

Foundations of Reinforcement Learning

Online RL: Monte Carlo, Sarsa and Q-learning with GLIE

Yuejie Chi

Department of Electrical and Computer Engineering

Carnegie Mellon University

Spring 2023

Announcement

- HW2 is posted, and will be due next Thursday (before spring break). Easier than HW1.
- Next week, we will discuss policy optimization (small order adjustment from the syllabus).
- Start thinking about midterm paper presentation NOW, which is scheduled for the week after the spring break.

Midterm paper presentation

- An in-class presentation on a selected paper (either self-choice upon approval of the instructor or selected from a given pool).
 - a tentative list is posted on the course website; still adding more papers
- 12 minutes presentation + 3 minutes questions.
 - Due to class size, we'll run this over 3 lectures. Presentation order will be generated randomly.
 - Participation is required - as your active participation in QA will count!
- "Critical" review:
 - what you like/ don't like the paper
 - what are the main take-aways
- highlight one result from the paper, by providing proof ideas or offering numerical simulations.
 - teaching your classmates something new!

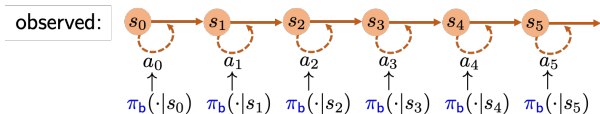
Outline

Online RL with exploration

ϵ -greedy exploration

Monte Carlo, Sarsa and Q-learning with GLIE

Recap: Q-learning following a behavior policy



To achieve $\|Q_T - Q^*\|_\infty \leq \varepsilon$, the sample size T needs to be at least above the order [Li et al., 2022]

$$\frac{1}{\mu_{\min}(1-\gamma)^5 \varepsilon^2} + \frac{t_{\text{mix}}}{\mu_{\min}(1-\gamma)},$$

where

- μ_{\min} is the minimum state-action occupancy probability

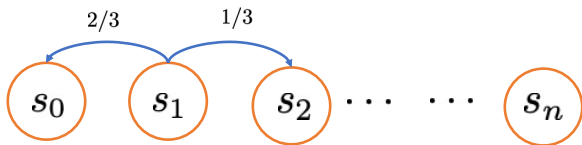
$$\mu_{\min} := \min \underbrace{\mu_{\pi_b}(s, a)}_{\text{stationary distribution}}$$

- t_{mix} is the mixing time, which captures the time to reach the steady state

Limitation

μ_{\min} need to be positive $\implies \pi_b$ visits the entire state-action space

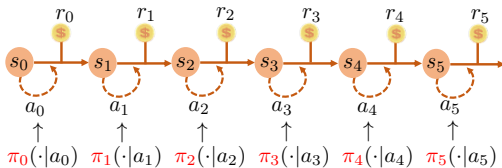
- π_b must be randomized
- Can we find such π_b for all MDPs?
- μ_{\min} can be exponentially small \implies need a lot of samples!



Can exploration helps to mitigate this issue?

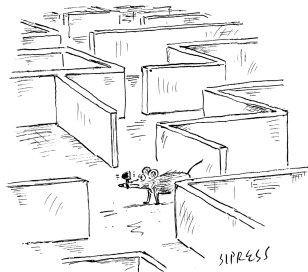
Online RL with exploration

Online RL



Exploration under an adaptive policy:

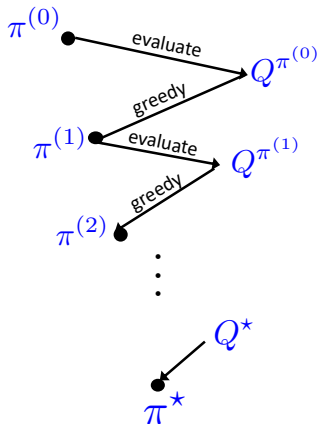
- trial-and-error
- sequential and online
- using samples



"Recalculating ... recalculating ..."

Model-free policy iteration?

Policy iteration



Can we run policy iteration using samples?

