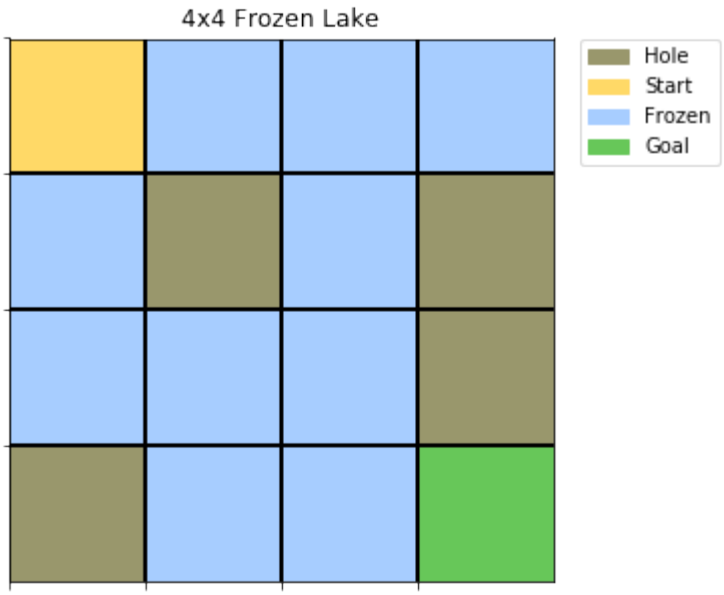
Reinforcement Learning

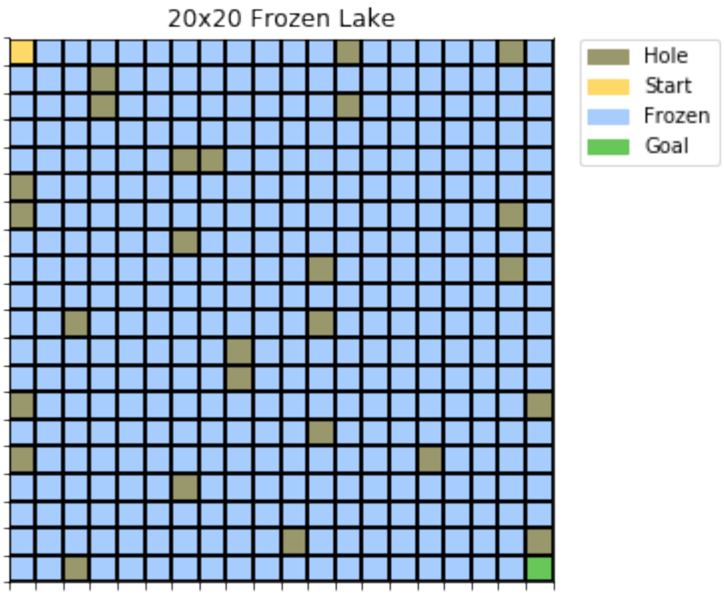
Tiera Lee

Chosen Markov Decision Processes

For this project I utilized GymAI Gym's FrozenLake environment. The agent must navigate the grid world that is the frozen lake, starting from the top left corner to its goal of the bottom right corner. The lake is not completely frozen, so there are holes that the agent must avoid falling into. While this seems straight forward, the lake is slippery. There is a 33.33% chance that the agent will go in its intended direction, and a 33.33% chance that it will slip and go in either perpendicular direction. When the agent reaches the goal state it will receive 100 points. There is a -1 point penalty for moving, which will encourage the agent to move in the shortest route possible. There is also a -10 point penalty for falling into a hole, which will encourage the agent to avoid these pitfalls.

The first lake is a 4x4 grid, which creates 16 possible states. Given the reward system and non-deterministic properties of the agent's actions, this will be interesting to how conservative or daring the agent chooses an optimal policy. The second lake is a 20x20 grid, creating 200 states, which is 25 times larger than the smaller lake. There are also more holes for the agent to fall into, which may cause the agent to be more conservative. A larger state space will be interesting experiment with in regards to exploration vs. exploitation during q-learning.



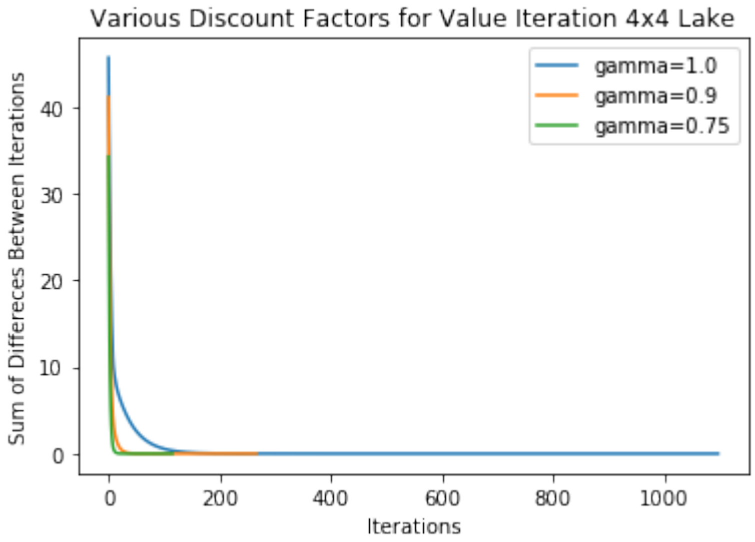


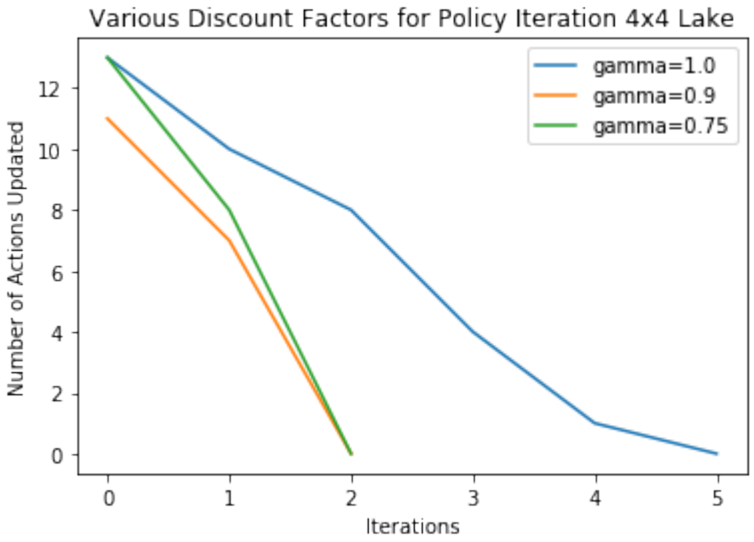
Solving MDPs with value iterations and policy iterations

