Keagan Ryder

CS 370

Project Two

Throughout Project Two, I developed a deep reinforcement learning agent that could intelligently navigate a maze to find a hidden treasure before a pirate opponent. This project required the integration of a neural network model with reinforcement learning principles to enable an agent to learn optimal behavior through repeated interactions with its environment. The maze was represented as a matrix, and the agent's goal was to maximize cumulative reward by discovering the most efficient path to the treasure while avoiding losing states. This project demonstrated how artificial intelligence can use experience and reward feedback to adapt and make decisions over time without explicit instructions.

In developing the Q-learning model, I implemented a deep neural network to approximate the Q-function that maps environment states and actions to expected rewards. The neural network utilized multiple dense layers with PReLU activation functions and the Adam optimizer to achieve stable learning. I also applied the Huber loss function to balance precision and robustness during training. To improve both speed and reliability, I incorporated a Double Deep Q-Network (DDQN) structure using a target network that periodically synchronized weights from the online model. This addition prevented overestimation of Q-values and allowed for smoother convergence. Furthermore, I optimized the training loop by reducing batch size, limiting steps per episode, and using reward shaping to encourage the agent to move closer to the treasure at every step. These enhancements greatly increased efficiency, allowing the agent to reach a one hundred percent win rate in significantly fewer epochs compared to the baseline configuration.

Through this project, I learned how to combine neural network design with reinforcement learning algorithms to create intelligent behavior from raw data and experience. The coding process deepened my understanding of how agents balance exploration and exploitation through epsilon-greedy strategies, and how replay memory can be used to stabilize learning by reusing past experiences. I also learned that tuning hyperparameters such as learning rate, epsilon decay, batch size, and gamma directly impacts the model's ability to generalize and reach optimal performance. Implementing Double Q-learning and experience replay provided practical insights into techniques that are commonly used in modern artificial intelligence systems such as autonomous robots and game-playing agents.

If given more time, I would experiment with alternative architectures such as convolutional neural networks or recurrent layers to handle larger and more complex environments. I would also test adaptive learning rates and prioritized experience replay to further improve training stability. This project not only strengthened my technical proficiency with machine learning frameworks but also reinforced the importance of iterative experimentation and analysis when developing intelligent systems. Overall, Project Two was a valuable opportunity to apply reinforcement learning concepts to a real-world simulation and to witness how an artificial agent can evolve from random movement to purposeful decision-making through experience and reward.