Practical Assignment 6: Learning Taxi-v3 task using Value Iteration

1. Problem Statement

In this practical assignment, students are required to implement the Value Iteration algorithm to compute the optimal policy for the Taxi-v3 environment available in the Gymnasium library. The objective is to determine the optimal sequence of actions that maximizes cumulative reward while transporting a passenger from a pickup location to a designated drop location in a 5x5 grid world.

2. Taxi-v3 Environment Description

Taxi-v3 is a discrete grid-world environment consisting of a 5x5 grid. There are four fixed landmark locations identified as R (Red), G (Green), Y (Yellow), and B (Blue). At the beginning of each episode, a passenger is randomly located at one of these four landmarks, and the destination is randomly chosen from the same set of landmarks.

The state space consists of 500 states computed as: 5 (rows) × 5 (columns) × 5 (passenger locations including inside taxi) × 4 (destinations).

The action space contains six discrete actions:
0 - Move South
1 - Move North
2 - Move East
3 - Move West
4 - Pickup Passenger
5 - Drop Passenger

The environment contains internal walls represented by barriers that prevent movement between specific adjacent cells. If the taxi attempts to move through a wall or outside the grid boundary, it remains in the same state.

Environment Dynamics: Taxi-v3 is a deterministic environment. For a given state and action, the next state and reward are fixed. There is no stochastic transition probability.

3. Reward Structure

-1 reward for every step taken.
+20 reward for successful drop-off at the correct destination.
-10 reward for illegal pickup (when passenger is not at taxi location).
-10 reward for illegal drop-off.

The episode terminates only after a successful drop-off of the passenger at the correct destination.

4. Value Iteration Algorithm

Value Iteration is a dynamic programming method used to compute the optimal value function for a Markov Decision Process. It directly applies the Bellman Optimality Equation iteratively until convergence.

Bellman Optimality Equation:

$$V(s) = \max_a \Sigma P(s'|s,a) [R(s,a,s') + \gamma V(s')]$$

The algorithm repeatedly updates the value of each state by taking the maximum expected return over all possible actions. The process continues until the maximum change in value across all states becomes smaller than a predefined threshold.

5. Pseudocode for Value Iteration

Initialize $V(s) = 0$ for all states
Repeat:
  $\Delta = 0$
  For each state s:
    $v = V(s)$
    $V(s) = \max$ over actions a of $[\Sigma P(s'|s,a) (R + \gamma V(s'))]$
    $\Delta = \max(\Delta, |v - V(s)|)$
Until $\Delta < \theta$ (small threshold)
Extract policy:
  $\pi(s) = \text{argmax}$ over actions a of $[\Sigma P(s'|s,a) (R + \gamma V(s'))]$

6. Advantages and Applications

Advantages:
- Guaranteed convergence to optimal policy for finite MDPs.
- Conceptually simple and mathematically sound.
- Suitable when full model of environment is known.

Applications:
- Robotics path planning
- Automated transport systems
- Resource allocation problems
- Game AI decision making

7. Expected Output

After implementation, the algorithm should:

- Converge to a stable value function.
- Generate an optimal policy for all 500 states.
- Successfully complete the Taxi task with positive cumulative reward.
- Reduce unnecessary movements compared to a random policy.

8. Conclusion

In this practical assignment, students implement the Value Iteration algorithm to solve the Taxi-v3 environment. This exercise strengthens understanding of Dynamic Programming, Bellman Optimality updates, and optimal policy extraction. The experiment demonstrates how mathematical concepts of MDPs are translated into practical reinforcement learning solutions.