Value iteration and policy iteration are both methods for offline planning, which assumes the agent has prior knowledge about the affect on the environment. Value iteration computes the optimal state value function by improving the utility for each state until convergence. For this project, the value table is considered converged when the difference between the sum of the absolute value between the current value table and the previous value table is less than 1e-20. Policy iteration instead iteratively calculates a new policy directly until convergence. Convergence in this case means that the currently policy names no more changes to the previous policy.

In addition to comparing the recommended policies between the two methods, I experimented with varying discount factors: 1.0, 0.9, 0.75. The discount factor prevents the total reward from going to infinity and gives the model preference for determining immediate vs. future rewards. A discount factor of 1.0 tells the agent to only consider long-term rewards and disregard its immediate reward. A discount factor of less than 1.0 gives preference to immediate rewards.

4x4 Lake



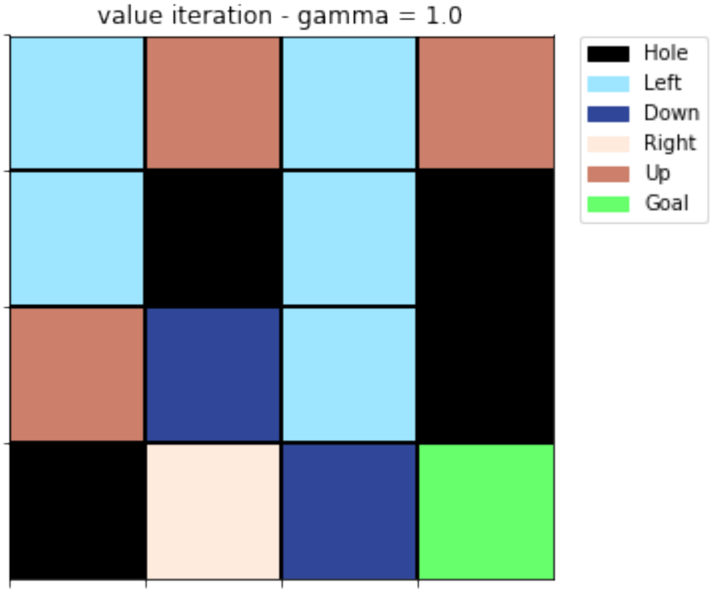


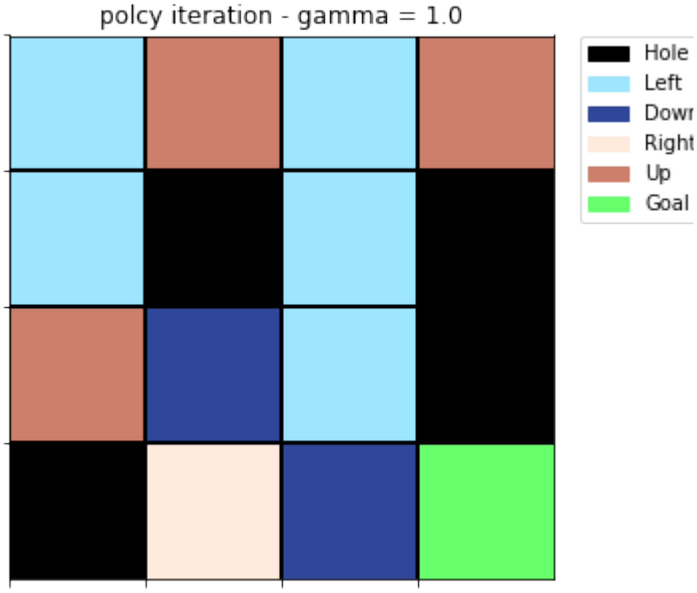
It is clear to see that policy iteration converges much more quickly than value iteration does. The fastest converging value iteration test, with a discount factor of 0.75, took 116 iterations to converge, while the slowest converging policy iteration test, with a discount factor of 1.0, took five iterations to converge. This is to be expected due to the terminating factor of each algorithm. As stated above, value iteration will continue until the changes in the state values fall below some threshold, and then that final value table is converted to a policy. Policy iteration, while more computationally expensive, cuts out the middleman of calculating utility values of each state.

