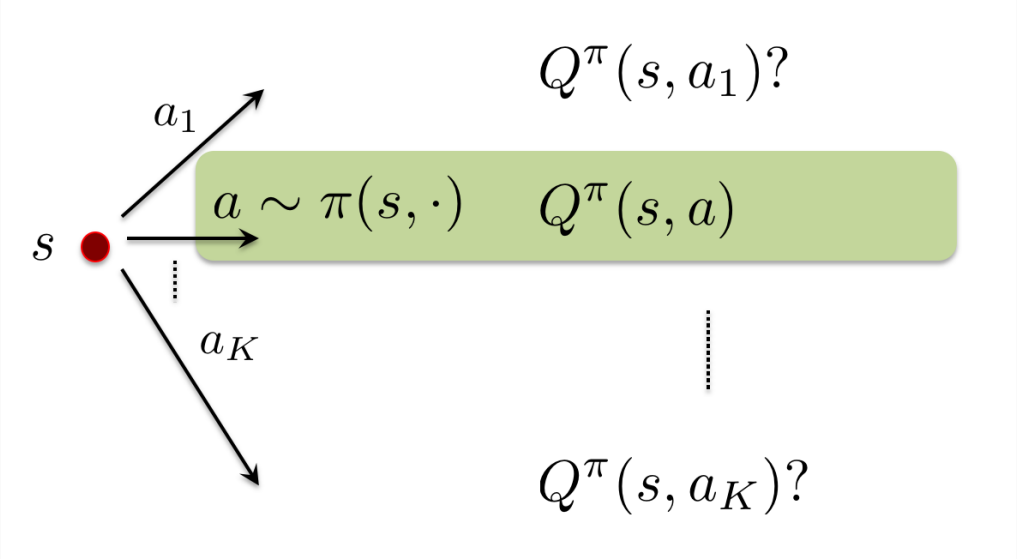
$$\forall s, a, \quad Q^{\pi^{(k)}}(s,a) = r(s,a) + \lambda \sum_j p(j|s,a) V^{\pi^{(k)}}(j)$$

2. Policy improvement: for all s, $\pi^{(k+1)}(s) \in \arg\max_{a \in A_s} Q^{\pi^{(k)}}(s,a)$

MC policy evaluation. (For Step 1.)

Run episodes under $\pi^{(k)}$ and collect n returns $(G_i)_{i=1,\dots,n}$ for the first-visits at (s,a) to estimate $Q^{\pi^{(k)}}(s,a) \approx \frac{1}{n} \sum_{i=1}^{n} G_i$.

Monte Carlo evaluation of the (state, action) value function



Generating episodes under π provides estimates of $V^\pi(s) = Q^\pi(s, a)$ where $a \sim \pi(s, \cdot)$ only, not of $Q^{\pi^{(k)}}(s, a)$ for all a.

Solution 1. Exploring starts. Initially select a random (state, action) pair, and from there, run π after the first step

Solution 2. ϵ-soft policies. Restrict the attention to ϵ-soft policies: when in state s, w.p. ϵ, use a random action, and w.p. $1 - \epsilon$ use the base policy

Monte Carlo control with Exploring Starts

Monte Carlo ES:

1. **Initialization:** π, $\forall s, a$, $Q(s, a) = 0$
   Returns$(s, a) \leftarrow$ empty list, for all s, a
2. **Iterations:** for episode $i = 1, \ldots, n$
   select $(s_{1,i}, a_{1,i})$ uniformly at random
   generate $\tau_i = (s_{1,i}, a_{1,i}, r_{1,i}, \ldots, s_{T_i,i}, a_{T_i,i}, r_{T_i,i})$ under π
   $G = 0$
   for $t = T_i, T_i - 1, \ldots, 1$:
   a. $G = r_{t,i} + \lambda G$
   b. Unless $(s_{t,i}, a_{t,i})$ appears before in the episode:
   append G to Returns$(s_{t,i}, a_{t,i})$
   $Q(s_{t,i}, a_{t,i}) \leftarrow$ average(Returns$(s_{t,i}, a_{t,i})$)
   $\pi(s_{t,i}) \leftarrow \arg\max_a Q(s_{t,i}, a)$

Monte Carlo control for ϵ-soft policies

Monte Carlo for ϵ-soft policies:

1. **Initialization:** π ϵ-soft, $\forall s, a$, $Q(s,a) = 0$
   Returns$(s,a) \leftarrow$ empty list, for all s, a
2. **Iterations:** for episode $i = 1, \ldots, n$
   generate $\tau_i = (s_{1,i}, a_{1,i}, r_{1,i}, \ldots, s_{T_i,i}, a_{T_i,i}, r_{T_i,i})$ under π
   $G = 0$
   for $t = T_i, T_i - 1, \ldots, 1$:
   a. $G = r_{t,i} + \lambda G$
   b. Unless $(s_{t,i}, a_{t,i})$ appears before in the episode:
      append G to Returns$(s_{t,i}, a_{t,i})$
      $Q(s_{t,i}, a_{t,i}) \leftarrow$ average(Returns$(s_{t,i}, a_{t,i})$)
      update π: $a \sim \pi(s_{t,i}, \cdot)$ such that

$$a = \begin{cases} \arg\max_b Q(s_{t,i}, b) & \text{w.p. } 1 - \epsilon, \\ \text{uniform} & \text{w.p. } \epsilon \end{cases}$$

Policy improvement theorem

Theorem 1. Let π be an ϵ-soft policy. If π' is an ϵ-greedy policy w.r.t. Q^π. Then for all $s \in \mathcal{S}$, $V^\pi(s) \leq V^{\pi'}(s)$.

