CS 7641 Machine Learning

Assignment 4 – Markov Decision Processes

Liyue Hu (lhu81)

Abstract

This assignment seeks to explore various learning algorithms including Markov Decision Process (MDP) and reinforcement learning. For MDP, I used both using value iteration and policy iteration methods to find the optimal policy. For reinforcement learning, I used Q Learner. Comparison analysis was done between the methods on two different environments, Frozen Lake and Cliff Walking.

1. Introduction

For this assignment, I explored two simple MDP problems – Frozen Lake and Cliff Walking. First, I used MDP to solve for the optimal solutions to these problems, comparing and contrasting the convergence steps, time per step for value iteration method and policy iteration method. The key assumptions underlying these MDPs is that the models are non-deterministic, and that the reward and transition functions are known to the agent. Furthermore, I also explored the impact of having a smaller or larger number of states by looking at two versions of the Frozen Lake problem with different number of states.

Lastly, I chose a reinforcement learning algorithm, Q-learning to solve the two MDPs. The main differentiator for Q-learning is that it works in cases where the reward and transition functions are unknown.

2. MDPs

2.1. Frozen Lake

The Frozen Lake problem is a game in which the agent (e.g. robot) is trying to navigate from one side of the semi-frozen lake ("S") to a goal on the other side ("G"). The agent has to do so without falling into the many holes in between, only stepping on the frozen parts, otherwise the game ends as failed. As well, the agent must try to win the game with the minimal number of actions. There are four possible types of states: starting state "S", frozen space "F", hole "H" and the goal state "G". The features of the game are represented through the rewards that are associated with different states. Moving into the Goal state carries +1 point, and it also ends the game, with no other states to move to, or actions to take from there. Moving into any "H" states carries -1 point, and will also end the game. Aside from the aforementioned states, every other move carries a small negative reward of -0.1 to encourage the agent to win the game with minimal number of steps but also not at the expense of falling into a hole.

The environment is non-deterministic by design, since the ice surface is slippery. For each intended action, there is an 80% likelihood of the actual action being the intended, and 10% probability for each adjacent action. Various discount factors (0.1 – 0.9 in 0.1 increments) are tested to explore the impact of varying discount rate.

I used at 15*15 Frozen Lake grid setup as a representation of a “large” number of state environment to contrast with the “small” number of state environment of Cliff Walking.

2.2. Cliff Walking

The second task that I implemented was the cliff walking problem, where the agent is walking within a grid, similar to that of the Frozen Lake problem, trying to get from start to goal. There are also 4 different types of states that the agent can find itself in, start state “S”, cliff state “C”, road “R”, and goal state “G”. The start state is located on the bottom left of the 4*12 grid (with 48 possible states), while the goal state is located on the bottom right. Along the bottom are all “C” states, which all carry -1 point and has the power to terminate the game if the agent ends up there. The rest are R states that carry a slight negative reward of -0.1 to favor shorter routers. If the agent ends up in the goal state, then it is rewarded with +1 point, and the game ends in a win.

Cliff Walking is also a non-deterministic environment, where the wind can potentially move the agent in a downward direction 50% of the time, either one position or two position down. Different discount factors are also tested for this environment to demonstrate impact.

2.3. Why are these MDPs interesting?

The two MPDs I have selected are interesting to analyze together because they are two fairly simple environment, however they are set up differently in some key aspects to show how performance is impacted by these factors.

The first difference is the placement of the goal versus states that should be avoided. The second difference is grid size, which can be seen as a way to represent problem complexity. The third difference is the non-deterministic probability of the unintended actions and their respective consequences, which should affect the final policy.

