

## **Project 2**

Sydney D Porter

Southern New Hampshire University

CS 370: Current/Emerging Trends in CS

October 27<sup>th</sup> 2024

## Project 2

For this project, I developed a solution using a Deep Q-Network (DQN) to help a pirate agent navigate a maze toward treasure. This implementation was written in Python and used two primary components: GameExperience and TreasureMaze. The pirate could move forward, backward, left, and right. Rewards were given when the pirate reached the treasure, while penalties were applied for invalid actions, returning to previous squares, or getting stuck. Reinforcement learning (RL) allowed the pirate agent to learn the best actions to take at each state through trial-and-error. Two key strategies—exploration and exploitation—guided this learning process.

In RL, exploration involves selecting actions that are new or uncertain to gather more knowledge, ensuring the agent finds better strategies in the long term. In contrast, exploitation uses the knowledge already gathered to select actions with the highest reward. Striking a balance between these strategies is critical, as too much exploration wastes time, while excessive exploitation can limit performance by narrowing the agent's learning.

To transition smoothly from exploration to exploitation, a learning rate decay was employed. According to Wilson (2022), adjusting the learning rate ensures the model explores early on, but gradually relies on learned patterns to improve efficiency. In this project, the decay rate was combined with epsilon ( $\epsilon$ ), a parameter controlling randomness in the agent's decisions. As the alpha value—computed as a function of epoch number and decay rate—dropped below  $\epsilon$ , the agent stopped acting randomly and started using its accumulated experience.

The balance between exploration and exploitation was manually adjusted by fixing epsilon at 0.1 and fine-tuning the decay rate. The agent initially explored the maze through random actions and progressively shifted toward more informed decisions. This approach mirrors how humans solve problems: starting with trial-and-error to understand the environment,

then using knowledge to plan better moves. However, unlike a human, the pirate agent lacked the ability to foresee obstacles several steps ahead, making its learning purely dependent on immediate feedback.


This limitation stems from the agent's model-free nature, meaning it relies entirely on experiences gathered during training and cannot predict future outcomes (Sutton & Barto, 2018). Humans, by contrast, can think several moves ahead to avoid dead ends before reaching them. In model-free RL, the agent learns incrementally and makes decisions one step at a time, which constrains its ability to strategize effectively.

At the start of training, exploration was prioritized to encourage the agent to try various paths and learn the maze's structure. As the number of epochs increased, the agent shifted toward exploitation to use the knowledge it had gained. Setting the decay rate to 0.1 meant the agent spent about 90 epochs primarily exploring before focusing on exploiting experience-based predictions. Although a human could likely solve a maze faster given their foresight, the agent's performance gradually improved within its constraints.

The primary objective was for the agent to learn a policy that maps states—representing positions in the maze—to actions. As it navigated the maze, each movement either earned rewards or penalties, guiding the agent toward maximizing total rewards over time. Positive feedback was provided for moving closer to the treasure, while penalties discouraged unnecessary backtracking or invalid moves. The agent's policy evolved over time as it refined the association between states and optimal actions through repeated interactions with the environment.

The DQN model used a neural network to learn the best action for each state-action pair. Here's the code that determined the rewards based on the pirate's behavior:

python

 Copy code

```
def get_reward(self):
    pirate_row, pirate_col, mode = self.state
    nrows, ncols = self.maze.shape

    if pirate_row == nrows - 1 and pirate_col == ncols - 1:
        return 1.0
    if mode == 'blocked':
        return self.min_reward - 1
    if (pirate_row, pirate_col) in self.visited:
        return -0.25
    if mode == 'invalid':
        return -0.75
    if mode == 'valid':
        return -0.04
```

Initially, the agent acted randomly to explore the maze, but over time, it switched to exploiting its learned strategies. This transition was guided by the following logic:

```
if 1 / (1 + epoch * decayRate) < epsilon:
    action = np.argmax(experience.predict(previous_envstate))
else:
    action = random.choice(validActions) # Select randomly from valid actions
```

The agent began successfully finding the treasure as early as the second epoch, and by epoch 248, it was able to locate the treasure with 100% success.

This project demonstrates how reinforcement learning can teach an agent to navigate a maze efficiently through the combination of exploration and exploitation. By controlling the decay rate and epsilon, the agent gradually improved its ability to find the optimal path. While the agent's model-free nature limits its ability to plan, the use of DQN allowed it to learn from past experiences and achieve consistent results over time.

## References

- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.
- Wilson, J. (2022, August 18). Understanding learning rate decay in machine learning. Towards Data Science. <https://towardsdatascience.com/understanding-learning-rate-decay-in-machine-learning-fc7bd0f26c1c>