535514 Spring 2024: Reinforcement Learning **(Due: 2024/03/20, Wednesday, 21:00)**

Homework 1: Warm-Up – Fundamentals of MDPs and RL

Submission Guidelines: Your deliverables shall consist of 2 separate files – (i) A PDF file: Please compile all your write-ups into one .pdf file (photos/scanned copies are acceptable; please make sure that the electronic files are of good quality and reader-friendly); (ii) A zip file: Please compress all your source code into one .zip file. Please submit your deliverables via E3.

Problem 1 (Q-Value Iteration) (15+15=30 points)

(a) Recall that in Lecture 3, we define $V^*(s) := \max_\pi V^\pi(s)$ and $Q^*(s,a) := \max_\pi Q^\pi(s,a)$. Suppose $\gamma \in (0,1)$. Prove the following Bellman optimality equations:

$$V^*(s) = \max_a Q^*(s,a) \tag{1}$$

$$Q^*(s,a) = R_{s,a} + \gamma \sum_{s'} P^a_{ss'} V^*(s'). \tag{2}$$

Please carefully justify every step of your proof. (Hint: For (1), you may first prove that $V^*(s) \le \max_a Q^*(s,a)$ and then show $V^*(s) < \max_a Q^*(s,a)$ cannot happen by contradiction. On the other hand, (2) can be shown by using the similar argument or by leveraging the fact that $Q^\pi(s,a) = R_{s,a} + \gamma \sum_{s'} P^a_{ss'} V^\pi(s')$)

(b) Based on (a), we thereby have the recursive Bellman optimality equation for the optimal action-value function Q_* as:

$$Q^*(s,a) = R_{s,a} + \gamma \sum_{s'} P^a_{ss'} \big(\max_{a'} Q^*(s',a') \big) \tag{3}$$

Similar to the standard Value Iteration, we can also study the *Q-Value Iteration* by defining the Bellman optimality operator $T^* : \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ for the action-value function: for every state-action pair (s,a)

$$[T^*(Q)](s,a) := R_{s,a} + \gamma \sum_{s'} P^a_{ss'} \max_{a'} Q(s',a') \tag{4}$$

Show that the operator T^* is a γ-contraction operator in terms of ∞-norm. Please carefully justify every step of your proof. (Hint: For any two action-value functions Q, Q', we have $\|T^*(Q) - T^*(Q')\|_\infty = \max_{(s,a)} \big|[T^*(Q)](s,a) - [T^*(Q')](s,a)\big|$)

Problem 2 (Regularized MDPs) (10+10=20 points)

In Lecture 4, we formally describe the regularized MDP, which is a direct extension of the classic MDP with a regularizer Ω. In this problem, for simplicity, suppose we use the Shannon entropy as our regularizer, i.e., $\Omega(\pi(\cdot|s)) \equiv H(\pi(\cdot|s)) := -\sum_{a\in\mathcal{A}} \pi(a|s) \ln \pi(a|s)$. Let us verify a few important properties mentioned in Lecture 4 as follows.

(a) Recall that we introduce the "regularized Bellman expectation operator" T^π_Ω as

$$[T^\pi_\Omega V](s) := R^\pi_s + \Omega(\pi(\cdot|s)) + \gamma P^\pi_{ss'} V. \tag{5}$$

Please verify that T^π_Ω is a contraction operator in L_∞ norm. (Hint: Try to extend the proof procedure of the contraction property of T^π in Lecture 3)

(b) Moreover, under regularized MDPs, we study the optimal value functions V^*_Ω and optimal Q functions Q^*_Ω

and learn the Bellman optimality equations as

$$V_\Omega^*(s) = \max_{\pi \in \Pi} R_s^\pi + \gamma P_s^\pi V_\Omega^* \tag{6}$$

$$Q_\Omega^*(s,a) = R_{s,a} + \gamma E_{s' \sim P(\cdot|s,a)}[V_\Omega^*(s')]. \tag{7}$$

Could you design an iterative algorithm that can obtain V_Ω^* and Q_Ω^*? Please clearly write down the complete pseudo code of your algorithm and provide comments on each line of your pseudo code. (Hint: Try to extend the Value Iteration for standard MDPs to the regularized MDPs based on Equation 6)