3. Analysis of Value Iteration v.s. Policy Iteration

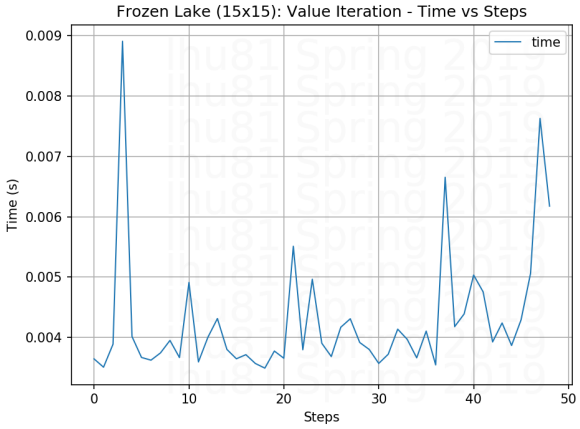
In this section, I will compare the performance of value iteration and policy iteration on the two MDPs. Value iteration and policy iteration are two different ways to solve for the optimal policy, they are both based on the Bellman Equation. Value iteration aims to solve the problem by computing and iteratively updating the utility value of each state. Policy iteration aims to solve the solve problem by testing and improving policies by updating the utility function. Value iteration is simpler while policy iteration will usually converge with less iteration.

3.1. Time v.s. Steps

As show below in Figure 1-4, value iteration takes less time per step but policy iteration takes less steps to converge. We can also note the difference between number of steps it takes for “large” number of states problem (frozen lake 15*15) over “small” number of states problem (cliff walking 4*12). It is also

important to note that for policy iteration, each additional step takes longer to run than the previous, whereas for value iteration the time requirement fluctuates from step to step.

For the large Frozen Lake problem, the amount of time per iteration is more than 100x for policy iteration versus value iteration, which makes value iteration a lot faster to converge even though it requires a lot more steps to converge. For Cliff Walking, time per step around 0.0007 seconds for value iteration and between 0.14 and 0.22 seconds for power iteration, again power iteration takes a lot longer than value iteration. For small number of state environment, it is more advantageous to use value iteration over policy iteration because significant time advantage of value iteration.



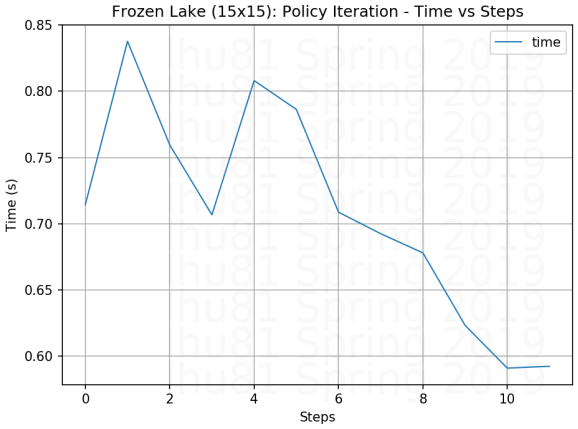
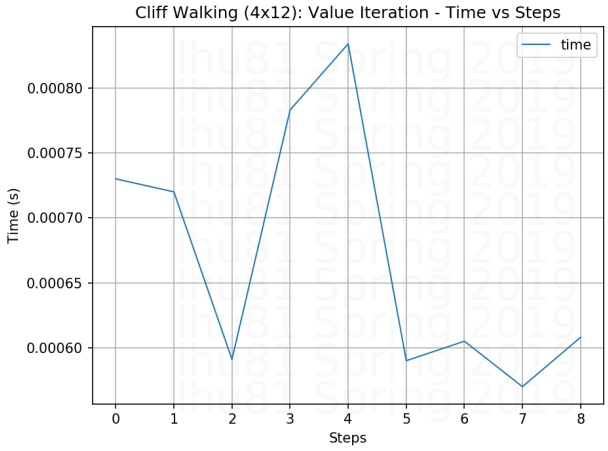


Figure 1 (left); Figure 2 (right)



