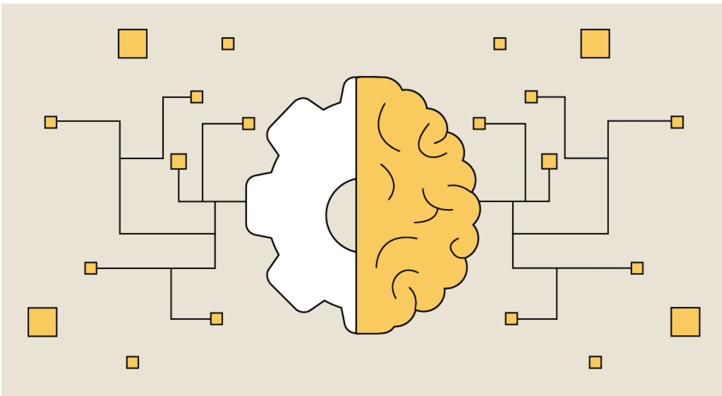
A Dynamic Maze Solver using Q-Learning

A Reinforcement Learning Approach to Autonomous Navigation

DAMODARANE Jean-Baptiste



March 2025

Abstract

This paper presents a dynamic maze-solving algorithm using Q-Learning, a reinforcement learning technique. The agent explores a dynamically generated perfect maze and learns the optimal policy to reach the exit using trial-and-error. The Q-Learning approach allows the agent to balance exploration and exploitation, leading to improved decision-making over time. We discuss the mathematical foundations, algorithmic implementation, and experimental results.

Contents

1 Introduction

Maze-solving is a well-known problem in artificial intelligence and robotics. Reinforcement Learning (RL) provides a promising approach to navigating mazes by enabling an agent to learn an optimal path based on rewards. In this work, I implement a Q-Learning-based agent to solve a dynamically generated maze using an epsilon-greedy strategy.

2 Motivation and Purpose

As a Master of Science student at Epitech, I was eager to explore the field of Reinforcement Learning and its applications in autonomous navigation. My objective in this project was to gain a deeper understanding of how agents can learn optimal strategies in complex environments. Specifically, I wanted to investigate:

- How an agent can navigate a dynamically generated maze efficiently using Q-Learning.
- The impact of different hyperparameters (learning rate, discount factor, and exploration rate) on the learning process.
- The convergence behavior of the Q-table and how long it takes for an agent to learn an optimal path.
- The potential improvements and future enhancements, such as Deep Q-Networks (DQN) or multi-agent collaboration.

This project serves as both a practical implementation of reinforcement learning and a foundation for future research in AI-driven decision-making processes.

3 Q-Learning Algorithm

Q-Learning is a model-free reinforcement learning algorithm that estimates the optimal action-selection policy for a given state-space. It updates Q-values using the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1}$$

where:

- $Q(s,a)$ is the Q-value for state s and action a
- α is the learning rate
- r is the reward obtained
- γ is the discount factor
- $\max_{a'} Q(s',a')$ represents the highest Q-value of the next state s'

4 Maze Representation and Environment

The maze is represented as a 2D grid where:

- 0 represents free space
- 1 represents walls
- 2 represents the exit

A perfect maze is generated using a depth-first search (DFS)-based recursive backtracking algorithm, ensuring a single unique solution.

5 Agent Behavior and Action Selection

The agent's movement follows an ϵ-greedy policy:

$$\text{Choose action} = \begin{cases} \text{random action}, & \text{with probability } \epsilon \\ \arg\max Q(s,a), & \text{otherwise} \end{cases} \tag{2}$$

where ϵ controls the trade-off between exploration (random moves) and exploitation (choosing the best-known action).

6 Reward Function

The agent receives a reward based on the following conditions:

- $+1$ if it reaches the exit
- -1 if it collides with a wall
- 0 for all other moves

7 Implementation Details

The implementation uses Python with Tkinter for visualization and NumPy for data manipulation. The Q-table is initialized with zeros, and the maze updates dynamically based on the agent's movement. The main steps include:

1. Initializing the Q-table and maze.
2. Selecting actions using ϵ-greedy policy.
3. Updating Q-values using the Bellman equation.
4. Rendering the maze and agent position.
5. Iterating until the agent reaches the goal.

8 Experimental Results

We tested the algorithm on a 10x10 maze. Over multiple episodes, the agent initially explores but gradually converges to an optimal path. The Q-table values stabilize, indicating learning convergence.

9 Conclusion and Future Work

This project demonstrates how Q-Learning can effectively solve mazes through reinforcement learning. Future work could explore:

- Dynamic obstacles
- Multi-agent collaboration
- Deep Q-Networks (DQN) for continuous environments

10 References

- GeeksforGeeks. (n.d.). What is Reinforcement Learning? Retrieved from https://www.geeksforgeeks.org/what-is-reinforcement-learning/
- Stanford University. (n.d.). Readings. Retrieved from https://web.stanford.edu/class/psych209/Readings