Problem 3 (A Property Used in Policy Gradient) (10 points)

Show the following useful property discussed in Lectures 5-6: for any function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

$$\mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}} \Big[\sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \Big] = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\mu^{\pi_\theta}} \mathbb{E}_{a \sim \pi_\theta(\cdot|s)} \big[f(s,a) \big] \tag{8}$$

(Hint: It might be slightly easier to go from the RHS to LHS. Specifically, you may first expand the RHS of (8) into a sum of $f(s,a)$ over s and a and then apply the definition of $d_\mu^{\pi_\theta}$, which involves a sum of probability over t. Next, try to reorganize the triple summation into the form of the LHS of (8))

Problem 4 (Implementing Policy Iteration and Value Iteration) (35 points)

In this problem, we will implement Policy Iteration and Calue Iteration for a classic MDP environment called "Taxi" (Dietterich, 2000). This environment has been included in the OpenAI Gym: https://gym.openai.com/envs/Taxi-v3/. To accomplish this task, you may take the following steps:

- Get familiar with the Taxi environment by reading the Gym documentation at https://www.gymlibrary.dev/environments/toy_text/taxi/. The state space consists of 500 possible states as there are 25 taxi positions, 5 possible locations of the passenger (including the case when the passenger is in the taxi), and 4 destination locations. Moreover, the agent has 6 possible actions (namely, 0: move south; 1: move north; 2: move east; 3: move west; 4: pickup passenger; 5: drop off passenger). The rewards are: (i) -1 per step unless other reward is triggered; (ii) +20 for delivering passenger; (iii) -10 for executing "pickup" and "drop-off" actions illegally.

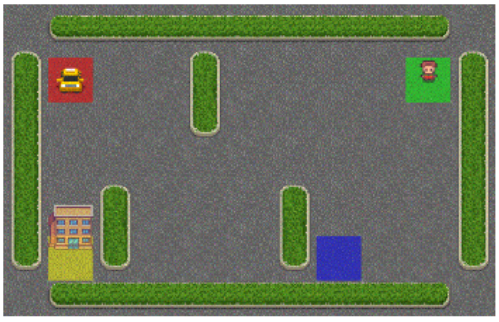

Figure 1: An illustration of the Taxi environment.

- Read through **policy_and_value_iteration .py** and then implement the two functions **policy_iteration** and **value_iteration** based on the pseudo code of PI and VI provided in the lecture slides.
- Note: Please set $\gamma = 0.9$ and the termination criterion $\varepsilon = 10^{-3}$. Moreover, you could use either Taxi-v2 or Taxi-v3 environment (Taxi-v3 is recommended). Note that discrepancy $= 0$ is a necessary condition (but not sufficient) of correct implementation, and with the default $\varepsilon = 10^{-3}$, you shall be able to observe zero discrepancy between the policies obtained by PI and VI.

Problem 5 (Installing and Getting Familiar With D4RL Dataset) (10 points)

In this problem, you will be asked to install and investigate the D4RL Dataset, which is currently the standard dataset for Offline RL. To accomplish this task, you shall take the following steps:

- Please first take a careful look at the GitHub repo of D4RL https://github.com/Farama-Foundation/D4RL and the paper https://arxiv.org/abs/2004.07219. Then, install **d4rl** by following the installation guide on the GitHub repo (you may need to install several other packages, such as Gymnasium and MuJoCo).
- Read and run the provided **d4rl_sanity_check.py** and finish the following tasks:
  - (1) Generate an offline dataset by directly running **d4rl_sanity_check.py**.
  - (2) Describe the format of the dataset that you just obtained.
  - (3) Modify the **d4rl_sanity_check.py** and generate another offline dataset of one of the MuJoCo tasks (e.g., Hopper, Walker, or Halfcheetah). Similarly, describe the format of the MuJoCo dataset that you just obtained.
- One final remark: We will keep using the D4RL dataset for HW2, HW3, and the team final project. Therefore, it would be very helpful to now set up the environment that you could easily reuse subsequently.