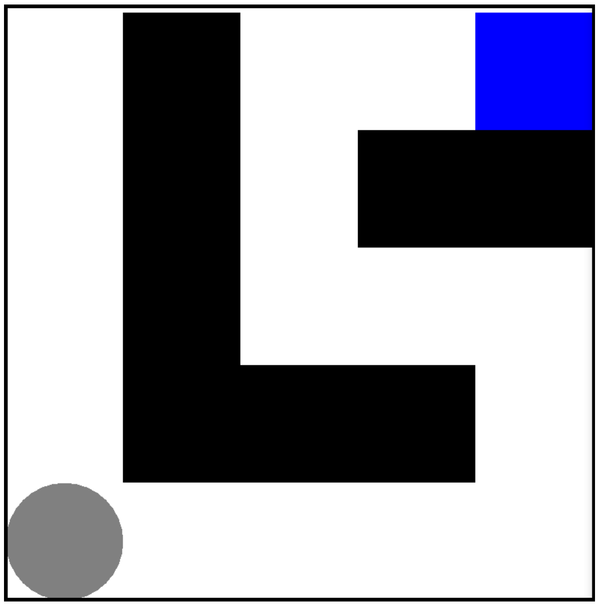
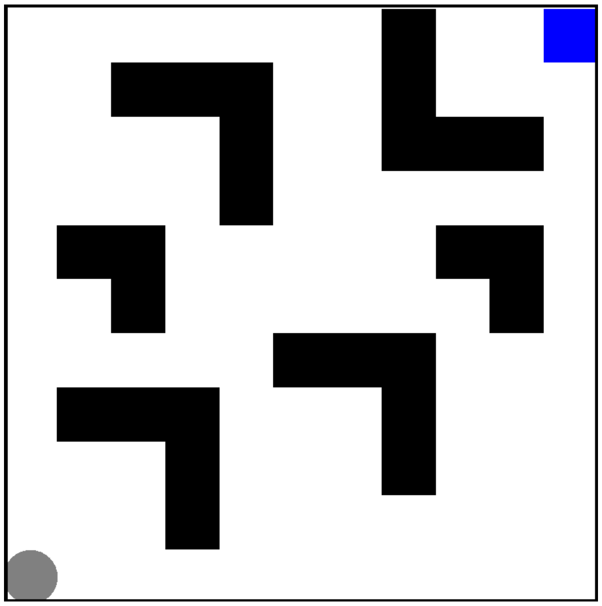
Reinforcement Learning

Zhijian Li

1 Markov Decision Problems

The MDPs will be from a group of problems called Grid World. Basically, it is a 2D grid in which an agent must navigate from start to finish. There are also obstacles in the grid. I will be using two different sized grids, a 5x5 grid and a 10x10 grid (below). The easy problem will be the 5x5 grid, while the harder problem will be the 10x10 grid.

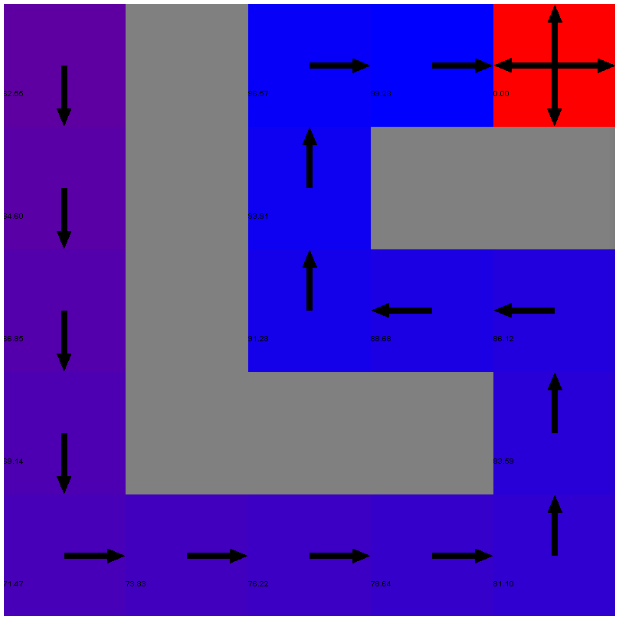
The grey circle represents the start and the blue square represents the goal. The reward function is defined as such, for every move the circle makes, there is a reward of -1 and for reaching the goal, there is a reward of +100.
The transition model is when the agent decides to move in any direction, there is an 80% chance that it will move in that direction, and 20/3 or 6.67% that he will move it any of the other three directions.
The algorithms I will be analyzing are value iteration, policy iteration and Q learning.

