Vyorius Test - 2D Autonomous Agent (Reinforcement Learning Approach)

Kirtiraj Jamnotiya

April 7, 2025

1 Introduction

The task involves building an autonomous agent that can navigate a 2D grid environment using Reinforcement Learning (RL) techniques. The agent starts from a fixed point, avoids obstacles, and reaches a goal point. The environment contains both static and dynamic obstacles, which the agent must avoid to successfully navigate the grid. We use Q-learning, a popular RL algorithm, to train the agent in this task.

2 Reinforcement Learning Approach

2.1 Q-Learning Algorithm

We used Q-learning, a model-free RL algorithm, for this task. In Q-learning, an agent learns the value of taking an action in a particular state. The goal is to learn the optimal policy that maximizes the cumulative reward over time. The Q-value for each state-action pair is updated using the Bellman equation:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

Where:

- $Q(s_t, a_t)$ is the Q-value for state s_t and action a_t
- α is the learning rate

- r_t is the reward for taking action a_t in state s_t
- γ is the discount factor
- $\max_{a'} Q(s_{t+1}, a')$ is the maximum Q-value for the next state

The agent uses an epsilon-greedy policy for action selection, where ϵ is the exploration rate. At the start, the agent explores more (higher ϵ) and gradually reduces exploration as it learns (lower ϵ).

2.2 State Representation

The state space is represented as a grid where each cell corresponds to a state. The agent's position is represented by coordinates (x, y), and its action space consists of four possible movements: up, down, left, and right.

2.3 Rewards

The agent receives the following rewards:

- $+100$ for reaching the goal.
- -10 for colliding with an obstacle.
- -1 for each step taken.

The agent aims to maximize its total reward by reaching the goal while avoiding obstacles.

2.4 Exploration vs Exploitation

In the early stages of training, the agent uses exploration to discover the environment by randomly choosing actions. Over time, the exploration rate ϵ decays, allowing the agent to exploit its learned knowledge and take the most optimal actions.

3 Challenges Faced

3.1 Dynamic Obstacles

One of the major challenges was implementing dynamic obstacles. Unlike static obstacles, dynamic obstacles change their positions during the simulation, adding a layer of complexity to the agent's navigation. The agent had to continuously adapt to these moving obstacles, which required careful handling of the state space and reward system.

3.2 Exploration vs Exploitation

Balancing exploration and exploitation during training was another challenge. Early on, the agent explored too much, which slowed down the learning process. As epsilon decayed, the agent gradually started exploiting its learned knowledge, but it sometimes got stuck in suboptimal paths. Fine-tuning the epsilon decay rate was crucial to finding a good balance.

3.3 Performance Monitoring

Tracking the agent's performance (time taken, collisions, and distance traveled) in real-time while ensuring smooth visualization was challenging. We used a CSV log file to track performance metrics across different episodes. This allowed us to analyze the agent's progress and optimize its training process.

4 Ideas for Improvement

4.1 Deep Q-Networks (DQN)

Currently, the agent uses a Q-table, which is suitable for small state spaces. However, for larger or more complex environments, using a neural network to approximate the Q-values (Deep Q-Networks, DQN) would allow the agent to handle larger state spaces more effectively and generalize better to unseen states.

4.2 Multi-Agent Systems

The environment could be enhanced by adding multiple agents with different behaviors and reward structures. This would simulate more complex interactions and could lead to interesting results in terms of agent cooperation, competition, or learning from others.

4.3 Dynamic Difficulty Adjustment

We could introduce dynamic difficulty adjustments to the environment. For instance, the number of obstacles or their speed could increase based on the agent's performance, creating more challenging training scenarios and potentially accelerating the learning process.

4.4 Better Obstacle Generation

The current grid-based obstacles are simple, but in a real-world scenario, obstacles can have more complex shapes and movements. We could implement non-grid-based, more irregular obstacle shapes or even obstacles that have complex movement patterns to make the task more realistic and challenging.

5 Conclusion

The Q-learning-based agent successfully learns to navigate a grid environment, avoiding obstacles and reaching its goal. Despite facing challenges like dynamic obstacles and exploration-exploitation balance, the agent shows promising performance. Future improvements could focus on scaling up the environment, using more advanced RL techniques, and enhancing the agent's learning capabilities.