Lemma. For all $s \in \mathcal{S}$, $V^\pi(s) \leq \sum_a \pi'(s,a) Q^\pi(s,a)$. Or written differently:

$$V^\pi(s) \leq \mathbb{E}_{\pi'}[Q^\pi(s_1, a) | s_1 = s]$$

Theorem 2. Let π be an ϵ-soft policy. If π' is an ϵ-greedy policy w.r.t. Q^π. If $V^\pi = V^{\pi'}$, then π is an optimal ϵ-soft policy.

The MC control algorithm with ϵ-soft policies converges to an optimal ϵ-soft policy.

Proof of the lemma

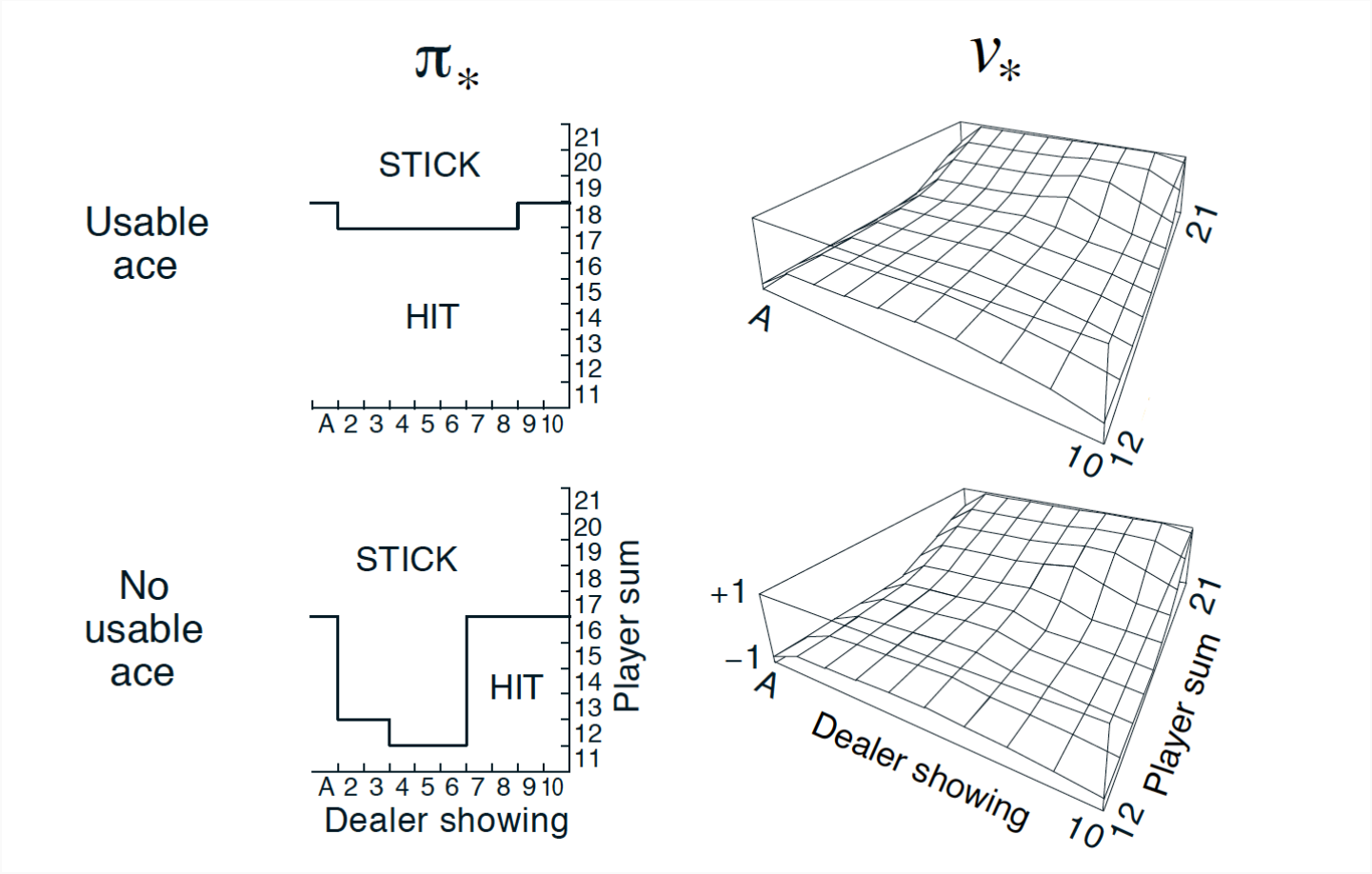
$$\begin{aligned}
\sum_a \pi'(s,a) Q^\pi(s,a) &= \frac{\epsilon}{|A_s|} \sum_a Q^\pi(s,a) + (1-\epsilon) \max_a Q^\pi(s,a) \\
&\geq \frac{\epsilon}{|A_s|} \sum_a Q^\pi(s,a) + (1-\epsilon) \sum_a \frac{\pi(s,a) - \frac{\epsilon}{|A_s|}}{1-\epsilon} Q^\pi(s,a) \\
&= \frac{\epsilon}{|A_s|} \sum_a Q^\pi(s,a) + \sum_a \pi(s,a) Q^\pi(s,a) - \frac{\epsilon}{|A_s|} \sum_a Q^\pi(s,a) \\
&= V^\pi(s)
\end{aligned}$$

