Homework 2

Due: November 25, 2021

Questions in this Homework appear ⟦ *framed and in italics* ⟧.

Theoretical questions are marked with the symbol ✎ and are worth 4 points in total. Questions that require implementation are marked with the symbol 🖥 and are worth 6 points in total. You should submit a `pdf` file with your answers, including all relevant information necessary to reproduce your results. As an appendix to your document, make sure to include the code used to generate your answers.

Do not be excessively concerned with the evaluation facet of the homework. If you have any difficulties in understanding a question or coming up with an answer, come forward and discuss with me and the rest of the class—treat the homework more as a tool to help you learn than to grade you.

1   The cliff problem (3 pts.)

Consider the following problem. An agent must navigate the grid world represented in Fig. 1, where the grey area corresponds to a cliff that the agent must avoid. The goal state corresponds to the cell marked with a G, while the cell marked with an S corresponds to a starting state. At every step, the agent receives a reward of -1 except in the cliff region, where the reward is -100. Whenever the agent steps into a cliff state, its position is reset back to the start state. When the agent steps into the goal state, the episode ends. The agent has available four actions: up (U), down (D), left (L) and right (R), all of which move the agent deterministically in the corresponding direction.
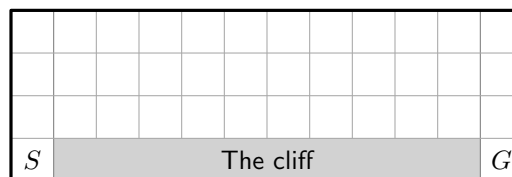

Figure 1: The cliff problem. The agent must navigate from the cell marked S to the cell marked G without stepping in the cliff (the grey area).

In this question, you will compare the performance of SARSA and Q-learning on the cliff task. To do so, assume that the agent follows an ε-greedy policy, with $\varepsilon = 0.15$. Run both algorithms for 500 episodes,

making sure that the Q-values for both methods are initialized to 0. Consider throughout that $\gamma = 1$ and use a step-size of $\alpha = 0.5$.

📇   **Question 1.** *Compare:*

- *The total reward in each episode for Q-learning and SARSA, plotting the two in a single plot.*

- *The resulting policy after the* 500 *episodes.*

Comment any differences observed.

Note: *To mitigate the effect of noise on the plot, perform multiple runs and average the result across runs.*

◇

In Question 1 you implemented Q-learning and SARSA in a simple grid world domain, where exact representations for q^* were possible. However, many domains are not amenable to such exact representations, due to the large size of the corresponding state spaces. In such domains, some form of function approximation is necessary. In the remainder of the homework, you will look at the problem of *function approximation* in RL.

2   TD learning with function approximation (3 pts.)

Consider the MDP $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where

- $\mathcal{S} = \{1, 2, 3, 4, 5, 6, 7\}$ is the state space;

- $\mathcal{A} = \{A, B\}$ is the action space;

- The transition probabilities are summarized in the matrices

$$
\mathbf{P}_A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{P}_B = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \end{bmatrix}.
$$

- $r(s, a) = 0$ for all pairs $(s, a) \in \mathcal{S} \times \mathcal{A}$.

- $\gamma = 0.99$.

In this question, you will again observe the performance of SARSA and Q-learning in the MDP above when function approximation is used. Both methods should represent the Q-function as a linear combination of features, i.e., the Q-functions computed by the two algorithms should take the form

$$
q_{\mathbf{w}}(x, a) = \sum_{k=1}^{15} \phi_k(x, a) w_k,
$$

where ϕ_k is the kth feature and w_k the associated weight. In particular, given the matrices

$$\mathbf{\Phi}_A = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{\Phi}_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

make sure to use the features defined, for every state $s \in \mathcal{S}$ and every action $a \in \mathcal{A}$, as

$$\phi_k(s,a) = [\Phi_a]_{s,k}.$$

Then, given a transition $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$, the update equations for Q-learning and SARSA become, respectively,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \alpha\phi(s_t, a_t)\big(r_t + \gamma \max_{a \in \mathcal{A}} q_{\mathbf{w}^{(t)}}(s_{t+1}, a) - q_{\mathbf{w}^{(t)}}(s_t, a_t)\big),$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \alpha\phi(s_t, a_t)\big(r_t + \gamma q_{\mathbf{w}^{(t)}}(s_{t+1}, a_{t+1}) - q_{\mathbf{w}^{(t)}}(s_t, a_t)\big),$$

where $\phi(s_t, a_t)$ is a column vector with kth component given by $\phi_k(s_t, a_t)$ and α is the step-size.

In running the two algorithms, assume that the agent follows a policy that selects action A with probability $\frac{1}{7}$ and action B with probability $\frac{6}{7}$. Run both algorithms for 500 time steps, making sure that the parameter vectors for both methods are initialized to

$$\mathbf{w}^{(0)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 10 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^\top.$$

Consider throughout a step-size of $\alpha = 0.01$.

✎ **Question 2.** *Compare the norm of the parameter vector at each time step for Q-learning and SARSA, plotting the two in a single plot. Comment any differences observed.*

Note: *To mitigate the effect of noise on the plot, perform multiple runs and average the result across runs.*

---

3 The policy gradient theorem (2 pts.)

As seen in Question 2, when you combine function approximation with some temporal difference methods, convergence guarantees no longer hold and divergence may occur. Additionally, even if the methods converge, the approximation that you obtain to the Q-function may lead to a poor policy.

Policy gradient methods constitute an alternative to value-based methods such as TD-learning. In this class of methods, we select a *parameterized family of policies*, Π_θ, and seek the *best* policy in Π_θ, instead of directly approximating the optimal Q-function. Policy gradient methods work by adjusting the parameter θ controlling the policy so that the overall performance of the agent improves.

