Stanford CME 241 (Winter 2021) - Midterm Exam

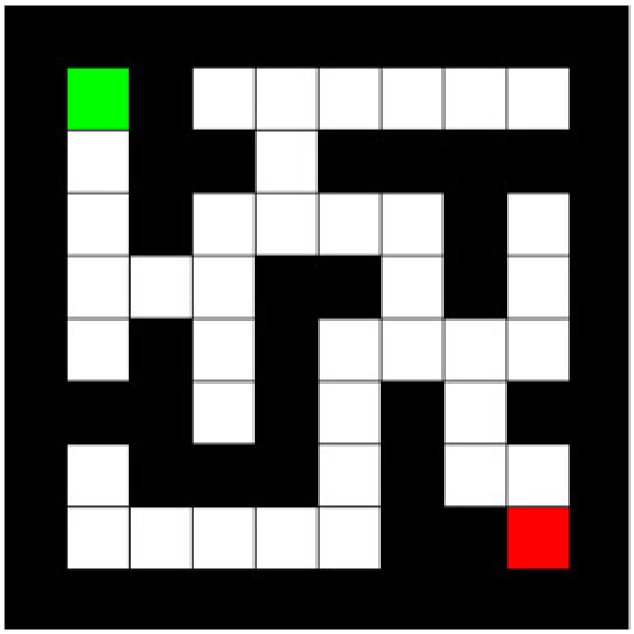
Instructions:

- We trust that you will follow The Stanford Honor Code.
- You have about 74 hours to work on this test, and need to submit your answers by 11:59pm Wednesday February 17. However, the estimated amount of work for this test is about 3-4 hours, depending on your proficiency in typesetting mathematical notations and in writing code. There are 3 problems (with subproblems), with a total of 30 points.
- Include all of your writing, math formulas, code, graphs etc. into a single answers-document. Be sure to write your name and SUNET ID in your answers-document. You can do your work in LaTeX or Markdown or Jupyter or any other typesetting option, which should finally be converted to the single answers-document PDF to be submitted. Note: Code can be included in LaTeX using the *lstlisting* environment, and graphs can be included using *includegraphics*.
- Submit your answers-document on Gradescope. Please ensure you have access to Gradescope before starting the exam so there is ample time to correct the problem. The Gradescope assignment for this midterm is here.

Problems:

1. **Value Iteration Optimization.** We have considered the Value Iteration algorithm in this class to solve finite state / action MDPs. In this problem we will consider how reward function choices effect the convergence of the algorithm.

   Consider the grid maze below:



   Each square on the grid is either an open space (white squares), or a block (black squares). You can move between open spaces, you cannot move through blocks. We have provided a dictionary of the maze shown in grid_maze.py. We would like to use an MDP formulation and solution to calculate the optimal path through the maze from every point. The goal square in the displayed example grid maze is (7,7) at the bottom right corner. At each step, if you are not in the goal square, you must move one of (Up, Down, Left, Right) to a neighboring space. The maze ends when you reach the goal square.

   - **5 points:** Express with clear mathematical notation of the state space, action space, state transition probability function $\mathcal{P}$, **two formulations** of the reward transition function $\mathcal{R}_T$ and discount rate of an MDP so that the above *maze* problem is solved by arriving at the Optimal Value Function (and hence, the Optimal Policy) of this MDP. Hint: *One formulation can have a **reward for each transition** with a discount factor of $\boldsymbol{\gamma = 1}$, while another formulation can have a **reward only for transitioning into the goal state** with a discount factor of $\boldsymbol{\gamma < 1}$.*

