Project Two – Design Defense

CS-370-R3355 Current/Emerging Trends in CS

Raphael Coloma

February 21, 2024

Starting out, a human being would have the advantage of solving the maze puzzle and locating the treasure over an intelligent agent. Humans can think critically, letting us evaluate our environment and quickly make decisions that appear advantageous. We are also more curious and less averse to risk than an intelligent agent would be. Humans would not make all the optimal decisions through the maze, but we would adapt to the puzzle easier and probably complete it on the first run. Whereas an intelligent agent would require multiple episodes of trial and error before reaching the end and finding the treasure.

A human being would start by first evaluating their environment. They would then begin exploring the maze and gain experience to help them navigate and find the treasure. They would develop strategies to reach their goal such as exploiting maze boundaries or making mental/physical notes of places they’ve traversed. They would continue to explore and map their position until they reach the end and find the treasure.

An intelligent agent first evaluates its state and then determines what actions it can take. In the beginning, they will explore the environment and choose actions randomly. They will receive rewards varying from –1 point to 1 point depending on their actions. After they have acted, they’ll observe the outcome and evaluate the reward (Lamba, 2018). The episode will be stored in its experience replay and be used to update the learning model. This repeats until the intelligent agent wins or loses. The training ends when the intelligent agent reaches a 100% win rate. At this point, the intelligent agent will beat the human every time.

Both humans and intelligent agents have the same approach to solving the treasure maze but execute it differently. Both evaluate their environments while exploring. Both develop strategies from their experiences and use that to navigate the maze. Both take risks to make progress. Humans can process possible decisions and calculate the best option immediately. They can adapt to challenges and have an advantage when they start because of their prior knowledge about the world around them. The intelligence agent learns and gains experience as it trains through episodes. Like a human, it can evaluate what actions to take based on its experiences, but it must make those decisions and learn from them by calculating the max rewards. Without prior knowledge, it will act randomly to learn and gain experience. But intelligent agents have the “mathematical ability” to calculate their moves (Ryther, 2021). With practice and policy updates, intelligent agents can anticipate their actions and devise the most optimal route for the max rewards with the Epsilon Greedy algorithm (Ryther, 2021).

To understand the Epsilon Greedy algorithm, we must first look at what exploitation and exploration are. In reinforcement learning, exploitation and exploration are tactics an intelligent agent would use when interacting with its environment. “Exploration is all about finding more information about the environment, whereas exploitation is exploiting already known information to maximize the rewards” (ADL, 2018). There needs to be a ratio of these two tactics; explore too much and the intelligent agent will wander aimlessly, exploit what is known will not take risks in search of greater rewards.

The Epsilon Greedy algorithm is an algorithm that starts with a constant between 0 and 1; this is the epsilon. For the code, the epsilon is set to 0.1. This is the threshold for determining if the intelligent agent is going to explore for more information or exploit the information it knows. For each episode, a random number is chosen and if it is lower than the epsilon, the intelligent agent will explore; if it is higher, then it will exploit. Once the win rate of the intelligent agent becomes greater than 90%, the epsilon changes to 0.05. At 90%, the intelligent no longer needs to explore the maze and can exploit what it knows to increase its win rate to 100%. The Epsilon Greedy algorithm allows the reinforcement learning policy to develop a basis in safe decision making while allowing risk to find better outcomes (Ryther, 2021). Finding the ideal proportion of exploitation and exploration is difficult, but the Epsilon Greedy algorithm is one of the main strategies to tackle this problem. Having an algorithm that dictates when the intelligent agent should explore or exploit ensures that it is the model is receives adequate information to make optimal decisions.

Reinforcement learning works by executing the best possible actions to receive the maximum number of rewards (ADL, 2018). Reinforcement learning uses rewards earned through exploitation and exploration to make sure that the path to the treasure is optimized. If the pirate is always making safe choices by exploiting its knowledge, it may not be receiving the highest possible rewards resulting in poor performance and optimization. By exploring and taking risks, the pirate can reap higher rewards to enhance its policy; this way, when it exploits, it will have better opportunities (Ryther, 2021).

To implement the deep Q-learning I first established the number of epoch and setup a loop for each one. The loop begins with the pirate starting on a free cell at the top left of the maze and the environment is reset. While the game status is “not\_over”, the model determines the action of the pirate for that state (whether it explores or exploits). Once an action is taken, the reward is calculated, and the experience is stored as an episode that is remembered as a “replay object.” This gets stored as the input and target and is passed to the model. The loop repeats using and building the pirate's experiences until the pirate either wins or loses. Once the win rate reaches 90% the epsilon decreases to 0.05 to increase the probability of the intelligent agent to exploit what it knows until it reaches 100% win rate.

References

ADL. (2018, August 27). *A brief introduction to reinforcement learning*. FreeCodeCamp. Retrieved February 24, 2024, from <https://www.freecodecamp.org/news/a-brief-introduction-to-reinforcement-learning-7799af5840db>

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>

Ryther, J. (2021, December 4). *Treasure Maze Reinforced Learning*. Medium. Retrieved February 23, 2024, from <https://medium.com/@jryther91/treasure-maze-reinforced-learning-744a9aa80a71>