- We can perform the step of policy evaluation via Monte Carlo or TD-learning.
- However, when $\pi^{(t)}$ is deterministic, can only evaluate $Q^{\pi^{(t)}}(s, a)$ for $a = \pi^{(t)}(s)$ following policy $\pi^{(t)}$.
- Need to perform exploration!

ϵ -greedy exploration

ϵ -greedy exploration

- With probability $1 - \epsilon$ choose the greedy action
- With probability ϵ choose an action at random

$$\pi(a|s) = \begin{cases} \epsilon/|\mathcal{A}| + 1 - \epsilon & \text{if } a^* = \arg \max_{a \in \mathcal{A}} Q(s, a) \\ \epsilon/|\mathcal{A}| & \text{otherwise} \end{cases}$$

Theorem 1

For any ϵ -greedy policy π , the ϵ -greedy policy π' with respect to Q^π is an improvement, i.e. $V^{\pi'}(s) \geq V^\pi(s), \forall s \in \mathcal{S}$.

- assumes exact policy evaluation

Proof of greedy policy improvement

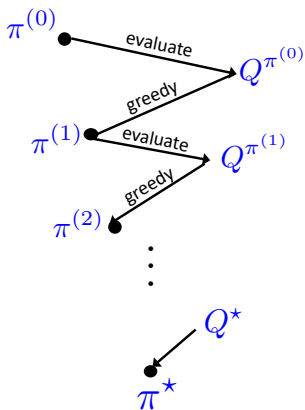
Proof: Let $m = |\mathcal{A}|$.

$$\begin{aligned} Q^\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) Q^\pi(s, a) \\ &= \frac{\epsilon}{m} \sum_{a \in \mathcal{A}} Q^\pi(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} Q^\pi(s, a) \\ &\geq \frac{\epsilon}{m} \sum_{a \in \mathcal{A}} Q^\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \epsilon/m}{1 - \epsilon} Q^\pi(s, a) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) Q^\pi(s, a) = V^\pi(s). \end{aligned}$$

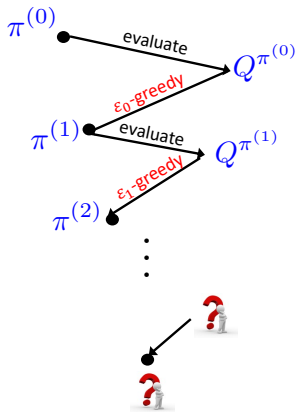
Therefore, by the policy improvement lemma, $V^{\pi'}(s) \geq V^\pi(s)$.

Generalized policy iteration

Policy iteration



Greedy policy iteration



Does greedy policy iteration converge to the optimal Q-function/policy, particularly when using samples?

Monte Carlo, Sarsa and Q-learning with GLIE

Greedy in the Limit with Infinite Exploration (GLIE)

Definition 2 (GLIE)

- **Infinite exploration.** All state-action pairs are explored infinitely many times,

$$\lim_{k \rightarrow \infty} N_k(s, a) = \infty$$

- **Greedy in the limit.** The policy converges on a greedy policy,

$$\lim_{k \rightarrow \infty} \pi_k(a|s) = \mathbb{I}(a = \operatorname{argmax}_{a' \in \mathcal{A}} Q_k(s, a'))$$

- ϵ -greedy is GLIE with $\epsilon_k = 1/k$.
- Boltzmann exploration is GLIE: $\pi_k(a|s) \propto e^{\beta_k(s)Q_k(s,a)}$ for appropriate choice of $\beta_k(s)$ [Singh et al., 2000].

GLIE Monte-Carlo control

Monte-Carlo control requires episodic environment (since it works only with complete sequences).

- 1 Sample a new episode using π ;
- 2 Update the Q-estimate using Monte-Carlo;
- 3 Update policy π using Q , e.g., ϵ -greedy.

