Assignment 1

Reinforcement Learning Programming - CSCN 8020

September 29, 2023

Problem 1 [10]

Pick-and-Place Robot: Consider using reinforcement learning to control the motion of a robot arm in a repetitive pick-and-place task. If we want to learn movements that are fast and smooth, the learning agent will have to control the motors directly and obtain feedback about the current positions and velocities of the mechanical linkages.
Design the reinforcement learning problem as an MDP, define states, actions, rewards *with reasoning.*

Problem 2 [20]

Problem Statement

2x2 Gridworld: Consider a 2x2 gridworld with the following characteristics:

- State Space (S): s_1, s_2, s_3, s_4.
- Action Space (A): up, down, left, right.
- Initial Policy (π): For all states, $\pi(up|s) = 1$.
- Transition Probabilities $P(s'|s,a)$:
  - If the action is valid (does not run into a wall), the transition is deterministic.
  - Otherwise, $s' = s$.
- Rewards $R(s)$:
  - $R(s_1) = 5$ for all actions a.

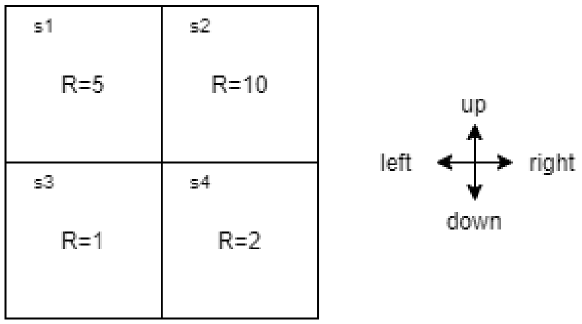

Figure 1: 2x2 Gridworld

- $R(s_2) = 10$ for all actions a.
- $R(s_3) = 1$ for all actions a.
- $R(s_4) = 2$ for all actions a.

Tasks

Perform two iterations of Value Iteration for this gridworld environment. Show the step-by-step process (**without code**) including policy evaluation and policy improvement. Provide the following for each iteration:

- Iteration 1:
    1. Show the initial value function (V) for each state.
    2. Perform value function updates.
    3. Show the updated value function.
- Iteration 2: Show the value function (V) after the second iteration.

Problem 3 [35]

Problem Statement

5x5 Gridworld: In Lecture 3's programming exercise (here), we explored an MDP based on a 5x5 gridworld and implemented Value Iteration to estimate the optimal state-value function (V_*) and optimal policy (π_*).
The environment can be described as follows:

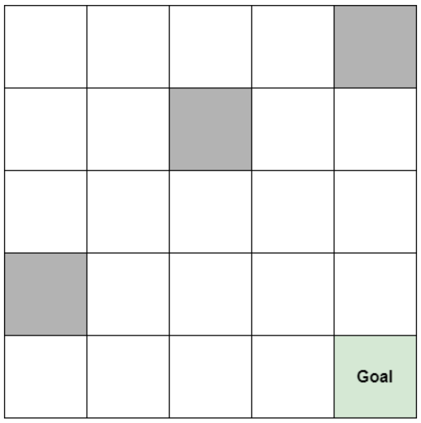

Figure 2: 5x5 Gridworld

- States: states are identified by their row and column, the same as a regular matrix. Ex: the state in row 0 and column 3 is $s_{0,3}$ (Figure: 2)
  - Terminal/Goal state: The episode ends if the agent reached this state. $s_{Goal} = s_{4,4}$
  - Grey states: $\{s_{2,2}, s_{3,0}, s_{0,4}\}$, these are valid but non-favourable states, as will be seen in the reward function.
- Actions: a_1 = right, a_2 = down, a_3 = down, a_4 = up for all states.
- Transitions: If an action is valid, the transition is deterministic, otherwise $s' = s$
- Rewards $R(s)$:

$$\begin{aligned} R(s) &= +10 & s &= s_{4,4} \\ &= -5 & s \in S_{grey} &= s_{2,2}, s_{3,0}, s_{0,4} \\ &= -1 & s \in S &\neq s_{4,4}, S_{grey} \end{aligned}$$

Tasks

Task1: Update MDP Code

1. Update the reward function to be a list of reward based on whether the state is terminal, grey, or a regular state.
2. Run the existing code developed in class and obtain the optimal state-values and optimal policy. Provide a figures of the gridworld with the obtained V_* and π_* (You can manually create a table).

Task 2: Value Iteration Variations

Implement the following variation of value iteration. Confirm that it reaches the same optimal state-value function and policy.

1. **In-Place Value Iteration**: Use a single array to store the state values. This means that you update the value of a state and immediately use that updated value in the subsequent updates.

Deliverables

- Full code with comments to explain key steps and calculations.
- Provide the estimated value function for each state.
- **Important:** Compare the performance of these variations in terms of optimization time, number of episodes, and provide comments on their computational complexity.

Problem 4 [35]

Problem Statement

Off-policy Monte Carlo with Importance Sampling: We will use the same environment, states, actions, and rewards in Problem 3.

Task

Implement the off-policy Monte Carlo with Importance sampling algorithm to estimate the value function for the given gridworld. Use a fixed behavior policy $b(a|s)$ (e.g., a random policy) to generate episodes and a greedy target policy.

Suggested steps

1. Generate multiple episodes using the behavior policy $b(a|s)$.

2. For each episode, calculate the returns (sum of discounted rewards) for each state.

3. Use importance sampling to estimate the value function and update the target policy $\pi(a|s)$.

4. You can assume a specific discount factor (e.g., $\gamma = 0.9$) for this problem.

5. Use the same main algorithm implemented in lecture 4 in class.

Deliverables

- Full code with comments to explain key steps and calculations.

- Provide the estimated value function for each state.

- **Important** Compare the estimated value function obtained from Monte Carlo with the one obtained from Value Iteration in terms of optimization time, number of episodes, computational complexity, and any other aspects you notice.