Assignment 1 COMP 767
January 9, 2020 **Reinforcement Learning** **Due: Jan 29, 2020**

All the questions are mandatory unless otherwise stated. For assignment instructions and submission guidelines please refer to the instructions document.

1 Theory

Question 1. [15 points] **Multi-Arm Bandits**
Consider the *pure exploration* setting where a learner is confronted with a *multi-arm* bandit problem defined by K arms with reward distributions $\{R_i\}_{i=1}^{K}$. Each R_i has finite set of points not mapped to 0 bounded support in $[0, 1]$ and mean μ_i. The learner explores the arms for a number of steps and has to pick the best one after T steps: $\mu^* = \max_{i=[K]} \mu_i$. Imagine the simple setting where the agent chooses each arm uniformly for a fixed number of trials T and then picks the one with the highest mean (each arm will be pulled T/K times). Derive the value of T, i.e. the number of arm pulls - needed to guarantee that $\mu^* - \mu_{\widehat{i}} \leqslant \varepsilon$, $\forall \varepsilon \geqslant 0$ with probability $1 - \delta$, $\delta \in (0, 1)$, where $\widehat{i}$ is the arm the agent chooses after T steps of interaction. T should be in big O notation as a function of ε and δ. (*Tip: use union bound and a concentration inequality, e.g. Hoeffding's inequality*).

Question 2. [10 points] **Markov Decision Processes**
In this question we consider two MDPs which differ slightly. You will analyze how the value function differs for these MDPs.

i Consider two MDPs $M = (S, A, P, R, \gamma)$ and $\bar{M} = (S, A, P, \bar{R}, \gamma)$, which only differ in their reward functions. We have for any $s \in S$, $a \in A$, $\bar{R}(s, a) = R(s, a) + \mathcal{N}(\mu, \sigma^2)$, where $\mathcal{N}$ is a Gaussian with constant mean μ, and variance σ^2 that does not depend on s or a. For any policy π, let V_M^π denote its value function in M and $V_{\bar{M}}^\pi$ denote its value function in $\bar{M}$. For any $s \in S$, express $V_{\bar{M}}^\pi(s)$ in terms of μ and $V_M^\pi(s)$.

ii In the previous part, we saw how the values differs if the reward function is changed. In this question we explore the effects if the transition matrix is changed. Consider two MDPs $M = (S, A, P, R, \gamma)$ and $\bar{M} = (S, A, \bar{P}, R, \gamma)$ where $\bar{P} = (\alpha P + \beta Q)$, $\alpha \in [0, 1]$, $\beta \in [0, 1]$ are constants such that $\alpha + \beta = 1$, Q is another transition matrix with the same dimension as P. Let V_M^π denote the value function in M and $V_{\bar{M}}^\pi$ denote the value function in $\bar{M}$. For any $s \in S$, express $V_{\bar{M}}^\pi(s)$ in terms of $V_M^\pi(s)$.

Question 3. [15 points] **Policy Evaluation and Improvement**
Consider a discrete, finite state MDP whose optimal value is $V^\star$. Let $\gamma \in [0, 1)$ be

the discount factor. Consider any value function $\widehat{V}$ such that $|V^\star(s) - \widehat{V}(s)| \leqslant \varepsilon$ for all states $s \in S$. Prove that $L_{\widehat{V}}(s) \leqslant \frac{2\gamma\varepsilon}{1-\gamma}$ for all s, where $L_{\widehat{V}}(s) = V^\star(s) - V_{\widehat{V}}(s)$, $V_{\widehat{V}}$ is the value function obtained after evaluating the greedy policy with respect to $\widehat{V}$.

2 Coding

In addition to the instructions provided for each question, please make sure you abide by the following general rules of reproducible and meaningful research:

- When applicable please do a thorough hyperparameter search. A detailed report of the hyperparameter search applicable to the algorithm has to be included in the submission *(Justify the choice of hyperparameters with supporting plots).*
- Each experiment has to be run on different seeds as specified in each individual question. Mean curve along with variance should be highlighted in the plots.

For more information regarding reproducibility and reporting of results, please refer to the instructions file posted along with the assignment.

Question 1. [30 points] **Explore-Exploit in Bandits** *(Choose* ***one*** *of the following two tracks)*

A. Implement Boltzmann (softmax) exploration, the UCB algorithm and Thompson sampling algorithm. Compare the performance of these algorithms on the 10-armed problem described in the course textbook (which we also discussed in class). Please plot three graphs: average training return (averaged over 10 independent runs), average regret per step (which you can compute in this case, as we know apriori the best arm) and testing plots (showing the reward of the policy, computed after every 10 pulls, and averaged over 5 pulls). Explain what you expected to see and what you actually saw.

