**7-3 Project Two Submission**

Tomas J. Estanislao

Southern New Hampshire University

CS 370: Current and Emerging Trends in Computer Science

Professor Timothy Alexander

August 20, 2023

**7-3 Project Two Submission**

A human could potentially solve a pathfinding problem by randomly choosing open paths to take. Throughout, they would build a mental map in their head to remember unexplored paths. Another option would be having some basic rule to follow, such as always choosing the leftmost path, and still using a mental map. The intelligent agent uses a trial-and-error approach to gradually learn how to solve the maze. It explores the maze, updates Q values, and learns which actions to take—maximizing total rewards over time. This agent uses a deep neural network in conjunction with Q-learning, “A DQN agent is a value-based reinforcement learning agent that trains a critic to estimate the return or future rewards” (MathWorks). Both approaches require exploration to discover untread paths. However, DQN’s use algorithmic processes to learn and exploit the environment— finding optimal solutions to the maze. Humans use memory, reasoning, and intuition to solve the maze.

Exploitation is using the agents best guess, regarding maximum reward, to make an action. Exploration is making a random action and using that information to refine its learning algorithm. A greedy reinforcement learning method always chooses the best decision given current knowledge, “The main drawback of this behaviour lies in its lack of exploration: if our knowledge is not accurate enough, we can be « stuck » and keep choosing a sub-optimal decision forever without any chances to discover better options” (Rocca, 2021). This agent uses an e-greedy algorithm to make decisions. It explores randomly at a .1 probability, allowing the agent to exploit 90% of the time. When the agent gets good enough to have a win rate of 90%, epsilon decreases to .05, expediting the agent by allowing it to exploit 95% of the time. This method of reinforcement learning allows for more learning in the beginning, and efficient solving at the end. Different problems require different strategies. Reinforcement learning may require simulating different strategies in order to efficiently solve a given problem.

The DQN uses a neural network model with 2 hidden layers to approximate Q-values. There are N number of neurons, where N equals the size of the maze, composing each input and hidden layer. The output layer contains N neurons for each N number of actions available. There are two stages: sampling and training. In sampling, values are used to predict an optimal action to take when exploiting. Episode information is inserted into an experience object, including values for: previous state, action taken, reward received, current state, and game status. This experience data is used to train the model using the model.fit method. The purpose of this is to update the neural network weights, allowing the model to make better predictions. This is an implementation of the bellman equation to minimize the error between predicted and target Q-values. Experience replay is an implementation used to stabilize training, “Experience replay helps by using the experiences of the training more efficiently. We use a replay buffer that saves experience samples that we can reuse during the training. This allows the agent to learn from the same experiences multiple times” (Simonini, 2018). To summarize, the goal is to gradually learn an optimal Q-value function for solving the maze. The agent predicts cumulative rewards to determine an action, saves its experience data, trains on it, and is better able to predict subsequent actions. At a 90% success rate, it accelerates its exploitation. Tuning hyperparameters is essential for Machine Learning; Neurons, batch sizes, and epsilon values all can contribute to massive performance effects. It’s also important to run extensive tests to ensure the agent doesn’t get stuck at a local maxima— solving extremely fast in optimal tests— but not converging in other tests. A policy based, or hybrid, approach may lead to better performance for this pathfinding problem.

**References**

MathWorks. (n.d.). Deep Q-Network (DQN) Agents. https://www.mathworks.com/help/reinforcement-learning/ug/dqn-agents.html

Rocca, J. (2021, April 18). The exploration-exploitation trade-off: intuitions and strategies. Medium. https://towardsdatascience.com/the-exploration-exploitation-dilemma-f5622fbe1e82

Simonini, T. (2018). The deep Q-learning algorithm - hugging face deep RL course. Hugging Face. https://huggingface.co/learn/deep-rl-course/unit3/deep-q-algorithm