Delivery Robot

With Q-learning Algorithm

Nourhan deif

Abdulkhaliq Sarwat
Kareem Ezzaldin

Ali Adel

1. Introduction:

- The goal of this project is to **implement the Q-Learning algorithm** for the idea of **robot delivery**, which allows the robot to navigate the grid and reach a specific goal, **this algorithm uses exploration and exploitation** until it finds the optimal path.

2. Project description

★ Environment Description:

  - The environment is defined as a grid of 10X10, and the agent starts from the initial state (0,0) and aims to reach the goal(9,9) in each episode. The agent navigates while avoiding random obstacles.

★ Action space

  - UP
  - Down
  - Left
  - Right

❖ The actions move the agent to adjacent cells in the grid. If any action leads to an invalid state (out of boundaries or into an obstacle), the agent remains in its current position.

3. Algorithms used

- **Q-Learning**→This algorithm learns the value of actions taken in specific states to optimize future decisions.
- **Parameters:**
  1. Alpha(0.1) : Learning rate is responsible for the learning rate and the ability to update current information.

  2. Gamma(0.9): Discount factor It determines the importance of current Vs future rewards for the agent, if they are close to 1, it prioritizes future rewards.

  3. Epsilon(1.0): It is the rate of exploration, where the exploration of the agent determines the environment, if it is 1, it will discover the entire environment

4. Epsilon Dency(0.99): The decay rate reduces the value of epsilon after each training and here decreases by 1%.

5. Minimum Epsilon(0.1): The minimum exploration rate ensures that the agent continues to explore even after training to prevent him from relying on only the current options and may find a better option.

6. Number of Epsilon(1000): Total training episodes.

4. Implementation

❖ Key functions:

- is_valid_state(state): using helper function to determine if the state is valid or not to be in the grid not out of it this means if the given state is within grid bounds and not an obstacle.

- take_action(state, action): Function to get the next state by updating the agent position based on the chosen action, maintaining valid movement.

- get_reward(state): Assigns rewards based on the agent's state (100) for reaching the goal,(-10) for obstacles, and (-1) for each step token.