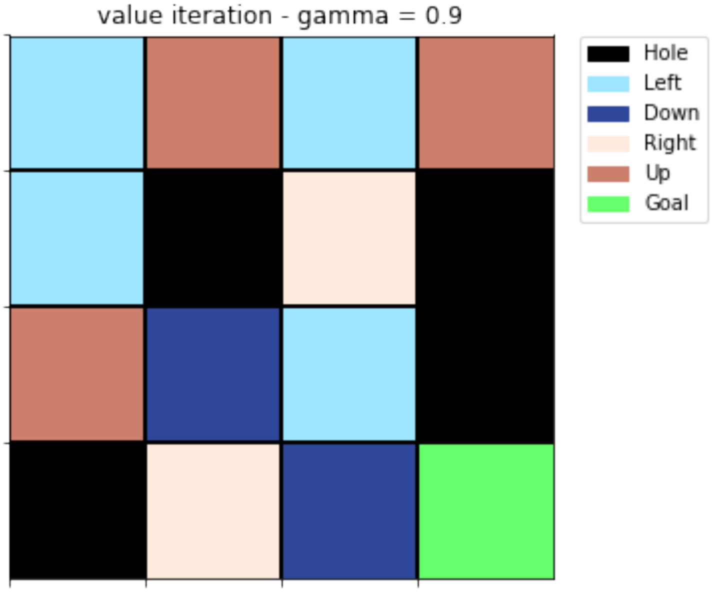
Changing the discount factor in value iteration greatly affects how many iterations it takes to converge. A discount factor of 1.0, which favor long-term rewards over immediate values, takes 1097 iterations to converge. This is because by heavily favoring long-term rewards its not learning from immediate values rewards. Just by dropping the discount factor to 0.9, it takes 267 iterations for converges, a drastic increase in performance. Making the discount factor 0.75, giving even more preference to immediate rewards, only takes 116 iteration to converge. The shape of the curve also implies this immediate reward preference. Within the first 25 iterations, the sum of differences drops to almost zero.

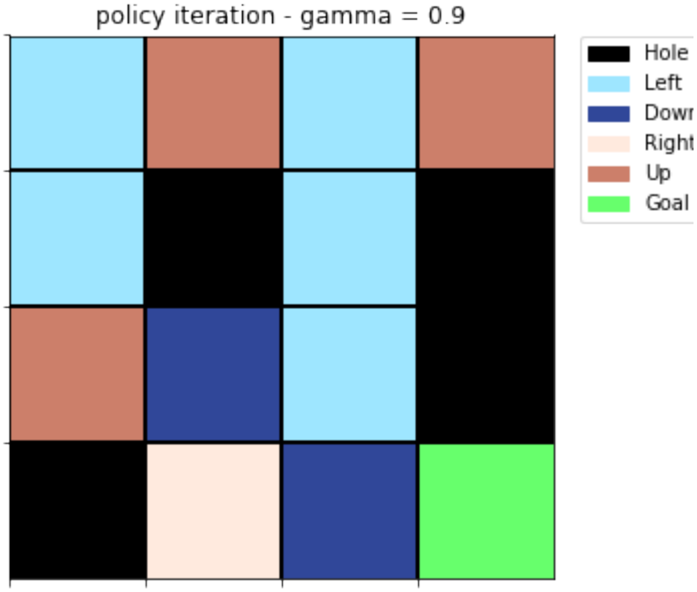
Similar behavior can be seen in policy iteration graph. A discount factor of 1.0 takes the longest to converge because it's not learning from immediate rewards. The initial policy is randomly selected, and despite this fact a discount factor of 0.9 and 0.75 converged at two iterations. This is directly attributed to the preference towards immediate values.

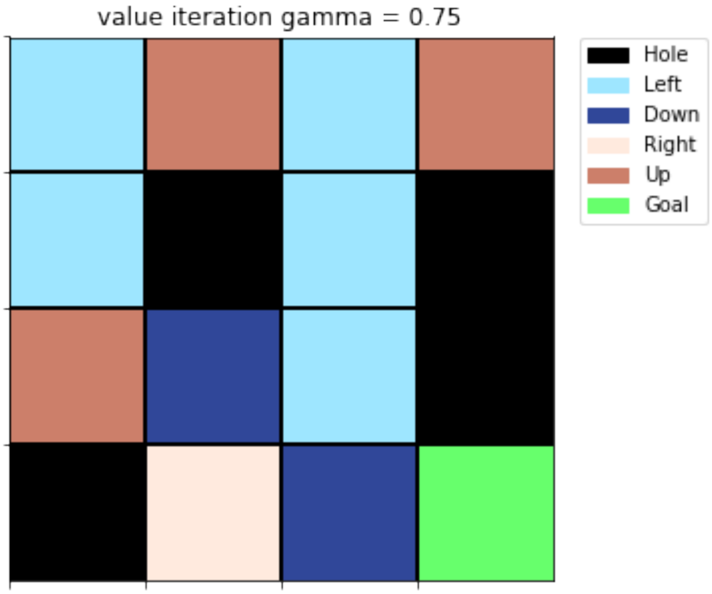
Now to examine the resulting polices between these experiments.

