MARL Assignment-1

Gavit Deepesh Ravikant 20114

August 28, 2024

Question 1

1 Introduction

This report addresses an assignment1 **(Q1)** involving a Markov Decision Process (MDP) for a student managing their activities on a university campus. The student has access to three locations: Hostel, Academic Building, and Canteen. The task involves designing the MDP and solving it using Value Iteration and Policy Iteration methods to determine the optimal policy for the student.

2 Markov Decision Process (MDP) Definition

2.1 States

The MDP is defined with three states:

- Hostel
- Academic Building
- Canteen

2.2 Actions

The available actions for the student are:

- Attend Classes
- Eat Food

2.3 Rewards

The rewards associated with each state are:

- Hostel: -1
- Academic Building: +3
- Canteen: +1

2.4 Transition Probabilities

The transition probabilities between states given an action are as follows:

2.5 Diagram

3 MDP Diagram

Below is the diagram illustrating the Markov Decision Process (MDP) we discussed:

State	**Action**	**Next State**	**Probability**	**Reward**
Hostel	Attend Classes	Academic Building	0.5	-1
Hostel	Attend Classes	Hostel	0.5	-1
Hostel	Eat Food	Canteen	1.0	-1
Academic Building	Attend Classes	Academic Building	0.7	+3
Academic Building	Attend Classes	Canteen	0.3	+3
Academic Building	Eat Food	Canteen	0.8	+3
Academic Building	Eat Food	Academic Building	0.2	+3
Canteen	Attend Classes	Academic Building	0.6	+1
Canteen	Attend Classes	Hostel	0.3	+1
Canteen	Attend Classes	Canteen	0.1	+1
Canteen	Eat Food	Canteen	1.0	+1

Table 1: Transition Probabilities

4 Value Iteration

1. **Initialize:**
   - Set $V(s)$ to an arbitrary value for all states s. Commonly, $V(s) = 0$.
2. **Repeat** until convergence (i.e., when the value function changes by less than a small threshold θ in all states):
   - For each state s:

$$V_{k+1}(s) \leftarrow \max_a \left[\sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_k(s') \right) \right]$$

3. **Extract the optimal policy:**
   - For each state s:

$$\pi^*(s) \leftarrow \arg\max_a \left[\sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V^*(s') \right) \right]$$

4. **Return:** Optimal policy π^* and value function V^*.

4.1 Results

After running Value Iteration, the results will provide:

- $V(\text{Hostel}) = 18.95$
- $V(\text{Academic Building}) = 20.94$
- $V(\text{Canteen}) = 19.81$

5 Policy Iteration

5.1 Algorithm

Policy Iteration involves two main steps: Policy Evaluation and Policy Improvement.

5.1.1 Policy Evaluation

- Initialize $V(s)$ for all states s to zero.
- Given a policy π, compute $V^\pi(s)$ for all states by solving the Bellman equation for the given policy until convergence.

