Victor Feight

CS-370

Design Defense

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

Describe the steps a human being would take to solve this maze.

I can imagine a human being taking a depth first approach, with little regard as to penalties for moving to adjecent cells. That is, one might take a long hallway til the end, then backtrack, and go down the next cooridor, with no regard for movement penalties.

Describe the steps your intelligent agent is taking to solve this pathfinding problem.

In the environment, TreasureMaze.py, there is a maze object defined as a matrix. The agent can move in one of four directions—left, right, up, down. The agent receives a reward with positive or negative points (between -1 to 1) for every movement. The highest reward being granted when the agent reaches the treasure; if the treasure hits an occupied cell or attempt to go outside the maze boundary, it will incur the highest penalty, and a smaller penalty will be applied if the agent revisits a cell. To prevent the agent from wandering, add a minimal penalty.

What are the similarities and differences between these two approaches?

The human being does not rationally think of the greatest discounted reward at every step, nor does he think of penalties. He’s naturally drawn to cooridors without thinking of wandering outside the maze and might revisit cells several times haphazardly. An AI is optimizing his movement in every turn for the greatest discounted reward possible, thus they aim to win fast and learn from mistakes at every moment.

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

What is the difference between exploitation and exploration? What is the ideal proportion of exploitation and exploration for this pathfinding problem? Explain your reasoning.

In GameExperience.py, all the states between initial state and terminal state are stored so the agent can later learn by exploitation (experience). This agent learns primarily through exploitation (experience), but it can choose to explore the environment and discover new paths (exploration). 15000 epochs with an exploration factor of 0.1 was required to train the network to a 100% win rate, any higher exploration factor and the time taken would have dramatically increased, any lower epochs and we might not have achieved 100%. This is because 0.1 represents for every 10 attempts, the agent will attempt to learn by experience 9 times and explore (learn a new path) 1 time, this resulted in an optimized algorithm.

How can reinforcement learning help to determine the path to the goal (the treasure) by the agent (the pirate)?

With epsilon, the exploration factor, we use a policy that takes a maze snapshot (envstate), as input (consisting of full cell maze), and returns the action to be taken by the agent. Upon starting, the agent uses a random policy, after thousands of games our reward policy provides feedback on how the agent should improve itself. We enhance the learning by varying exploitation with exploration, this is based on Markov Decision Processes. After each action and each state, the agent collects a reward or penalty, so the more the agent wanders and wastes time, the less reward. A similarly implemented rat solving maze states: “The agent's goal is to collect the maximal total reward during a "game". The greedy policy of choosing the action that yields the highest immediate reward at state s, may not lead to the best possible total reward as it may happend that after one "lucky" strike all the subsequent moves will yield poor rewards or even penalties” (Deep reinforcement learning for maze solving).

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

How did you implement deep Q-learning using neural networks for this game?

As the goal of the Deep-Q learning implementation is to find the best possible navigation sequence that results in reaching the treasure cell, while maximizing the reward – the Deep-Q learning network was given, but the Q-training algorithm needed to be created and implemented.

First, I had to determine an optimal number of epochs to help achieve 100% win rate, and run all the cells with Tensorflow backend to train the network. The final cell shows a test model for a game – we can see the map the pirate takes through the maze, avoiding the path obstacles.

Specifically, the agent selects a free cell at random to start 🡪 epsilon-greedy value determines the probability that an agent should select the random action, rather than an action to maximize expected utility 🡪 agent takes the action, storing the episode, reward, and game status 🡪 if the pirate has won that game, update the win rate, else update the loss, finish when win\_rate is greater than epsilon.

Deep reinforcement learning for maze solving. qmaze. (n.d.). Retrieved December 10, 2021, from https://www.samyzaf.com/ML/rl/qmaze.html.