

Q-Learning Algorithm for Maze Traversal

by David Allen 

Q-Learning Implementation

My Q-Learning algorithm has very little information about its environment. The Q-table is initialized to 0 in every cell, meaning that the model learns about the maze solely through the rewards it receives. It frequently tries to move into walls, step out of bounds, or retrace its steps as it updates the Q-table. The algorithm works by first choosing an action. With probability $1-\epsilon$ the model will choose the best action given information of the table; otherwise, it will select randomly. If there is no single best action to take (meaning there is a tie), the model selects randomly among the actions with the maximum Q-value. The model is not allowed to cross walls or step out of bounds. The algorithm continues selecting actions until it finds an exit. Each episode, the model retains the Q-table it generated in the previous episode. After running for multiple episodes, the model generally improves in its ability to navigate the maze.

Analysis

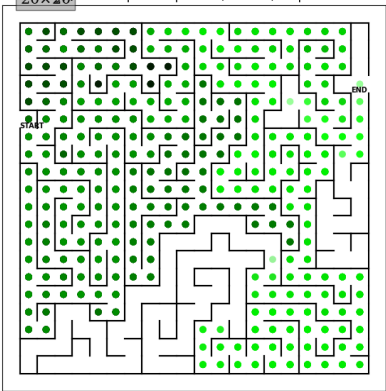
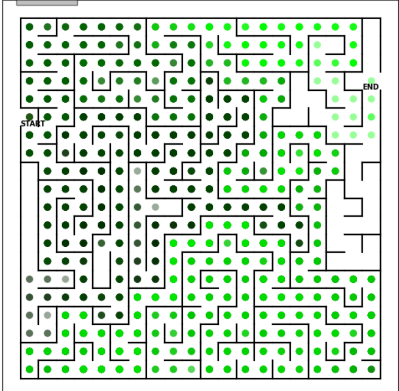
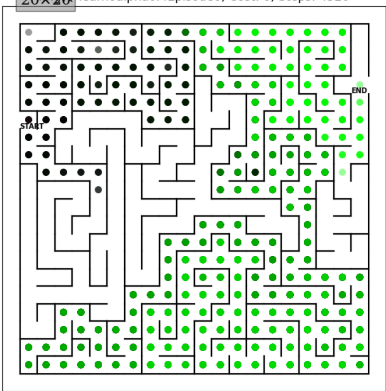
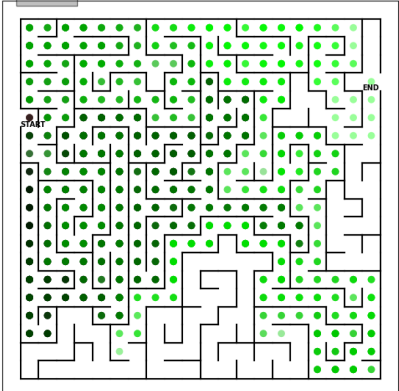


The algorithm usually found the exit within a maximum of 10000 steps. Most of the early episodes are spent learning about the environment, such as where walls, dead-ends, and paths to the exit are located. As the model runs over multiple episodes, it gets better at navigating the maze (takes fewer steps).

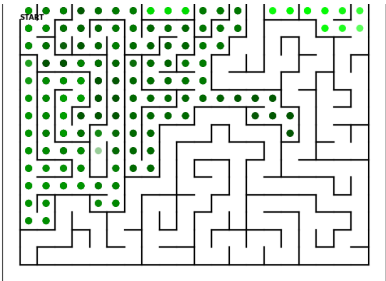
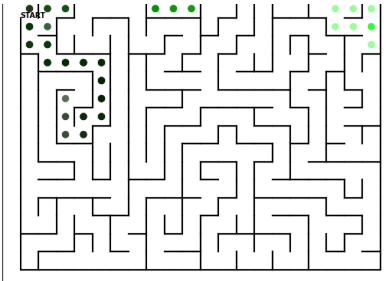
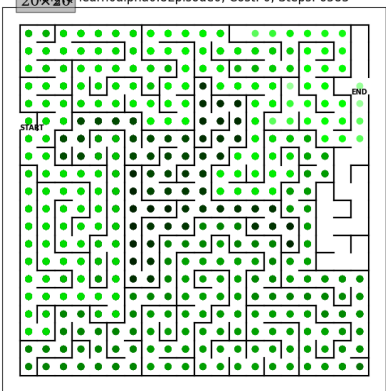
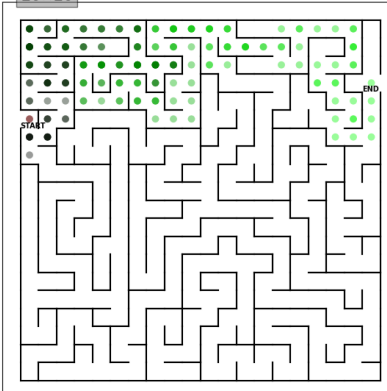
Strengths: The model does not need any initial information about the maze to find the exit. All it needs are the state, the actions, and the reward for those given parameters. The algorithm can almost always find the exit without getting stuck (whether or not it is guaranteed to find the exit is not something I can prove). The algorithm can vastly improve itself over the course of a few episodes.