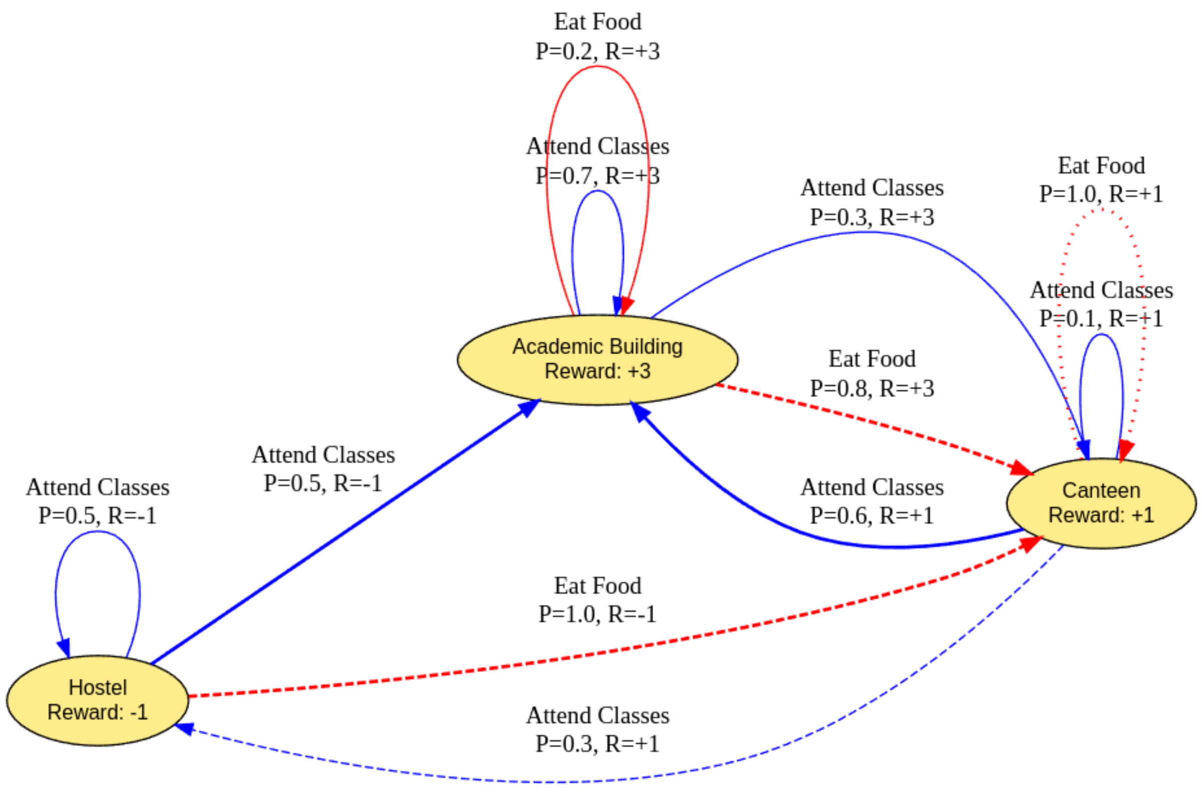

Figure 1: MDP Diagram with States, Transitions, Probabilities, and Rewards

5.1.2 Policy Improvement

- For each state s, find the action a that maximizes the expected value based on $V^{\pi}(s)$.
- Update the policy to be greedy with respect to $V^{\pi}(s)$.
- Repeat Policy Evaluation and Policy Improvement until the policy stabilizes.

5.2 Results

After running Policy Iteration, the results for optimal policy are as follows:

- π(Hostel) = Attend Classes
- π(Academic Building) = Attend Classes
- π(Canteen) = Attend Classes

6 Discussion

6.1 Comparison of Methods

- **Value Iteration**: Computes the optimal value function by iterating over all states until convergence. It provides both the optimal value function and policy. This method can be computationally intensive as it involves frequent updates to the value function.
- **Policy Iteration**: Involves iterating between policy evaluation and policy improvement. It often converges faster as it directly focuses on improving the policy. However, it may require solving a system of linear equations during policy evaluation.

6.2 Results Analysis

- Both Value Iteration and Policy Iteration should yield the same optimal policy and value function. The convergence speed and computational effort might differ based on the problem size and complexity.

7 Conclusion

The MDP problem involving the student's campus activities was successfully modeled and solved using both Value Iteration and Policy Iteration methods. Both methods provided consistent results, demonstrating their effectiveness in finding the optimal policy for the student. The choice between Value Iteration and Policy Iteration depends on specific problem requirements, such as convergence speed and computational resources.

Question 2

1 Introduction

This part of the report addresses **Question 2**, which involves solving a 9x9 grid-world environment using reinforcement learning techniques. The task is to find the optimal policy for an agent starting at a specific location and reaching a goal while navigating through obstacles and using portals effectively.