Proof of Theorem 1

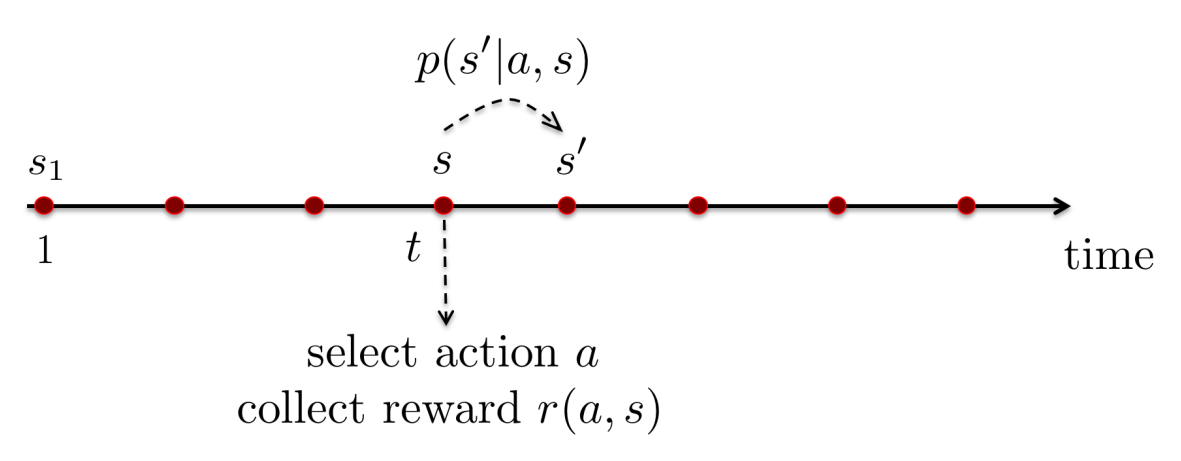
$$
\begin{aligned}
V^{\pi}(s) &\leq \mathbb{E}_{\pi'}[Q^{\pi}(s_1, a)|s_1 = s] \\
&= \mathbb{E}_{\pi'}[r_1 + \lambda V^{\pi}(s_2)|s_1 = s] \\
&\leq \mathbb{E}_{\pi'}[r_1 + \lambda \mathbb{E}_{\pi'}[Q^{\pi}(s_2, a_2)]|s_1 = s] \\
&= \mathbb{E}_{\pi'}[r_1 + \lambda r_2 + \lambda^2 V^{\pi}(s_3)|s_1 = s] \\
&\dots \\
&\leq \mathbb{E}_{\pi'}[r_1 + \lambda r_2 + \lambda^2 r^3 + \dots |s_1 = s] \\
&= V^{\pi'}(s)
\end{aligned}
$$

Example: Blackjack

The optimal policy and its value function obtained using MC control algorithm:



3. Q-learning



Discounted RL problems:

- **Unknown** stationary transition probabilities $p(s'|s,a)$ and rewards $r(s,a)$, uniformly bounded: $\forall a, s,\ |r(s,a)| \leq 1$
- Objective: for a given discount factor $\lambda \in [0,1)$, from the data, find a policy $\pi^\star \in MD$ maximizing (over all possible policies)

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{T} \lambda^{t-1} r(s_t^\pi, a_t^\pi,)|s_1^\pi = s\right]$$

Bellman's equations

Value function and optimal policy: $V^\star(s) = \sup_{\pi \in MD} V^\pi(s)$ obtained by solving Bellman's equations (through VI or PI algorithm):

$$\forall s, \quad V^\star(s) = \max_{a \in A_s} \underbrace{\left[r(s,a) + \lambda \sum_{j \in S} p(j|s,a) V^\star(j) \right]}_{Q(s,a) \text{ optimal reward from state } s \text{ if } a \text{ selected}}$$

An optimal policy π is stationary $\pi = (\pi_1, \pi_1, \ldots)$ where $\pi_1 \in MD_1$ is defined by: for any s,

$$\pi_1(s) = \arg\max_{a \in A_s} Q(s,a)$$

Q is referred to as the Q-function.

Bellman's operator

(Non-linear) **Bellman operator** $\mathcal{L} : \mathcal{V} \to \mathcal{V}$ defined by:
for all $V \in \mathcal{V}$, $\mathcal{L}(V) = \sup_{\pi_1 \in MD_1} (r_{\pi_1} + \lambda P_{\pi_1} V)$ or equivalently by

$$\forall s \in S,\ \mathcal{L}(V)(s) = \max_{a \in A_s} \left[r(s,a) + \lambda \sum_j p(j|s,a) V(j) \right]$$

The value function is the unique fixed point of Bellman's operator:
$\mathcal{L}(V^\star) = V^\star$.

Q-learning algorithm

- Off-policy learning: we observe a trajectory of the system under the behavior policy π_b:
  At step $t \geq 1$: (s_t, a_t, r_t, s_{t+1}) state, action, reward, next state

- Can we apply R-M stochastic approximation algorithm to find the root of $h(V) = \mathcal{L}(V) - V$?
  (Reminder. R-M algorithm: $V^{(t+1)} = V^{(t)} + \alpha_t(h(V^{(t)}) + M_t)$)

Why not V-learning?

