**Project Two: Design Defense of Pirate Agent**

Sean Bruyere

Southern New Hampshire University

CS-370 Current/Emerging Trends in CS

Dr. Obafemi Balogun

April 16, 2023

For many decades, scientists have been approaching ways to develop artificial intelligence in computers. From the early beginnings of the subject, they have looked towards humanity’s own memory infrastructures and thought processes to achieve its goal. However, the shear complexity of the human mind makes it difficult to fully replicate intelligence in AI agents. In this project, I took a basic step toward the goal of AI through reinforcement learning for a pathfinding agent. The overall purpose of the agent is to ‘learn’ the optimal path to find a treasure in a maze before a human player. Using a learning method called deep Q-learning, this project shows how the right combination of learning can create an AI that can outperform even the best human in specific tasks. AI still has a way to go before it can truly replicate the emotional, empathetic, and logical complexities of the human mind, but it certainly can perform many times greater in certain applications.

When it comes to mazes and puzzles, humans take a more visual approach to solving problems. In a pathfinding mission such as the treasure hunt game, a human may first recognize the starting point of the maze and the goal endpoint to get a sense of the layout. From the starting point, they can recognize which paths lead to dead ends or obstacles, and which are free to move through. It is a basic exploration of the maze prior to any decision making. Once they move to the next space on the board they can once again make these observations. They would largely explore the maze this way, gradually eliminating dead-end paths and narrowing down the options until the treasure is found.

Meanwhile, the agent designed for this project takes a reward based and random action approach to learning the maze. It may take a few runs or epochs to learn, but it can be a very effective means for the machine to optimize the path. How this works is that the agent starts at the starting point just like any other player. It will then generate a random action to change the state of the environment such as move left, right, up, or down. Once it makes this action, a reward is given based on the state. If a dead-end or obstacle is found the agent loses a point while if it enters a free space gains a point. The end goal is to explore and memorize how the state changes to maximize the reward given. The algorithm used for this model was deep Q-Learning, a reinforcement learning algorithm that gives an optimal future value to these actions. Each of which are place in a Q-table that is constantly updated for future decision making. As the agent learns, it can exploit these values to ultimately find the ultimate path to beating the human player. Within my model, this optimized at epoch 275, getting a 100%-win rate.

The largest similarities between these two methods of solving the maze is the use of exploration and exploitation to solve the problem. As they move along the maze both humans and AI need to reassess their state in the maze. They both will analyze any potential obstacles or dead ends that prevent from moving forward in the maze and remember the path that got them there. This will lead to exploitation of the maze were they to play another game. Both humans and computers will remember where the obstacles were to avoid them. However, it is arguable that the more complex the maze gets the better the AI agent would perform in the exploitation due to the memory capacities of humans. A large difference between the two approaches is the human ability to visualize each obstacle ahead and the use of a reward-based approach to decision making of the agent. These two different approaches, along with the capabilities of computers, provide a certain advantage the AI agent when it comes to solving the maze. Humans can be prone to errors that can cause them to take longer to solve the maze, even through multiple iterations especially as it gets more complex. Meanwhile, the reward-based system and Q-table memory capacity allows the agent to find tune and learn the path at greater speeds and with more accuracy. It shows this ability in the model, as it is able consistently obtain a 100%-win rate.

Exploration is an important part of any kind of learning process be it human or AI. It is an important part of gathering any unknown knowledge simply by visualizing or analyzing one’s current state. It is a way to determine what types of actions caused what reaction to learn an optimal result. Meanwhile, exploitation is the next step in learning and uses the knowledge provided in exploration to predict future actions. The problem with using both is determine the correct balance in the agent model to optimize the agent. If there were too much exploration, the model may just move aimlessly on the board. If there were too much exploitation there may not be enough previous knowledge for the model to even change state correctly. To provide a balance to the decision making the model provides an e-greedy approach to reinforcement learning. With this greedy approach “the idea is simply to take either the action that seems to be optimal with probability (1-epsilon) or a completely random action with probability epsilon” (Rocca) which is the exploration part of the decision making. The e value for my model is set at 0.1, which means that for every tenth episode, the agent will explore an action. This helps the model optimize its learning in case the exploitation portion of the Q-values are not accurate, and the low exploration value prevents the agent from simply wandering in the maze.

In reinforcement learning, an agent learns to take actions in an environment to maximize its reward signal. The agent interacts with the environment by taking actions, analyzing its current state, and receiving feedback from the environment in the form of rewards or penalties based on the action taken. The end goal of the agent is to learn a policy that maximizes the cumulative reward over time (Bhatt). Applying this method to the pathfinding agent, the policy needs to be one that determines the best action to take at each location in the maze. These actions could include moving in a particular direction or path. To start the learning, the agent initializes a random action to take and receives the reward/penalty feedback from the environment. A move to an empty space that has not been visited yet would provide a reward and the Q-values would be updated. A move to an obstacle or previously visited space would give a penalty. Finally, over time the agent would adjust its policy to reflect actions that lead to the highest rewards. Using this method, the agent is able to learn without ever needing past data to optimize its path. It simply learns from trial and error and provides the best and quickest possible means to find the treasure.

The training model for this pirate agent in particular applied the use of deep Q-learning algorithms. A large difference between this algorithm and what has been previously discussed is the use of a neural network in lieu of a Q-table to optimize policy. A way to understand how this model works is that “rather than mapping a state-action pair to a q-value, a neural network maps input states to action/Q-value pairs” (Wang). The first step in the learning process for the agent is to initialize the neural network. Here, there is an input layer that lakes in the maze size, a hidden layer that analyzes actions, and an output layer the outputs the actions. Next is to initialize the training algorithm. The goal is to find the optimal number of epochs to reach a 100%-win rate with the initial number of epochs set to 15000. First, a random action is chosen, and the model can begin its training. Once it begins the game, the actions and states are sent to the input layer of the neural network. Here it will begin to work on the policy that optimizes the model where each output node in the network layer represents an action. At the output layer of the network the action’s Q-value is updated. As the agent’s network updates, “the action that has the largest predicted Q-value at the output is the best-known action at that given state” (Wang). This network will continually be updated with each action until the agent wins or loses. For my particular model, the network optimized at the 275th epoch and was quickly able to win consistently well prior to that number.

Reference:

Beysolow, I. T. (2019). *Applied reinforcement learning with python : With openai gym, tensorflow, and keras*. Apress L. P..

Bhatt, S. (2019, April 19). *Reinforcement learning 101*. Medium. Retrieved April 15, 2023, from https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

Gulli, A., & Pal, S. (2017). *Deep learning with keras : Get to grips with the basics of keras to implement fast and efficient deep-learning models*. Packt Publishing, Limited.

Rocca, J. (2021, May 23). *The exploration-exploitation trade-off: Intuitions and strategies*. Medium. Retrieved April 14, 2023, from https://towardsdatascience.com/the-exploration-exploitation-dilemma-f5622fbe1e82

Wang, M. (2021, October 3). *Deep Q-learning tutorial: Mindqn*. Medium. Retrieved April 13, 2023, from https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc