**Module 7: Project Two**

**CS-370**

**Brandon Hobbs**

**April 11, 2023**

I have been asked to design an approach to solving a pirate maze. For this task I created a Deep Q Network (DQN) in Python that drives the *GameExperience* and *TreasureMaze* classes. The pirate was able to move forward, backward, left and right. The pirate agent was rewarded for reaching the treasure and penalized for being blocked, vising the same square, and trying an invalid move. The pirate was trained through reinforcement learning on the optimum action for each state. The strategies adopted for learning fall into two categories: exploration and exploitation.

Exploration in reinforcement learning refers to a strategy of choosing actions that the agent has not yet tried, or that it believes will lead to new information or knowledge that can improve its long-term performance. Exploration focuses on acquiring new information and knowledge that can lead to better decisions and improved performance in the future.

Exploitation refers to a strategy of choosing actions that the agent believes will result in the maximum immediate reward based on the knowledge it has learned so far. Exploitation focuses on maximizing the agent's short-term performance by making decisions based on its current knowledge.

To have the agent explore a lot in the early attempts but use gained knowledge in later attempts a learning rate decay was used (Haswani, 2020). A simple alpha value was calculated using the reciprocal of the epoch number multiplied by the decay rate. If alpha was below the threshold, epsilon (ε), the agent stopped randomly wandering and used gained knowledge.

Without resorting to hyperparameter sensitization, epsilon remained at 0.1 and only the decay rate was manually tuned.

This early, randomized wandering and then relying on gained knowledge is like similar to how a human may solve the maze. They may start off wandering early to discover possible paths and then stringing that gained knowledge into useful ideas. Where the pirate agent and a human differ is forward visibility and intuition. That is, the pirate agent cannot see beyond its current square and may walk into an obstacle blindly – a human probably would not. This is because reinforcement learning (RL) with DQN is model-free.

Without a “model of the universe” the agent “can be thought of as [using] an "explicit" trial-and-error algorithm” (“Model Free,” 2023). Meaning that the agent cannot think beyond or plan ahead the current move while in learning mode. This does not pertain to a human – who could easily look many moves ahead and decide that the remining forward moves all result in a dead end and change path before reaching the obstacle.

In general, at the beginning of the learning process, exploration should be given more priority to allow the agent to explore different paths and learn about the environment. As the agent gains more experience and knowledge, the balance can shift towards exploitation to make use of that knowledge and achieve better performance.

To achieve a balance between the strategies the decay rate was manipulated. Eventually a decay rate of 0.1 was settled on for the epsilon value of 0.1. This would allow for 90 epochs of random learning (trial-and-error) before experience-based predictions were used. A human may not need 90 attempts to solve a simple maze given their foresight, but if that visibility were restricted to a single move – hard to say.

***How can reinforcement learning help to determine the path to the goal (the treasure) by the agent (the pirate)?***

As mentioned previously, the agent learns through actions. At each state the agent learns a policy that maps that state to an action. In the context of pathfinding, the states correspond to the different locations on the map, and the actions correspond to the movements that the agent can make to navigate the map.

After each action the agent is rewarded or punished with a goal of maximizing the total expected reward over time. As stated before, exploration and exploitation were used to define the actions available to choose within the states

During the learning process, the agent explores the environment and updates its action-value function based on the feedback it receives. As the agent gains more experience, its action-value function converges to the optimal values, and the policy it learns becomes more accurate and efficient at finding the optimal path to the goal.

***How did you implement deep Q-learning using neural networks for this game?***

DQN was used to map the action-state pairs. With each action a reward or punishment was determined using this code block:

*def get\_reward(self):*

*pirate\_row, pirate\_col, mode = self.state*

*nrows, ncols = self.maze.shape*

*if pirate\_row == nrows-1 and pirate\_col == ncols-1:*

*return 1.0*

*if mode == 'blocked':*

*return self.min\_reward - 1*

*if (pirate\_row, pirate\_col) in self.visited:*

*return -0.25*

*if mode == 'invalid':*

*return -0.75*

*if mode == 'valid':*

*return -0.04*

As the agent transitions from exploration to exploitation these rewards guide the agent on the next best possible move. This decision, as stated before, was based on a decaying learning function. Below a threshold (ε) exploitation is used where exploration is used above this threshold. The code is shown below:

*if 1 / (1 + epoch \* decayRate) < epsilon:*

*Action = np.argmax(experience.predict(previous\_envstate))*

*else:*

*Action = random.choice(validAction )#randomily pick from the validActions*

The agent began “winning” (finding the treasure) in the second epoch and by epoch 248 the agent was winning 100% of the time. See Figure 1 for the final path of the pirate agent taken to find the treasure.

A black and white checkered background

Description automatically generated with low confidence

**Figure 1: Final Path (80% shading) Taken by the Pirate Agent to Find the Treasure (50% shading)**

**Citations:**

Haswani, V. (2020, September 3). *Learning rate decay and methods in deep learning*. Medium. https://medium.com/analytics-vidhya/learning-rate-decay-and-methods-in-deep-learning-2cee564f910b

Hobbs, B. (2023, March 25). *Project one*. [Unpublished report]. SNHU

Model-free (reinforcement learning). (2023, March 3). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Model-free\_(reinforcement\_learning)&oldid =1142602733

OpenAI. (n.d.). *Part 2: kinds of RL algorithms*. https://spinningup.openai.com/en/latest/spinningup/rl\_intro2.html#a-taxonomy-of-rl-algorithms

Yang, A. (2022, July 24). *What is exploration vs. exploitation in reinforcement learning?* Medium. https://angelina-yang.medium.com/what-is-exploration-vs-exploitation-in-reinforcement-learning-a3b96dcc9503