Solution:

$$
\begin{aligned}
\mathcal{N} &= \{(x,y) | (x,y) \in ([0,M) \times [0,N)),\ \text{maze}[(x,y)] = "SPACE"\} \\
\mathcal{T} &= \{(x,y) | (x,y) \in ([0,M) \times [0,N)),\ \text{maze}[(x,y)] = "GOAL"\} \\
\mathcal{S} &= \mathcal{T} \cup \mathcal{N} \\
\mathcal{A} &= \{L, R, U, D\} \\
\mathcal{P}((x,y), a, (x',y')) &= \begin{cases} 1 & \text{if } a = L,\ x' = x+1,\ y' = y,\ (x',y') \in \mathcal{S} \\ 1 & \text{if } a = R,\ x' = x-1,\ y' = y,\ (x',y') \in \mathcal{S} \\ 1 & \text{if } a = U,\ x' = x,\ y' = y-1,\ (x',y') \in \mathcal{S} \\ 1 & \text{if } a = D,\ x' = x,\ y' = y+1,\ (x',y') \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases} \\
\mathcal{R}_T((x,y), a, (x',y')) &= \begin{cases} -1 \end{cases}
\end{aligned}
$$

Sparse reward

$$
\mathcal{R}_T((x,y), a, (x',y')) = \begin{cases} 1 & \text{if } (x',y') \in \mathcal{T} \\ 0 & \text{otherwise} \end{cases}
$$

- **5 points:** For both of the MDP formulations defined in part 1: write working Python code (with type annotations and comments) that models the MDP and solves the Optimal Value Function and Optimal Policy. Your implementations should take as input a dictionary of $(x,y) \rightarrow \{"STATE", "BLOCK", "GOAL"\}$, corresponding to an arbitrary $n \times m$ grid maze; we have provided the input for the grid maze above in the file grid_maze.py. Your implementations should also track the number of of iterations it takes for value iteration to converge.

Solution: Soln 1 using the course libraries
Soln 2 written from scratch, in place value iteration

- **2 points:** Demonstrate that the optimal policies for both of your implementations match. Which required less iterations to converge?

Solution: Soln 1: 16 / 16 iterations of value iteration with the course repository default convergence settings.
Soln 2: 384 / 832 updates to the value function for *in place VI* for sparse / dense rewards

2. **MRP Value Function Approximation**. **6 points:** Consider a finite MRP with n states, specified as: $(\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma)$ where $\mathcal{P}$ represents the state-transition probability function and $\mathcal{R}$ represents the reward function. Assume there are no terminal states (so, $\mathcal{N} = \mathcal{S}$). Assume $\mathcal{P}$ is specified in the usual form as a $n \times n$ matrix, and $\mathcal{R}$ is specified in the usual form as a column vector of length n. We wish to approximate the value function of this MRP with a linear function based on m features that are specified as a $n \times m$ matrix Φ (each column gives a particular feature's value for all states, and each row is the feature vector for a particular state). Give as minimal as possible conditions on the features matrix Φ such that we can *exactly represent* the true value function (These conditions should depend only on given values $(\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma)$).

Solution: We start with the matrix bellman equations

$$V = \mathcal{R} + \gamma \mathcal{P} V$$

We see that if we are using linear value function approximation then we can write

$$\tilde{V} = \Phi \cdot w$$

where Φ is the $n \times m$ matrix of features for all states.

If $\tilde{V}$ is able to exactly capture V then we must have

$$\tilde{V} = \mathcal{R} + \gamma \mathcal{P} \cdot \tilde{V} \rightarrow (I - \gamma \mathcal{P}) \cdot \Phi \cdot w = \mathcal{R}$$

We see that with $\gamma < 1$, $I - \gamma \mathcal{P}$ is full rank. So a minimally sufficient condition for this to hold is that the vector $(I - \gamma \mathcal{P})^{-1} \cdot \mathcal{R}$ must lie in the span of the m vectors comprising the columns of the matrix Φ

3. **Career Optimization**. **12 points:**

Imagine you live in a world where every job is an hourly-wage job. You have H available hours in a day (for some fixed $H \in \mathbb{Z}^+$), and each morning when you leave your house, you can decide to split those H hours into:

- Hours spent on learning to get better at your current job (call it $l \in \mathbb{Z}_{\geq 0}$),
- Hours spent on searching for another job (call it $s \in \mathbb{Z}_{\geq 0}$), and
- Remaining $H - l - s$ hours spent on actually working on your current job.

If your job currently pays you at an hourly-wage of w dollars, then at the end of that day, you will be given a cash amount of $w \cdot (H - l - s)$ dollars. We assume that any hourly-wage w in our world is an integer in the finite set $\{1, 2, \ldots, W\}$ for some fixed $W \in \mathbb{Z}^+$.