Weaknesses: Due to the very little information available to the model, the algorithm often takes a long time, especially in early episodes, to find the exit and exhibits strange behaviors, such as frequently retracing its steps and staying in one area for a long period of time.

By hyper-tuning parameters, the algorithm's performance was significantly altered across multiple episodes of training.

Learning Rate (Alpha): For these tests the discount rate was held at 0.8 and exploration rate at 0.2. Through testing of the algorithm with different learning rates, it seems that in general a higher learning rate allows the model to find the exit much faster. In the figures below, when alpha was set to only 0.2, the algorithm improved from 4428 to 3313 steps in 10 episodes. However, when the learning rate was set to 0.8, the performance improved phenomenally from 6583 to only 244 steps in 10 episodes. The term "learning rate" is a fitting name for the parameter. This is likely because the model is much more greatly impacted by the reward of different states, allowing it to remember the location of walls better and clear paths as well.

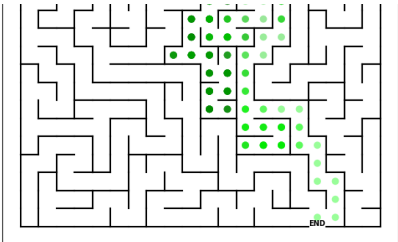
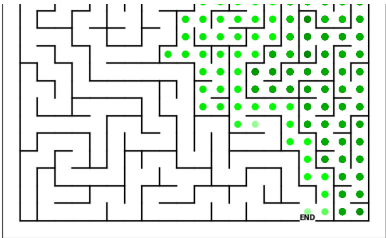
Learning Rate	Episode 0	Episode 9
$\alpha = 0.2$	<div>209200-learn0alpha0.2Episode0; Cost: 0; Steps: 4428</div> 	<div>209200-learn0alpha0.2Episode9; Cost: 0; Steps: 3313</div> 
$\alpha = 0.4$	<div>209200-learn0alpha0.4Episode0; Cost: 0; Steps: 4320</div> 	<div>209200-learn0alpha0.4Episode9; Cost: 0; Steps: 2327</div> 
	<div>209200-learn0alpha0.6Episode0; Cost: 0; Steps: 2469</div> 	<div>209200-learn0alpha0.6Episode9; Cost: 0; Steps: 456</div> 

$\alpha = 0.6$		
$\alpha = 0.8$		

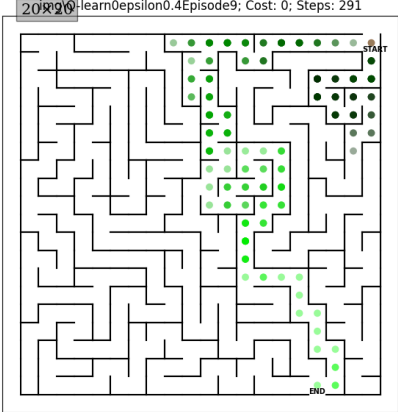
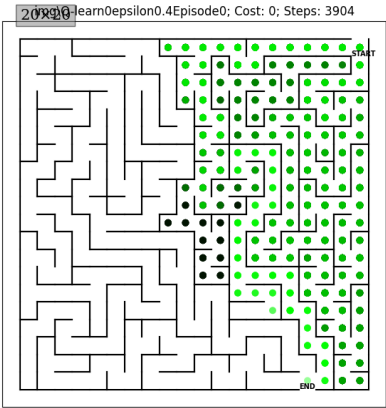
Exploration Rate (Epsilon): I chose exploration rate as my next parameter to test. For these tests the learning and discount rate were held at 0.8. The exploration rate seemed to have less of an impact than learning rate did across 10 epochs. In the figures below, increasing the exploration rate caused the number of steps in the first episode to increase, from 2959 when epsilon = 0.2 to 4425 when epsilon = 0.8. In each case, the algorithm improved over 10 episodes. However, when the exploration rate was 0.8, the algorithm took much longer to find the exit, 2823 steps. In the animation, it almost appears as if the model is trying to fight back against its own randomness. More cells are explored than in the other tests. Yet it still manages to find the exit faster than in the first episode.