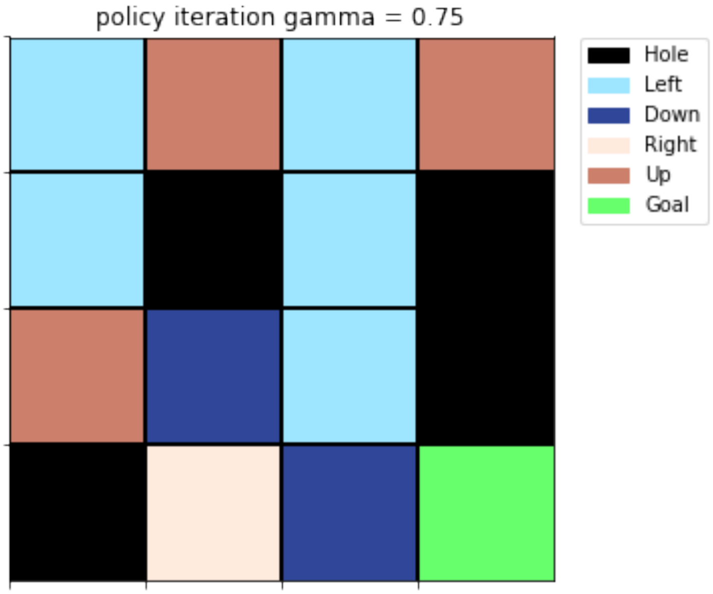






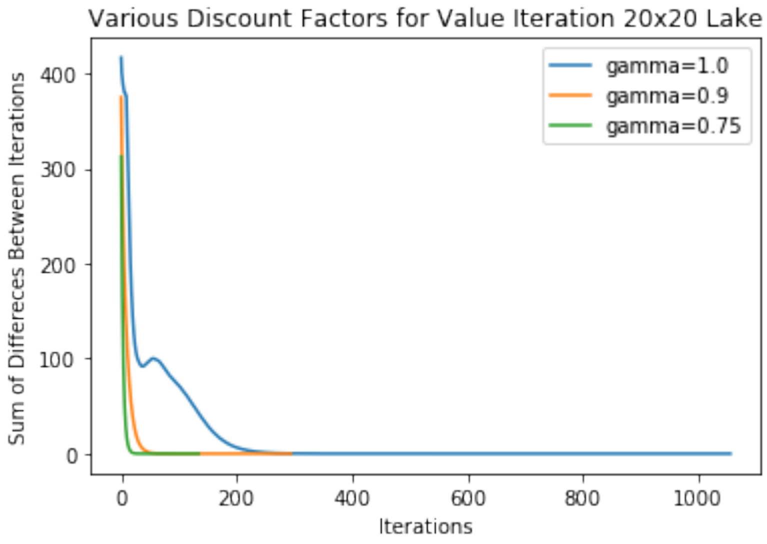


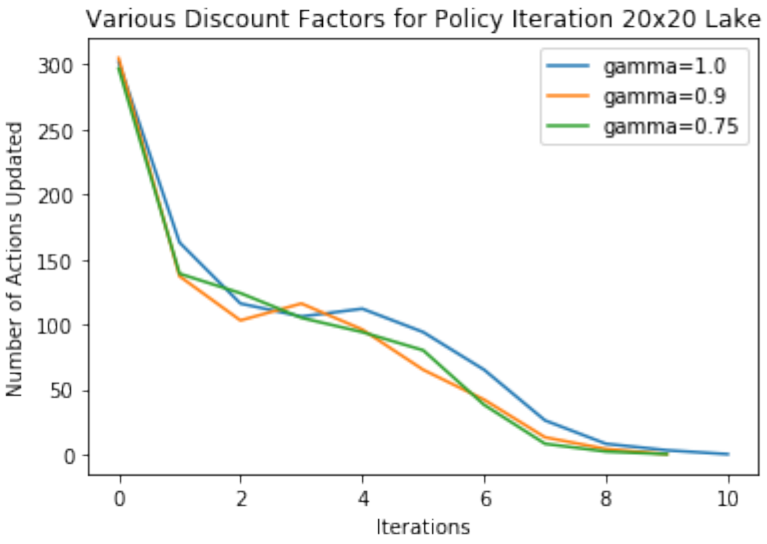




It’s clear to see that all of the experiments converged to the same optimal policy. Since the state space is so small, there is reason to believe that there are very few, and maybe only one, optimal policy. At first glance it may seem that the policy chosen is counter-intuitive, as most of the actions seem to be pushing the agent away from the goal state. This is actually the modeling choosing actions make it improbable for the agent to fall in the goal, while still making its way to the goal state. As stated in the MDP description, there is only a 33.33% that the agent will go in the direction it intends to go, 33.33% chance that it will go to the left of that action, and 33.33% chance that it will go to the right of that action. For example, the optimal action for [1,2] of the grid is ‘up’. There is a 33.33% chance that the agent will go up, a 33.33% change it will go left, which causes it to stand still, and a 33.33% chance that it will go right, which is the actual preferred next state. Notice how there is 0% chance that it will fall in the hole.