Theorem 3

GLIE Monte-Carlo control converges to the optimal Q-function, $Q(s, a) \rightarrow Q^(s, a)$.*

Using bootstrapping for policy evaluation

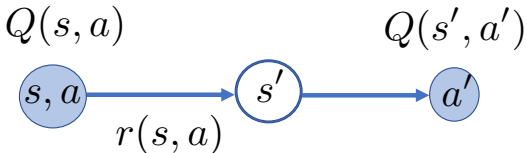
TD has several advantages over Monte-Carlo (MC):

- Lower variance
- Online
- Incomplete sequences

Natural idea: use TD instead of MC for policy evaluation

- apply TD to $Q(s, a)$
- use ϵ -greedy policy improvement
- update every time step (we don't need to wait for TD to converge)

Sarsa for on-policy Q-update



On-policy Q-update:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(\underbrace{r(s, a) + \gamma Q(s', a')}_{\text{target}} - Q(s, a) \right)$$

- The name comes from updating using a 5-tuple: (s, a, r, s', a') .
- The method is an on-policy method because it tries to learn about policy π from experience sampled from π .
- Can combine TD(λ) in Sarsa for the target part.

Sarsa for on-policy control

① **Initial phase:** initialization $Q(s, a)$ arbitrarily. Set $Q(s, \cdot) = 0$ if s is a terminal state. Initial state s and initial policy π (e.g., uniform).

② For each round $t = 1, 2, \dots$:

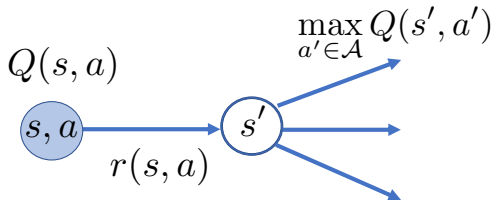
- Choose $a \sim \pi(\cdot|s)$ for the current state s ;
- Take action a , observe $r = r(s, a)$ and next state s' ;
- Choose $a' \sim \pi(\cdot|s')$ from the next state s' and update the Q-value

$$Q(s, a) \leftarrow Q(s, a) + \alpha_t (r(s, a) + \gamma Q(s', a') - Q(s, a))$$

- $s \leftarrow s', a \leftarrow a'$.
- Update policy π using Q , e.g., ϵ -greedy.

Repeat this for every episode if in an episodic environment.

Q-learning for off-policy Q-update



Off-policy Q-update:

$$Q(s, a) \leftarrow Q(s, a) + \alpha_t \left(\underbrace{r(s, a) + \gamma \max_{a' \in \mathcal{A}} Q(s', a')}_{\text{target}} - Q(s, a) \right)$$

- The method is an off-policy method because it tries to learn about an improved policy from experience sampled from π .

Q-learning for off-policy control

- 1 **Initial phase:** initialization $Q(s, a)$ arbitrarily. Set $Q(s, \cdot) = 0$ if s is a terminal state. Initial state s and initial policy π (e.g., uniform).
- 2 For each round $t = 1, 2, \dots$:
 - Choose $a \sim \pi(\cdot|s)$ for the current state s ;
 - Take action a , observe $r = r(s, a)$ and next state s' ;
 - **Update the Q-value**

$$Q(s, a) \leftarrow Q(s, a) + \alpha_t \left(r(s, a) + \gamma \max_{a' \in \mathcal{A}} Q(s', a') - Q(s, a) \right)$$

- $s \leftarrow s'$.
- Update policy π using Q , e.g., ϵ -greedy.

Repeat this for every episode if in an episodic environment.

Convergence of Sarsa/Q-learning with GLIE

Theorem 4 ([Singh et al., 2000])

Sarsa/Q-learning converges to the optimal action-value function, $Q(s, a) \rightarrow Q^(s, a)$, under the following conditions:*

- *GLIE sequence of policies $\pi_t(a|s)$;*
- *Robbins-Monro sequence of learning-rates*

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

- Due to GLIE, the policy will also converge to the optimal policy.

