**7-3 Project Two Submission**

Mark Meyer

Prof. Kalysa Wilson

CS-370-H7202

April 9, 2022

**7-3 Project Two Submission**

The steps a human being would take to solve this ‘treasure hunt maze’ problem depends on what perspective the person has for an environment. If a person can see the maze as we do on the screen, it would be extremely easy because the solution could be seen and would likely find the treasure on the first try of the maze. If a person were inside the maze and could only see walls in front of them, it would still be quite easy as the maze is rather elementary, and a person would treat it like walking down a winding hallway with a treasure at the end and would find it the first try as well.

However, if a human could only see what the AI model sees which is a reward value provided by every adjacent block and only having the view of a single block to determine the next best move, then it is likely that a human would follow the same methodology as the AI model in seeking the treasure as the highest reward and choosing future moves based on what the person has learned since they started the maze. The difference is that a human would not necessarily be able to remember every single move they previously made or every single reward that led them to make a correct move or wrong move for that matter. With that said, it is highly possible that over the course of multiple games, the AI would prove to be more efficient in solving the maze providing both the AI and human were given the same information to solve the maze. The steps an intelligent agent takes, in this case using a REINFORCEMENT algorithm called ‘Q-Learning’, is to be provided possible actions relative to the agent position and, until the treasure is found, will formulate a Q-value using the Bellman equation that gives the agent a reward for the current position on the maze and select an action which ‘maximizes the expected return’ on the next move based on the policy of accumulated experience (an array of previous states, actions, and rewards) (OpenAI, 2018). Additionally, according to OpenAI (2018), when the Bellman equation includes the optimization of ‘max over action’, an action must be selected leading to the highest reward—as shown in this model:

“action = np.argmax(experience.predict(envstate))”.

The differences between the approach of a machine versus that of a human, assuming the same information is given, are that a machine will not have the interference of human error, memory loss, distraction, and other opportunities of inconsistency unless the values set in the AI model allow such. Otherwise, a human has the advantage of sight which far exceeds any computing advantages in a simple maze like this. The similarities of approaches between machine versus human is that given any current position and knowledge of possible rewards and actions, both machine and human agent will use experience to make the best possible decision for predicting and/or choosing the best action for accumulating the largest future rewards. Once a move is made, while the treasure has not been found, both agent types will also update their ‘memory’ or experience with an assessment of their latest move. With that said, another difference would be that a human can quickly change their overall and/or move-to-move strategy on the fly using any sort of abstraction, whereas a machine is limited by its model.

The difference between exploration and exploitation in reinforcement learning is exploration allows an AI model to expand its knowledge outside the bounds of its calculations knowing that new—usually random—introductions of experience can ultimately lead to a higher reward; by contrast, exploitation is where an AI model knows something works and becomes greedy in its calculations as far as using a successful—or at least predicted to be successful—method repeatedly knowing very well it will lead to a higher reward (Lindwurm, 2019). The ideal proportion of exploitation versus exploration for the model of this pathfinding problem was 20:80—that is 20% exploitation and 80% exploration. The reason this ratio works for this maze is because there are very few paths to take. The maze is very linear in design (versus an open world design), so for the agent to venture out and explore, assuming the agent is told the location of walls (which it is), can only lead the agent closer to the treasure. In doing so, the agent quickly learns the path to follow to get to the end rather than relying on the tunnel vision of exploitation. Instead, exploitation is utilized effectively after the environment is quickly learned through exploration.

Reinforcement learning helps to determine the path to the treasure by the agent when the agent successfully advances through the maze and the policy, network of state-action pairs with their corresponding Q-values, is updated to reflect that success (Wang, 2021). The more success the agent experiences, the higher the Q-value becomes in its calculations, and the better values the model can provide to the agent to make better decisions. As the accumulation of rewards increases, the probability of making decisions that advance toward the treasure increases. As more games are played and treasure is found, the agent learns to adapt from various starting points and its policy becomes increasingly optimized to manage unexpected starting points and situations (Yoon, 2019).

The deep Q-learning algorithm was implemented in this game by utilizing an *experience replay object* to store and update the agent’s experience of previous state, action, reward, current state, and game status to be applied to the neural network. Once the neural network receives the update, the model is trained and may calculate any penalties of a bad prediction—often referred to as ‘loss’. Importantly, the model takes in inputs of actions paired with Q-values, each of which is associated with a state. The output of the network is the new action determined by the Q-value. In this model, the Q-value is updated using a greedy Epsilon algorithm where random actions are chosen based on a specified probability value (epsilon), and if a best action can be determined, it is executed; otherwise, another random action takes place (Wang, 2021). Between observing past games via experience replays, observing loss, and updating Q-values, the model is trained on the best possible path to equip the agent with a neural network of intelligence for each subsequent game.

**References**

Lindwurm, E. (2019, November 3). Intuition: Exploration vs exploitation. Medium. Retrieved April 9, 2022, from <https://towardsdatascience.com/intuition-exploration-vs-exploitation-c645a1d37c7a>

OpenAI. (2018). Part 2: Kinds of RL algorithms — Spinning up documentation. Welcome to Spinning Up in Deep RL! — Spinning Up documentation. Retrieved April 9, 2022, from <https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html>

Wang, M. (2021, October 3). Deep Q-learning tutorial: MinDQN. Medium. Retrieved April 9, 2022, from <https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc#:~:text=Critically%2C%20Deep%20Q%2DLearning%20replaces,process%20uses%202%20neural%20networks>

Yoon, C. (2019, May 23). Deriving policy gradients and implementing reinforce. Medium. Retrieved April 9, 2022, from <https://medium.com/@thechrisyoon/deriving-policy-gradients-and-implementing-reinforce-f887949bd63>