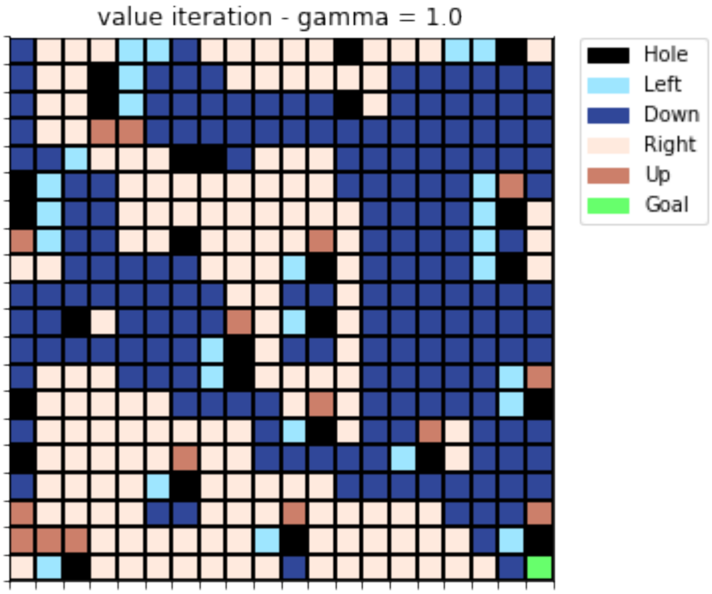
20x20 Lake

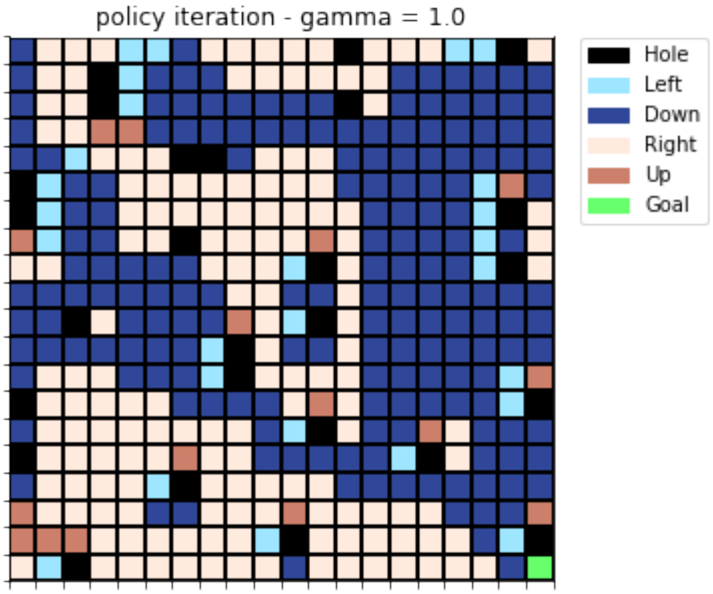


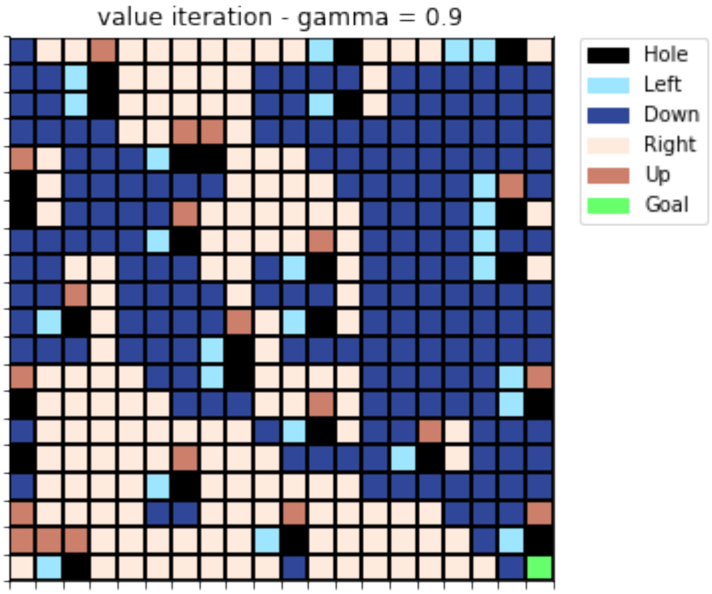


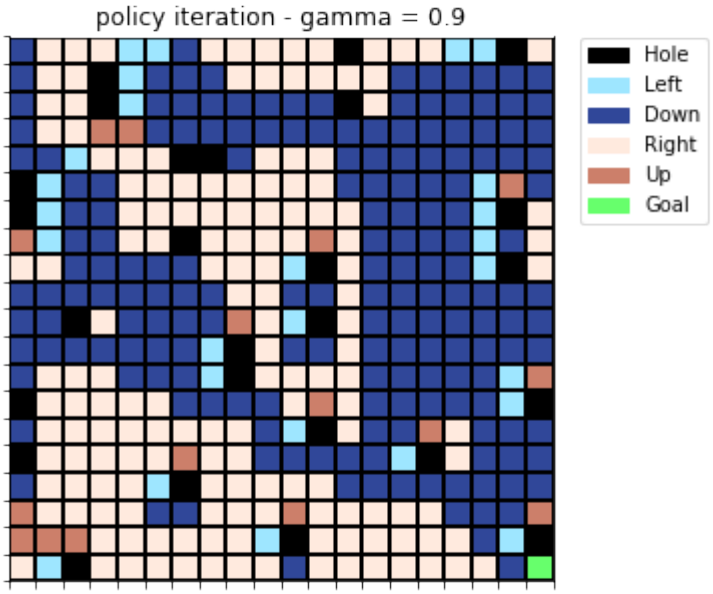
The same patterns of convergence seen in the 4x4 lake are exhibited in the 20x20 lake. For value iteration, as the discount factor increases, the number of iterations it takes for convergence increases. The number of iterations for convergences was not too dissimilar to the 4x4 lake. A discount factor of 1.0 took 1056 iterations, a factor of 0.9 took 295 iterations, and a factor of 0.75 took 135 iterations to converge. I will discuss the reasoning behind this in the next section comparing the optimal policies. That being said, the shapes of the convergence curves have a less drastic drop off compared to the 4x4. This is due the much larger size of 20x20, and it just takes more tries for the agent to navigate a larger world.

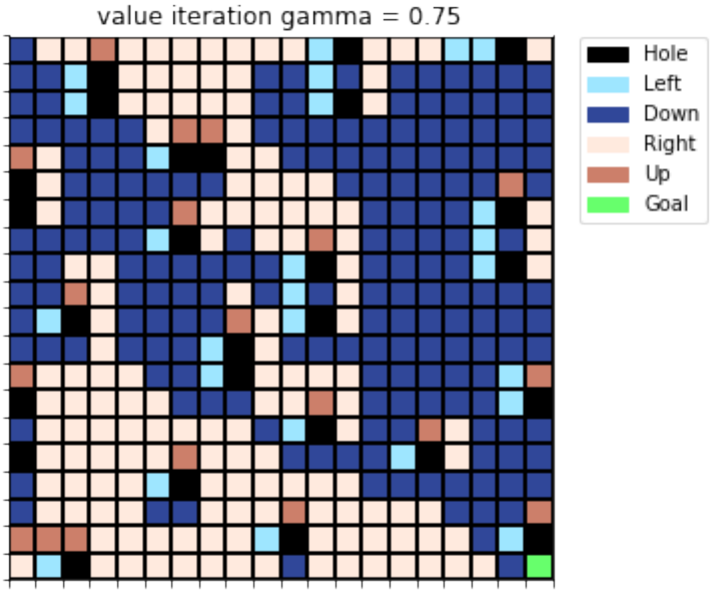
For policy iteration it takes 10 iterations to converge for a discount factor of 1.0. Both discount factors for 0.9 and 0.75 converge at 9 iteration. This is again due to learning from immediate rewards. Increasing the state space by 25 times only doubled the amount of iterations it took to reach an optimal policy. This too will be discussed in the next section.