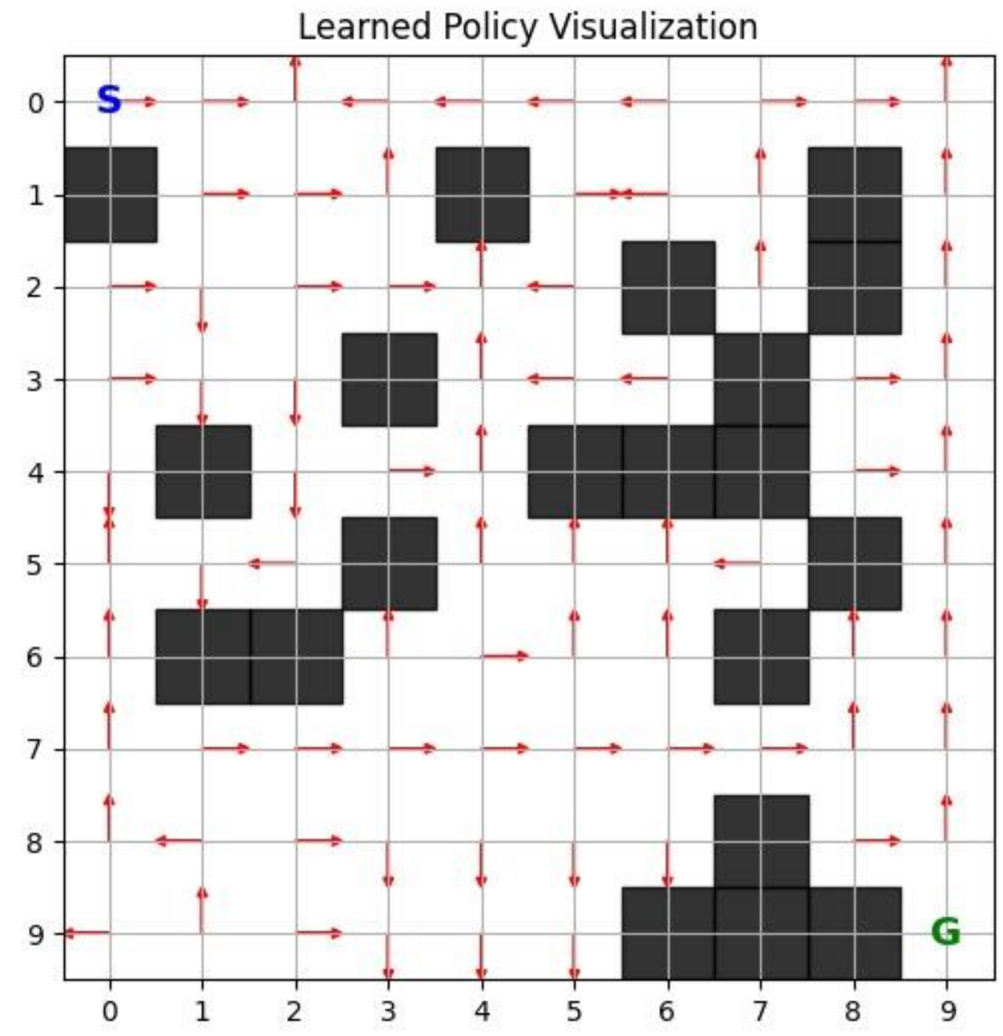
5. Performance analysis

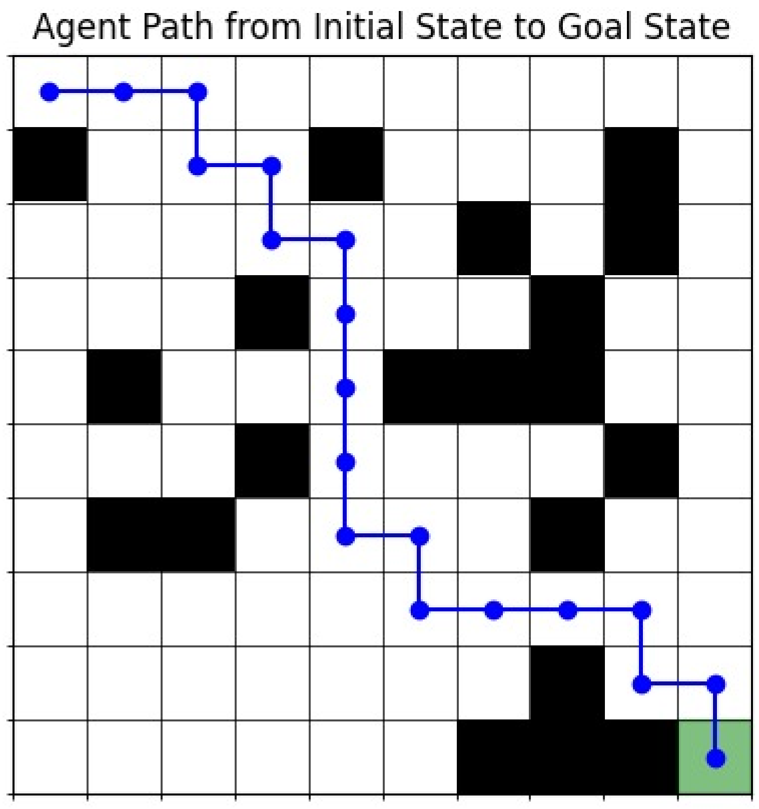
- The agent was trained in over 1000 episodes with performance being evaluated every 100 episodes. The exploration strategy allowed the agent to learn and adequately improve his policy over time, as the rate of effective exploration decreased.

6. Results and visualization

❖ Performance: The average rewards were calculated, showing that the agent got better over time. It learned to reach the goal more efficiently and made fewer mistakes by avoiding obstacles.

❖ Final Policy: The Q-table shows the learned action values for each state and highlights the best path to reach the goal.


Learned Policy Visualization
S
G
0
1
2
3
4
5
6
7
8
9



7. Conclusion

- The project shows how the Q-learning algorithm can be used to navigate a grid with obstacles. The agent learns to find the best path to the goal.