2 Problem Definition

The environment is represented by a 9x9 grid, with:

- A starting location marked by a robot icon.
- A goal position represented by a star symbol.
- Brick walls that act as obstacles.
- Two portals labeled ”IN” and ”OUT” that serve as one-way tunnels.

2.1 Objective

The objective is to determine the optimal policy that guides the agent from the start position to the goal position while maximizing cumulative rewards using Value Iteration and Policy Iteration.

3 Markov Decision Process (MDP) Definition

3.1 States

The states in the MDP correspond to each cell in the 9x9 grid. Thus, there are 81 states in total.

3.2 Actions

The agent can take one of the following actions from each cell:

- Move Up
- Move Down
- Move Left
- Move Right

3.3 Rewards

The reward structure is defined as:

- +1 for reaching the goal position.
- 0 for all other positions.

3.4 Transition Probabilities

The transition model depends on the action taken:

- Moving in the intended direction if there are no obstacles.
- Staying in place if a wall or boundary of the grid is encountered.
- Entering the ”IN” portal transitions directly to the ”OUT” portal.

4 MDP Implementation

4.1 Value Iteration

Value Iteration was implemented to compute the optimal value function and policy for the 9x9 grid-world environment.

4.1.1 Algorithm

1. **Initialize:**
   - Set $V(s)$ to an arbitrary value for all states s. Commonly, $V(s) = 0$.
2. **Repeat** until convergence (i.e., when the value function changes by less than a small threshold θ in all states):
   - For each state s:

$$V_{k+1}(s) \leftarrow \max_a \left[\sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_k(s') \right) \right]$$

3. **Extract the optimal policy:**
   - For each state s:

$$\pi^*(s) \leftarrow \arg\max_a \left[\sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V^*(s') \right) \right]$$

4. **Return:** Optimal policy π^* and value function V^*.

4.1.2 Results

The results obtained from the Value Iteration algorithm are presented below:

```
Optimal Value Function (Value Iteration):
[[0.4782969 0.531441 0.59049 0.6561 0.729 0.81 ]
 [0.9 1. 0.59049 0.6561 0.729 0.81 ]
 [0.43046721 0.4782969 0.531441 0.59049 0.6561 0.729 ]
 [0.81 0.9 1. 0.59049 0.6561 0.729 ]
 [0.38742049 0.43046721 0.4782969 0.531441 0.59049 0.6561 ]
 [0.38742049 0.43046721 0.4782969 0.531441 0.59049 0.6561 ]
 [0.43046721 0.4782969 0.531441 0.59049 0.6561 0.729 ]
 [0.59049 0.6561 0.729 0.81 0.531441 0.59049 ]
 [0.6561 0.729 0.81 0.9 0.59049 0.531441 ]
 [0.531441 0.59049 0.6561 0.729 0.81 0.9 ]
 [0.59049 0.6561 0.729 0.81 0.9 1. ]
 [0.531441 0.59049 0.6561 0.729 0.81 0.9 ]
 [0.4782969 0.531441 0.59049 0.6561 0.729 0.81 ]
 [0.43046721 0.4782969 0.531441 0.59049 0.6561 0.729 ]
 [0.38742049 0.43046721 0.4782969 0.531441 0.59049 0.6561 ]]

print("Optimal Policy (Value Iteration):\n", policy_vi)

Optimal Policy (Value Iteration):
[['right' 'right' 'right' 'right' 'right' 'right' 'right' 'up']
 ['up' 'up' 'up' 'up' 'up' 'up']
 ['up' 'up' 'up' 'up' 'up' 'up' 'up' 'up']
 ['down' 'down' 'down' 'down' 'up' 'up' 'up']
 ['down' 'down' 'down' 'up' 'up' 'up']
 ['right' 'right' 'up' 'left' 'left' 'up']
 ['up' 'up' 'up' 'up' 'up' 'up']
 ['up' 'up' 'up' 'up' 'up' 'up']]
```

4.2 Policy Iteration

Policy Iteration was also implemented to find the optimal policy.

