IOC5262 Spring 2020: Reinforcement Learning **(Due: 2020/04/10, 23:55pm)**

Assignment 1: MDPs, Policy Gradient, and Model-Free Policy Evaluation

Submission Guidelines: Please compress all your write-ups (photos/scanned copies are acceptable) and source code into one .zip file and submit the compressed file via E3.

Problem 1 (Q-Value Iteration) (8+8=16 points)

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

$$V_*(s) = \max_a Q_*(s,a) \tag{1}$$

$$Q_*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a V_*(s'). \tag{2}$$

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. Similarly, (2) can be shown by using the similar argument)

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

$$Q_*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a \big(\max_{a'} Q_*(s',a') \big) \tag{3}$$

Similar to the value iteration, we can 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)

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

Show that the operator T^* is a γ-contraction operator with L_∞-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 (Distributional Perspective of MDPs) (8+8+8=24 points)

Recall that given a policy π, the distributional Bellman operator $B^\pi : \mathcal{Z} \to \mathcal{Z}$ is defined as

$$[B^\pi Z](s,a) :\overset{D}{=} r(s,a) + \gamma P^\pi Z(s,a), \tag{5}$$

where $\gamma \in (0,1)$. In the first two subproblems (a)-(b), we would like to show that the B^π is a contraction operator in the maximal form of the Wasserstein metric (i.e. $\bar{d}_p$ defined in Lecture 4). For ease of exposition, we further consider the following notations: Given any two random variables U, V with CDFs F_U, F_V, we write $d_p(U,V) := d_p(F_U, F_V)$.

(a) To begin with, show that the Wasserstein metric has the following property: Let U and V be two random variables. Let A be another random variable that is independent of U and V. Then, we have

$$d_p(A+U, A+V) \le d_p(U,V). \tag{6}$$

(b) By using the result in (a), show that B^π is a γ-contraction operator in $\bar{d}_p$.

(c) Provide a counterexample with 3 states and 2 actions showing that the distributional Bellman optimality operator may NOT be a contraction operator in the the maximal form of the Wasserstein metric. (Hint: You may construct your counterexample based on the example provided in Lecture 4)

Problem 3 (Implementing Policy Iteration and Value Iteration) (20 points)

In this problem, we will implement policy iteration and value 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-v2/. Read through **policy_and_value_iteration .py** and then implement the two functions **policy_iteration** and **value_iteration** (Note: please set $\gamma = 0.9$ and the termination criterion $\varepsilon = 10^{-3}$).

Problem 4 (Policy Gradient) (8+8=16 points)

(a) Show the following useful property discussed in Lecture 6: for any function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

$$\mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}} \Big[\sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \Big] = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_\theta}} \mathbb{E}_{a \sim \pi_\theta(\cdot|s)} \big[f(s, a) \big] \tag{7}$$

(Hint: It might be slightly easier to go from the RHS to LHS)

(b) Show that for episodic environments, the policy gradient can be expressed using the advantage function A^{π_θ} as follows:

$$\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}} \Big[\sum_{t=0}^{T-1} \gamma^t A^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \Big] \tag{8}$$

Problem 5 (Implement Monte-Carlo and Temporal Difference Policy Evaluation) (24 points)

In this problem, we will apply both **first-visit MC** and **TD(0)** to reproduce the plots of the value function of the Blackjack problem under the fixed policy that one will stick only if the sum of cards is greater than or equal to 20. The Blackjack environment has been included in the OpenAI Gym: https://gym.openai.com/envs/Blackjack-v0/. Please read through **mc_td_policy_evaluation.py** and then implement the two functions **mc_policy_evaluation** and **td0_policy_evaluation** (Note: please set $\gamma = 1.0$).