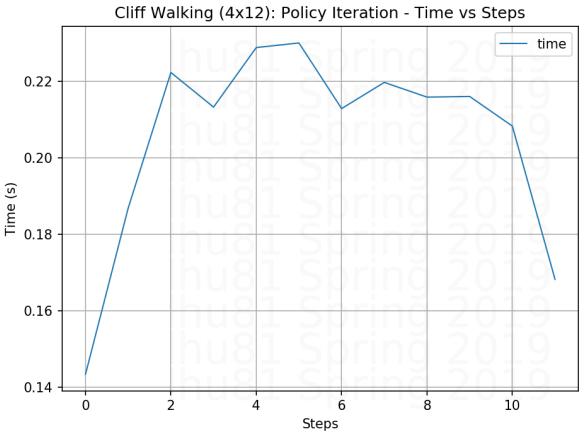


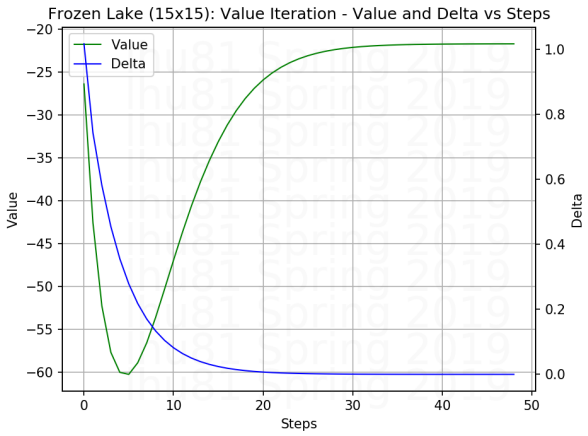
Figure 3 (left); Figure 4 (right)

3.2 Value and Delta vs Steps

As shown in Figure 5 -8, value increases with steps, while the delta changes over all states between steps approaches zero with increase in steps. This makes sense because with each update (step), for both value and policy iteration, we aim to improve the value function, which is seen as the consistent

increase in reward value for each step. As well, the delta approaches 0 as we get closer to the optimal policy and each step yield less and less changes.

It is notable to see the shape difference between the curves for value iteration and policy iteration, for policy iteration, convergence occurs fairly fast, even for large number of state problems (Frozen Lake), after the 1st step, we got farily close to the optimal policy. In contrast, the improvement for value iteration comes at a more gradual pace.



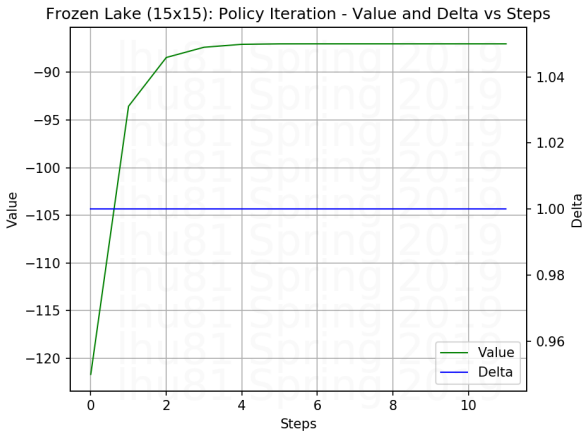
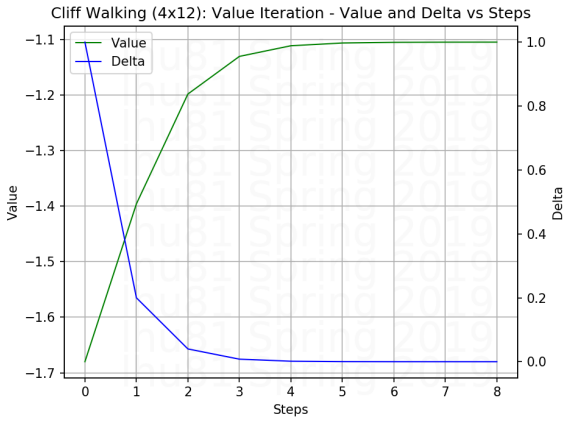


Figure 5 (left); Figure 6 (right)



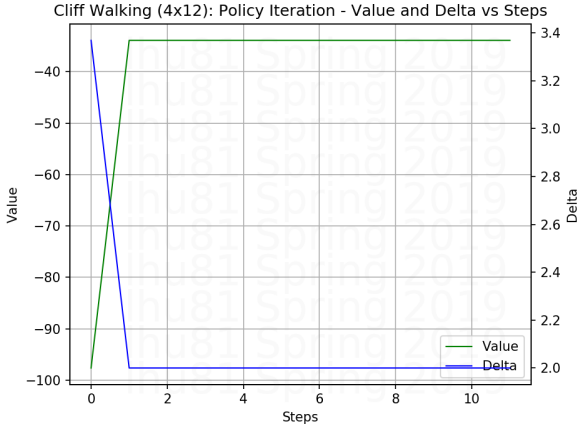


Figure 7 (left); Figure 8 (right)

Cliff Walking took policy iteration 12 steps to converge, and took value iteration 8 steps to converge. For large Frozen Lake, it took policy iteration 12 steps to converge, and it took value iteration 49steps to converge. It took Cliff Walking less iterations to converge for value iteration than policy iteration because of the different definition of convergence between value iteration and policy iteration, where value iteration is considered to have converged when delta change is less than a certain value, and policy iteration convergence is defined as 10 steps of non-changing policy, and as a result policy

iteration can not be less than 12. In general, this makes sense since often times policy would converge before values converge.