- At step t, assume we have an estimate $V^{(t)}$ of $V^\star$.
  Asynchronous R-M algorithm:

$$V^{(t+1)}(s) = V^{(t)}(s) + 1_{\{s_t=s\}}\alpha_{n_s^{(t)}}(\mathcal{L}(V^{(t)})(s) - V^{(t)}(s) + M_t)$$

  where $\mathbb{E}[M_t] = 0$, and where $n_s^{(t)}$ is the number of visits of state s up to t.

- We can apply an asynchronous R-M algorithm to estimate $V^\star$ provided that from the trajectory

$$(s_1, a_1, r_1, \ldots, s_t, a_t, r_t, s_{t+1})$$

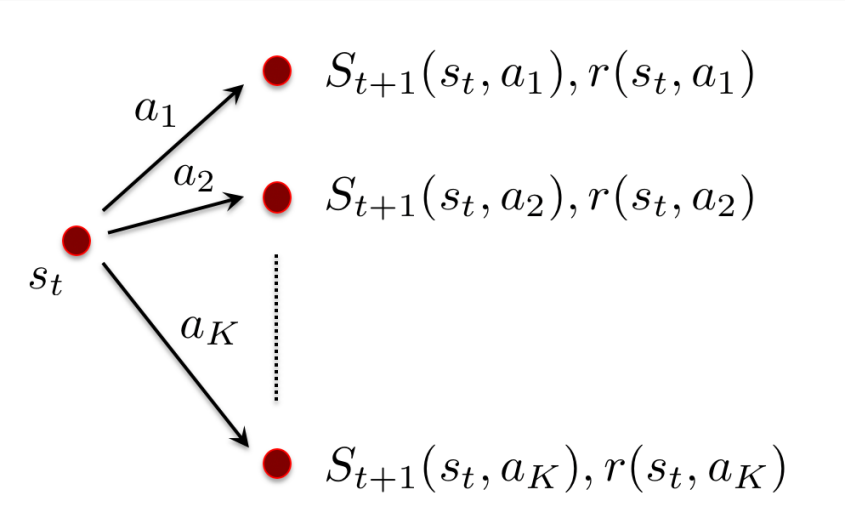
  we can build an unbiased estimator of

$$\mathcal{L}(V^{(t)})(s_t) - V^{(t)}(s_t) = \max_{a \in A_{s_t}} (r(s_t, a) + \lambda \sum_j p(j|s_t, a)V^{(t)}(j) - V^{(t)}(s_t).$$

Why not V-learning?

- At step t, assume that we can try all actions a and observe the new state $S_{t+1}(s_t, a)$ a r.v. such that $\mathbb{P}[S_{t+1} = j | s_t, a] = p(j|s_t, a)$. We can compute

$$y_t = \max_{a \in A_{s_t}} (r(s_t, a) + \lambda V^{(t)}(S_{t+1}(s_t, a))) - V^{(t)}(s_t)$$



Why not V-learning?

To apply the R-M algorithm to evaluate $V^\star$, we would need that $\mathbb{E}[y_t|s_t] = \mathcal{L}(V^{(t)})(s_t) - V^{(t)}(s_t)$. However:

$$\begin{aligned}\mathcal{L}(V^{(t)})(s_t) - V^{(t)}(s_t) &= \max_{a \in A_{s_t}} (r(s_t, a) + \lambda \sum_j p(j|s_t, a) V^{(t)}(j)) - V^{(t)}(s_t) \\ &= \max_{a \in A_{s_t}} (r(s_t, a) + \lambda \mathbb{E}[V^{(t)}(S_{t+1}(s_t, a))|s_t]) - V^{(t)}(s_t)\end{aligned}$$

whereas

$$\mathbb{E}[y_t|s_t] = \mathbb{E}[\max_{a \in A_{s_t}} (r(s_t, a) + \lambda V^{(t)}(S_{t+1}(s_t, a)))|s_t] - V^{(t)}(s_t)$$

and $\mathbb{E}[\max(X, Y)] \neq \max(\mathbb{E}[X], \mathbb{E}[Y])$.

V-learning is not possible.

Remark: extension to random rewards

The above problem remains when rewards are random.

$r(s_t, a_t)$ is a r.v. with mean $\mu(s_t, a_t)$.
Bellman's equations:

$$\forall s \in S,\ V^\star(s) = \max_{a \in A_s} \left[\mu(s,a) + \lambda \sum_j p(j|s,a) V^\star(j) \right]$$

Observations: (additional randomness in rewards)

$$y_t = \max_{a \in A_{s_t}} \left(r(s_t, a) + \lambda V^{(t)}(S_{t+1}(s_t, a)) \right) - V^{(t)}(s_t)$$

Example: bandit optimization

No state, two actions a_1, a_2 with Bernoulli rewards of respective means $\mu_1 > \mu_2$.
The optimal policy is to select action a_1 (always).
Value function: $V^\star = \mu_1 + \lambda\mu_1 + \lambda^2\mu_1 + \ldots = \frac{\mu_1}{1-\lambda}$.

