Report on Implementing a Reinforcement Learning Agent for a Gridworld Environment

1. Introduction

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to interact with an environment by taking actions and receiving rewards. This project implements an RL agent using **Q-learning**, a model-free, off-policy, value-based algorithm, to navigate a **5x5 Gridworld environment** efficiently while avoiding obstacles and maximizing rewards.

2. Problem Definition

The objective of this project is to develop an RL agent that can autonomously learn an optimal policy for navigating the Gridworld environment. The agent must learn to find the best path from a **start position** to a **goal position**, avoiding obstacles and minimizing negative rewards.

Environment Details:

- **Grid Size:** 5x5
- **Start State:** (0,0)
- **Goal State:** (4,4)
- **Obstacles:** Placed at predefined positions

Rewards System:

- **+10** for reaching the goal
- **-5** for hitting an obstacle
- **-1** for each step taken

3. Q-learning Algorithm

Q-learning is a **model-free** RL algorithm that estimates Q-values for state-action pairs using the **Bellman equation**:

Q(s,a)←Q(s,a)+α(r+γmax⁡a′Q(s′,a′)−Q(s,a))Q(s, a) \leftarrow Q(s, a) + \alpha (r + \gamma \max_{a'} Q(s', a') - Q(s, a))
where:

- Q(s,a)Q(s, a) is the Q-value for state ss and action aa
- α\alpha is the learning rate
- γ\gamma is the discount factor
- rr is the reward received
- s′s' is the next state after taking action aa
- a′a' is the next best action
-

Hyperparameters Used:

- **Learning Rate (α\alpha)** = 0.1
- **Discount Factor (γ\gamma)** = 0.9
- **Exploration Rate (ε\varepsilon)** = Starts at 1.0 and decays
- **Episodes:** 5000
- **Exploration Decay:** 0.995 per episode

4. Implementation

4.1 Environment Representation

The Gridworld is implemented as a 5x5 grid with predefined rewards and obstacles. The agent interacts with the environment using four possible actions:

1. **Up (0)**
2. **Down (1)**
3. **Left (2)**
4. **Right (3)**

4.2 Q-learning Implementation

- A **Q-table** is initialized with zeros for all state-action pairs.
- The agent chooses actions using an **ε-greedy strategy**, where it either explores (random action) or exploits (chooses the best action based on Q-values).
- Q-values are updated using the **Bellman equation**.
- The exploration rate (ε) decays over time to transition from exploration to exploitation.

4.3 Training Process

- The agent is trained over **5000 episodes**.
- The **total reward per episode** is recorded to track learning progress.
- The **Q-table** is updated iteratively to improve the learned policy.

4.4 Evaluation Metrics

- **Training Progress**: Analyzed using total reward per episode.
- **Q-values Heatmap**: Visual representation of learned Q-values.
- **Policy Visualization**: A grid showing the optimal action for each state.

5. Results and Analysis

5.1 Training Performance

The agent initially explores randomly, but as training progresses, it learns an optimal policy. The total reward per episode increases as the agent gains experience.

5.2 Visualization Results

Q-value Heatmap:

A heatmap of Q-values is generated, where higher values indicate more favorable states.

Policy Visualization:

The optimal policy is represented using directional arrows:

- **↑ (Up)**
- **↓ (Down)**
- **← (Left)**
- **→ (Right)**

This policy guides the agent efficiently towards the goal.

6. Challenges and Limitations

- **Exploration-Exploitation Tradeoff:** Finding the right balance between exploration and exploitation was challenging.

- **Parameter Sensitivity:** The choice of hyperparameters significantly impacted learning speed and performance.

- **Local Minima:** The agent sometimes converged to suboptimal paths depending on the obstacle placement.

7. Conclusion and Future Work

The Q-learning agent successfully learned to navigate the Gridworld efficiently. Future improvements could include:

- Implementing **alternative RL algorithms** (e.g., SARSA, Deep Q-Networks, Double Q-Learning, Dyna-Q, Monte Carlo Policy Iteration, Policy Gradient Methods (REINFORCE), Advantage Actor-Critic (A2C)).

- Testing with **larger Gridworld environments**.

- Optimizing **hyperparameters using automated tuning methods**.

- Exploring **dynamic environments** where obstacles change positions.

8. References