The objective of this project is to design an intelligent pirate agent for a treasure hunt game. This agent represents (NPC) Non-player character whose goal is to find the treasure before the human player. Overall challenge is to learn how to navigate a maze-like environment filled with obstacles. In the end, I will attempt to implement a deep Q-learning algorithm that enables the pirate to learn optimal strategies through trial and error.

Human vs machine approach in problem solving sounds crazy if you think about. Trying to understand who is smarter, in a fundamental way. Human solving a maze might just rely on intuition. We can agree that some percentage of humans can adapt quickly, avoid repeating mistakes, and often use spatial reasoning to visualize paths. On the other hand, machines rely on mathematical models and other data-driven numbers. My pirate agent uses deep Q-learning, involving estimation of value-based actions given states and updating those estimates based on rewards received. Machines don’t understand the maze; they learn through repeated interactions and give feedback.

Human vs Machine Approach

|  |  |
| --- | --- |
| Identify entry points and scan surroundings | Start from a random free cell |
| Choose direction based on gut feelings | Observe the environment state |
| Explore paths, backtrack from dead ends | Choose an action using ε-greedy strategy |

* Are they similar? Yes, because both use trail and error and learn from feedback
* How can they be different? Humans generalize quickly; agents require many iterations and structured rewards.

The pirate’s agent’s purpose is to autonomously navigate the maze and reach the treasure efficiently. Exploration means trying new actions to discover better strategies while exploitation is using known actions that yield high rewards. In my previous attempts, I was using incorporating high exploration to gather diverse experiences. I wanted to make significant changes to see what happens. As the agent improves, ε should decay toward exploitation. The importance of finding a balance is catching the early mistakes.

Reinforcement learning (RL) enables the agent to learn optimal paths without explicit instructions. The pirate receives rewards for reaching the treasure and nothing for invalid moves or losing. (Gharbi, 2024). I attempted to implement deep Q-Learning using a neural network with input layers. Was this easy? Absolutely not. The challenges included:

* Indentation Errors and logic bugs in the training loop (Example 1)
* Undefined variables
* Balancing exploration and exploitation to avoid premature convergence

Example 1: [Codio - Pirate Intelligent Agent](https://codio.com/rjackson761161531/pirate-intelligent-agent/tree/TreasureHuntGame_Rolesshania_Jackson.ipynb) user: Jackson, Rolesshania

A screenshot of a computer

AI-generated content may be incorrect.

Despite encountering multiple indentation errors and runtime issues during development, I was able to make steady progress toward a deep functional Q-learning agent. Through repeated debugging and refinement, the pirate agent began to demonstrate successful pathfinding behavior, achieving several wins during training. Each iteration brought me closer to a clean, error-free implementation, and helped reinforce my understanding of Python’s structure, reinforcement of learning logic, and model training dynamics. These challenges ultimately strengthened my problem-solving skills and the full effect of feedback from instructors and peers.

Sources:  
Gharbi, A. (2024). Dynamic reward-enhanced Q-learning for path planning. <https://www.emerald.com/aci/article/doi/10.1108/ACI-10-2023-0089/1250201/A-dynamic-reward-enhanced-Q-learning-approach-for>

[CS 370 Pirate Intelligent Agent Specifications](https://learn.snhu.edu/content/enforced/2019746-CS-370-10424.202581-1/course_documents/CS%20370%20Pirate%20Intelligent%20Agent%20Specifications.pdf?ou=2019746)