Drina Baptiste

CS 370

Design Defense

For this project, I built an intelligent agent that learns how to navigate a maze using reinforcement learning, specifically through Deep Q-Learning (DQN). The goal was to design an agent that could find the treasure in the maze without having any prior knowledge of its layout. This paper explains how I approached the problem, how the agent works, and why this algorithm was the right choice. It also compares how machines and humans solve problems like this and reflects on what makes reinforcement learning effective for pathfinding.

Humans and machines approach problem-solving in very different ways. Humans rely on intuition, experience, and spatial reasoning. We can look at a maze and mentally visualize a path to the end. If we hit a dead end, we quickly backtrack and try a new route based on logic. On the other hand, a machine like the agent I built does not know anything about the maze at first. It must learn everything through trial and error. It moves around, collects feedback in the form of rewards or penalties, and over time builds an understanding of what actions lead to better outcomes.

If a person were placed in this maze, they would likely start at the beginning, look around to see where they can go, and then choose a direction that seems promising. They would probably aim for the direction of the goal and adjust if they ran into obstacles. My intelligent agent follows a similar approach, but instead of thinking in abstract terms, it learns patterns through repeated experience. At first, it explores the maze randomly, but as it gains experience, it starts to rely more on its internal model, a neural network that estimates the best action to take in each situation. This transition from exploration to exploitation helps the agent move from guessing to planning.

The key difference is that humans can reason about the future without needing to try every possibility. Machines, however, need data from every possible outcome to learn what works. But both systems rely on learning from mistakes and adjusting behavior over time. In this project, I tried to bridge that gap by designing an agent that learns efficiently while still behaving in a way that is like human problem-solving.

The purpose of the intelligent agent is to solve the maze on its own without being told what to do. It learns by interacting with the environment and receiving feedback in the form of rewards. If it reaches the treasure, it gets a positive reward. If it revisits cells or hits a wall, it gets penalized. Over time, it learns to choose actions that maximize its total reward. This kind of learning is incredibly useful in environments where hardcoding rules is not practical, and it reflects how intelligent systems can adapt to challenges.

One of the most important parts of reinforcement learning is balancing exploration and exploitation. Exploration means trying new things to learn more about the environment, while exploitation means using what you already know to make the best choice. At the start of training, the agent explores a lot, but as it gets better, it starts exploiting its knowledge more. I used a strategy that slowly reduces exploration over time. In this maze problem, that balance was key. Too much exploration slows down learning, while too little means the agent might miss better paths. I found that gradually lowering exploration once the win rate passed 90 percent helped the agent reach a perfect success rate efficiently.

Reinforcement learning is a great fit for this kind of pathfinding task because it allows the agent to learn through experience. Every action has a consequence, and over time, those consequences shape the agent’s behavior. By adjusting its internal model using feedback from the environment, the agent starts to figure out the best way to reach the treasure with the fewest mistakes.

To implement deep Q-learning, I used a neural network built with Keras. The network takes in the maze’s current state and outputs a score for each possible action. The agent chooses actions based on those scores. I also used an experience replay system that stores past experiences and samples from them to train the model. This helps the learning process stay stable and prevents the agent from forgetting important lessons. The model updates itself using a formula called the Bellman equation, which helps it estimate future rewards based on its current state and action.

In the end, the agent was able to reach the treasure 100 percent of the time. This showed that the learning process worked and that reinforcement learning, especially with deep Q-learning, is a powerful tool for solving complex pathfinding problems. It was exciting to see the agent improve over time, and this project really helped me understand how intelligent systems can be designed to learn and succeed on their own.