Exploration Rate	Episode 0	Episode 9
		

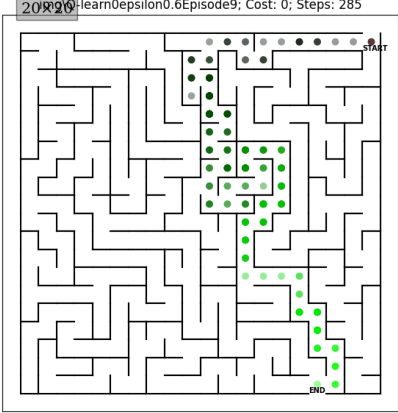
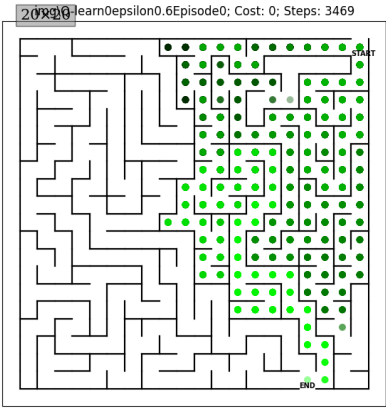
$\epsilon = 0.2$



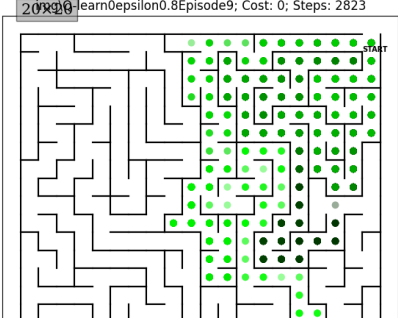
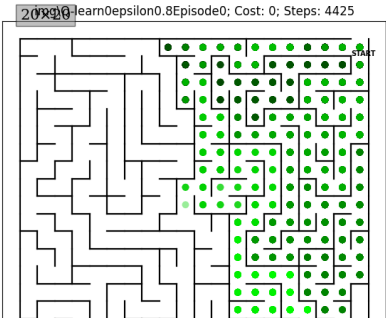
$\epsilon = 0.4$

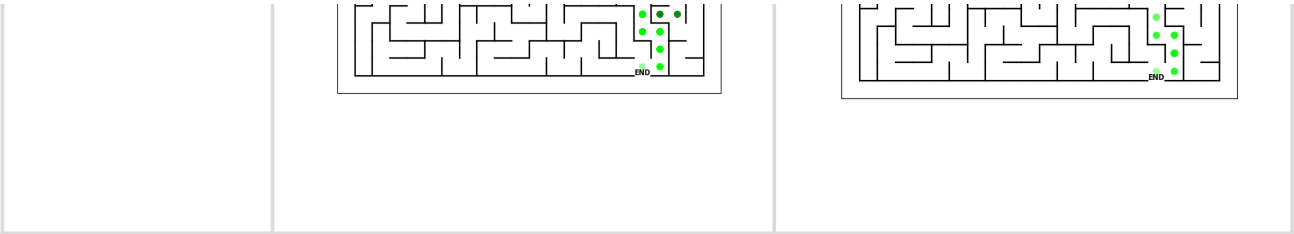


$\epsilon = 0.6$



$\epsilon = 0.8$



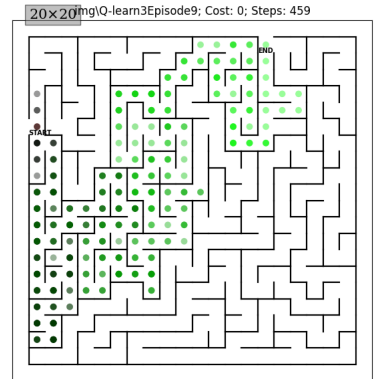
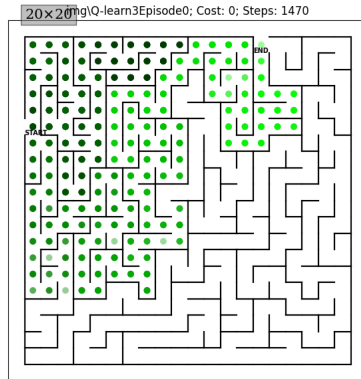
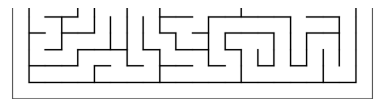
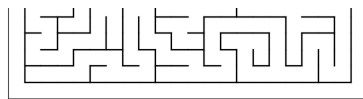


Results

Here are 3 generated mazes and the resulting path taken by the algorithm in the first and last episode among 10 episodes. Learning and discount rates are set to 0.8 while exploration rate is set to 0.2. The grey/green dots represent squares that the algorithm explored. Lighter shades of green are where the algorithm explored most recently, while darkest shaded dots are the cells that the algorithm explored earliest. At the top of each diagram the number of steps taken is shown. As you can see, while the model improved greatly over 10 episodes in the 2nd and 3rd maze, in the first maze the algorithm actually became much worse. Depending on the layout of the maze and the random actions the model takes, it seems that 10 episodes is not enough to guarantee that the model will significantly improve.



3



Conclusion

Overall the algorithm did a decent job learning the layout of the maze. It took several episodes to understand the layout of the maze, especially due to the restrictions it had of limited information and only being able to move one step at a time. Because of the amount of randomness at work in the model, a simple decision to go right or left at the beginning of a maze could seriously affect its performance, which can be seen in the examples above. Unfortunately it was never able to find the optimal path in 10 episodes. However, in general, the model was able to learn about the maze and perform better over time.