July 2, 2023

I’m going over the Reinforcement learning documentation from Geeksforgeeks and going over the syntax when setting up a network diagram using pylab, numpy, and network. Also using Q-learning to find the quickest path from one state to another. So far I’ve documented how they set up the m-table (immediate reward) using all points and edges from the matrix (edges define the immediate reward). Within the main matrix take out a row of a certain state then find the correlating action based on the index. This is used to find the value given the selected action:

```
max_value = Q[action, max_index]                                    #
Q[current_state, action] = M[current_state, action] + gamma * max_value      #
```

The entire code runs for selecting state and max-action runs until it is told to stop. Afterwards it returns the path with the best q-value. The structure is very similar to the adjacency matrix. Previously, I mentioned how I was able to develop an environment for the agent, making the robot with a local camera from scratch and adding object detection. Chance had pointed out there was a limitation because my method involves using the recognition color, which only detects objects with the same calling. It doesn’t include distance tracking. He recommended I used the faris robot because it includes the RGB camera which enables distance detection based on black and white lighting. Also the world is based on teaching other students how to set up Reinforcement learning. Even though I have the environment set up, I still don’t know how to code RL. I'm planning to go over the Faris code and its own Webots files to find a way to setup an RL syntax code for my own project. Also going over previous examples from “geeks for geeks” because of the comments in syntax.and the research project you sent me.