We can see that the difference in steps to converge is more noticeable for larger number of state problems between policy and value iteration. Policy iteration is therefore good for real life problems in which the cost of iteration/steps is high.

4. Q-Learner

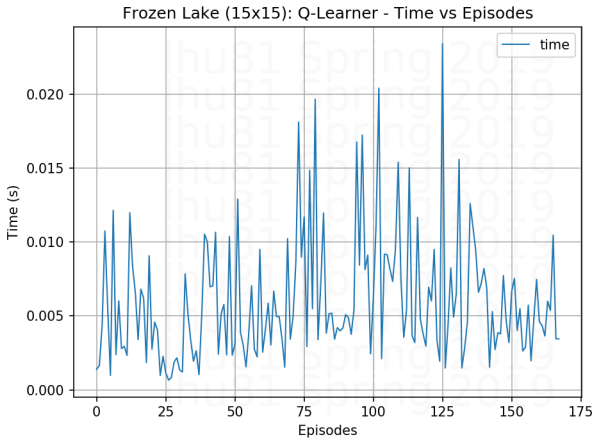
Q-learner is a reinforcement learning algorithm that seeks to find the optimal policy by updating an estimate of the actual Q value through each iteration (also called episode). Q-learner uses alpha update, which is an example of temporal difference learning to estimate Q. The main advantage of Q-learner is that it does not require that the agent know the reward and transition functions ahead of time, which is the case for a lot of real-life problems. However, it takes Q-learner more iterations and computing time to converge than through value iteration or policy iteration. I will delve deeper into the performance of Q-learner on the two MPDs below.

For q-learner, I tested different values for learning rate, alpha (0.1, 0.5, 0.9), different q initiation rules (random or none), different values for epsilon (0.1, 0.3, 0.5) and the same set of discount factors used for policy iteration and value iteration.

4.1. Frozen Lake

For the frozen lake setup, the best parameters were alpha of 0.9, random q initiation, epsilon of 0.1 and discount factor of 0.8. Alpha of 0.9 is the highest value tested, implying that a high learning rate that favors new learned value over old value is the best approach. Epsilon of 0.1 is the lowest value tested, which implies that a strategy that does more exploitation than exploration is preferred for this environment. Epsilon decay rate is set at 0.001.

As shown below in Figure 9 and Figure 10, it took the Q-learner 168 episodes to converge, a number that is significantly higher than the number for value iteration and policy iteration, which is expected. On the bottom right, delta actually fluctuated quite a bit between 0 and 1 even for later episodes.



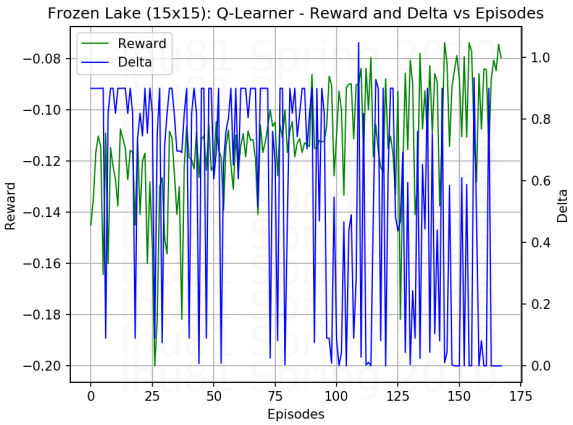
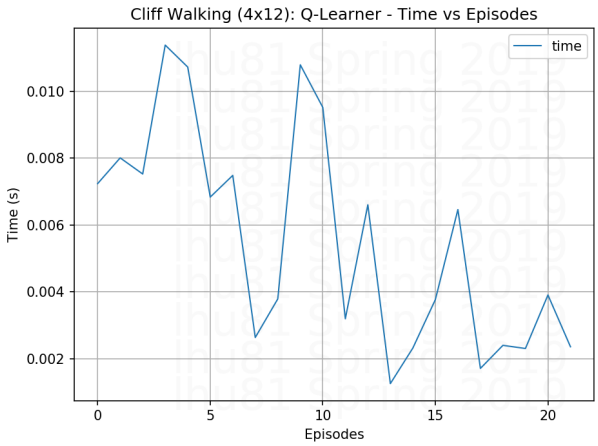