Case study: Q-learning versus Sarsa

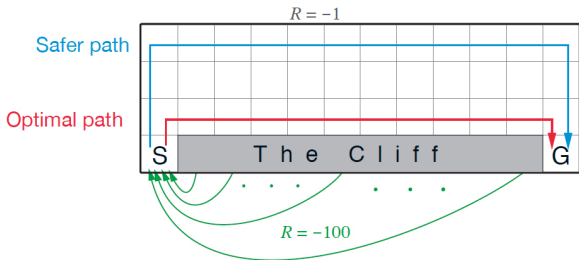


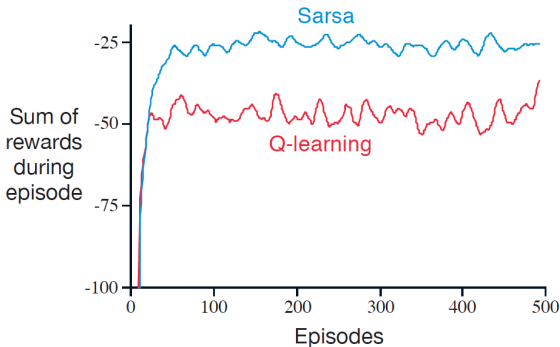
Figure credit: Sutton and Barto

Cliff walking:

- undiscounted, episodic task, with start and goal states, and the usual actions causing movement up, down, right, and left.
- Reward is -1 on all transitions except "The Cliff". Stepping into this region incurs a reward of -100 and sends the agent instantly back to the start.

Case study: Q-learning versus Sarsa

Both are executed with ϵ -greedy policy, $\epsilon = 0.1$.



- Q-learning learns the optimal policy but results in its occasionally falling off the cliff because of the “ ϵ -greedy” action selection.
- Sarsa takes the action selection into account and learns the longer but safer path through the upper part of the grid.

Double Q-learning with GLIE

- 1 **Initial phase:** initialization Q_1, Q_2 arbitrarily. Initial state s and initial policy π (e.g., uniform).
- 2 For each round $t = 1, 2, \dots$:
 - Choose $a \sim \pi(\cdot|s)$ for the current state s ;
 - Take action a , observe $r = r(s, a)$ and next state s' ;
 - Update the Q-value using one of the following randomly

$$Q^1(s, a) \leftarrow Q^1(s, a) + \alpha_t \left(r(s, a) + \gamma Q^2(s, \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q^1(s', a)) - Q^1(s, a) \right),$$

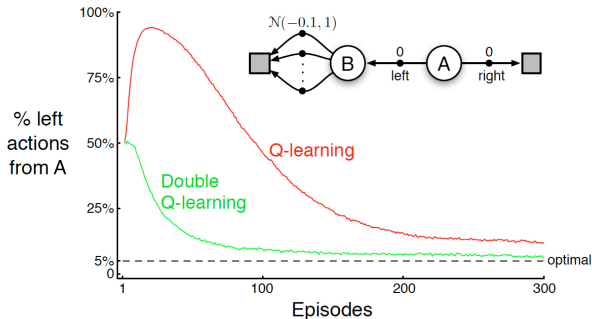
or

$$Q^2(s, a) \leftarrow Q^2(s, a) + \alpha_t \left(r(s, a) + \gamma Q^1(s, \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q^2(s', a)) - Q^2(s, a) \right).$$

- $s \leftarrow s'$.
- Update policy π using $(Q^1 + Q^2)/2$, e.g., ϵ -greedy.



Repeat this for every episode if in an episodic environment.

Mitigating overestimation via double Q-learning



- Q-learning initially learns to take the left action much more often than the right action, and always takes it significantly more often than the 5% minimum probability enforced by ϵ -greedy action selection with $\epsilon = 0.1$.
- Double Q-learning is essentially unaffected by the overestimation bias.

References I

-  Li, G., Wei, Y., Chi, Y., Gu, Y., and Chen, Y. (2022).
Sample complexity of asynchronous Q-learning: Sharper analysis and variance reduction.
IEEE Transactions on Information Theory, 68(1):448–473.
-  Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000).
Convergence results for single-step on-policy reinforcement-learning algorithms.
Machine learning, 38:287–308.