Given an MDP $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, let μ_0 represent the initial state distribution, *i.e.*,

$$\mu_0(s) \triangleq \mathbb{P}\left[S_0 = s\right].$$

We define the global performance of a policy $\pi_\theta \in \Pi_\theta$ as

$$\begin{aligned} J(\theta) &\triangleq \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^\infty \gamma^t r_t \mid S_0 \sim \mu_0\right] \\ &= \sum_{s \in \mathcal{S}} \mu_0(s) v_{\pi_\theta}(s) \\ &= \sum_{s \in \mathcal{S}} \mu_0(s) \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) q_{\pi_\theta}(s, a). \end{aligned}$$

Policy gradient performs a gradient ascent update on the parameters θ defining the policy. In other words, policy gradient methods rely on the update

$$\theta^{(k+1)} = \theta^{(k)} + \alpha_k \nabla_\theta J(\theta^{(k)}).$$

You will show that the gradient $\nabla_\theta J(\theta)$ has a form that is convenient for computation, in what is known as the *policy gradient theorem*.

✎ **Question 3.** *Show that*

$$\nabla_\theta J(\theta) = \sum_{s \in \mathcal{S}} \mu_\theta(s) \sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) q_{\pi_\theta}(s, a),$$

where

$$\mu_\theta(s') \triangleq \sum_{s \in \mathcal{S}} \mu_0(s) \sum_{t=0}^\infty \gamma^t p_{\pi_\theta}^t(s' \mid s)$$

and we write $p_\pi^t(s' \mid s)$ to denote the t-step transition probability from state s to state s' when policy π is followed. Such probabilities are defined inductively as

$$p_\pi^0(s' \mid s) = \mathbb{I}[s = s'] \qquad p_\pi^{t+1}(s' \mid s) = \sum_{s'' \in \mathcal{S}} \sum_{a \in \mathcal{A}} p(s' \mid s'', a) \pi(a \mid s'') p_\pi^t(s'' \mid s).$$

Suggestion: *For a fixed state $s \in \mathcal{S}$, start by showing that*

$$\nabla_\theta v_{\pi_\theta}(s) = \sum_{s' \in \mathcal{S}} \sum_{t=0}^\infty \gamma^t p_{\pi_\theta}^t(s' \mid s) \sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a \mid s') q_{\pi_\theta}(s', a),$$

using the fact that

$$v_{\pi_\theta}(s) = \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) q_{\pi_\theta}(s, a).$$

---

4   The actor-critic architecture (2 pts.)

One important aspect that follows from Question 3 is that, in the computation of the gradient, it is necessary to compute the function q_{π_θ}, which brings us back to the need for value-based methods. And while computing

q_{π_θ} can be done by resorting to SARSA, we are left with the challenge of selecting adequate features to represent q_{π_θ}. However, as will soon become apparent, the policy-gradient framework provides us with a sound solution to that specific difficulty. Consider, once again, the expression for the gradient

$$\nabla_\theta J(\theta) = \sum_{s \in \mathcal{S}} \mu_\theta(s) \sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) q_{\pi_\theta}(s, a).$$

We can equivalently write

$$\nabla_\theta J(\theta) = \sum_{s \in \mathcal{S}} \mu_\theta(x) \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} q_{\pi_\theta}(s, a),$$

where we simply multiplied and divided the innermost expression by $\pi_\theta(a \mid s)$. Using the equality

$$\nabla_\theta \log \pi_\theta(a \mid s) = \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)},$$

we get

$$\nabla_\theta J(\theta) = \sum_{s \in \mathcal{S}} \mu_\theta(s) \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) \nabla_\theta \log \pi_\theta(a \mid s) q_{\pi_\theta}(s, a)$$
$$= \mathbb{E}_{S \sim \mu_\theta(\cdot), A \sim \pi_\theta(\cdot \mid S)} \left[\nabla_\theta \log \pi_\theta(A \mid S) q_{\pi_\theta}(S, A) \right].$$

✎ **Question 4.** *Show that, by setting* $\phi(s, a) = \nabla_\theta \log \pi_\theta(a \mid s)$,

$$\nabla_\theta J(\theta) = \mathbb{E}_{S \sim \mu_\theta(\cdot), A \sim \pi(\cdot \mid X)} \left[\phi(S, A) (\Gamma_\phi q_{\pi_\theta})(S, A) \right]$$

where, for an arbitrary function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

$$(\Gamma_\phi f)(s, a) = \phi^\top(s, a) \mathbf{\Phi}^{-1} \mathbb{E}_{S \sim \mu(\cdot), A \sim \pi(\cdot \mid X)} \left[\phi(S, A) f(S, A) \right]$$

and

$$\mathbf{\Phi} = \mathbb{E}_{S \sim \mu_\theta(\cdot), A \sim \pi(\cdot \mid S)} \left[\phi(S, A) \phi^\top(X, A) \right].$$

The result in Question 4 settles the difficulty of selecting the features to estimate q_{π_θ}, since by using $\phi(s, a) = \nabla_\theta \log \pi_\theta(a \mid s)$ we recover an unbiased estimate of the gradient, even if an approximation of q_{π_θ} is used. The features $\phi = \nabla_\theta \log \pi_\theta$ are known in the literature as *compatible features* or *compatible basis functions*, and unify in a common framework a policy-based approach (policy-gradient) and a value-based approach (temporal-difference learning). This framework, which combines policy improvement and evaluation, is known as the *actor-critic architecture*.