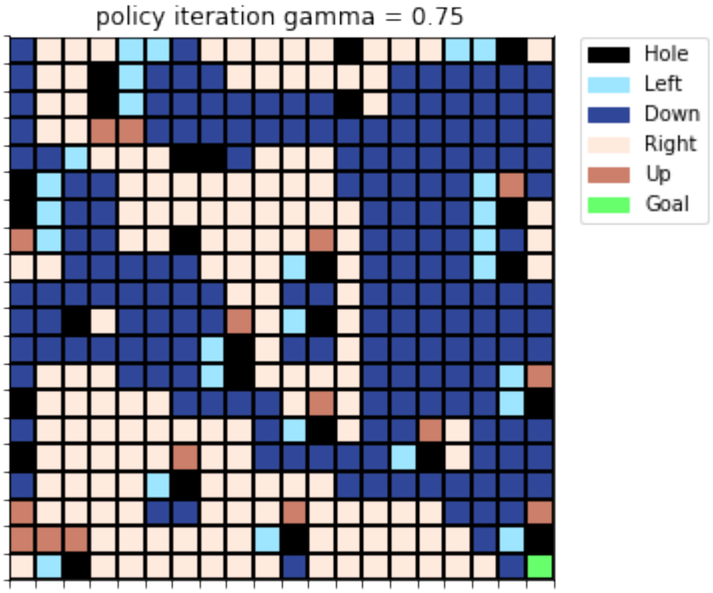












Unlike the 4x4 lake, there are minor differences between all the experiments. Comparing value iteration, a discount factor of 1.0 has the most differences between the three plots, especially in the top left quadrant. This makes sense because the top left quadrant locations will be visited first, but the agent is only considering long-term rewards. The initial values computed are not being considered in the final policy, so the agent doesn't really care what they are. The difference in optimal policy between the discount factors of 0.9 and 0.75 are minimal. They are just affected by how much the agent uses immediate values. Returning to the topic of convergence in the previous section; the optimal policy for the 30x20 lake is much less conservative than the 4x4 lake. Even though there are more holes to fall into, because of the exponential growth in grid space, there are a lot more safe locations. The agent can afford to take a more direct path to the goal without worrying about falling in a hole. The "safer" larger lake allows the model to converge at around the same time the smaller "more dangerous" lake does.

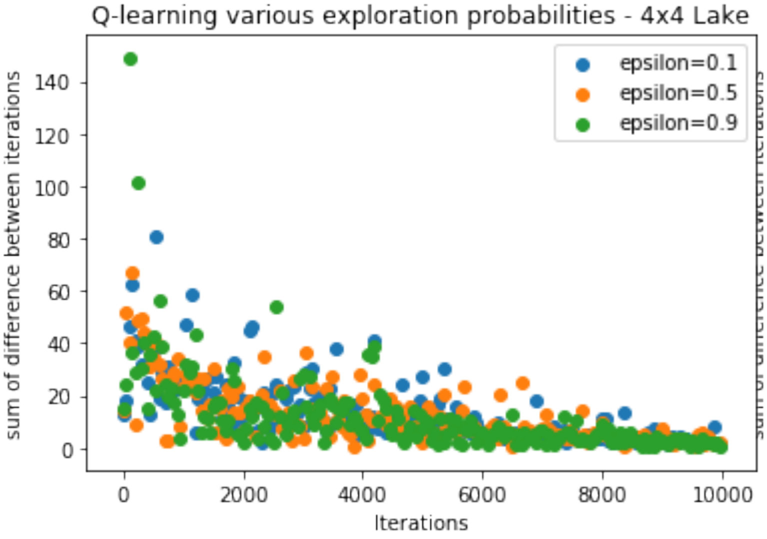
As for policy iteration, all the optimal policies are identical. This is because the policy iteration algorithm is doing a lot more computational work per iteration to find a policy. If there is indeed one optimal policy, policy iteration is more likely to find it than value iteration.

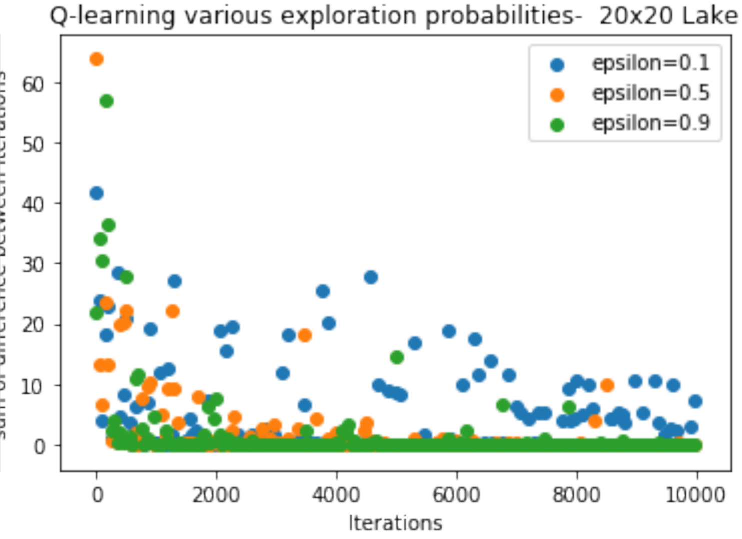
When comparing the results between two methods with discount factor of 1.0, they are identical. This is due to only looking at the long-term reward. There differences between policy and value iterations with a discount factor of 0.9 and 0.75. This is due to the different termination factors between the two algorithms. Value iterations can still change for some states if the optimal policy can have multiple paths. Again, do the larger state space, this lake if "safer" and the agent does not have to take a conservative policy. This is allows policy iteration to converge fairly quickly given the relative increased state size.

Q-learning

Q-learning is a model free algorithm. That is, the agent does not have any knowledge of the state transition functions or the reward models. However the agent will discover what are good and bad actions by trial and error. It will do this by approximating the state-action pairs of the Q-function by observing the interaction with the environment over time. Because this MDP is non-deterministic, the learning rate decreases proportionally to 1/visit(s,a), where visit(s,a) is the number of time the agent visits that state-action pair.