Figure 9 (left); Figure 10 (right)

4.2. Cliff Walking

For the Cliff Walking environment, the best parameters are alpha value of 0.9, initialize q-values at random, epsilon of 0.3 and discount factor of 0.6. Convergence occurred on the 22nd episode. This is higher than the 8 and 12 for both VI and PI, which is expected, since Q-learner has no prior knowledge of the reward or the transition functions. Cliff Walking also favors high alpha value, as with Frozen Lake. However, Cliff Walking prefer a higher epsilon value of 0.3 versus 0.1 for Frozen Lake, implying that more exploration and less exploitation is beneficial for this set up.

As shown in Figure 11 and Figure 12 below, time per episode exhibit a decreasing trend, with a range of around 0.002 to 0.010, this is in between the time per step for value iteration and policy iteration. On the reward and delta versus episode graph, we can see that reward did not increase as was the case for value iteration and policy iteration, this is because Q-learner does not require that the reward improves with every episode, so long as every state-action pairs are visited infinite times.



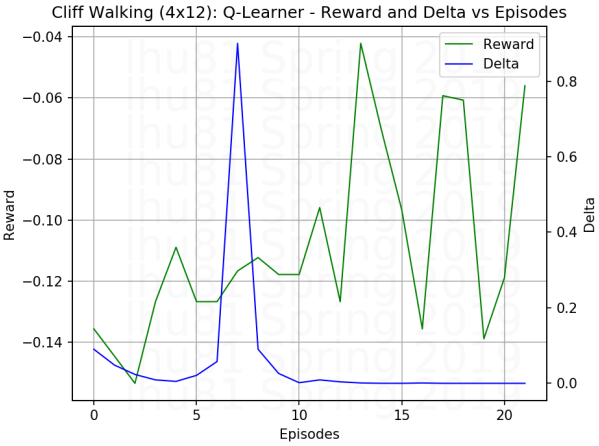


Figure 11 (left); Figure 12 (right)

5. Comparison

5.1. Overall Time

Environment	Algorithm	Best Params	Time
Frozen Lake (15*15)	Policy Iteration	*{"discount_factor": 0.8}*	12.54
Frozen Lake (15*15)	Value Iteration	*{"discount_factor": 0.8}*	4.59
Frozen Lake (15*15)	Q-Learning	*{"alpha": 0.9, "q_init": "random", "epsilon": 0.1, "epsilon_decay": 0.0001, "discount_factor": 0.8}*	5.48
Cliff Walking (4*12)	Policy Iteration	*{"discount_factor": 0.2}*	2.75
Cliff Walking (4*12)	Value Iteration	*{"discount_factor": 0.9}*	2.58
Cliff Walking (4*12)	Q-Learning	*{"alpha": 0.9, "q_init": "random", "epsilon": 0.3, "epsilon_decay": 0.0001, "discount_factor": 0.6}*	2.53

Table 1

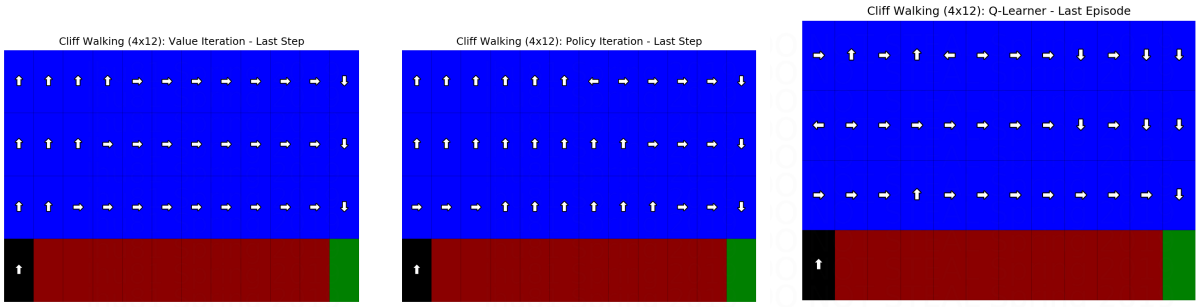
In Table 1 above, I summarized time performance of all three algorithms with their best parameters. For the Frozen Lake problem, value iteration required the least time to converge, and Q-learning required slightly more time, whereas policy iteration required significantly more time to converge. This is proof that policy iteration is the most time expensive algorithm to use for harder MPDs with large number of possible states. For Cliff Walking, Q-learning and value iteration had similar time while policy iteration had slightly higher time. This shows that the time requirement is not as prominent for easier MDPs with smaller number of possible states.

