ECS 427/627: Multi-agent Reinforcement Learning:

Assignment 1

Instructions

- Use of existing MDP solvers or planning libraries is **not permitted**.
- Submit:
  - Source code
  - A short report (PDF, maximum 6 pages)
- Clearly explain assumptions, design choices, and experimental observations.

1 Question 1: Differential-Drive Robot with Orientation-Dependent Planning (35 Marks)

Consider a differential-drive mobile robot navigating a known indoor environment represented as a 2D grid with obstacles (10×10 grid with 5 obstacles places randomly). Unlike a standard gridworld, the robot's orientation is part of the state. Pick one of the cells as the goal.

State Space

$$s = (x, y, \theta)$$

where:

- x, y are grid coordinates
- $\theta \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$ is the robot orientation

Action Space

- `Forward`
- `TurnLeft`
- `TurnRight`

Transition Model

- `Forward`:
  - Moves one grid cell forward with probability 0.8
  - Slips sideways (relative to orientation) with probability 0.1 each
- `TurnLeft` and `TurnRight` are deterministic orientation changes

Collisions with obstacles or map boundaries result in a terminal state.

Reward Structure

- Each action: -1
- Collision: -100 (terminal)
- Reaching the goal: $+50$ (terminal)

Tasks

(a) Formulate the problem explicitly as an MDP.

(b) Evaluate Value iteration for 3 iterations and then implement the same in code. Show the first 3 iterations and compare the results with that of the hand written solution. Find the optimal policy.

(c) Evaluate Policy iteration for 3 iterations and then implement the same in code. Show the first 3 iterations and compare the results with that of the hand written solution. Find the optimal policy.

(d) Compare Value Iteration and Policy Iteration in terms of:

- Number of iterations to converge
- Runtime
- Memory usage

(e) Visualize the final policy using orientation-aware arrows.

(f) Change the reward structure to two different values and compare using the previous question metrics. Basically create a table for comparison and show that in code also.

(g) Implement the Monte-Carlo Method and compare with the value iteration and policy iteration

2 Question 2: Battery-Aware Robot Navigation (30 Marks)

A mobile robot must reach a goal location while managing a limited battery supply.

State Space

$$s = (x, y, b)$$

where:

- x, y are grid coordinates
- $b \in \{0, 1, \ldots, B\}$ denotes battery level

Action Space

- `Move`: consumes one unit of battery
- `Recharge`: allowed only at designated charging stations

If the battery reaches zero away from a charging station, the robot enters a terminal failure state.

Reward Structure

- Each move: -1
- Recharge action: -2
- Battery depletion failure: -100 (terminal)
- Goal reached: $+100$ (terminal)

Tasks

(a) Define the MDP including terminal conditions.

(b) Solve the MDP using Value Iteration.

(c) Repeat the solution for discount factors $\gamma = 0.99, 0.9$, and 0.7.

(d) Visualize and compare the resulting policies.

(e) Explain how battery-awareness emerges for different values of γ.

(f) Implement the Monte-Carlo Method and compare with the value iteration and policy iteration

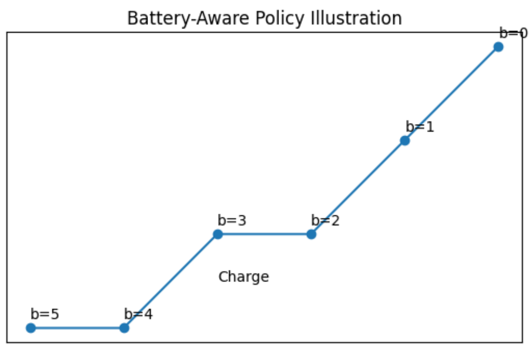

Figure 1: Conceptual illustration of battery-aware navigation. Battery level decreases along the trajectory, and recharging decisions emerge as part of the optimal policy.

3 Question 3: Risk-Sensitive Robot Navigation Near Hazards (35 Marks)

A robot must reach a goal while navigating near hazardous regions such as cliffs or fragile terrain.

State Space

$$s = (x, y, h)$$

where h indicates proximity to a hazardous region.

Transition Model

- In normal regions, actions succeed with high probability.
- Near hazards, actions have a small probability of slipping into a catastrophic terminal state.

Reward Structure

- Step cost: -1
- Goal reached: $+50$ (terminal)
- Catastrophic failure: -200 (terminal)

Tasks

(a) Solve the MDP using Value Iteration with the given reward structure.

(b) Plot the optimal policy and identify risk-taking regions.

(c) Increase the slip probability near hazards and recompute the policy.

(d) Compare the policies and explain observed changes.

(e) Discuss why shortest-path intuition fails in this scenario.

(f) Implement the Monte-Carlo Method and compare with the value iteration and policy iteration

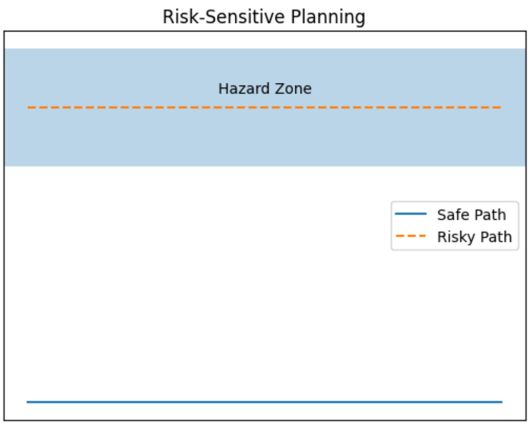

Figure 2: Risk-sensitive navigation problem illustrating a short but risky path near a hazard zone versus a longer but safer alternative. Small changes in slip probability can change the optimal policy.

Report Requirements

The report must include:

- Clear MDP formulation for each problem
- Implementation details

- Convergence plots
- Policy visualizations
- Comparative and robotics-oriented discussion