Michael Lorenz

CS-370 Emerging Trends in Computer Science

Dr. Habibi

20 October 2024

Design Defense Report

**Analyze the differences between human and machine approaches to solving problems.**

Human Maze Solving

The approach a human would take to solve a top-down maze, such as in this maze application, involves a reliance on visual information from the eye and hand-eye coordination to solve the maze. First, a human identifies the starting position and end position. Then, according to a study on maze solving by Rutgers University, the steps humans use to solve a maze involve a balance of two separate hand-eye coordination strategies: “exploration” and “guidance.” Assuming the human solves the maze using a cursor, exploration is when a human is “scanning ahead with the eye for more information,” to find the correct path while the cursor is stationary, relying on memory to store the correct path (Zhao and Marquez, 2013). Guidance uses eye movement saccading along the path while leading the mouse, which uses less internal memory and “strong reliance on immediate visual cues” (Zhao and Marquez, 2013). Humans use a varying mix of both until reaching the target with the cursor.

Intelligent Agent Maze Solving

The intelligent agent uses a specific approach to solving the path-finding problem for the maze using an algorithm that can be described as follows:

1. The agent is given a neural network model and starts at a randomly chosen cell in a new *TreasureMaze* environment with initialized states and properties.
2. The agent obtains initialized state and environment properties with *observe()*.
3. The agent decides to explore a random action or exploit learned knowledge depending on the *epsilon* value. If exploring, it randomly selects a direction to move. If exploiting, it uses the model to predict Q-values for possible actions and selects the action with the highest Q-value.
4. The agent obtains a new state, reward, and game status by taking the action with *act()*. This episode is stored in memory with *experience.remember()*
5. The agent samples a batch of experiences with experience replay and trains the model using the *fit()* method to improve Q-value predictions in different environments. Rewards in each experience determine which actions lead to the best outcome, improving Q-value accuracy.
6. The agent repeats steps 3-5 until the game ends (maze is solved or agent reached max moves)
7. The agent evaluates its win history after a game is won or lost and reduces the exploration rate to 5% if the win rate is above 90%
8. The agent repeats steps 1-7 until achieving a 100% win rate over a specific amount of games and passes the completion\_check() method (meaning the agent can win from any cell), indicating the agent has solved the problem.

Similarities and Differences

Both approaches use a combination of exploration and exploitation-type strategies, relying on sensors and memory. A human and an agent both gather information about the environment by scanning the maze state and then using exploration to discover a path or using learned information in exploitation to select the best action to move toward the target. However, they differ in how information is obtained and how choices are made. Humans rely on visual sensory information to perceive the maze, possible pathways, and the end goal, while an agent relies on a numerical representation of the state, possible actions, and previous states to determine moves. While both approaches “optimize” between exploration and exploitation, a human will optimize between using more time and memory load to solve with fewer errors or using guidance to solve with less time and memory load, relying on immediate information to solve faster with more errors and back-tracking (Zhao and Marquez, 2013). The agent optimizes using previously stored experiences, a reward system, and many epochs of maze-solving to predict actions and improve predictions or explore at a specified rate by choosing paths randomly. A human uses prior problem-solving intuition, sensory perception and coordination, and reasoning to solve a maze much more rapidly (in 1 game), while the agent uses rewards, predicted Q values, and many iterations of refinement to solve the problem without prior intuition.

**Assess the purpose of the intelligent agent in pathfinding.**

Exploration and Exploitation

The agent uses both exploitation and exploration in pathfinding. In my implementation, the agent uses an epsilon value to determine the chance that it chooses exploration to follow a path randomly. If the agent chooses exploration, it randomly selects one of the four possible directions to try, encouraging learning with a new path. In exploitation, the agent uses learned knowledge by calculating the Q-value for each possible path and selecting the one with the best-predicted outcome, reinforcing paths that lead to better outcomes. The below table shows the data I collected to determine the best proportion of exploitation and exploration.

| epsilon | Best Epochs | Average Epochs to 100% Win Rate | Variation (how much epoch counts differed across runs) |
| --- | --- | --- | --- |
| 0.1 | 198 | 243 | Medium |
| 0.25 | 100 | 123 | Low |
| 0.325 | 110 | 156 | Medium-High |
| 0.4 | 52 | 154 | High |

The best epsilon value is around .25; this rate had the least average number of epochs to solve the problem with little variation. Although I had an epoch count as low as 52 for 0.4, other trials varied wildly, leading to inconsistent results.

Reinforcement Learning

Reinforcement learning helps my agent determine the best path to the treasure by learning from rewards and penalties. The only reward is +1 when reaching the treasure, encouraging the agent to seek the treasure. Each valid move incurs a -0.04 penalty, discouraging wondering, and visiting previously visited cells results in a higher -0.25 penalty to further discourage pointless wondering. A penalty of -0.75 occurs when trying to move to maze walls. Using this reward structure helps the agent avoid repetitive or incorrect actions while navigating to the treasure. Using rewards in deep Q-learning helps optimize the path to the treasure and maximize cumulative rewards

**Evaluate the use of algorithms to solve complex problems.**

Deep Q-Learning

The pathfinding agent is an implementation of a deep-Q learning algorithm, which utilizes a neural network to optimize the Q-value function, helping the agent predict the best action to take in each state. This algorithm utilized a Sequential neural network model, consisting of an input layer that processes all maze cells as the state using neurons and several dense hidden layers activated with the PReLU function. The final output layer uses 4 neurons to represent each action, outputting the Q-values for each possible action, determined by the maximum predicted cumulative reward. As the agent uses experience replay to store game experiences, the neural network is optimized using previous experiences and reinforcement learning, such as stored action and reward pairs, to learn an optimal policy for reaching the treasure. This optimizes the agent's ability to predict Q-values of moves and find the shortest path to the goal while maximizing the reward.

**References**

*GameExperience.py*. (2020). Southern New Hampshire University.

Lorenz, M. (2024) *Lorenz\_Michael\_ProjectTwoMilestone.ipynb.* Southern New Hampshire University.

*TreasureMaze.py*. (2020). Southern New Hampshire University.

Zhao, M., & Márquez, A.G. (2013). Understanding Humans' Strategies in Maze Solving. Rutgers University. ArXiv, <https://arxiv.org/abs/1307.5713>