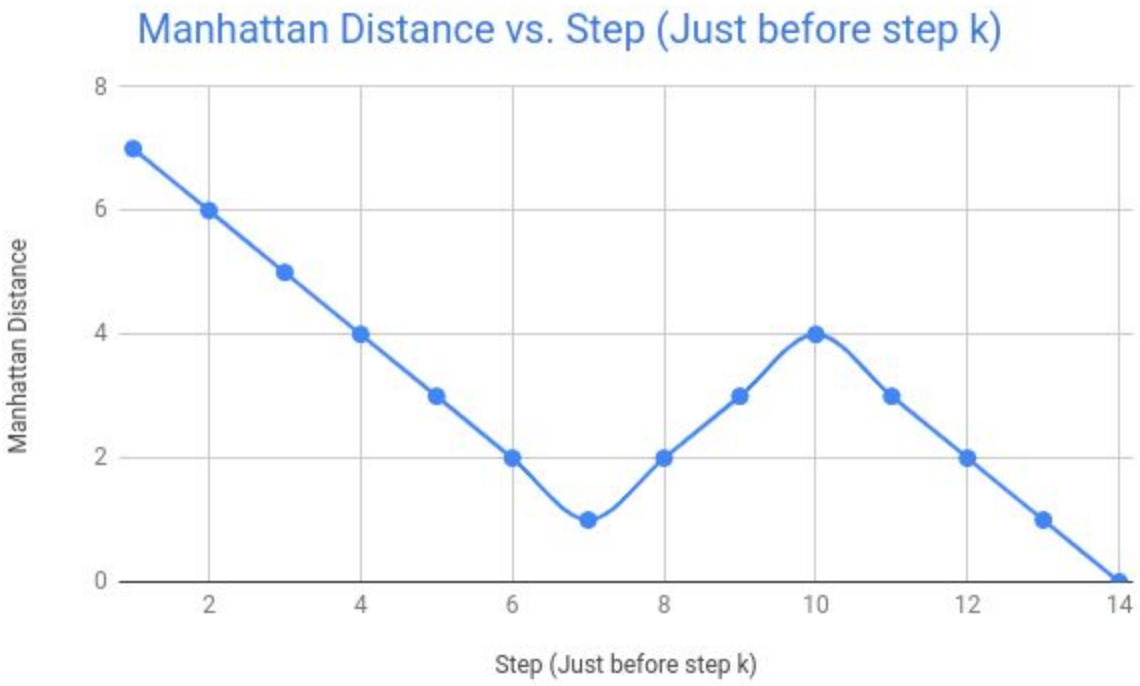
AI, Assignment 4
Katari Saketh
2015045

Analysis: link

3.a. Performance of Q-Learning:
At each position of an agent in the maze, Manhattan distance is calculated, which decreases as shown in the below graph -



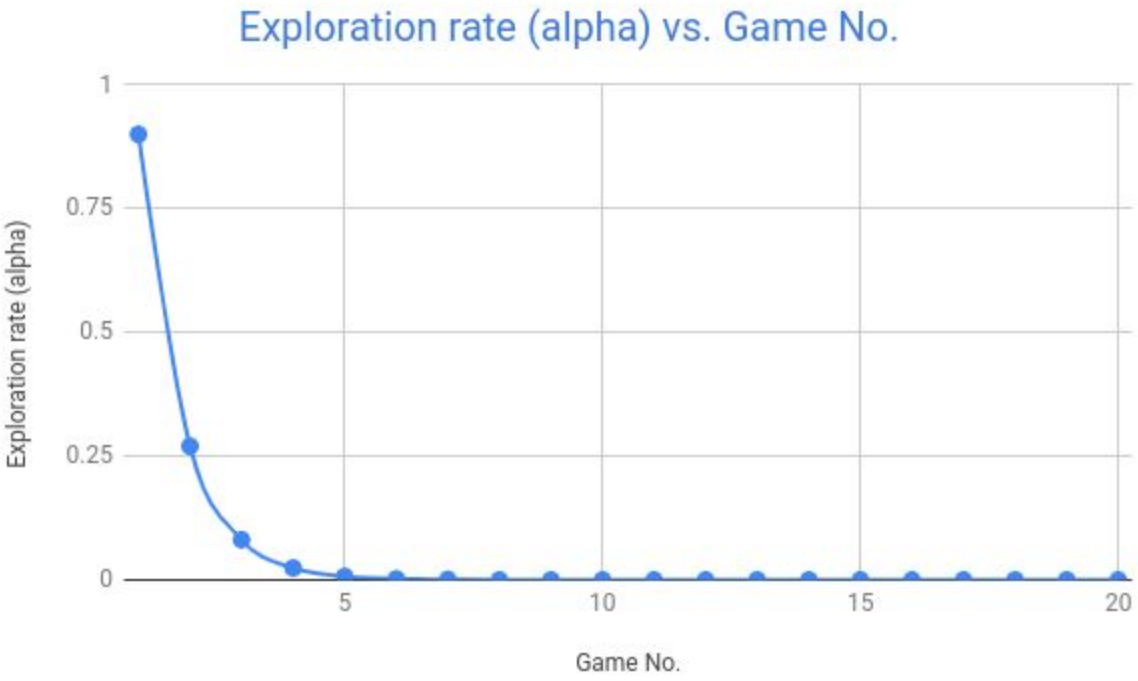
During the training phase, the total steps taken in each game is computed; after some no. of games, the model reaches the saturation point (shortest path), as shown below -

3.b. and 3.d. Exploration Rate (α):

Start with high exploration rate and slowly decrease (at the start of the game) the exploring rate, like 70% after every game.