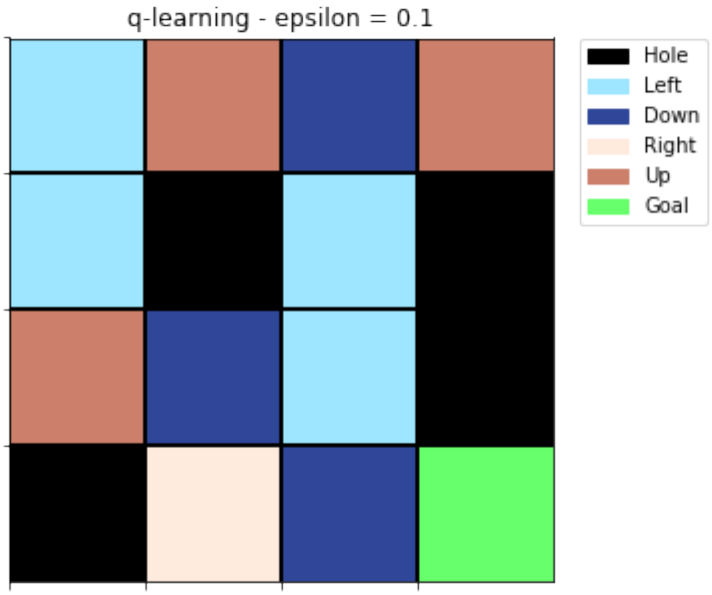
The agent must decided between exploiting known actions in each state and exploring new actions for a possible better reward. Q-learning is solved using a epsilon-greedy approach. Epsilon is the probability of the agent exploring a new, random action. I experimented with three difference epsilon values: 0.1, 0.5, and 0.9. Each experiment was ran for 10000 iterations with a disount factor of 0.9 to observe their convergence.

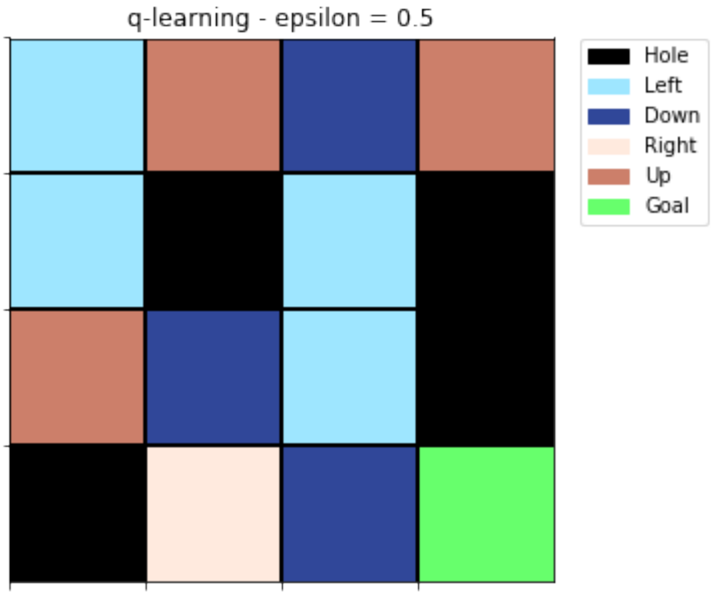


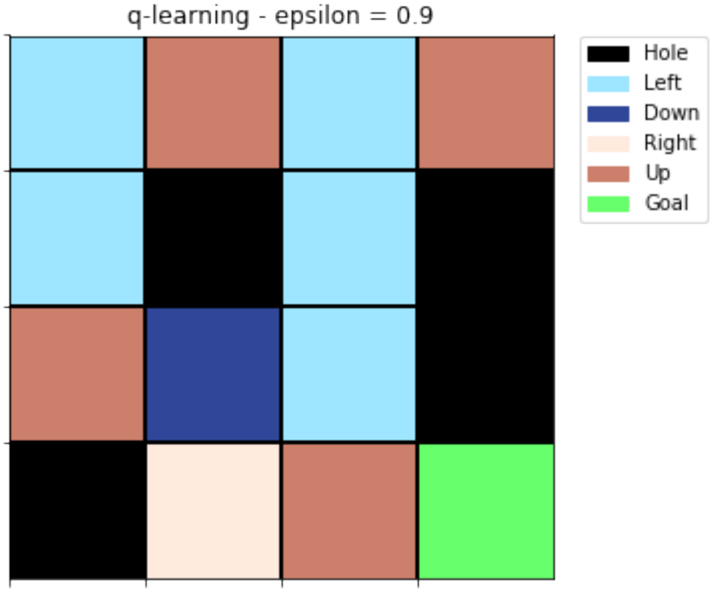


For the 4x4 lake, the convergences for the various epsilons are very similar. This is because how small the state space is. Whether the agent chooses to explore or exploit, it is most likely going to visit a large amount of space state combinations in 10000 iterations. The 20x20 lake is much more interesting, Epsilon of 0.9 converges fastest to a final form at about 2000 iterations. Since the space is so large, the agent benefits from exploring new states freely and gets a better sense of what the best actions are. Epsilon of 0.5 converges at around 4000 iterations because it exploits known good paths 50% of the time. The epsilon of 0.1 does not reach convergence in 10000 iterations because the agent simply cannot explore the space sufficiently. It constantly chooses when it knows to be good, but its most likely a locally optimal solution instead of a globally optimal solution.