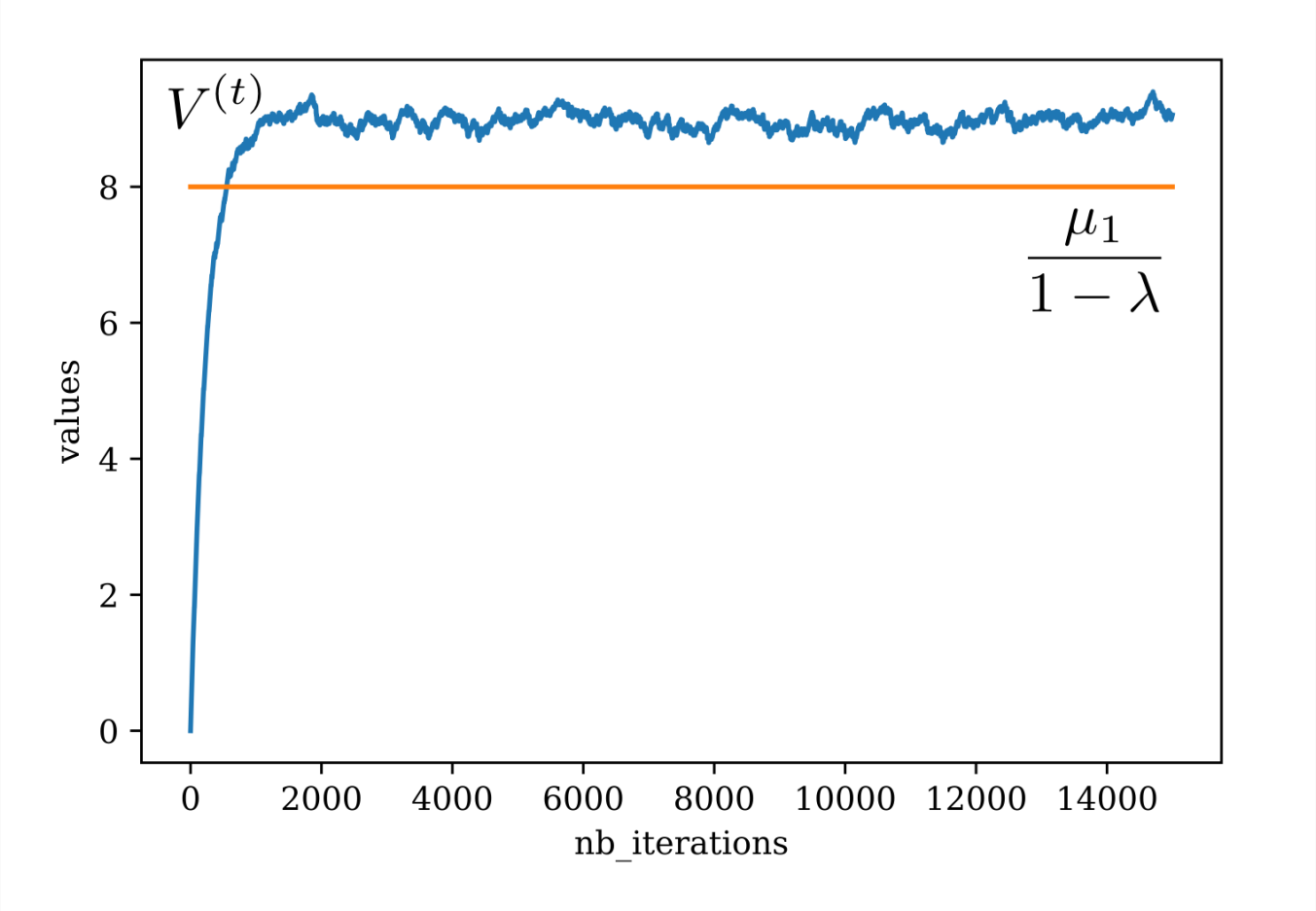
In each round t, we can try each action and observe the random rewards $R_{1,t}$ and $R_{2,t}$.

$$y_t = \max\{R_{1,t}, R_{2,t}\} + (\lambda - 1)V^{(t)}, \quad \mathbb{E}[y_t] = \mu_1 + \mu_2 - \mu_1\mu_2 + (\lambda - 1)V^{(t)}$$

The R-M algorithm $V^{(t+1)} = V^{(t)} - \alpha_t y_t$ converges to $\frac{\mu_1 + \mu_2 - \mu_1\mu_2}{1-\lambda}$.

Example: bandit optimization

$\mu_1 = 0.8$, $\mu_2 = 0.5$, $\lambda = 0.9$. Step sizes $= 0.04$.



Q-function

We apply the R-M algorithm to the estimate the Q-function instead. $Q(s,a)$ is the maximum expected reward starting from state s and taking action a:

$$Q(s,a) = r(s,a) + \lambda \sum_j p(j|s,a) V^\star(j)$$

Note that $V^\star(s) = \max_{a \in A_s} Q(s,a)$, and hence

$$Q(s,a) = r(s,a) + \lambda \sum_j p(j|s,a) \max_b Q(j,b)$$

Q is the fixed point of an operator H (defined on $\mathbb{R}^{S \times A}$)

$$(HQ)(s,a) = r(s,a) + \lambda \sum_j p(j|s,a) \max_b Q(j,b)$$

Q-function and Stochastic Approximation

- At time t, we have an estimated Q-function, $Q^{(t)}$. The system is is state s_t and action a_t is selected (under the behavior policy). We observe s_t, a_t, r_t, s_{t+1}. We can compute:

$$y_t = r(s_t, a_t) + \lambda \max_b Q^{(t)}(s_{t+1}, b)) - Q^{(t)}(s_t, a_t).$$

- y_t can be used in an asynchronous R-M algorithm converging to Q, because

$$\begin{aligned}
\mathbb{E}[y_t|(s_t, a_t)] &= r(s_t, a_t) + \lambda \mathbb{E}[\max_b Q^{(t)}(s_{t+1}, b))|(s_t, a_t)] - Q^{(t)}(s_t, a_t) \\
&= r(s_t, a_t) + \lambda \sum_j p(j|s_t, a_t) \max_b Q^{(t)}(j, b) - Q^{(t)}(s_t, a_t) \\
&= H(Q^{(t)})(s_t, a_t) - Q^{(t)}(s_t, a_t)
\end{aligned}$$

