AIR5066/MDS5121: Reinforcement Learning **CUHKSZ**

Assignment 2

TA: Qibing Bai

Due Date: November 11th, 11:59 pm
Total points available: 100 pts.

Note: *Please note that external references are totally allowed only if you give appropriate reference. There is no required format of reference. Please elaborate on your answers as well. (not just give a number etc)*

Problem 1: Markov Decision Process [30 pts.]

Consider a treacherous mountain with $n+1$ ledges, sequentially numbered from 0 to n. Ledge 0 is the base camp, and ledge n is the summit. A climber finds themselves on a ledge i (where $1 \leq i \leq n-1$). From any of these intermediate ledges, the climber must decide on their next move.

They have two techniques available: a **'Careful Step'** or a **'Risky Grapple'**.

- If they choose a **'Careful Step'** from ledge i, they attempt a calculated move to an adjacent ledge. However, the mountain is perilous. The higher they are, the greater the effects of wind and fatigue. A slip will send them down to ledge $i-1$ with probability $\frac{i}{n}$, while a successful maneuver moves them up to ledge $i+1$ with probability $\frac{n-i}{n}$.
- If they choose a **'Risky Grapple'** from ledge i, they throw a grappling hook towards a distant point. This is a wild, unpredictable move. The hook can anchor onto any of the other n ledges $(0, \ldots, i-1, i+1, \ldots, n)$ with a uniform probability of $\frac{1}{n}$ for each.

Landing on ledge 0 (the base camp) means the climb has failed, and the climber must retreat. Reaching ledge n (the summit) means the climber has succeeded and escaped the dangers of the mountain.
What technique should the climber use on each of the ledges $1, 2, \ldots, n-1$ to maximize the probability of reaching the summit (ledge n) before failing the climb (landing on ledge 0)? We will solve this by modeling it as a Markov Decision Process (MDP) and finding the Optimal Policy.

1. Express with clear mathematical notation the state space, action space, transition function, and reward function of an MDP so that the above *climber's ascent* problem is solved by arriving at the Optimal Value Function (and hence, the Optimal Policy) of this MDP.
2. Write working Python code (with type annotations and comments) or pseudo-code that models this MDP and solves for the Optimal Value Function and Optimal Policy.

Problem 2: Markov Reward Process [15 pts.]

Imagine an autonomous agent operating in a discrete, stochastic environment with n distinct states. The agent has a perfect model of this world, which is an infinite-horizon Markov Reward Process (MRP) defined by the tuple $(\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a finite state space with n states, $\mathcal{P}$ is the $n \times n$ state-transition matrix, $\mathcal{R}$ is an $n \times 1$ reward vector, and $\gamma \in [0, 1)$ is the discount factor.

The true value function for this MRP, denoted $\mathcal{V}^*$, is an $n \times 1$ vector. We wish to approximate this value function using a linear model based on a set of m features. These features are provided as an $n \times m$ matrix Φ, where each row corresponds to the feature vector for a particular state. The approximated value function is thus given by $\hat{\mathcal{V}} = \Phi w$, where w is an $m \times 1$ vector of weights.

What is the necessary and sufficient condition on the feature matrix Φ that guarantees the existence of a weight vector w such that the linear approximation is exact (i.e., $\hat{\mathcal{V}} = \mathcal{V}^*$)? Express your condition in terms of the given MRP parameters ($\mathcal{P}$, $\mathcal{R}$, γ).

Problem 3: Value and Policy Iteration [20 pts.]

Consider a simple MDP with 3 states s_1, s_2, s_3 and 2 actions a_1, a_2. The transition probabilities and expected rewards are:

$$s_1 : \begin{cases} a_1 : \big(\{s_1 : 0.2, s_2 : 0.6, s_3 : 0.2\}, 8.0\big), \\ a_2 : \big(\{s_1 : 0.1, s_2 : 0.2, s_3 : 0.7\}, 10.0\big) \end{cases}$$

$$s_2 : \begin{cases} a_1 : \big(\{s_1 : 0.3, s_2 : 0.3, s_3 : 0.4\}, 1.0\big), \\ a_2 : \big(\{s_1 : 0.5, s_2 : 0.3, s_3 : 0.2\}, -1.0\big) \end{cases}$$

$$s_3 : \begin{cases} a_1 : \big(\{s_3 : 1.0\}, 0.0\big), \\ a_2 : \big(\{s_3 : 1.0\}, 0.0\big) \end{cases}$$

E.g. The transition probability of $P(s_1, a_1, s_2) = 0.6$, and the reward of $R(s_1, a_1) = 8.0$. Assume discount factor $\gamma = 1$.

1. Initialize the value function for each state to be it's max (over actions) reward, i.e., we initialize the Value Function to be $V_0(s_1) = 10.0, V_0(s_2) = 1.0, V_0(s_3) = 0.0$. Then manually performs two iterations of value iterations. Show the final values for each state and the computation process.
2. Let $\pi_k(s)$ denotes the extracted policy after k iterations of value iteration. Argue that $\pi_k(s) = \pi_2(s)$ for all states s.
3. What are the key distinctions between the value iteration and policy iteration algorithms, and when might you prefer one to the other?
4. What changes if during policy iteration, you only run one iteration of Bellman update instead of running it until convergence? Do you still get an optimal policy?

Problem 4: Value and Policy Iteration Implementation [35 pts.]

In this problem, you are going to implement value and policy iteration and perform experiments with OpenAI Gym environment. Skeleton codes (in python) are provided through Google Colab (you can use it for free with the computation demands of this assignment), and here is the link to the skeleton code. You will need to **create a copy of this colab notebook in your own google drive**.

0.1 Implementation

Please implement both value and policy iteration.

Submit your code a share link from google drive. No mark will be rewarded if your code cannot be ran.

0.2 Experiment

Please try to play around with the Frozen Lake environment with different configurations (gamma, improvement/evaluation iterations, random seed) and with value and policy iteration. Please try your best to get a policy with a score as high as possible (though your grades will not be affected by this score). Repeat each experiment with 5 different random seeds, then report the highest score you've achieved across the 5 repeats with 1 standard deviation and write (a few sentences) to compare the two iteration methods.