Stanford CME 241 (Winter 2020) - Takehome Midterm Exam

Jupyter notebook with answers to be turned in by 12:00AM Wed Feb 12 (by Piazza private message)

1. **(Estimated Difficulty: 1 hour of work) Coding Optimal Actions on Lilypads.** Consider an array of $n+1$ lilypads on a pond, numbered 0 to n. A frog sits on a lilypad other than the ones numbered 0 or n. When on lilypad i ($1 \leq i \leq n-1$), the frog can take one of two actions A and B. If it takes action A when on lilypad i ($1 \leq i \leq n-1$), it is thrown to lilypad $i-1$ with probability $\frac{i}{n}$ and is thrown to lilypad $i+1$ with probability $\frac{n-i}{n}$. If it takes action B when on lilypad i ($1 \leq i \leq n-1$), it is thrown to one of the lilypads $0, 1, \ldots, n$ with equal probability $\frac{1}{n+1}$ (note one of these $n+1$ lilypads is the lilypad i itself). A snake, perched on lilypad 0, will eat the frog if the frog lands on lilypad 0. The frog can escape the pond (and hence, escape the snake!) if it lands on lilypad n.

   What actions must the frog take at each of the lilypads $1, 2, \ldots, n-1$, in order to maximize the probability of escaping the pond (i.e., reaching lilypad n before reaching lilypad 0)? Although there are more than one ways of solving this problem, we'd like to solve it by modeling it as an MDP and identifying the Optimal Policy.

   - **3 points:** Express with clear mathematical notation the state space, action space, transitions function and rewards function of an MDP so that the above *frog-escape* problem is solved by arriving at the Optimal Value Function (and hence, the Optimal Policy) of this MDP.
   - **6 points:** Write working Python code (with type annotations and comments) that models this MDP and solves the Optimal Value Function and Optimal Policy (you can re-use any code you have written previously as part of this course's suggested assignments).
   - **3 points:** Plot a graph of the Optimal Escape-Probability and of the associated Optimal Action, as a function of the states of this MDP, for $n = 3, n = 10, n = 25$. Make qualitative and informal arguments for why the frog makes these optimal actions in the various states.

2. **(Estimated Difficulty: 30 minutes of work) Job-Hopping and Wage-Maximization**. You are a worker who starts every day either employed or unemployed. If you start your day employed, you work on your job for the day (one of n jobs, as elaborated later) and you get to earn the wage of the job for the day. However, at the end of the day, you could lose your job with probability $\alpha \in [0,1]$, in which case you start the next day unemployed. If at the end of the day, you do not lose your job (with probability $1-\alpha$), then you will start the next day with the same job (and hence, the same daily wage). On the other hand, if you start your day unemployed, then you will be randomly offered one of n jobs with daily wages $w_1, w_2, \ldots w_n \in \mathbb{R}^+$ with respective job-offer probabilities $p_1, p_2, \ldots p_n \in [0,1]$ (with $\sum_{i=1}^n p_i = 1$). You can choose to either accept or reject the offered job. If you accept the job-offer, your day progresses exactly like the *employed-day* described above (earning the day's job wage and losing the job at the end of the day with probability α). However, if you reject the job-offer, you spend the day unemployed, receive the unemployment wage $w_0 \in \mathbb{R}^+$ for the day, and start the next day unemployed. The problem is to identify the optimal action at the start of each unemployed day in a manner that maximizes the infinite-horizon *Expected Discounted-Sum of Wages Utility*. Assume the daily discount factor for wages (employed or unemployed) is $\gamma \in [0,1)$. Assume CRRA utility function $U(w) = \frac{w^{1-a}}{1-a}$ for CRRA risk-aversion parameter $a \in \mathbb{R}$ (for $a = 1$, $U(w) = \log w$). So you are looking to maximize

$$\mathbb{E}[\sum_{u=t}^{\infty} \gamma^{u-t} \cdot U(w_{i_u})]$$

at the start of a given day t (w_{i_u} is the wage earned on day u, $0 \le i_u \le n$ for all $u \ge t$).

- **4 points**: Express with clear mathematical notation the state space, action space, transition function, reward function, and write the Bellman Optimality Equation customized for this MDP.
- **3 points**: You can solve this Bellman Optimality Equation (hence, solve for the Optimal Value Function and the Optimal Policy) with a numerical iterative algorithm (essentially a Dynamic Programming algorithm customized to this problem). Write pseudo-code for this numerical algorithm. Clearly define the inputs and outputs of your algorithm with their types (int, float, List, Mapping etc.). You can choose to write actual Python code to implement this algorithm (if you so wish), but you will get full points for simply writing clear pseudo-code.

3. **(Estimated Difficulty: 30 minutes of work) Solving a continuous states/actions MDP analytically**. **6 points:** Consider a continuous-states, continuous-actions, discrete-time, infinite-horizon MDP with state space as $\mathbb{R}$ and action space as $\mathbb{R}$. When in state $s \in \mathbb{R}$, upon taking action $a \in \mathbb{R}$, one transitions to next state $s' \in \mathbb{R}$ according to a normal distribution $s' \sim \mathcal{N}(s, \sigma^2)$ for a fixed variance $\sigma^2 \in \mathbb{R}^+$. The corresponding cost associated with this transition is $e^{as'}$, i.e., the cost depends on the action a and the state s' one transitions to. The problem is to minimize the infinite-horizon *Expected Discounted-Sum of Costs* (with discount factor γ). For the purpose of this exam, solve this problem just for the special case of $\gamma = 0$ (i.e., the myopic case) using elementary calculus. Derive an analytic expression for the optimal action in any state and the corresponding optimal cost.