CS-370  
Professor Balogun  
Michael Turco  
22 February 2025

Project Two - Design Defense

In this project, we have created a Pirate Intelligent Agent capable of solving a small 8x8 maze with obstacles. The agent was trained using a deep Q-Learning algorithm, and was able to achieve a 100% win rate of the maze.

To solve this maze, a human player able to see the whole maze would be able to see the finish line, and navigate around the obstacles to get the pirate to the treasure. It would be fairly obvious to a human player that they need to go around the obstacles, and the player is probably already familiar with solving maze puzzles, even much bigger than this one. The agent, on the other hand, has no starting information about how to play the game, and only knows the possible actions it can take. When it starts, “all of its weights and thresholds are initially set to random values”, which basically means that it has never seen a maze before and doesn’t understand how any of it works (Hardesty, 2017). The pirate agent starts off by making moves at random, and trying to form connections between the situation that the action was taken, and the reward value received for doing so. A key difference between these two types of players, is that human players typically don’t start playing with 0 information about how to solve a maze, and even so, they could have other life experiences that would help them not make completely random moves. Human players don’t function off of the same reward system that the pirate agent does, where each individual move provides feedback. The agent also learns from thousands of trials of playing the game, whereas a human can solve the maze with no more than 2.

In pathfinding, ‘exploration’ is known as going to a new area instead of going for something you already know. Exploitation on the other hand, is choosing something that you already know the value of. Exploration can “gather fresh data” and “reduce uncertainty” by learning more about the potential rewards outside of where they have previously visited (GeeksforGeeks, 2024). Exploitation “concentrates on known high-reward actions” and “maximizes the immediate or short-term reward”, which can avoid uncertainty by sticking with known rewards (GeeksforGeeks, 2024). In the code, “epsilon = 0.1”, which means the agent will select an action using exploration 1/10 of the time, and will use exploitation 9/10 of the time. This value helps keep the agent stable since too much exploration can cause instability, and this value helps maintain a steady rise in performance over time. Reinforcement learning is able to help determine the path to the treasure by providing rewards for getting closer, and penalties for bad or invalid moves. This is able to ‘reinforce’ good decisions, and makes the agent more likely to pick actions that will eventually lead to it getting to the treasure. Reinforcement learning helps assign a value to each action or environment state, which leads the agent to make the most worthwhile actions that lead to the most benefit.

To implement deep Q-learning with neural networks for a similar pathfinding game, you must first define your maze as an array, and also define the set of actions available in any move. You must also select an epsilon value, which determines how often the agent will pick a move based on exploration (from 0 to 1). For a neural network, you must also define the number of epochs to run, the maximum amount of memory, and the data size to use for training. To code Q-learning, you start by making a loop that iterates through each of the training epochs. In this loop, a random starting cell is selected, and the agent is placed at that location. Until the game ends, the agent continually chooses random moves, either through exploration or exploitation (by comparing the epsilon value to a random float). After this, a reward is calculated, and the agent has the ‘experience’ added to their memory to use for later. Lastly, the model is updated based on the reward and experience, and then the game is either counted as a win or a loss. This concludes one epoch, and this is looped for as many epochs as you have set (I ran 1,000). I achieved a 100% win rate at epoch 996 out of 1000

References:  
  
GeeksforGeeks. (2024, May 18). *Exploitation and exploration in machine learning*. <https://www.geeksforgeeks.org/exploitation-and-exploration-in-machine-learning/>  
  
Hardesty, L. (2017, April 14). *Explained: Neural networks*. MIT News. <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>