Homework 2: Statistical Foundations of Reinforcement Learning (Spring 2021)

University of California, Santa Barbara

Assigned on April 26, 2020 (Monday)

Due at 11:59 pm on May 10, 2020 (Monday)

Notes:

- Be sure to read "Policy on Academic Integrity" on the course syllabus.
- There are ***[105 points]*** in this homework, and a bonus ***[30 points]***.
- You need to submit your homework via Gradescope.
- Contract the instructor if you spot typos. Any updates or correction will be posted on the course Announcements page and piazza, so check there occasionally.

---

0. Acknowledgment *[0 points]*

For each question in this HW, please list all your collaborators and reference materials (beyond those specified on the website) that were used for this homework.

1. **List of Collaborators** List the names of all people you have collaborated with and for which question(s).
2. **List of Acknowledgements.** If you find an assignment's answer or use a another source for help, acknowledge for which question and provide an appropriate citation (there is no penalty, provided you include the acknowledgement). If not, then write "none".

1 Balancing an inverted pendulum (25 Points + 10 Bonus)

Have you tried balancing a pen tip down on your palm? This is a very challenging problem for our reflexes, but a relatively easy one for AI. In this problem, we will try using what we learned to build an RL agent to learn a strategy for such a maneuver. For a similar example of learning-to-control in practice, check out the physical implementation in this video: `https://www.youtube.com/watch?v=hGSn9DyvZDU`

(a) (5 Points) On your platform, set up OpenAI gym (`https://gym.openai.com/`). In most platforms, you could get away with "pip install gym".

Then use the provided code snippet to load a CS292 customized version of Cartpole. It is now a finite horizon episodic RL problem with continuous states and discrete action. We will

focus on learning a stationary policy, e.g., the policy chooses an action based only on the immediate state observation.

What is the state (observation) space, action space in this environment? Implement a naive policy that takes Action 0 if the angle of the rod is negative and Action 1 otherwise.

Implement a function that conducts on-policy evaluation of a given policy π. Run this naive policy for 1000 episodes and report the mean and standard deviation of the (undiscounted) total reward of each episode. (if you comment out `env.render()`, if will run more quickly.)

(b) (10 points) Discretize the state-space and action space. Implement Q-learning with ϵ-greedy in the tabular case.

Using the provided "`discretize_state`" function to discretize the observed continuous states into a discrete states so we can model the environment as a tabular MDP.

Write two functions: 1. "`Update_Q`" that update the Q value matrix according to the (infinite horizon, tabular MDP version of) Q-learning update; 2. "`policy`' which takes the observation and select the greedy action according to a given Q-value function.

Run Q-learning with (ϵ-greedy exploration) for 400 episodes (we have provided some default parameters on ϵ schedule, learning rate, and discounting parameter γ; feel free to tweak them if they do not work well for your implementation).

Plot the expected reward of the greedy policy learned at Episode $0, 50, 100, ..., 400$ (checkpoints) using your on-policy evaluation function from Part (a).

(c) (10 points) Implement Q-learning with linear function approximation with the provided linear state feature extractor $\phi(s) \in \mathbb{R}^d$.

The standard trick to convert state-feature extractor to a state-action feature extractor for discrete actions is to output $\phi(s,a) = [\phi(s)^T, \mathbf{0}^T]^T$ if $a = 0$ and $\phi(s,a) = [\mathbf{0}^T, \phi(s)^T]^T$ if $a = 1$ where $\mathbf{0}$ is an all zero column vector with the same length of $\phi(s)$.

Implement the Q-learning updates with linear Q-value function approximation as well as the policy.

Plot the learning curve just like in Part (b).

(d) (Bonus 10 points) Replace the Q-learning in Part (b) and Part (c) with expected SARSA. Compare the online average of the episode rewards with that of Q-learning as well as the corresponding greedy-policy at the Episode checkpoints. Plot all four lines on the same figure.

