Project 2 Design Defense

Text

Description automatically generated I am an AI developer working for a gaming company. I have been tasked to design an intelligent agent for an NPC character in a treasure hunt game that players will race against to solve a maze. At the end of the maze is the hidden treasure. To make the maze easy for the computer to understand, we represent it as an 8x8 matrix.

Text

Description automatically generated The way a human might approach solving the maze is by picking a start point and testing the direction they feel has the best chance of leading them to the end. As they make each move, they find out whether their moves were good or bad. Eventually, they find their way through the maze. The intelligent agent takes a slightly different approach in that is more programmatically methodical. The agent picks a random cell and places themselves there.

The learning loop starts, and the agent begins to take actions through the maze. For each action, a number is generated at random. If that number is less than the epsilon value, then a random action is selected from the valid actions and the loop continues. If the number is greater than the epsilon value, then the next action is selected based on the previous reward history. The A screenshot of a computer

Description automatically generatedaction with the highest reward is selected.

Text

Description automatically generated Finally, the agent records the reward results of its action into its experience history and evaluates whether the agent has won or lost. This whole process is looped until the agent’s win rate is greater than the epsilon value.

Both the computer and human approaches start at an arbitrary spot in the maze. They both look around at possible directions to proceed and try to move in the direction that will bring them closer to the treasure. The difference is that the computer’s approach is much more methodical while the human is going off of intuition. The decisions they make with be different because of this.

There are two main types of decisions that can be made by the intelligent agent. The first is an explorative action. Exploration is when an agent takes an action that may not provide significant short-term value but will provide future value. The opposite of this is exploitation. Exploitative actions offer more short-term value in exchange for less value over a longer period of time. Finding the balance between these types of actions will allow for optimal learning. (Lamba 2018) The Exploitative actions make sure we make a good action next, and explorative actions help to keep making good decisions about the next actions to take.

For the agent created in this design, the epsilon value that controls the balance between the two decision types was set to 0.1. This means that there was a one in ten chance that and explorative action would be take next. If there is a small epsilon value, less explorative actions will be taken. This could negatively impact the learning. If there is a large epsilon value, more explorative action will be taken. Since the explorative actions are randomly selected from the list of valid actions, there could be too much random action and not enough action based on experience. The 0.1 value for epsilon offers a good starting point to balance to two types of actions.

Reinforcement learning is the type of learning when good or correct actions are rewarded. Punishment can also be given for bad or incorrect actions. By doing this, we can train a computer model to do certain things. (Yoon 2019) For this situation, we wanted the intelligent agent to make it through the maze. To determine the path to the treasure, we could use a point system that rewards the agent as it takes steps that lead it closer to the treasure. The points rewarded for each action would be stored and referred to when the next decision must be made in that position.

For the pirate agent in this design, a deep Q-learning neural network was implemented. First, the maze and possible actions were all defined. Then, a start point was selected at random and the agent in put into that cell. The agent calculates a potential reward for each of the actions that it can take, and the highest scoring action is taken. Once the action is made, the process repeats until the agent reaches the end of the maze. By doing this, we create what is called a Q-table map. (Lamba 2018) This process is looped until the intelligent agent created had a satisfactory win rate. Each iteration of the main loop, where the agent travels through the maze. is has its history recorded. This is done so that the next time the agent must pick an action, it can refer to the rewards it was granted previously. By iterating through the maze and refining the Q-table map, the agent will being to take a more consistent route as the only time it will deviate from the optimal route is when an explorative action is generated. The ability for these algorithms to increase their efficiency by learning from their previous history is what makes them useful tools in the field of computer science.

References:

Lamba, A. (2018, September 3). *An introduction to Q-learning: Reinforcement learning*. Medium. Retrieved October 13, 2022, from https://medium.com/free-code-camp/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc

Yoon, C. (2019, July 17). *Understanding actor critic methods*. Medium. Retrieved October 13, 2022, from https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f