B. Implement UCB and Thompson sampling for a simple 2-arm bandit setting described in the following parts.

i Consider the setting where the reward distributions of the bandit arms are independent Bernoulli (rewards for each arm i is 1 with probability p_i and 0 with probability $1-p_i$), where the parameter p_i for each arm are sampled independently and uniformly over $[0, 1]$. Test UCB and Thompson sampling algorithms in this setting.

ii In this part, consider Bernoulli distributions where the parameters p_1 and p_2 of the two arms are correlated. Specifically, $p_2 = 1 - p_1$. Consider the following settings for selecting the parameter p_1: (a) *uniform* - means that $p_1 \sim U([0, 1])$ (uniform distribution over the unit interval), (b) *easy* - means that $p_1 \sim U(0.1, 0.9)$ (uniform distribution over those two possible values) (c) *medium* - $p_1 \sim U(0.25, 0.75)$ and (d) *hard* - $p_1 \sim U(0.4, 0.6)$. For each setting, please plot the following graphs: average training return (averaged over 10 independent runs), average regret per step (which you can compute in this case, as we know a priori the best arm) and a testing plots (showing the reward of the optimal policy, computed after every 10 pulls, and averaged over 5 pulls). Explain what you expect to see and what you actually saw.

Question 2. [30 points] **Dynamic Programming** *(Choose* **one** *of the following two tracks)* Pick two tabular environments of your choice where the transition probabilities and the reward function are known. Some examples of tabular environment include Frozen lake (Brockman et al., 2016), Taxi environment (Dietterich, 1998), grid world, MiniGrid, towers of Hanoi (Abel, 2019), etc. You can either use gym environments or design your own environment in python.

A. Implement tabular policy iteration and value iteration to find the optimal value function and optimal policy.

i Report the training performance of these methods with the following plots: (i) cumulative reward per episode obtained by the agent, and (ii) the number of timesteps to solve the task per episode of experience, both reported over 5 different seeds (as specified in the beginning of the coding section).

ii For the test performance, after every 10 episodes of training (for every seed), run the estimated policy in the environment for 5 test episodes and report the mean over (i) cumulative reward per episode obtained by the agent, and (ii) the number of timesteps required to solve the task per episode of experience. You will extend the reporting scheme over seeds from training.

iii In both training and testing plots, highlight the optimal performance (maximum return in an episode / minimum number of timesteps per episode) in a dotted line.

iv Please provide your own observations with respect to the results obtained. Also, provide the time complexity, space complexity of these algorithms.

B. Implement the following variants of value iteration and policy iteration to find the optimal value function and optimal policy.

i Modified policy iteration: instead of fully (to convergence) evaluating a policy, just compute a few Bellman backups and then improve the policy (See section 6.5 of (Puterman, 1994)).

ii Asynchronous value iteration, Gauss-Seidel and Jacobi variants of value iteration (See section 6.3.3 of (Puterman, 1994)).

For both the parts,

- Report the training performance of these methods with the following plots: (i) cumulative reward per episode obtained by the agent, and (ii) the number of timesteps required to solve the task per episode of experience, reported over 5 different seeds (as specified in the beginning of the coding section).
- For the test performance, after every 10 episodes of training (for every seed), run the estimated policy in the environment for 5 test episodes and report the mean over (i) cumulative reward per episode obtained by the agent, and (ii) the number of timesteps required to solve the task per episode of experience. You will extend the reporting scheme over seeds from training.
- In both training and testing plots, highlight the optimal performance (maximum return in an episode/minimum number of timesteps per episode) in a dotted line.
- Please provide your own observations with respect to the results obtained, specifically in terms of speed of convergence to the right optimal solutions.

References

Abel, David (2019). "simple rl: Reproducible Reinforcement Learning in Python". In:

Brockman, Greg et al. (2016). *OpenAI Gym.* eprint: `arXiv:1606.01540`.

Dietterich, Thomas G. (1998). "The MAXQ Method for Hierarchical Reinforcement Learning". In: *Proceedings of the Fifteenth International Conference on Machine Learning.* ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 118–126. ISBN: 1558605568.

Puterman, Martin L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* 1st. USA: John Wiley and Sons, Inc. ISBN: 0471619779.