2 Arm Elimination: An alternative to UCB (30 Points + 20 Bonus points)

In this question, we will study an alternative algorithm for multi-armed bandits, which works in pre-defined batches.

Define a sequence $\epsilon_i = 2^{-i}$ for $i = 1, 2, 3, ...$ The algorithm proceeds in batches $i = 1, 2, ...$ Each Batch i receives a subset $\mathcal{A}_i \subset \mathcal{A}$ sample each arm by N_i times (we will figure out what a good choice of N_i is soon).

Let $\hat{Q}_a^i = \frac{\sum_{j=1}^{N_i} R_{a,j}}{N_i}$ where $R_{a,j}$ is the jth sample within the ith batch that uses Arm a. Then we set

$$\mathcal{A}_{i+1} = \left\{ a \in \mathcal{A}_i \;\middle|\; \max_{a' \in \mathcal{A}i} \hat{Q}_{a'}^i - \hat{Q}_a^i \leq 2\epsilon_i \right\}$$

To say it differently, based on the outcome of Batch i, we elimination those Arms that are no longer "contenders" for the optimal arm.

(* Notice that we will *throw away all data from the previous batch*, for the simplicity of the analysis.)

(a) (5 Points) Given $\mathcal{A}_i$ and N_i, apply Hoeffding's inequality to show that with probability $1-\delta_i$, for all $a \in \mathcal{A}_i$

$$\left| \hat{Q}_a^i - Q(a) \right| \leq \sqrt{\frac{\log(\frac{2|\mathcal{A}_i|}{\delta_i})}{2N_i}}.$$

(Why don't you need the Azuma-Hoeffding's inequality in this case like in UCB?)

(b) (5 points) What is a good choice of $N_1, N_2, ...$ such that we can show that with probability $1-\delta$,

$$\sup_{a \in \mathcal{A}_i} \left| \hat{Q}_a^i - Q(a) \right| \leq \epsilon_i$$

for all $i = 1, 2, 3, 4, 5, ...$

(Hint: the algorithm is supposed to be an any-time algorithm, i.e., we do not know the horizon T ahead of time. So in order for the infinite sequence of δ_i to sum to δ, how should be design δ_i as a function of i? We have seen this in the lecture.)

(c) (5 points) Under the (high probability) event above, what is the regret of Epoch i?

(d) (5 points) Show that the total regret of this algorithm is $\tilde{O}(\sqrt{kT})$, where $\tilde{O}$ hides the logarithmic factors.

(e) (5 points) Show that the algorithm also satisfy the gap-dependent regret bound of $\tilde{O}(\sum_{a \neq a^*} \frac{1}{\Delta_a})$.

(f) (5 points) Let the switching cost be defined to be the total number of times you change your actions throughout the T steps. What is the best switching cost (in Big O notation, without hiding logarithmic factors) you can obtain with this algorithm?

(g) (Bonus 10 points) Consider the alternative algorithm without a pre-defined sequence of batches, but rather go by a round-robin fashion throughout based on the remaining action space $\mathcal{A}_t$. Let

$$\hat{Q}_t(a) = \frac{1}{\max\{1, N_t(a)\}} \sum_{i=1}^{t-1} \mathbf{1}(A_t = a) R_t.$$

where $N_t(a)$ is the total number of times arm a was pulled within the first $t-1$ steps.

Set $\mathcal{A}_1 = \mathcal{A}$. We update $\mathcal{A}_t$ in every iteration by

$$\mathcal{A}_{t+1} = \left\{ a \in \mathcal{A}_t \;\middle|\; \max_{a' \in \mathcal{A}t} \hat{Q}_t(a') - \epsilon_t(a') - (\hat{Q}_a^t + \epsilon_t(a)) \leq 0 \right\}$$

Show that there are appropriate choices of $\epsilon_t(a)$ that your algorithm can choose so you obtain qualitatively the same regret bound as before.

(Hint: Comparing to the standard batched arm-elimination algorithm, one of the challenges is that we cannot naively apply the standard Hoeffding's inequality any more.)