2 Importance

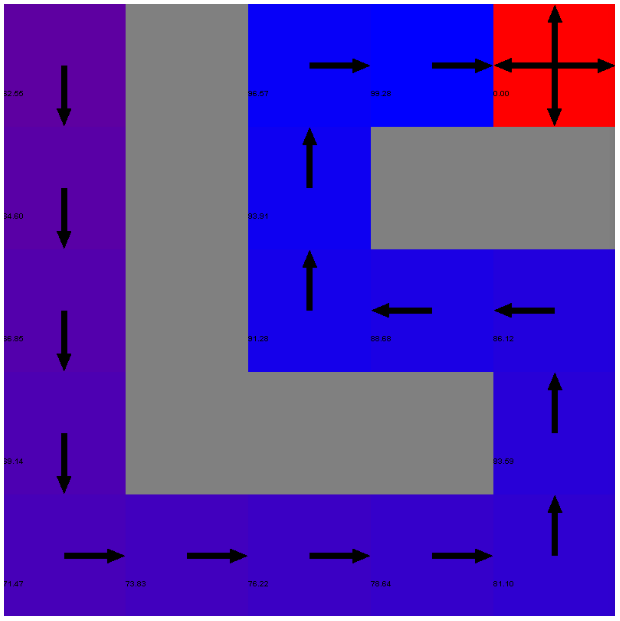
Grid World is an important problem because it can be linked to a lot of problems in the outside world. For example, the grid can represent a map, and the agent can be a robot trying to navigate from one place to another. For example, a Roomba would have to solve a very similar problem, as it has no idea what your house looks like, so must do some exploration and eventually plans its own shortest path to clean the house. So, Grid World can be translated to a navigation problem in the real world.

3 Easy MDP

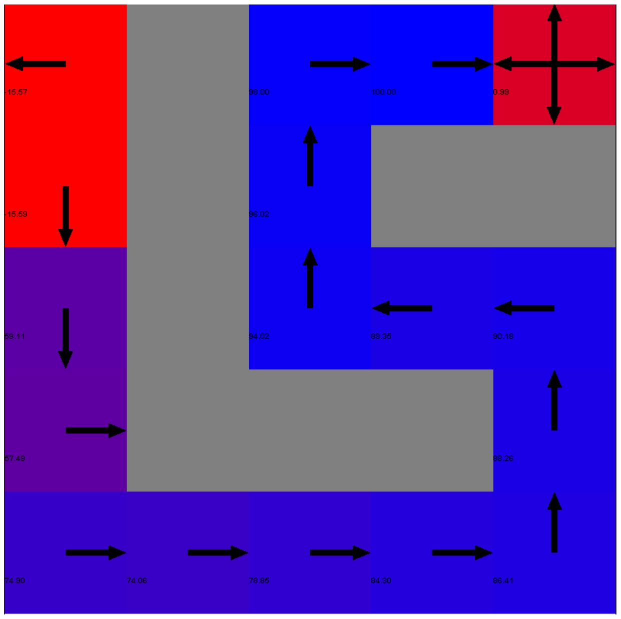
I ran all three algorithms on the easy MDP, setting the max number of iterations to be 100 for all three algorithms. I used the BURLAP library for all three algorithms. Below is the optimal policy generated for all three algorithms, along with graphs of the number of steps it took each algorithm to reach the goal for each iteration.



