Q-Learning for Maze Navigation

A. Mohamed[1], A. Tamer[2], S. Elsayed[3], and H. Amir[4]

[1]Department of Computer Science, Zewailcity University, Giza, Egypt

***Abstract*—This paper presents an implementation of reinforcement learning using the Q-Learning algorithm for maze navigation. The objective is to train an agent to navigate from a defined start to the goal while maximizing its total reward. The project focuses on generating a dynamic maze, designing a reward system, applying the Q-Learning algorithm, and visualizing the agent's progress. Evaluation metrics such as optimal path identification, accuracy, and reward optimization are used to assess performance.**

I. Introduction

Reinforcement learning has emerged as a powerful paradigm for solving complex decision-making tasks. In this project, we implemented Q-Learning to navigate a dynamically generated maze. The primary goal is to ensure that the agent finds an optimal path through the maze while minimizing penalties and maximizing rewards.

II. Problem Definition

The maze is defined as follows:

- A grid of size 15×15.
- Each cell has a value of zero (obstacle) or one (possible step).
- 25% of the maze cells are randomly distributed obstacles.
- The starting point is located at the top-left corner (0,0) and the goal is at the bottom-right corner (14,14).
- The agent can take one of four possible actions: right, left, up, or down.

III. Implementation

A. Problem Creation

A dynamic maze is generated for the agent to navigate, with 25% obstacles in a 15×15 grid. Each cell is either marked as 0 (obstacle) or 1 (possible step).

B. Reward Function

The reward function guides the agent by assigning points as follows:

- Reaching the goal cell = +100
- Landing on an obstacle cell = -10
- Moving to a valid non-goal cell = -1

C. Q-Learning Algorithm

The Q-table is structured as a 3D array [height, width, actions], updated using the rule:

$$Q[x, y, \text{action\_index}] = Q[x, y, \text{action\_index}] + \alpha \cdot (reward + \gamma \cdot \text{next\_max} - Q[x, y, \text{action\_index}]) \quad (1)$$

Fig. 1: Training Reward Function.

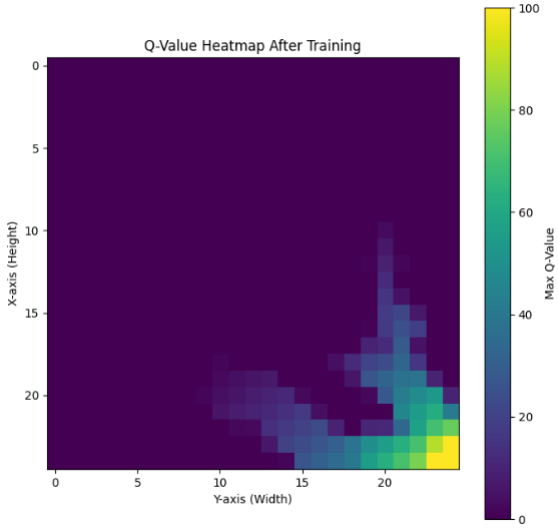

Fig. 2: Q-Value Heatmap After Training.

Exploration vs Exploitation:

- The agent explores with a probability of ϵ (0.2).
- Otherwise, the agent exploits its learned Q-table.

D. Visualization

The visualization demonstrates the agent's learning and navigation process in real-time, highlighting the optimal path after training.

E. Evaluation

Evaluation tracks the rewards, the number of steps, and evaluates if optimal actions were taken. It calculates accuracy as the percentage of optimal paths derived from the training phase.

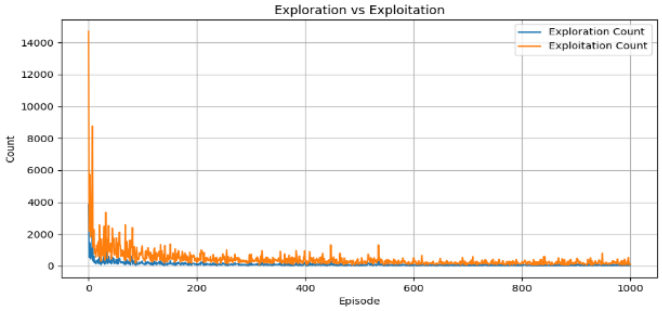

Fig. 3: Exploration vs Exploitation.

Fig. 4: final optimal path after training.

IV. Results

A. Training Results

- The agent successfully learned to navigate the maze.
- Improvement of the Q-table was observed through training.

B. Evaluation Results

- Did the agent succeed in finding the optimal path?
- Accuracy: Percentage of optimal steps taken.
- Rewards: Reflects agent training.
- Steps: Length of the path taken by the agent.

C. Visualization

Real-time visualization was performed using the Pygame library, highlighting the optimal path after training.

V. References

References

[1] Amine, A. (2020, December 19). Q-Learning Algorithm: From explanation to implementation. Medium. https://towardsdatascience.com/q-learning-algorithm-from-explanation-to-implementation-cdbeda2ea187

[2] Pygame Library: https://www.pygame.org/