ASSIGNMENT 02

Reinforcement Learning

Name	**Jawad Ali Shahid**
Reg no	**FA21-BAI-012**

Submitted to:

Dr. Abid Akber Gardezi

OCTOBER 26, 2024

COMSATS UNIVERSITY ISLAMABAD

1. Defining the Markov Decision Process (MDP) for the Grid World

The **Markov Decision Process (MDP)** for this problem is defined by five components, (S, A, P, R, γ) (S, A, P, R, \gamma) (S, A, P, R, γ), which describe the environment and how the agent interacts with it.

Components of the MDP

1. **State Space SSS**:
    - The agent can occupy any of the 9 cells in a 3x3 grid, giving us 9 states.
    - We represent each state as (imp) where imp ∈ {1,2,3}.
    - The start state is (1,1) (1,1) (1,1), and the goal state is (3,3) (3,3) (3,3).

    So, the state space SSS can be written as:

    S= {(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)}

2. **Action Space AAA**:
    - The agent has four possible actions:
        - **Up**: Moves the agent up by one cell.
        - **Down**: Moves the agent down by one cell.
        - **Left**: Moves the agent left by one cell.
        - **Right**: Moves the agent right by one cell.
    - If the action would take the agent outside the grid boundaries, the agent stays in the current cell.

3. **Transition Probability Function P (s ′|spa) P (s' | s, a) P (s ‘|spa)**:
    - This function describes the probability of moving to a new state s′s's′ given the current state sss and an action aaa.
    - Since this environment is deterministic, taking an action aaa in state sss results in a specific next state s′s's′ with probability 1 (if the action doesn't lead outside the grid).
    - Examples:
        - If the agent is in (1,1) (1,1) (1,1) and takes **Right**, it will move to (1,2) (1,2) (1,2).
        - If the agent is in (1,1) (1,1) (1,1) and takes **Up**, it will remain in (1,1) (1,1) (1,1) because it's already at the top boundary.

4. **Reward Function R (s, a, s′) R (s, a, s') R (s, a, s′)**:
    - Each move incurs a reward of −1-1−1, representing a cost of moving.
    - When the agent reaches the goal state (3,3) (3,3) (3,3), the episode terminates, and no further rewards are collected.

- Formally, we can define: R (s, a, s′) = {−10 if s! = (3,3), 0 if s= (3,3) }

5. **Discount Factor γ\gammaγ**:
   - Since we're only interested in the total steps (or cumulative reward) until reaching the goal, we can use a discount factor γ=1\gamma = 1γ=1, making this an **undiscounted MDP**.

2. Calculating the Value of the Given Policy π\piπ

The policy π\piπ in question is a **random policy**, where the agent chooses each of the four available actions (up, down, left, right) with equal probability (i.e., 1/4 for each action).

Value Function for a Policy

The value function Vπ(s)V^\pi(s)Vπ(s) for a policy π\piπ represents the expected cumulative reward (sum of rewards) from state sss while following π\piπ until reaching the goal state (3,3) (3,3) (3,3).

The **Bellman expectation equation** for a policy π\piπ is:

Vπ(s)=a∈A∑π(a|s) s′∑P (s′|s, a) (R (s, a, s′) +γVπ(s′))

In this case:

- π(as)=0.25\pi(as) = 0.25π(a|s)=0.25 for each action aaa.
- R (s, a, s′) =−1R (s, a, s') = -1R (s, a, s′) =−1 for every move until reaching (3,3) (3,3) (3,3).

Since calculating Vπ(S) analytically for each state would involve solving a system of linear equations, we can intuitively understand that:

- Because the agent is moving randomly, it will take more steps (on average) to reach (3,3) (3,3) (3,3).
- The expected value Vπ(S) will generally be the negative of the expected steps required to reach the goal from sss.

If we assume nnn is the expected number of moves to reach (3,3) (3,3) (3,3) under random moves, then:

Vπ(s)≈−n

where nnn will vary based on the starting position sss.

3. Suggesting an Optimal Policy π^*

An optimal policy $\boldsymbol{\pi^*}$ would minimize the total cost to reach the goal state (3,3) (3,3) (3,3), which is equivalent to minimizing the number of moves.

Optimal Policy π^*

A potential optimal policy $\boldsymbol{\pi^*}$ can be described as:

1. **At each state, choose the action that moves the agent closer to (3,3) (3,3) (3,3)**.
2. The goal is to make moves that either:
   - Decrease the row index (move down if the current row i<3i < 3i<3),
   - Or decrease the column index (move right if the current column j<3j < 3j<3).

Example Optimal Actions for Each State:

- (1,1) (1,1) (1,1): Move **Right** to (1,2) or **Down** to (2,1).
- (1,2) (1,2) (1,2): Move **Right** to (1,3) or **Down** to (2,2).
- (2,1) (2,1) (2,1): Move **Down** to (3,1) or **Right** to (2,2).
- (2,2) (2,2) (2,2): Move **Down** to (3,2) or **Right** to (2,3).
- (3,2) (3,2) (3,2): Move **Right** to (3,3) (goal state).

Explanation of Optimality

The optimal policy π∗\pi^*π∗ minimizes the number of steps by always moving closer to the goal, resulting in:

- A reduction in the expected number of moves from each state.
- A higher cumulative reward (less negative total reward) due to fewer steps with the constant -1 cost per move.

Under this policy, the value function Vπ∗(s)V^{\pi^*} (s)Vπ∗(s) will reflect the shortest path cost from each state to (3,3) (3,3) (3,3), with Vπ∗ ((3,3)) =0V^{\pi^*} ((3,3)) = 0Vπ∗ ((3,3)) =0 (since the goal state incurs no cost).