CS 370

Professor Smith

Julia Coronado

April 27, 2025

**Final Project**

The objective of this project was to design and implement a Pirate Intelligent Agent capable of navigating a maze environment in search of a treasure. Using Deep Q-Learning (DQN), the agent learns an optimal policy to maximize rewards while avoiding penalties from obstacles and out-of-bound moves. The project utilizes Python along with the TensorFlow and Keras libraries for building and training the neural network model. This document describes the agent's design, development decisions, challenges encountered, and overall functionality.

**Agent Design and Strategy**

The Pirate Intelligent Agent applies a Deep Q-Network (DQN) approach to learn optimal movements through the maze. The environment is modeled as an 8x8 grid, where each cell can be free, occupied, or the treasure (goal). The agent starts at the top-left corner and can move in four directions: up, down, left, and right.

Key components of the design include:

**State Representation:** Each environment state is captured as an 8x8 matrix, representing the pirate's current position within the maze.

**Action Space:** The agent can perform four possible actions: move up, down, left, or right.

**Neural Network Architecture:** A feedforward neural network was provided, containing multiple dense layers with activation functions. The output layer uses a linear activation to predict Q-values for each possible action. The model is trained using the Adam optimizer and mean squared error (MSE) loss function. Experience Replay An experience memory stores past episodes (state, action, reward, next state, done). Random batches are sampled from this memory to break correlation between consecutive moves and stabilize learning. **Reward Structure:** Rewards and penalties guide the learning process: reaching the treasure grants +1, moving into obstacles or outside the maze results in significant negative penalties, and each normal move incurs a small penalty to discourage unnecessary wandering.

**Q-Training Algorithm:** The training loop updates the network by minimizing the difference between predicted and target Q-values. An ε-greedy strategy ensures a balance between exploration (random moves) and exploitation (using learned knowledge).

**Design Decisions**

Several important design decisions influenced the development of the agent:

1. Deep Q-Learning Over Simple Q-Learning: A Deep Q-Network was chosen over basic Q-learning due to the high dimensionality of the state space (8x8 matrix). DQN allows the agent to generalize learning across similar states, enabling faster convergence.
2. **Experience Replay:** To ensure stable and efficient learning, an experience replay buffer was implemented. Sampling random mini batches prevents strong correlations between sequential states and reduces variance during training.
3. **Exploration-Exploitation Trade-off:** An ε-decay mechanism was implemented, where the probability of random action selection decreases over time. This ensures that the agent initially explores the environment widely but gradually shifts to exploiting learned strategies.
4. **Penalty-Based Movement:** A small penalty for each step encouraged efficient navigation toward the goal, preventing infinite loops or unnecessary exploration within the maze.
5. **Negative Threshold for Early Termination:** To avoid endless episodes where the agent accumulates negative rewards without making progress, a negative reward threshold was used to terminate unproductive games early.

**Challenges Encountered**

During development, the following challenges were encountered and addressed:

* **Balancing Exploration and Exploitation:** Initially, a high exploration rate caused the agent to behave randomly for too long. Tuning the ε-decay rate helped the agent shift appropriately toward exploitation.
* **Sparse Rewards Problem:** Finding the treasure yields the only major positive reward. Early episodes struggled with low learning signals. Increasing training epochs and batch sizes helped the agent gather enough experience to improve.
* **Handling Obstacles and Walls:** Without penalties, the agent often attempted invalid moves into obstacles or off the grid. By properly penalizing these actions, the agent quickly learned to avoid them.
* **Training Stability:** To prevent diverging Q-values during training, careful tuning of the learning rate, batch size, and discount factor (γ) was necessary.

**Conclusion**

The Pirate Intelligent Agent successfully learns to navigate the maze using Deep Q-Learning principles. Through effective use of experience replay, a well-tuned ε-greedy exploration strategy, and structured penalties for undesirable actions, the agent can consistently find the treasure with a high success rate after sufficient training. Future improvements could include implementing Double DQN or Prioritized Experience Replay to further enhance stability and performance, especially in larger and more complex mazes.

**References**

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

Python Software Foundation. (2024). *Python documentation*. <https://docs.python.org/3/>

Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.