Each employer has a wage model such that if you spend l hours on learning on a given day where your hourly-wage was w, then the employer sends you an email the next morning with that new day's hourly-wage of: $\min(w + x, W)$ where x is a Poisson random variable with mean $\alpha \cdot l$ for some fixed $\alpha \in \mathbb{R}^+$.

Each morning, with probability $\frac{\beta \cdot s}{H}$ for some fixed $\beta \in [0, 1]$, you will receive an email from another employer with a job-offer with hourly-wage of $\min(w + 1, W)$ where w was the hourly wage of the job you were on the previous day and s is the number of hours you spent on job-search the previous day.

You read all your emails before you leave your house in the morning. If another job is offered to you and if the hourly-wage of that job is greater than your current employer's hourly-wage stated in that morning's email, then you accept the other job. Otherwise you continue in your current job. Whichever job you decide to do, each morning when you leave your house, your decide how to split the H hours of that day into learning hours, job-searching hours and working hours.

Your goal is to maximize the Expected (Discounted) Wages earned over an infinite horizon (assume you never age and will live infinitely). Daily discount factor is a fixed $0 \leq \gamma < 1$).

- **3 points:** With proper mathematical notation, model this as a Finite MDP specifying the states, actions, rewards, state-transition probabilities and discount factor.
- **3 points:** Implement this MDP in python. If you wish, you may use the code in the git repo that you forked at the start of the course (eg: FiniteMarkovDecisionProcess), but if you prefer, you can implement it from scratch or using code you have written for the course previously (whichever is more convenient for you).
- **3 points:** Solve for the Optimal Value Function and Optimal Policy using Value Iteration. If you wish, you may use the code in the git repo that you forked at the start of the course (eg: value_iteration method in rl/dynamic_programming.py), but if you prefer, you can implement it from scratch or using code you have written for the course previously (whichever is more convenient for you).
- **3 points:** Plot a graph of the Optimal Policy (or Print the Optimal Policy) for the following configuration: $H = 10, W = 30, \alpha = 0.08, \beta = 0.82, \gamma = 0.95$. Provide an intuitive explanation for this Optimal Policy.

Solution:

Since this is an infinite-horizon problem, the state is simply the hourly-wage w of the current job. Hence, the state space is the finite set of integers: $\{1, 2, \ldots, W\}$. The action on any given day is the pair (l, s) and so, the action space is the set $\{(l, s) | 0 \leq l + s \leq H\}$. The reward r for this MDP on any given day, conditional on the state w and action (l, s) is equal to: $w \cdot (H - l - s)$. Let us denote the Poisson probability mass function with mean λ as f_λ and the corresponding cumulative probability function as F_λ. Let us denote the probability of job offer as $p(s) = \frac{\beta s}{H}$. Then the transition probability of the next state $w' \in \{1, 2, \ldots, W\}$, conditional on the state w and action (l, s) is given by the following:

$$\mathbb{P}[w' | (w, (s, l))] = \begin{cases} 0 & \text{if } w' < w \\ (1 - p(s)) \cdot f_{\alpha l}(0) & \text{if } w = w' < W \\ p(s) \cdot f_{\alpha l}(0) + f_{\alpha l}(1) & \text{if } w + 1 = w' < W \\ f_{\alpha l}(w' - w) & \text{if } w + 1 < w' < W \\ 1 & \text{if } w = w' = W \\ 1 - (1 - p(s)) \cdot f_{\alpha l}(0) & \text{if } w + 1 = w' = W \\ 1 - F_{\alpha l}(w' - w - 1) & \text{if } w + 1 < w' = W \end{cases}$$

The solution code is shown here. Running this code produces a plot of the Optimal Policy. From the plot, we see that for $w \leq 13$, it's optimal to spend all our time learning and for $w = 14$ or $w = 15$, it's optimal to spend all our time searching for a job, and for $w \geq 16$, it's optimal to spend all our time working to earn money. This makes intuitive sense if one thinks about our career journey through education phase to early stages of career and finally on to high-skill/high-experience stages.