Design Defense

Hunter Marx

6/23/2025

CS-370

When it comes to solving a maze, there are both similarities and differences to how humans and AI agents solve a maze to find a treasure.

Humans use inductive reasoning based on past knowledge of mazes to come up with possible paths. They then use trial and error and memory in combination with luck to eventually and hopefully solve the maze. Humans use intuition and pattern recognition, which is all inductive rather than explicitly known to humans.

The AI agent has a more streamlined approach. It use a combination of exploration and exploitation to systematically explore states and update q values. This eventually leads to knowledge of the correct path. The agent updates Q values and conducts mathematical optimization cyclically over time to solve the problem.

Both humans and AI agents are motivated by rewards. Humans are motivated by mental rewards, or at a lower level of abstraction, we are motivated by neurotransmitters which are released during and after we achieve a goal, which in this case happens to be solving the maze and finding the treasure. An AI agent is motivated by its utility function outlined in its program. This type of programming causes reliable reward seeking in the direction of said utility.

Exploitation defines the tendency to employ current knowledge to choose the best path;

*Exploitation np.argmax(mode.predict(state);*

While exploration defines the random chance to try new paths; *(random.choice(qmaze.valid\_actions());*

A balance of these programmed behaviors is key, and has to be fine tuned in a given problem to optimize efficient problem solving. The likelihood of my agent choosing exploration over exploitation is defined as epsilon in my code. I used the strategy of starting with a high rate of exploration with epsilon 0.1, instructing my agent to choose random moves 10% time. This rate of random moves was kept the same until a win rate of 90% was achieved, at which point epsilon was reduced to 0.05 (5% random moves).

Reinforcment learning was also an important strategy when designing the algorithm for this agent. The utility function of the agent in reinforcement learning is always to optimize reward while avoiding penalty. My agent received +1.0 points for finding treasure and winning, -0.75 points for running into walls, and –0.04 points per step to motivate it to take the shortest path possible. The agent stored these number of points as experiences, with the most recent one informing the maze state;

*GameExperience.remember());*

The agent used batches of moves for training to learn over time (experience.get\_data()).

The algorithm for my agent used a specific type of reinforcement learning called deep q learning. Deep is in reference to the use of a neural network. This contains an input layer (the maze state), hidden layers defined in build\_model, and an output layer (four numerical q values assigned to each direction left/right/up/down). The training steps of the agent consists of:

1. Play

The pirate explores the maze and stores moves in *GameExperience*.

1. Replay

Model selects fifty past moves (*data\_size = 50*) as a batch for training data.

1. Update

Agent adjusts Q values using the Bellman equation *GameExperience.get\_data()*

Overall, my algorithm was a robust implementation of reinforcement learning, particularly q learning. The agent is able to eventually find the path on its own using this strategy. The only downside is that this model requires many epochs/runs to be successful.