Value Iteration



Policy Iteration



Q-Learning

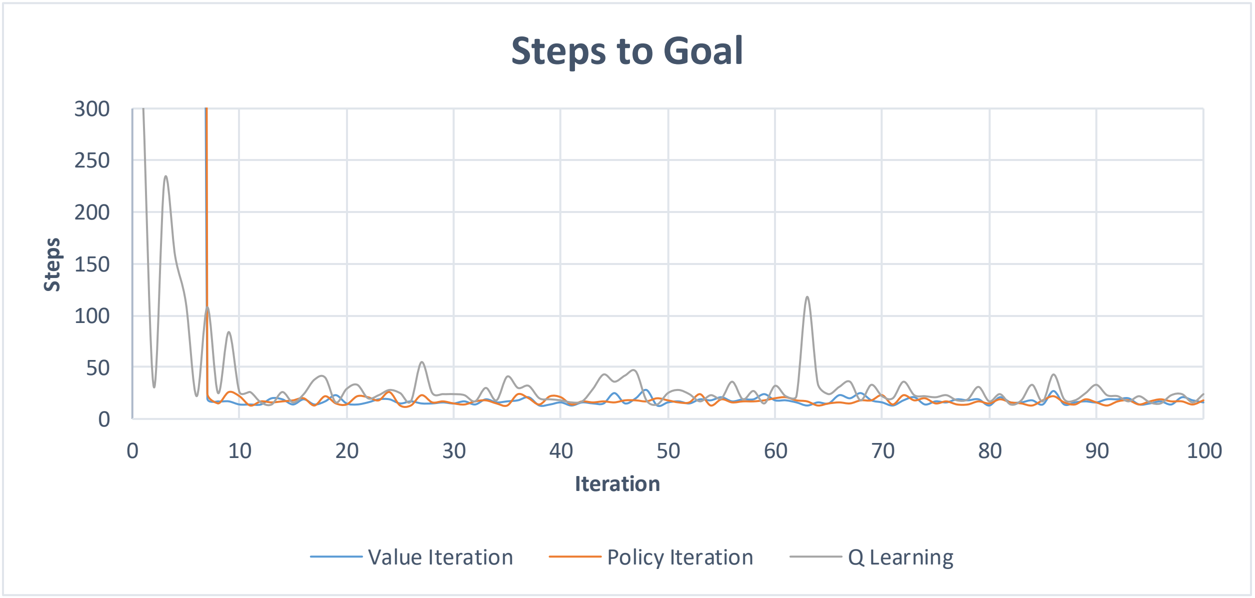
Both value iteration and policy iteration had the same optimal policy in the end. Given an easy problem, they both performed well, and produced perfect policies. This makes sense because it is proven that both algorithms will converge to the optimal solution given enough time. The policy is perfect, but the utilities are not. This is because for the top left corner, the utilities should be the lowest possible, because it is a dead end and there is no reason for the agent to ever move towards that space.

On the other hand, Q-learning almost performed well, with small errors such as the policies in the top left corner that tells the agent to run into the wall. But despite those errors, I would say Q-learning had better utilities (Q-values). This is because for the top left corner, the Q-values are much lower. This can be explained by the fact that as the agent initially explored, it realized that it was a dead end, and once it explored there, going in and coming out means double the energy required (as each action is -1).

What is also interesting is the color gradient that represents the utility of each space. The more purple color represented lower utility while the almost bright blue represented higher utility. As the agent moves closer to the goal, it makes sense that the utility increases. This is because of the way value iteration and policy iteration works. Once the agent reaches the finish, the +100 reward gets propagated back to the start, with it slowly being discounted.

