**Project Two: Design Defense**

Sawyer E. W. Hannel

Southern New Hampshire University

CS-370

Prof. Scott Gray

June 15, 2021

**Project Two: Design Defense**

The intricacy of human learning and development plays an influential role in artificial intelligence (A.I.). When a toddler is beginning to learn the world, the toddler is largely learning through the consequences, or rewards, of their actions (Poddiachyi, 2019). For example, when the toddler is beginning to learn to eat properly, they must first examine the environment and state they are in, such as the food on the plate, the fork and the spoon. The toddler will begin exploring the state and performing actions. They may decide to grab the spoon and launch it at the wall, this would be an action that generates a penalty resulting in correction. They may also begin to eat properly, which would result in a reward such as being told they did a good job or gaining permission to play. The more positive rewarding actions the toddler performs, the more we reinforce those actions. Therefore, implementing reinforcement learning was the perfect choice for the treasure hunt game.

If a human were to play the treasure hunt game, they would first examine their position on the board relevant to the treasure. Assuming they can only observe their immediate surroundings, they would analyze their surroundings before using this knowledge in deciding where to move. They might first pursue only the open spaces, or they may make a random selection of multiple open spaces. However, they would continue to move and correct themselves until they either found the treasure or the pirate beat them to the treasure and they lost the game. This is similar to how our reinforcement agent approached the treasure hunt game, however, the A.I. reinforcement agent has the advantage of playing hundreds of games within a few minutes.

The reinforcement agent was developed with first setting up our environment. Our environment includes setting up the location of the treasure and layout of the maze. Additionally, we set up our reward system which gave us -1.0 for moving into a blocked square, -0.25 for moving into an already visited square, -0.75 for moving into an invalid square and a -0.25 for moving into a valid square. One of the fundamental concepts that makes reinforcement learning successful, and the reason all the rewards are a negative number, is not achieving the highest reward but achieving the lowest overall penalty, or loss score (Brownlee, 2019). The basis for this is a mathematical notion that we are always closer to zero than we are to infinity. If we were to give positive rewards and chase a maximum, this could result in a never-ending computation. Lastly, actions are given q-values through a sequential deep learning model which will run through a sequence of actions to give us the maximum total reward (Ankit, 2020). This process is known as deep q-learning or a deep q-network (DQN) algorithm. The q-values are what give the agent the probabilities of making each move. The q-values are determined through fitting our machine learning model with all the previous states, actions the agent made and final results.

With our environment set up and our reward and action system established, we begin to run our agent on their quest to find the treasure. First, the agent retrieves the environment and current state information. Now that our agent has the current state information it is time to decide for an action. However, before implementing deep q-learning, we must first decide whether to explore or exploit the environment. The design behind exploration and exploitation lies within the epsilon greedy algorithm. The epsilon greedy algorithm will determine a ratio of selecting a random action (explore) to implementing deep q-learning and picking the highest rewarded action (exploit). If we were to completely exploit actions, we may miss potential alternatives that would have been better, so, by utilizing random actions we provide a way of possibly finding better routes we would have missed through exploiting. With the epsilon greedy algorithm, the best ratio is exploration 10% of the time and exploitation 90 percent of the time. This is optimized for early into the learning process. However, as our model becomes more accurate (once we hit 90 percent win rate) we have less of a risk of impacting the efficiency of model with random actions. So, we can move the exploration factor higher to better confirm there was no potential routes missed that would improve the learning model.

After running the program, the agent was able to achieve a 100 percent win rate within 170 episodes of playing the game. As we can see from the results, the utilization of reinforcement learning and deep q-learning was the best approach to efficiently creating an agent to solve the treasure hunt game.

**References**

Ankit, C. (2020, April 27). Deep Q-Learning: An Introduction to Deep Reinforcement Learning. *Analytics Vidhya*. [https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/](https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/%20)

Brownlee, J. (2019, October 22). *Loss and Loss Functions for Training Deep Learning Neural Networks*. Machine Learning Mastery. [https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/.](https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/.%20)

Poddiachyi, A. (2019, November 21). Reinforcement Learning, Brain, and Psychology: Introduction. *Towards Data Science*. <https://towardsdatascience.com/reinforcement-learning-brain-and-psychology-part-1-introduction-b5f79a0475ab>