Q-learning

Parameter. Step sizes (α_t)

1. **Initialization.** Select a Q-function $Q^{(0)} \in \mathbb{R}^{S \times A}$
2. **Observations.** (s_t, a_t, r_t, s_{t+1}) under the behavior policy π_b
3. Q**-function improvement.** For $t \geq 0$. Update the estimated Q-function as follows: $\forall s, a$,

$$Q^{(t+1)}(s,a) = Q^{(t)}(s,a)$$
$$+ 1_{(s_t,a_t)=(s,a)} \alpha_{n^{(t)}(s_t,a_t)} \left[r_t + \lambda \max_{b \in \mathcal{A}} Q^{(t)}(s_{t+1}, b) - Q^{(t)}(s_t, a_t) \right]$$

where $n^{(t)}(s,a) := \sum_{m=1}^{t} 1[(s,a) = (s_m, a_m)]$.

Q-learning convergence

Theorem. Assume that the step sizes (α_t) satisfy $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$. Further assume that the behaviour policy π_b visits every (state, action) pairs infinitely often. For any discount factor $\lambda \in (0, 1)$, under the Q-learning algorithm,

$$\lim_{n \to \infty} Q^{(t)} = Q, \quad \text{almost surely.}$$

The conditions required in the above theorem are met if $\alpha_t = \frac{1}{t+1}$ and if the behaviour policy yields an irreducible Markov chain (e.g. unichain model). They are also met for the ϵ-greedy behavior policy: it selects an action uniformly at random w.p. ϵ and $a_t \in \arg\max_{a \in A_{s_t}} Q^{(t)}(s_t, a)$ w.p. $1 - \epsilon$.

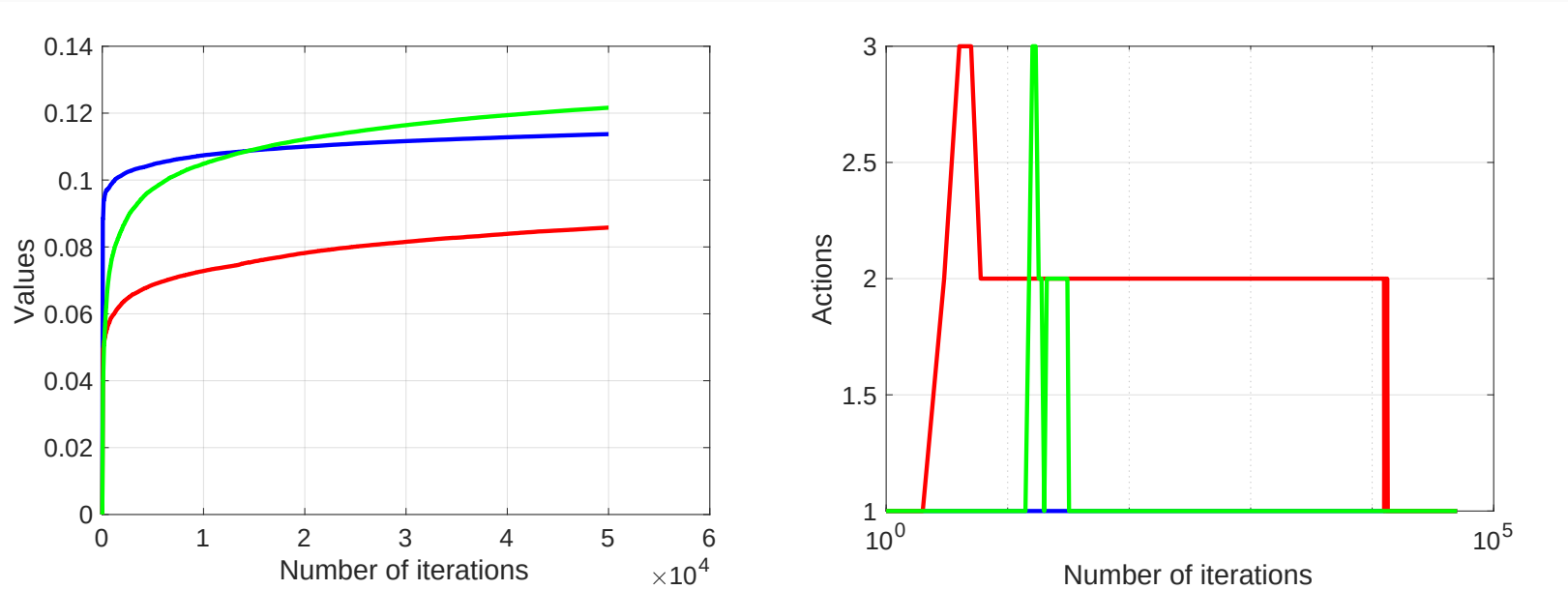
Q-learning: demo

The crawling robot ...