What is also interesting is that, unlike these two algorithms, Q-learning decided very different utilities for the top left. It seemed like it knew a little more about the problem, and didn't just mark those spots as an ok utility like the other two, but marked those spots as unfavorable spots. This is very good because the top left is a dead end. While value iteration and policy iteration did not label these spaces as such, Q-learning gave more insight to the problem, saying that those areas will almost be guaranteed dead zones, or areas with little reward, as illustrated by its utility.

Another interesting thing to look at is not only if the algorithm can come up with the optimal policy given enough time, but how fast it takes that algorithm to converge to the optimal policy. I have graphed the number of steps it took the agent to get to the goal at each iteration (1 to 100) for each algorithm.

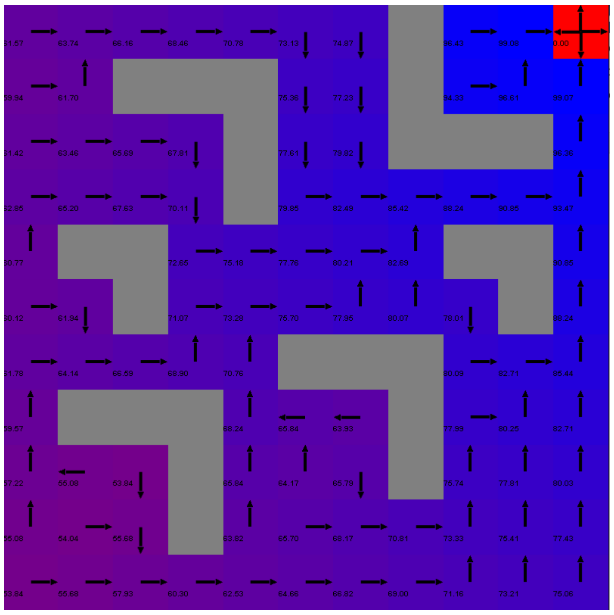


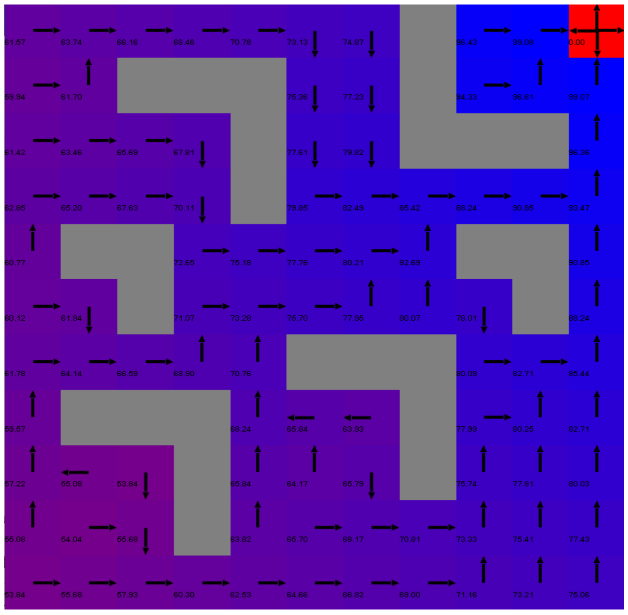
I set the maximum of the y-axis to be 300 because otherwise the rest of the data looked like straight lines. This is because policy iteration and value iteration started out with many steps, with value iteration and policy iteration having ~20,000 to 100,000 steps in the first 6 iterations. On the 7th iteration, value iteration and policy iteration converges to the optimal policy, as the number of steps drops to 20 and 25 respectively. As far as the optimal policy is concerned, both algorithms converged at the 7th iteration. After that iteration, they hovered around 15 to 25 steps to the goal.

Q-learning was different. It did not start out with as much steps as the other two algorithms. After 307 steps for the 1st iteration, Q-learning seemed to have already started converging. But, it took slightly longer to converge to an optimal policy than the other two algorithms. It seemed to converge after 10 iterations, but still had a higher variance in the number of steps as the number of iterations increased compared to the other two algorithms. This shows that Q-learning was not as confident in its policy as the other two algorithms.