Why? In the first game, agent has 0 learning i.e empty q-table; At the start of training, low exploration rate (0.1), which is nothing but fully reliable on q table, might make agent to get stuck at few states.

In first game, let the agent learn (updating q-table) by exploring the maze, with high exploration rate. Now after some learning, let the agent use his learning (updated q-table) and decide how to reach the goal state, with low exploration rate.



Discount Factor (γ):

High discount factor, like 0.9 is better. Small discount factor might make an agent not to reach the goal state, even if you have a very nice q-table. In other words, small discount factor might make an agent to not take advantage of the learning made in terms of q-table entries. Small

gamma, say 0, indicates you are just considering immediate reward (near future) and not the far future.

High gamma indicates, more importance to the far future (near future = immediate reward; far future = max q value of next state). Very high discount factor might make the immediate reward to have less impact on q value; ultimately, the long-term rewards matter the most, so, high discount factor, like 0.9 can be used.

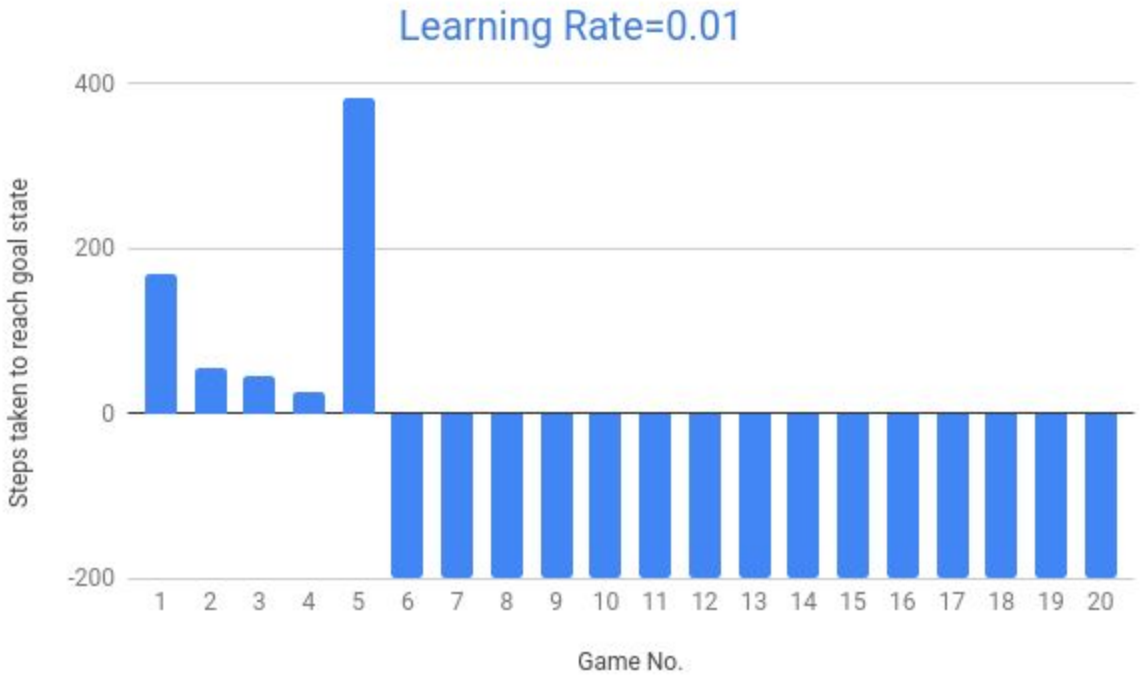
In the below chart, you can see that low discount factor, 0.1, didn't make use of nicely made q-table, due to which, it only considers immediate reward, thus, sometimes not reaching the goal state - the ones with negative no. of steps.

Learning Rate:

Higher the learning rate, better the model. High learning rate would update the q values quickly. Very less learning rate might not recognise the tiny changes in q values, which might lead to not reaching the goal state.

3.c. Equi-probable actions:

The analysis we did above (performance, hyperparameters), already has the property of choosing any action with equal probability as other actions.



The above graph shows how less learning rate (0.01) ignored the tiny changes in the q values while updating it, thus, not reaching the goal state (represented as -200, instead of 0 steps) most of the time.

3.e. Analysing any 2 states:

Best parameters: exploration rate=0.9, decreasing by 70% at the end of each game, discount factor=0.9, learning rate=0.9.

State 2|2 has been analyzed as shown in the spreadsheet. It took 12 iterations to converge to the best action. While converging, the model was not sure which action to select between action 1 and action 3. After the convergence, q table strongly suggests the model to select action 1.

State 3|3 is analyzed, which took 8 iterations to converge to the best action. While converging, the model was not sure which action to choose from action 0, 1, 2 and 3. After the convergence, q-table strongly suggests to choose action 0.

Observation: In both states, it can be seen that the last action selected in each game is always the optimal action.

4. New 5x5 maze (with different obstacles etc.):

It no more gives an optimal path (14-step path) to the goal state; sometimes it takes 100s of steps to reach the goal state and sometimes it takes more than 2000 steps to reach the goal state.

Why? Q-table is specific to a particular maze i.e every different maze has a different optimal q-table. Even though the size of maze is the same i.e 5x5, the properties are different like the placement of obstacles in the maze etc. This changes the optimal/shortest path to the goal state. Note that the Q-table we built during the training is for a different maze (obstacles are placed differently); So, the old q-table is not useful for a different maze, as the shortest path is now different for the new maze.

Problem: The q-table obtained from the training phase is not optimal for the new maze that we've created during our test phase.

Solution: Re-train the model on the new 5x5 maze and produce an optimal q-table corresponding to the new maze.