https://www.youtube.com/watch?v=2iNrJx6IDEo

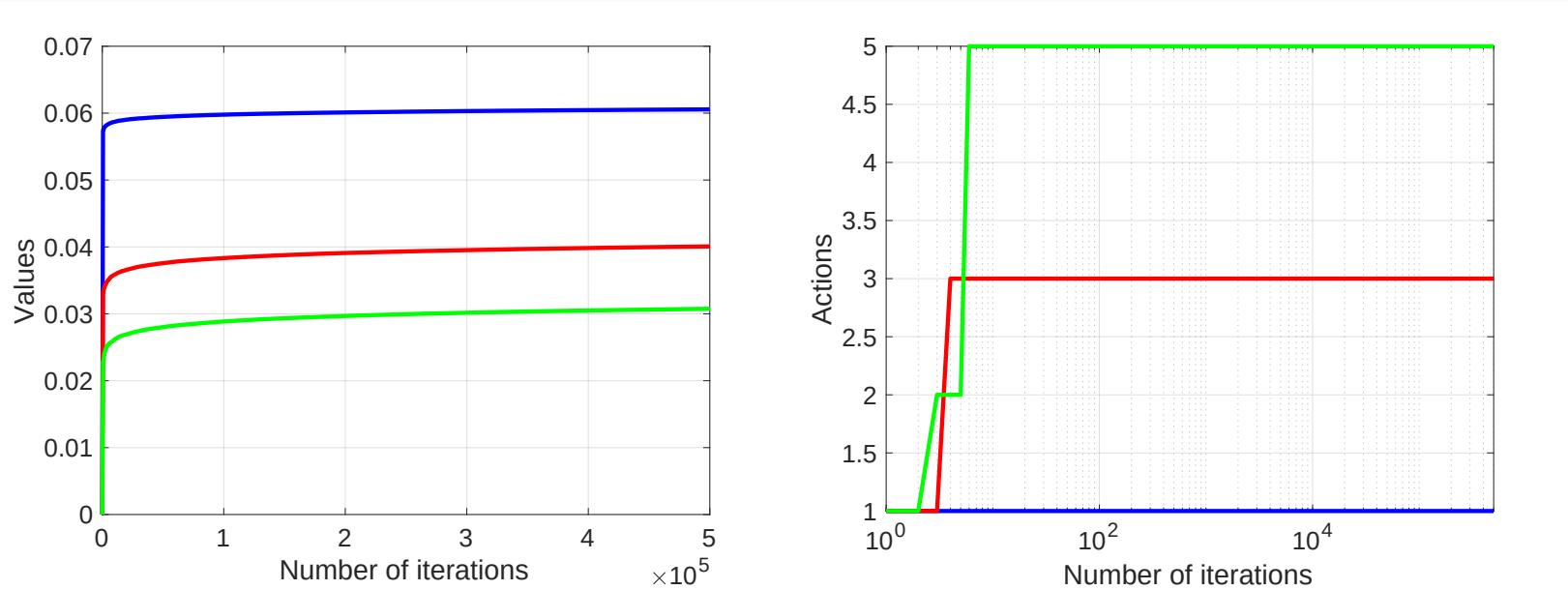
Q-learning: example

Randomly selected MDPs with 3 states and 3 actions.



Q-learning: example

Randomly selected MDPs with 3 states and 20 actions.



4. SARSA algorithm: On-policy learning

In step t:

1. We maintain a Q-function $Q^{(t)}$
2. We derive a randomized policy π_t from $Q^{(t)}$. Under π_t, in state s, the action a is selected with probability: $F(Q^{(t)}, s, a)$
3. We observe $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ under π_t (SARSA)
4. We update $Q^{(t+1)}$ so that it estimates the (state, action) value function of π_t

We hope that π_t converges to $\bar{\pi}$ such that its (state, action) value function $Q^{\bar{\pi}}$ satisfies:

$$\forall s, a, \quad Q^{\bar{\pi}}(s,a) = r(s,a) + \lambda \sum_{j \in S} p(j|s,a) \sum_{b \in A_j} F(Q^{\bar{\pi}}, j, b) Q^{\bar{\pi}}(j,b).$$

SARSA with ϵ-greedy

For example, the action can be selected as a ϵ-greedy policy based on $Q^{(t)}$: when in state s,

$$a = \begin{cases} \text{uniform}(\mathcal{A}_s) & \text{w.p. } \epsilon \\ \arg\max_{b \in A_s} Q^{(t)}(s, b) & \text{w.p. } 1 - \epsilon \end{cases}$$

In this case, we have:

$$F(Q, s, a) = \frac{\epsilon}{|A_s|} + (1 - \epsilon) 1_{\{a = \arg\max_{b \in A_s} Q(s,b)\}}$$

SARSA (state, action) value evaluation

- Asynchronous M-R algorithm:
  In step t, observe $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ under π_t: state, action, reward, state, action. We can compute:

$$y_t = r_t + \lambda Q^{(t)}(s_{t+1}, a_{t+1}) - Q^{(t)}(s_t, a_t).$$

  Then:

$$\mathbb{E}[y_t | s_t, a_t] = r_t + \lambda \sum_{j \in S} p(j | s_t, a_t) \sum_{b \in A_j} F(Q^{(t)}, j, b) Q^{(t)}(j, b) - Q^{(t)}(s_t, a_t).$$