4 Hard MDP

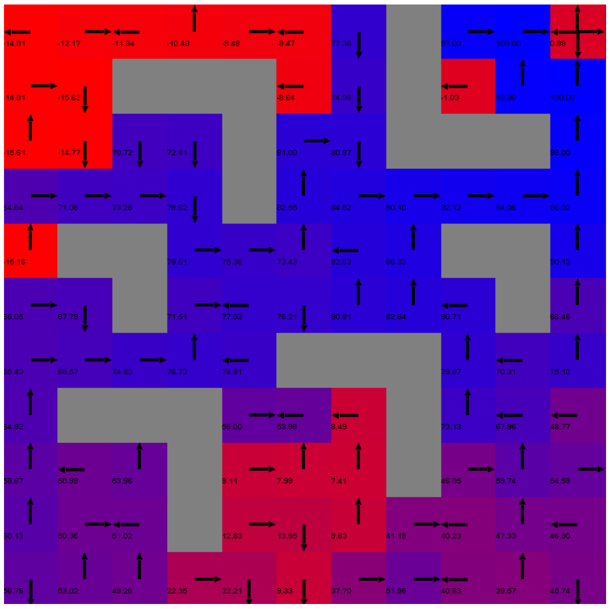
I ran the same tests on the hard MDP and recorded the optimal policy after 100 iterations. I also recorded the performance of each policy at every iteration (the number of steps).





Value Iteration

Policy Iteration



Q-Learning

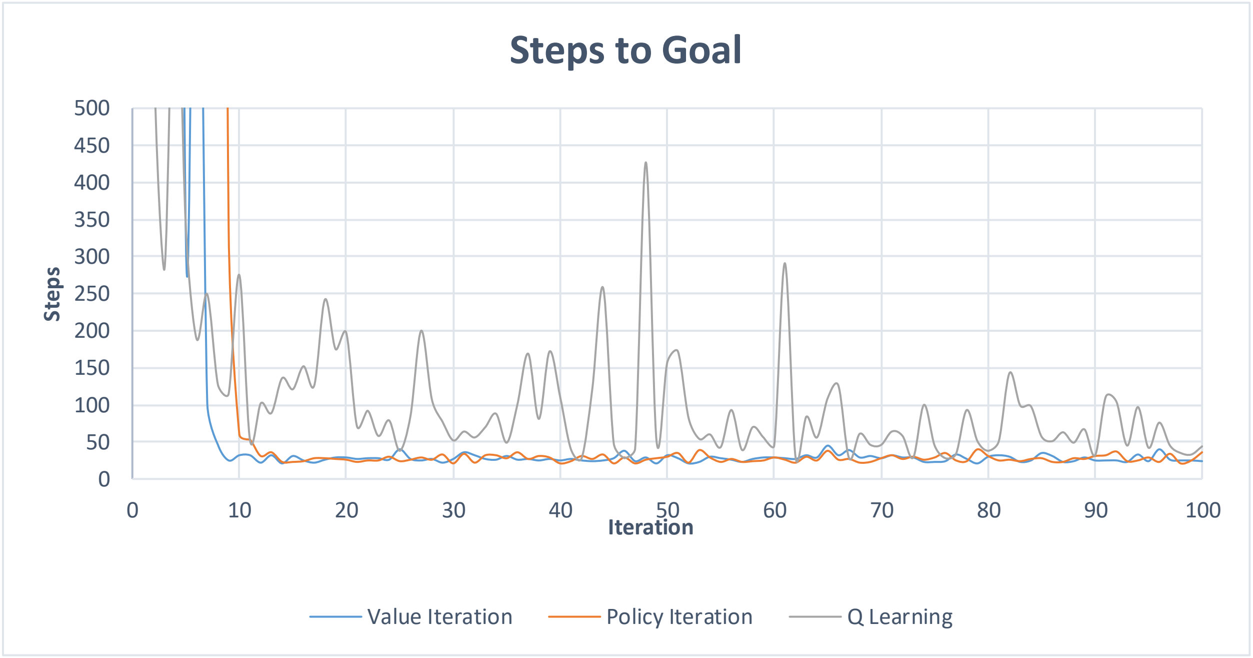
Just like the easy MDP, both value iteration and policy iteration have the same optimal policy. They both suggest that the optimal path is to go right and then up to reach the goal.