4x4 Optimal Solution

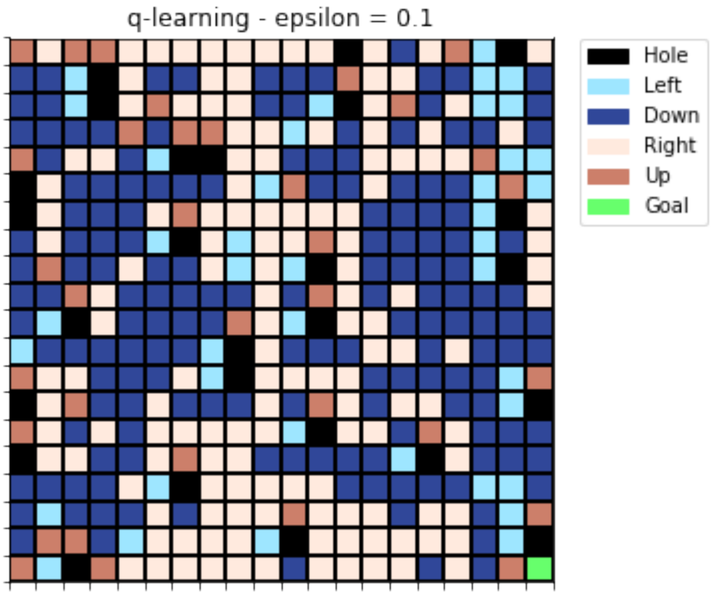


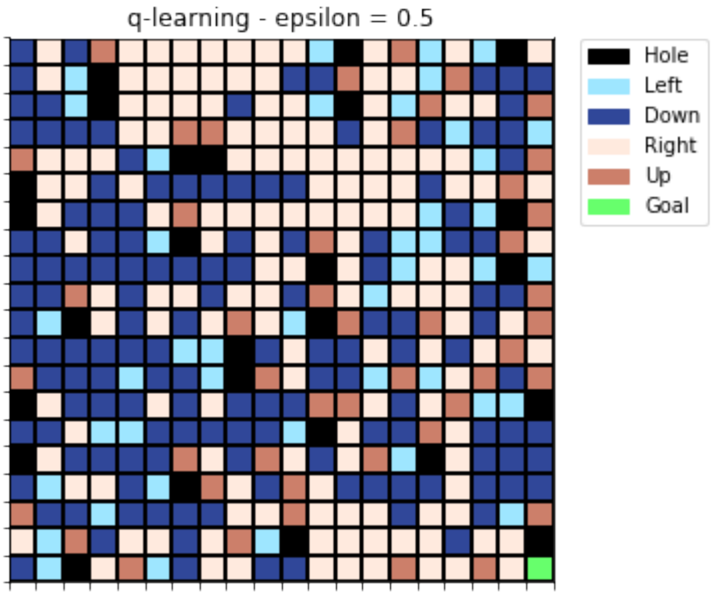


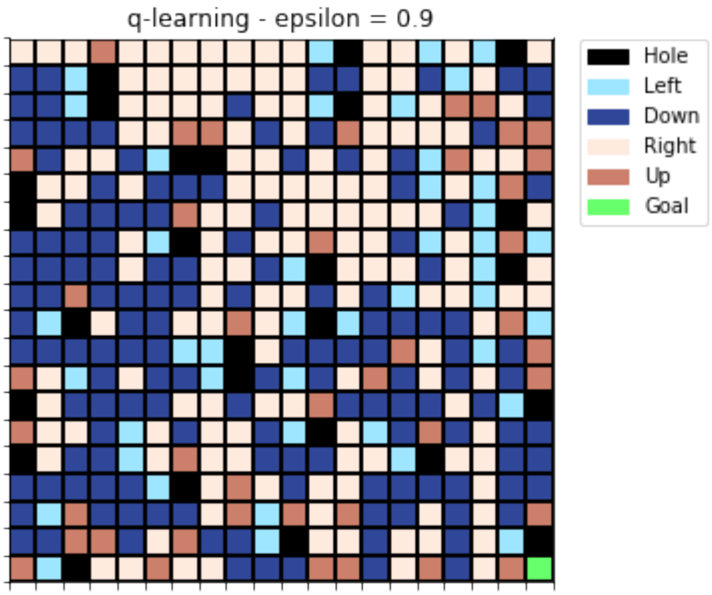


For the 4x4 lake, the optimal policy is just as conservative, and almost identical to the policies from value and policy iterations. This is because the state space is so small, with only 64 station-action pairs, that it can be explored reasonably within 10,000 iterations. Epsilon of 0.9 is actually the exact same policy. Epsilons of 0.1 and 0.5are the same, and is different by only one state. This is a minor difference and is most likely that the agent did not get to explore this state as often as it did others.

20x20 Optimal Solution







The different exploration rates give fairly different optimal policies. All of the polices seem to have learned to avoid falling into most the holes, but differ in the shortest path the goal. With 400 states with 4 possible actions each, there are 16,000 state - action pairs, so there are a large amount of possible polices to explore. The different exploration probabilities at 10,000 iterations are snapshots of the 16,000 state-action pairs. In future experiments I'd like to run q-learning for more than 10,000 iterations to see if the optimal polices would look similar. Unfortunately I ran into an overflow with iterations greater than 1000. All of the experiments are fairly different from the value and policy iteration counterparts. This is simply due to the model-free nature of 1-learning. The agent tries 10,000 times to learn the best policy, but because the state space is so large, it cannot figure it out as well without having prior knowledge to the model.

Conclusion

I explored three different methods for solving MDP: value iteration, policy iteration, and q-learning. Value iteration and policy iteration reliably can find the optimal solution for the 4x4 and 20x20 MDPs, because the agent has full knowledge about the models. Value iteration takes longer to sufficiently converge but is less computationally expensive to run. Policy iteration takes significantly less iterations to run, but is more computationally expensive and takes longer to run. The discount factor directly affects the both rates of convergence. Q-learning, on the other hand, assumes the agent knows nothing about its environment, and that said agent must understand its world through trial and error. (Q-learning is analogous to life, with ourselves as the agent.) Q-learning's advantage is that it does not need to know anything about the environment, but disadvantage is that it suffers from the exploration-exploitation dilemma and it takes many more iterations to converge. Optimal convergence depends on a good exploration probability for the given size of the MPD.

If given a choice between value iteration and policy iteration, I'd choose policy iteration for both MPD due to more consistent convergence to an optimal policy. If I could only perform q-learning, for the 4x4 lake I'd choose the an exploration rate of 0.9 in order to exploit the small space. For the 20x20 lake, I'd choose an exploration rate between 0.5 and 0.9 and run q-learning for more than 10,000 iterations.