AI 533 **Intelligent Agents and Decision Making** Winter 2025

OREGON STATE UNIVERSITY

School of Electrical Engineering and Computer Science

Total points: 100 Mini Project 1 Due date: Feb 10, 2025

Instructions: This project consists of two parts. Collaboration is not allowed on any part of this project. You are welcome to brainstorm coding practices (including data structures to use) or clarify your understanding of the question with your peers but you cannot code together or copy/re-use solutions. Document who you worked with for this project and cite websites from which you utilized any code (e.g., Stack Overflow, Stack Exchange, ChatGPT) in your code implementation. You must submit a *project report (.pdf file) and the code*.

Part A: Exact Methods for Solving MDPs (55 points)

Consider the gridworld in Figure 1, with two states covered in water and two states with wildfire. Each state is denoted as $\langle x, y, \text{water}, \text{fire}\rangle$. "Water" and "fire" are binary values, with 0 denoting the absence of water/fire and 1 denoting the presence of water/fire at location (x, y). The start state is denoted as $\langle 0, 0, 0, 0\rangle$ and the goal state is denoted by $\langle 3, 3, 0, 0\rangle$. The agent can move in all four directions. The agent succeeds with probability 0.8 and may slide to the neighboring cells with probability 0.1. Illustration of the transition probability for actions 'up' and 'right' are shown in Figure 1. If a move is invalid (such as moving into a wall), the agent will remain in that state with the corresponding probability. For example, when trying to move up in top-left cell (start state) of the grid, the agent will remain in that cell with probability 0.9 or move right with probability 0.1.

The agent receives a reward of $+100$ when it reaches the goal state. The agent receives a reward of -5 in the water states, -10 in widlfire states, and a reward of -1 in all other states. The process *terminates* when the agent reaches the goal state. The agent's objective is to maximize the expected reward it can obtain.

1. **(15 points)** Setup (code up) the environment for solving using value iteration and policy iteration.
2. **(25 points)** Implement value iteration for this problem with $\gamma = 0.3$ and $\gamma = 0.95$. Your code must output the policy and objective value for each setting. Include the resulting policies and the objective values in your report. Describe whether the policy in states leading to wildfire and water changed, as the discount factor is increased.
3. **(15 points)** Implement policy iteration for this problem with $\gamma = 0.95$. Your code must output the policy and objective value. Include the resulting policy and the objective value in your report. Is the policy and the objective value same as that of value iteration?

Part B: Imitation Learning (45 points)

1. **(15 points)** Policy simulation. Write a function to generate **an episode** for the grid in Part A, with $\gamma = 0.95$. Each episode must begin at the start state and ends at the goal state. An episode is of the form

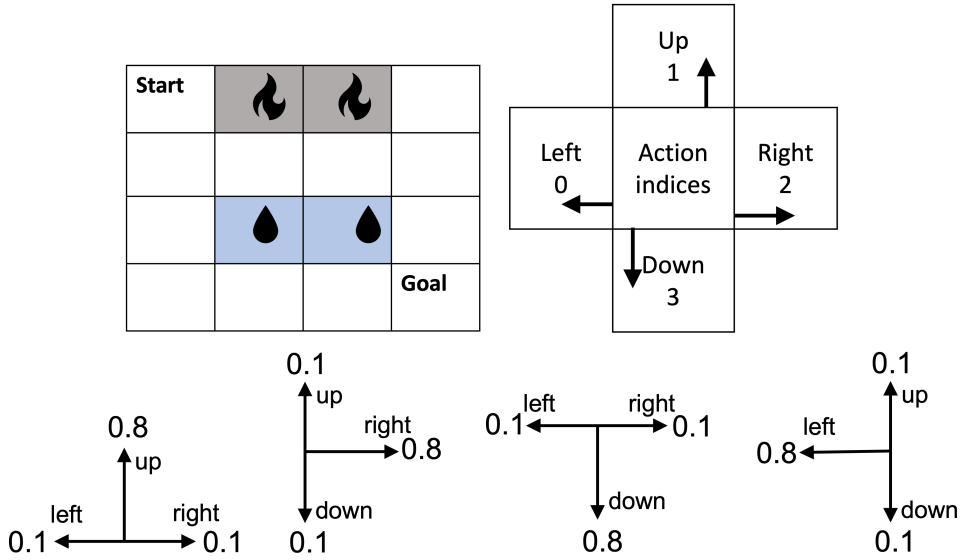

Figure 1: MDP

$\{(S_0, A_0, R_0), \ldots, (S_T, A_T, R_T)\}$ where S_0 is the start state, A_0 is $\pi(S_0)$, R_0 is the corresponding reward, S_T is the Goal state, A_T is $\pi(S_T)$ and R_T is the corresponding reward ($R_t(s, a), \forall t$). Given a policy π as input, the function must output an episode (or trajectory). The trajectory must follow the state transition function T. For example, when moving "right" at the start state, the function must select a *single* successor state among the fire state, moving down or remaining in the start state, proportional to their transition probabilities. Due to the stochastic transitions, the trajectories may be different each time you simulate the policy.

2. **(30 points)** We will implement DAGGER algorithm (Algorithm 3.1 in `https://arxiv.org/pdf/1011.0686`), with some modifications: (i) we will use the optimal policy from Part A to simulate the human expert and (ii) $\beta_i = 0$. The following are the high-level steps to follow in your implementation.

   (a) Initialize a random policy $\hat{\pi}$ such that is it not the optimal policy $\hat{\pi} \neq \pi^*$ and a empty dataset $D = \{\}$. For $i = 1$ to N, repeat steps (b) - (e).

   (b) Generate a trajectory τ_i using $\hat{\pi}$ and B.1. Create a data set D_i by annotating each state in trajectory τ_i with corresponding expert actions from π^* (policy computed using value iteration or policy iteration in Part A with $\gamma = 0.95$). Aggregate data set $D = D \cup D_i$.

   (c) Train a classifier of your choice using D. The training input is the states in D and the input labels are the corresponding actions in D.

   (d) For simplicity and consistency, the test set will consist of *all* states in the MDP.

   (e) Predict the action label for each state in the test set, using a classifier. This will produce a new policy $\hat{\pi}$.

Conduct experiments with $N = \{5, 10, 20, 30, 40, 50\}$ and plot accuracy. The x-axis of your graph is the N and the y-axis is the accuracy. The accuracy is calculated as the fraction of states for which the predicted action $\hat{\pi}$ in (e) matches the optimal policy π^* from Part A. Did the accuracy increase with an increase in N? Why or why not?