(h) (Bonus 10 points) What happens if, instead of round robin, you run a policy that chooses the arms in $\mathcal{A}_t$ uniformly at random? How many times do you need to change the (stochastic) policy in total?

3 Absorbing MDPs (50 points)

***Acknowledgment:** This question is taken from Wen Sun and Sham Kakade .

Consider an infinite horizon discounted MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \gamma, r, P, s_0\}$. Let us assume $\mathcal{S}$ and $\mathcal{A}$ are discrete here, and $r(s,a) \in [0,1]$. We will consider stationary policy π here, and recall $d^\pi(s,a)$ is the (normalized) state-action visitation of π under transition P, i.e.,

$$d^\pi(s,a) = (1-\gamma)\sum_{h=0}^{\infty} \gamma^h \mathbb{P}_h^\pi(s,a),$$

where $\mathbb{P}_h^\pi(s,a)$ is the probability of π hitting (s,a) at time step h under model P starting at s_0 at $h=0$.

Let us also assume that we are given a dataset $\{s_i, a_i, s_i'\}_{i=1}^N$ that consists of N state-action-next state triples, where $s_i' \sim P(\cdot|s_i,a_i)$. Following what we did in the tabular UCBVI lecture, we will define two statistics: $N(s,a) = \sum_{i=1}^N \mathbf{1}\{s_i, a_i = s, a\}$ and $N(s,a,s') = \sum_{i=1}^N \mathbf{1}\{s_i, a_i, s_i' = s, a, s'\}$, and we define $\widehat{P}$ as:

$$\widehat{P}(s'|s,a) = \frac{N(s,a,s')}{N(s,a)}, \forall s, a, s'.$$

For simplicity, let us assume $N(s,a) \geq 1$ for all s, a. We define a *known* set $\mathcal{K} \subset \mathcal{S} \times \mathcal{A}$ as follows:

$$\mathcal{K} = \{s, a \in \mathcal{S} \times \mathcal{A} : N(s,a) \geq k\},$$

where $k \in \mathbb{N}^+$ is some pre-defined constant integer and let us think about k as some reasonably big number. So from the lecture, we know that for any $s, a \in \mathcal{K}$, we have a reasonably good estimate of $\widehat{P}(\cdot|s,a)$, i.e., we can formulate the following high-probability event: *with probability at least* $1-\delta$*, for all* $s, a \in \mathcal{K}$*, (by a McDiarmid's inequality argument) we have that:*

$$\left\| \widehat{P}(\cdot|s,a) - P(\cdot|s,a) \right\|_1 \leq \sqrt{\frac{S \ln(SA/\delta)}{k}}, \forall s, a \tag{1}$$

Again let's think about k as a big number here, so that $\widehat{P}(\cdot|s,a)$ is accurate for any pair $s, a \in \mathcal{K}$ (In the generative model setting, $\mathcal{K} = \mathcal{S} \times \mathcal{A}$, but here some state-action pairs are not visited for sufficiently many times). For now, let us just condition on the above inequality being true.

Of course, outside $\mathcal{K}$, we cannot really guarantee a good accurate model. The question we want to study here is that *given the current data, how can we encourage the agent to visit unknown state-action pairs*?

Let us build the following absorbing MDP structure. Let us define an absorbing state $s^\dagger$ (note that $s^\dagger$ is some additional state we add to $\mathcal{S}$). Let us define the following two absorbing MDPs.

$$P^\dagger(\cdot|s,a) = \begin{cases} P(\cdot|s,a) & s,a \in \mathcal{K}, \\ \delta_{s^\dagger} & s = s^\dagger \text{ or } s,a \notin \mathcal{K}, \end{cases}, \quad \widehat{P}^\dagger(\cdot|s,a) = \begin{cases} \widehat{P}(\cdot|s,a) & s,a \in \mathcal{K}, \\ \delta_{s^\dagger} & s = s^\dagger \text{ or } s,a \notin \mathcal{K}, \end{cases}$$

where $\delta_{s^\dagger}$ is a delta distribution that has probability 1 at $s^\dagger$. Namely, in both models, whenever we hit a state-action pair not from $\mathcal{K}$, we transit to $s^\dagger$ immediately, and then we will just self-loop at $s^\dagger$ forever.

