Lawrence Artl III

CS 370 22EW1

Project Two

October 12, 2022

Pirate Intelligent Agent Design Defense

The Pirate Intelligent Agent project employs the use of a Deep Q-Learning network in order to guide an agent through a maze to a "treasure" goal. How an intelligent agent solves a simple maze is different from how a human agent approaches the same problem, but there are some similarities as well. This paper summarizes those differences, as well as the solution the intelligent agent uses in order to solve the maze as efficiently as possible.

When solving a maze, a human agent typically has the ability to see the whole maze, including the start and the finish and all paths in between. Taking in the whole picture allows the human to make decisions about which direction is the best to take. A human agent can see that they are going in generally the correct direction towards the finish. Additionally, human agents can avoid obstacles without considering them as a move that offers them a negative reward; the rules of the maze are to not go through walls, so the agent never has to waste time in thinking about it.

An intelligent agent solving the pirate treasure maze does not have an understanding of the concepts of “towards the finish” or “away from the finish” like a human agent does. The rules are much the same for the maze, but the agent must be taught in simple terms that borders are to be avoided, and certain patterns of spaces will lead to the finish quicker than others. For this to work, the intelligent agent uses a table or “Q-Table” that lists the values of each move the agent can make, and what value that move has (Venkatachalam, 2019). Moves that take away “points” are to be avoided, and moves that earn points are preferred. By assigning positive

points to moves that help the agent reach the goal and negative points to moves that place the agent in zones it is not supposed to go, the agent can make informed decisions about its approach.

The human agent and the intelligent agent both solve the maze in a similar fashion by "looking ahead" at what path will get it to the finish the fastest or in the most reliable (no back-tracking) manner (Venkatachalam, 2019). The human agent lacks the ability to tabulate all possible moves and their value, but that doesn't stop them from estimating the perceived value of a move based on other factors. A human agent is likely to move in a direction that moves them closer to the end-goal; similarly the intelligent agent will move towards the end-goal by picking the moves that indicate it is traveling in the right direction. A human agent need not be given negative points for moving into a "barrier" in the maze, as the rules of the maze are well known. The intelligent agent must learn this rule about barriers, but once it understands this it will avoid them just as well.

In any situation there is some level of decision making that involves going with what one knows versus taking a chance. This is the difference between exploitation and exploration as it applies to learning in artificial intelligence. An agent has the option to choose a move that it knows will garner it the greatest rewards based solely on current information. The action performed during exploitation is sometimes referred to as "greedy action", as it denotes an action that is designed to get the biggest payoff but in the short-term only (*Exploitation and Exploration in Machine Learning*, n.d.). Whether that action will result in larger gains overall is unknown to the agent. Exploration, however, is a set of actions that may not have immediate large benefits, but will likely have larger payouts in the end. Agents try to gather more information about the environment and the states that certain actions will result in in order to make the best overall decision (*Exploitation and Exploration in Machine Learning*, n.d.).

A good ratio between the two strategies is important for learning, but the overall goal of the learning is what can help to determine the best ratio. Increasing an agent's exploration rate to the highest (or near highest) possible value can result in much slower learning as they attempt to gather information about every possible move and resulting scenario that could transpire. However, forcing the agent to only use exploitation as a means of solving a problem can result in an agent that may find the solution to the problem, for instance the goal of a maze, but not in the best way(shortest path) possible. Therefore it is important to ensure that an agent has the opportunity to use both exploitation and exploration methods to solve the problem at hand; the specifics of the problem can help determine what this ratio should be.

Reinforcement learning algorithms aim to maximize the cumulative reward for an agent when the agent is placed in an unknown environment (Schoberg, 2020). In this way, RL algorithms can help to solve a maze like the pirate-treasure hunt by teaching the agent which square-sequences to follow in order to gain the maximum amount of points at the end. As the agent learns which moves / squares result in a loss of points and which moves / squares result in a gain of points, the agent can start to determine the best path towards the end goal, which triggers the end of the game.

The standard Q-Learning algorithm employs the use of a Q-table. The Q-table is created from all of the possible states and actions that are available to the agent. This table is used by the agent to determine the next move in solving the problem of navigating the environment (maze). This works well on small environments, but when environments are particularly large with potentially millions of Q-Values to populate the Q-Table, a Deep Q-Network is preferable. In the DQN, two neural networks work together to approximate the values that would normally occupy the Q-Table (Ausin, 2020). As one network adjusts its weights through the course of learning, the weights are transferred to the second network after a time, improving the approximations of the values that the networks compute.

The “Pirate Treasure Maze” implements a Deep Q-Network for the purpose of solving the maze. The designed model uses the previous environmental state to predict q-values for future actions. The agent then chooses an action to take, the environment is updated, rewards for the action are returned, and the next state is evaluated. As mentioned before, a degree of exploitation is implemented as well, giving the agent the ability to seek immediate high rewards in lieu of long-term rewards that may or may not materialize. However, some exploration is expected as well, which manifests itself in the comparisons of various final solutions to the maze. Given different runs, the agent will choose different paths to get to the same treasure square (see figure below). This likely is explained by the addition of exploration, as pure exploitation would, in theory at least, yield the same results every time.

While the intelligent agent may not work the same as a human agent when solving a maze such as the one in this project, the benefit is that the agent's learning can be saved and used for future projects. Additionally, the methods used can be refactored for other problems that may not be similar in their structure. The Deep Q-Learning network has a wide variety of applications in Artificial Intelligence problem solving, and it can be built fairly easily, as exemplified in this project.

References

Ausin, M. S. (2020, April 2). *Introduction to Reinforcement Learning. Part 3: Q-Learning with Neural Networks, Algorithm DQN*. Markel Sanz Ausin. Retrieved October 9, 2022, from https://markelsanz14.medium.com/introduction-to-reinforcement-learning-part-3-q-learning-with-neural-networks-algorithm-dqn-1e22ee928ecd

Exploitation and Exploration in Machine Learning. (n.d.). Javatpoint. Retrieved October 9, 2022, from https://www.javatpoint.com/exploitation-and-exploration-in-machine-learning

Schoberg, S. (2020, April 9). *Introduction to Reinforcement Learning (Q-Learning) by Maze Solving Example*. Medium. Retrieved October 9, 2022, from https://medium.com/analytics-vidhya/introduction-to-reinforcement-learning-q-learning-by-maze-solving-example-c34039019317

Venkatachalam, M. (2019, October 2). *Q-Learning - An introduction through a simple table based implementation with learning rate, discount factor and exploration*. gotensor. Retrieved October 9, 2022, from https://gotensor.com/2019/10/02/q-learning-an-introduction-through-a-simple-table-based-implementation-with-learning-rate-discount-factor-and-exploration/