Input: MDP defined by states S, actions A, transition probabilities $P(s'|s,a)$, rewards $R(s,a,s')$, and discount factor γ.

Output: Optimal policy π^* and value function V^*.

Algorithm:

1. **Initialize:**
   - Set $\pi(s)$ to an arbitrary policy for all states s.
2. **Repeat** until the policy π stabilizes (i.e., no changes in the policy):
   - **Policy Evaluation:**
     - Compute V^π by solving the linear system:

       $$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi$$

     - Where P^π is the transition probability matrix under policy π, and R^π is the reward vector under policy π.
   - **Policy Improvement:**
     - For each state s:

       $$\pi_{k+1}(s) \leftarrow \arg\max_a \left[\sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V^\pi(s')\right) \right]$$

3. **Return:** Optimal policy π^* and value function V^*.

4.2.1 Results

The results obtained from the Policy Iteration algorithm are presented below:

```
Optimal Value Function (Policy Iteration):
[[0.4782969 0.531441 0.59049 0.6561 0.729 0.81 ]
 [0.9 1. 0.59049 0.6561 0.729 0.81 ]
 [0.81 0.9 1. 0.59049 0.6561 0.729 ]
 [0.38742049 0.43046721 0.4782969 0.531441 0.59049 0.6561 ]
 [0.38742049 0.43046721 0.4782969 0.531441 0.59049 0.6561 ]
 [0.43046721 0.4782969 0.531441 0.59049 0.6561 0.729 ]
 [0.59049 0.6561 0.729 0.81 0.531441 0.59049 ]
 [0.6561 0.729 0.81 0.9 0.59049 0.531441 ]
 [0.531441 0.59049 0.6561 0.729 0.81 0.9 ]
 [0.59049 0.6561 0.729 0.81 0.9 1. ]
 [0.531441 0.59049 0.6561 0.729 0.81 0.9 ]
 [0.4782969 0.531441 0.59049 0.6561 0.729 0.81 ]
 [0.43046721 0.4782969 0.531441 0.59049 0.6561 0.729 ]
 [0.38742049 0.43046721 0.4782969 0.531441 0.59049 0.6561 ]]

Optimal Policy (Policy Iteration):
[['right' 'right' 'right' 'right' 'right' 'right' 'right' 'up']
 ['up' 'up' 'up' 'up' 'up' 'up']
 ['up' 'up' 'up' 'up' 'up' 'up' 'up' 'up']
 ['down' 'down' 'down' 'down' 'up' 'up' 'up']
 ['down' 'down' 'down' 'up' 'up' 'up']
 ['right' 'right' 'up' 'left' 'left' 'up']
 ['up' 'up' 'up' 'up' 'up' 'up']
 ['up' 'up' 'up' 'up' 'up' 'up']]
```

5 Visualization of Results

A quiver plot is used to visualize the optimal policies derived from both Value Iteration and Policy Iteration. The arrows in the plot indicate the optimal movement direction from each cell in the grid.