SARSA with ϵ-greedy

Parameter. Step sizes (α_t)

1. Initialization. Select a Q-function $Q^{(0)} \in \mathbb{R}^{S \times A}$

2. Observations. $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ under π_t ϵ-greedy w.r.t. $Q^{(t)}$

3. Q-function improvement. For $t \geq 0$. Update the estimated Q-function as follows: $\forall s, a$,

$$Q^{(t+1)}(s,a) = Q^{(t)}(s,a) + 1_{(s_t,a_t)=(s,a)} \alpha_{n^{(t)}(s_t,a_t)} \left[r_t + \lambda Q^{(t)}(s_{t+1}, a_{t+1}) - Q^{(t)}(s_t, a_t) \right]$$

where $n^{(t)}(s,a) := \sum_{m=1}^{t} 1[(s,a) = (s_m, a_m)]$.

SARSA convergence

Theorem. Assume that the step sizes (α_t) satisfy $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$. Further assume that the policy π_t is the ϵ-greedy policy w.r.t. $Q^{(t)}$. For any discount factor $\lambda \in (0, 1)$, under the SARSA algorithm,

$$\lim_{n \to \infty} Q^{(t)} = Q^{\bar{\pi}}, \quad \text{almost surely.}$$

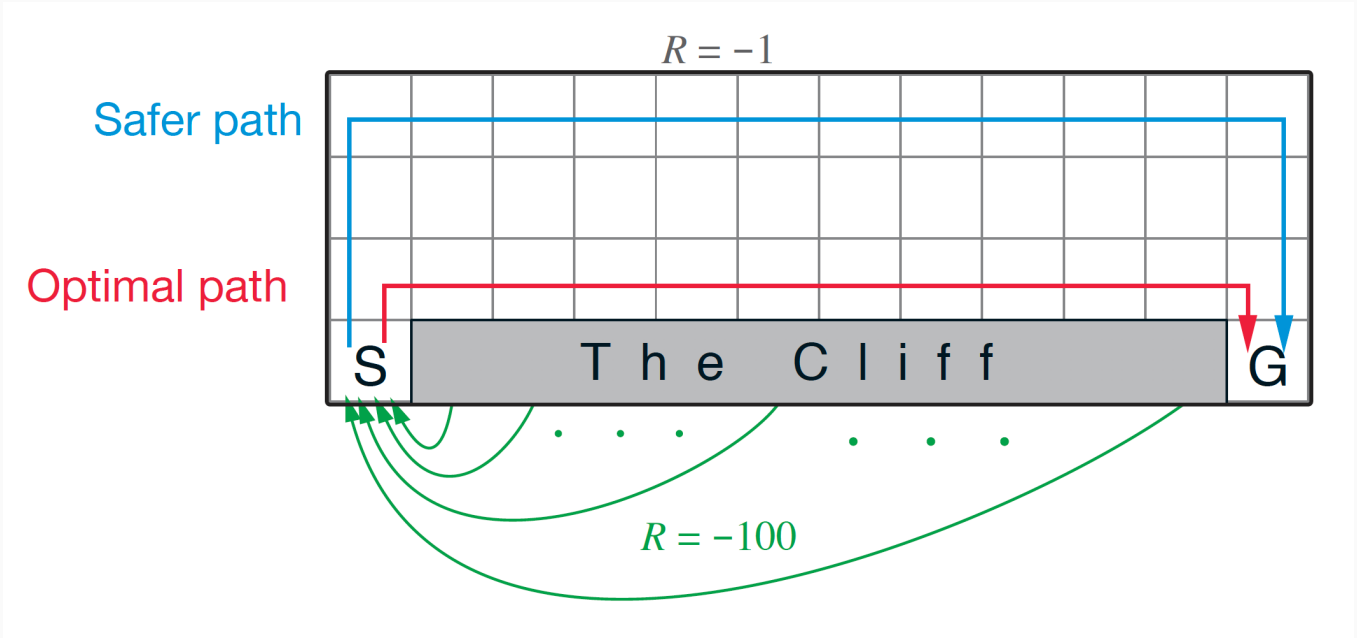
Observe that $Q^{\bar{\pi}}$ tends to Q as ϵ tends to 0^+.
We can select $\epsilon = 1/t$, in which case $Q^{(t)}$ tends to Q as $t \to \infty$.

Q-learning vs. SARSA

- Q-learning learns the true Q-function, and hence the optimal policy provided that the behavior policy explores all (state, action) pairs infinitely often. Q-learning does not learn the value function of the behavior policy.
- SARSA learns the (state, action) value function of the current behavior policy. It learns $Q^{\bar{\pi}}$ the (state, action) value function including the exploration phase (e.g. ϵ-greedy). SARSA is safer as it takes exploration into account.

Example: Cliff walking (cf. Sutton-Barto's book)

Goal: find the shortest path from 'S' to 'G' avoiding falling from the cliff. Episodic undiscounted MDP: reward $= -1$ moving in the grid, -100 if you fall in which case the episode ends.



Example: Cliff walking (cf. Sutton-Barto's book)

Q-learning with ϵ greedy exploration does not care if under the behavior policy, we fall from the cliff. And by learning the optimal path, it falls often!

SARSA behavior policy is safer, because it optimizes the value of the behavior policy and hence avoids falling too often.

Example: Cliff walking (cf. Sutton-Barto's book)

Online performance of Q-learning and SARSA (performance of the behavior policy)