Briana Long

10/21/2025

CS 370

Project Two

A human being's approach to solving a maze would be to first look at the overall goal. There would be an orientation for a starting point. Next, the goal post would be located, which is to find the treasure icon. A human would keep notice of the walls, paths, and dead ends of the maze. Then a plan would be formed on what paths to follow and look out for. The human being would then set to complete the task with their working memory. Dead ends would require backtracking and remembering visited cells in the maze. Then, as the human continues, it would refine its choices due to the mental map created until the goal is found.

An intelligent agent first starts with state representation to encode the maze state. It then sets up actions, this is the movements of up, down, left, and right. It then sets a policy for its actions. It can choose to explore by picking a random move or exploit by using a value action. Next, the agent applies the state and rewards with movements. A step is typically negative, while finding the treasure is positive. The experience is then stored and learned from in batches. This helps minimize errors and decreases run times. The learning is then stabilized through a target network. Over exploration episodes, the behavior becomes more greedy as opposed to exploratory. The agent’s Q-values encode a good path that it uses to find the treasure.

Human and intelligent agents can have similar approaches to finding the treasure in a maze. Both human and intelligent agents proceed incrementally. They react to feedback to

improve their actions. And they both avoid repeating mistakes. They both start with exploration and become more goal-oriented as the maze is iterated.

Human and intelligent agents tend to have differences when approaching a maze to complete the goal of getting the treasure. Humans plan with reason, and agents do not and learn to plan. Humans do not need as many trials to complete the mazes compared to an agent that requires many episodes.

The purpose of an intelligent agent in pathfinding is to autonomously discover a policy. That is mapping the states to actions to maximize the cumulative reward, such as finding the treasure without hard-coded rules or supervision during each step. Instead, it uses interaction to observe states, try actions, and learn from outcomes.

Exploration is the act of trying actions to gather information. This is done to prevent a suboptimal routine from being used consecutively in runs. It can take longer using this method to reach the treasure in runs, but it creates the best runs. Exploitation is when the best estimated action is used to maximize the reward now.

Reinforcement learning helps determine the path to the goal for the agent. This is done by assigning values to actions by crediting the steps that led to the treasure. Each transition gives a reward signal for the agent. The agent then updates its current Q-value to the reward and discounted value of the next state. As the state becomes closer to the return, there are higher Q-values. This creates a gradient policy to follow. The replay buffer and target network are used to help the agent learn the values in a stable manner from past experiences.

Implementing the Deep Q-Learning using neural networks for this game starts with assigning states, actions, and rewards. The state is the agent's position in the row and column.

The actions are up, down, left, and right. This prevents illegal moves such as diagonals or jumps. Then there is a set of rewards. For reaching the treasure, there is a +1.0, and a -0.01 per step to encourage shorter paths to find the treasure.

The network is a Q-function that states the input, architecture, loss, and optimization. The input is the state vector. The architecture is dense linear with outputs of Q. The loss is the mean squared TD error between the Q and target Y. An experience replay is introduced to create a replay buffer. This created a capacity in the stores. The mini-batch is set to sample each episode. Target networks update every C steps to stabilize the targets. The greedy policy creates a decay.

The training loop is started for each episode. The environment starts with a rest and the command to get to the treasure. Each step is chosen with a random action that is observed and stored in the replay buffer. The buffer is then sampled for compute targets of done or else. If not, then steps are continued and a decay is created to push the algorithm to find the treasure.