Zac McBride

CS 370 7-3

August 10, 2023

Project Two Pirate Game Design Defense

The Pirate Treasure Hunt Game is a board with a maze and treasure at the end of the maze. The point of the game is for the player to get to the treasure before the pirate does. This is essentially a pathfinding problem. The purpose of the code is to create an AI agent that can traverse this maze and reach the end goal using a deep Q-learning algorithm. The goal of the agent for this project is to have it reach the treasure every time it attempts to run the maze.

An article published in Computers & Graphics contrasts how a human and an AI agent would go about solving a pathfinding problem. For a human, if they know the environment, they will solve this problem by using a mental map of the area. If they are unfamiliar with the environment, they will still have a vague idea of how to get to the goal and will use this knowledge to reach it (Rahmani, 2022). Most humans would use previous knowledge of like environments to intuit how to reach the goal. They would also use outside resources, such as a map or asking some else for directions.

A deep learning agent would most likely not be able to use this approach. “While existing techniques give possible solutions for practical applications, they do not typically take into account human factors to closely simulate how humans behave in the real world” (Rahmani, 2022). The solution implemented for this project uses a Q-learning algorithm to have the agent find the treasure. Akshay Lamba, a writer for the Medium explains how this algorithm works; the agent uses a Bellman equation to predict the future reward of an action based on its current state (2022). As the agent performs actions the q-value is updated, which gives a more accurate prediction for the future reward of its next action. This q-value is used to create a q-table. “Q-Table is just a fancy name for a simple lookup table where we calculate the maximum expected future rewards for action at each state. Basically, this table will guide us to the best action at each state” (Lamba, 2022). The solution for this project uses Deep Q-learning. The difference between Q-learning and Deep Q-learning is that “Deep Q-learning approximates the values in the Q table rather than trying to populate them manually” (Beysolow, p. 65, 2019).

As humans we may begin by randomly exploring an environment, we are unfamiliar with. The Q-learning agent will have to do the same. The difference is the agent will have to do an exhaustive search of the environment to learn the best possible actions based on the current state. A human on the other hand will rely on knowledge they already have to intuit the best way to path find through an environment. This is like saving the action/reward tuple in memory for future use (Beysolow, 2019).

Because Deep Q-learning uses approximations instead of memorizing an entire data set, there needs to be a balance between exploration and exploitation. Initially, the agent will need to explore the environment to begin optimizing the q-values. It will take a random action and observe the reward, then use this to update the q-value. This is done by using the Epsilon-Greedy Algorithm (Beysolow, p.59, 2019). This is controlled by the epsilon value in the code for this project. When the epsilon value is 0.1, the algorithm will use the q-values 90% of the time and choose a random action 10% of the time. This allows the agent to reach the balance between exploring the environment and exploiting the Q-learning algorithm.

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

For the pirate game the agent has four actions it can take. It can go up, down, left, and right. The environment consists of a grid which has squares the agent can move to, and squares that represent obstacles where the agent cannot move to. There is also one square that contains the treasure. If the agent makes it to this square, it wins the game. When the agent moves to one of the square types, it receives an award. The agent is penalized for trying to move off the board, into an obstacle, moving into an empty square, and moving into a previously visited square. The value of the penalty depends on the severity of the move. It only receives a positive reward for landing on the treasure square. The Deep Q-learning algorithm is used so that the agent can maximize the total reward.

The model the agent uses has three dense layers which compute the q-values of the algorithm. The agent initially selects a random free cell at the beginning of each epoch and resets the loss, episode, and maze. It then loops until the game state ends (which happens when the reward decreases too low, or the agent gets to the goal). It then uses the model to predict the next action it should take. After the action is taken it observes the reward given. The q-values are updated, and this pattern continues until the game ends. The code will run until a set number of epochs is reached or the agent reaches a 100%-win rate.

Sources:

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

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.

Lamba, A. (2018, September 3). An introduction to Q-learning: Reinforcement learning. Medium. https://medium.com/free-code-camp/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc

Rahmani, V., & Pelechano, N. (2022, February 25). *Towards a human-like approach to path finding*. Science Direct. https://www.sciencedirect.com/science/article/abs/pii/S0097849321001849