Let us define a reward function for both MDPs,

$$r^\dagger(s,a) = \begin{cases} r(s,a) & s,a \in \mathcal{K}, \\ 1 & s = s^\dagger \text{ or } s,a \notin \mathcal{K}. \end{cases}$$

At this stage, we have setup all things we want. Let's see how we can use the absorbing structures we constructed above.

3.1 Probabilities and State action Visitations (30 points)

Given a policy π, let us denote $d^\pi_{P^\dagger}(s,a)$ as the state-action visitation under $P^\dagger$ and $d^\pi(s,a)$ as the state-action visitation under the true model P. We denote $d^\pi_{P^\dagger}(s) = \sum_{a \in \mathcal{A}} d^\pi_{P^\dagger}(s,a)$.

(a) (10 points) Prove that:

$$d^\pi(s,a) \geq d^\pi_{P^\dagger}(s,a), \forall s,a \in \mathcal{K}.$$

(b) (10 points) Recall that for $P^\dagger$, we have an extra absorbing state $s^\dagger$. Prove that:

$$d^\pi_{P^\dagger}(s^\dagger) = \frac{\gamma}{1-\gamma} \sum_{s,a \in (\mathcal{S} \times \mathcal{A}) \setminus \mathcal{K}} d^\pi_{P^\dagger}(s,a),$$

where $d^\pi_{P^\dagger}(s^\dagger) = \sum_a d^\pi_{P^\dagger}(s^\dagger, a)$

(c) (10 points) Let us consider $s,a \in (\mathcal{S} \times \mathcal{A}) \setminus \mathcal{K}$. Prove that:

$$d^\pi_{P^\dagger}(s,a) \leq d^\pi(s,a).$$

Hint: Remember that $\mathcal{M}$ does not contain the state $s^\dagger$. That is $\forall s,a \in \mathcal{S} \times \mathcal{A}$, $P(s^\dagger|s,a) = 0$. It may be helpful to think about what kind of (s,a) an agent needs to be in for $P^\dagger(s^\dagger|s,a) \neq 0$.

3.2 Optimism from the Absorbing MDPs (10 points)

Let us consider an arbitary stationary stochastic policy $\pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$. Denote $V^{\dagger,\pi}$ as the policy π's expected discounted total reward under $P^\dagger$ and $r^\dagger$. Prove the following inequality: for any π.

$$V^{\dagger,\pi} \geq V^\pi.$$

Namely we have found an upper bound of V^π.

3.3 Not Too Much Optimism (10 points)

Being optimism itself is not enough for efficient learning, after all, $+\infty$ is always a valid upper bound of V^π. We want the gap between $V^{\dagger,\pi}$ and V^π not too big. Prove the following:

$$V^{\dagger,\pi} \leq V^\pi + \frac{1}{(1-\gamma)^2} \sum_{s,a \in (\mathcal{S}\times\mathcal{A})\setminus\mathcal{K}} d^\pi(s,a),$$

i.e., the gap is upper bounded by the probability of π escaping the known set $\mathcal{K}$.

Hint: first write $V^{\dagger,\pi}$ and V^π using the state-action distributions. Second, group state-action pairs into three groups: the known, the unknown, and the additional $s^\dagger$. Reason each term, and use the results you proved in Section 3.1.

Remark: So in summary, we have seen another approach of providing optimism: we construct absorbing MDP where for any unknown state-action pair, we simply let it be absorbed into $s^\dagger$ and give the maximum possible reward one for the unknown state-action pair and $s^\dagger$. We also show that the optimism gap is related to the probability of escaping the known set $\mathcal{K}$. We will continue in Homework 3 to see how this construction gives an a polynomial sample complexity exploration algorithm in tabular MDPs.