On the other hand, Q-Learning produced a very different policy. By looking at Q-Learning's policy, we get picture of what might be a better policy. Q-Learning decides that the best route (shortest route) to take from start to finish is the path through the middle. Therefore, the utilities of the top and bottom are very low. This is to say that both the top and bottom paths are not the optimal path to take, so don't even try to go in those directions. Unlike Q-Learning, value iteration and policy iteration do not have such low utilities. It seems like they do not give any more information than that and say that despite the bottom and top paths being not as good, they still mark it as if they are likely paths, when it is clear both paths will lead to a lower reward in the long run.

This might not be the case because if the world was deterministic, it is very clear that the best path to take is right first the up to reach the goal. But because there is a 20% chance that the agent will move incorrectly. There is a very low chance that given the policy, the agent will be able to move along that path at all. Therefore, for Q-Learning, through experience, it believed that the best path was the diagonal, despite it not having the actual shortest path, there is much less that could go wrong when taking the middle, or so it thought.

Below is a table for the first 10 iterations and a graph of the number of steps it took the agent to get to the goal using each policy of each algorithm at each iteration.

Iteration	1	2	3	4	5	6	7	8	9	10
Value Iteration	1854	16342	5871	2931	325	1107	103	46	25	32
Policy Iteration	3503	4254	1124	4791	29324	4396	5112	3038	339	60
Q Learning	2057	595	283	764	339	189	249	127	115	275



Like the graph for the easy MDP, I set the max number of steps to be 500, because there was a lot of outliers for the beginning 10 iterations.

All three algorithms took roughly 2000-3000 steps to get to the goal in the first iteration, after that, Q Learning decreases rapidly, while value iteration and policy iteration take longer to decrease. From the graph, you can see that value iteration converges at the 7th iteration and policy iteration converges at the 9th iteration. After they converge, both policies do consistently well, with barely any variance after the convergence.

Unlike those two algorithms, Q-Learning converges sooner, but the variance is much greater, as the number of steps to the goal fluctuates between under 50 to peaks of ~425 steps. huge variance can be described as the agent trying to make a move, but moves incorrectly (20% of the time). If the agent was along the diagonal path, and gets thrown off its path, it might make a long detour through either the top or bottom path, because of the policy it comes up with. For example, if the policy represented currents and rivers, if the agent makes a wrong move, it would be swept through the top or bottom river and make a much longer detour, thus the huge increase in steps for different iterations. It could also get stuck in the red zones, where the policy seems to almost be random, and there is a high chance the agent will be wandering aimlessly inside there.

Despite this, there are some times where Q-Learning does perform better than value iteration and policy iteration, but just barely.

5 Easy vs. Hard

For both MDPs, value iteration and policy iteration performed relatively the same. This is because there is a very simple reward function and only one goal. Because of this, eventually both algorithms, given enough time, will propagate the reward from the goal back to the start. There isn’t anything that will throw off the algorithm.

For Q-Learning, it is more clear that it performed worse on the harder MDP. For the easier MDP, not only did it converge in a shorter amount of time (iterations), the variance in number of steps was much less.

The reason Q-Learning took longer to converge is because it had to take more time exploring before deciding on an optimal policy. This is because the number of states is much greater for the harder MDP.

References

1. https://github.com/juanjose49/omscs-cs7641-machine-learning-assignment-4

2. http://burlap.cs.brown.edu/