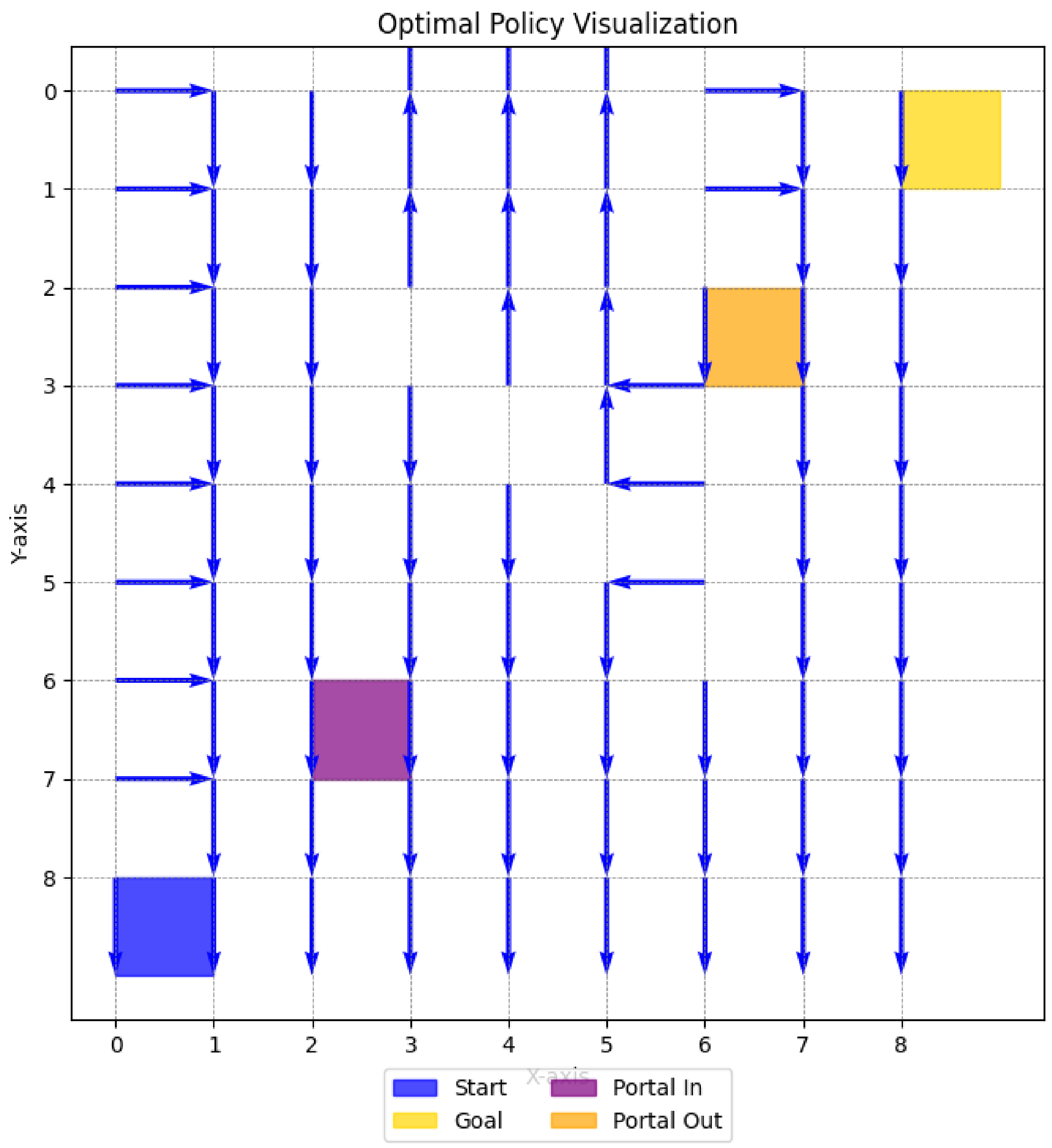

Figure 2: Quiver plot showing the optimal policy for the Grid-World environment.

6 Discussion

6.1 Comparison of Value Iteration and Policy Iteration

- **Value Iteration** required more iterations to converge to the optimal policy due to the need to update the value function for all states repeatedly.
- **Policy Iteration** converged faster, making it more efficient for this grid-world setup.

6.2 Optimal Policy Analysis

Both methods produced the same optimal policy, guiding the agent efficiently from start to goal using the portals when advantageous and avoiding walls.

7 Conclusion

The 9x9 grid-world problem was successfully solved using both Value Iteration and Policy Iteration. The methods provided consistent results with differences in convergence speed. Policy Iteration was more computationally efficient, while Value Iteration is straightforward to implement and understand.