5.2 Optimal Discount Factor (Gamma)

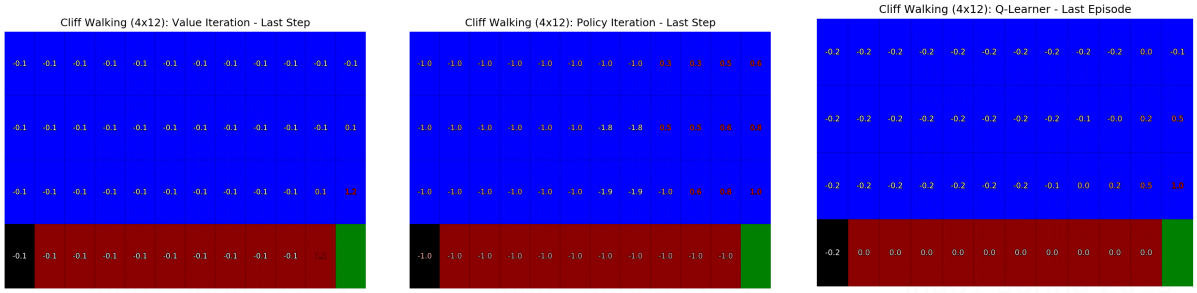
As shown in Table 1 above, the best performing discount rate was 0.2 for Cliff Walking policy iteration, 0.9 for Cliff Walking value iteration and 0.6 for Cliff Walking Q-learning. The best performing discount rate is 0.8 for Frozen Lake for all the algorithms. Discount rate is a number between 0 and 1, its purpose is to make sure the sum is finite even for infinite amount of future rewards. For the Frozen Lake problem, a relatively large discount rate of 0.8 was selected, placing higher importance on sum of future rewards. This is also the case for Cliff Walking under the value iteration algorithm. It is interesting to note that Cliff Walking problem had different optimal gamma for the three different algorithms. This could be d

5.3. Optimal Policy

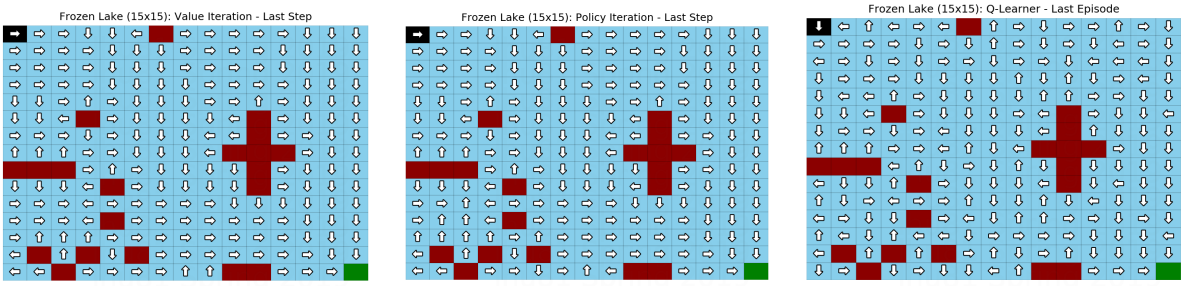
As shown in Figure 13 – Figure 24 below, I compared the final optimal policy and value for the various algorithms on the two different MDPs. Surprisingly, the policy and values for states from the last step/episode were quite different across the different algorithms. There are three possible reasons I can think of that could explain the discrepancy: the non-deterministic set-up of the MDPs, more than one possible optimal policy and the different discount factors. To further confirm my hypothesis, I ran a deterministic version of the two MDPs and compared the resulting final policy.



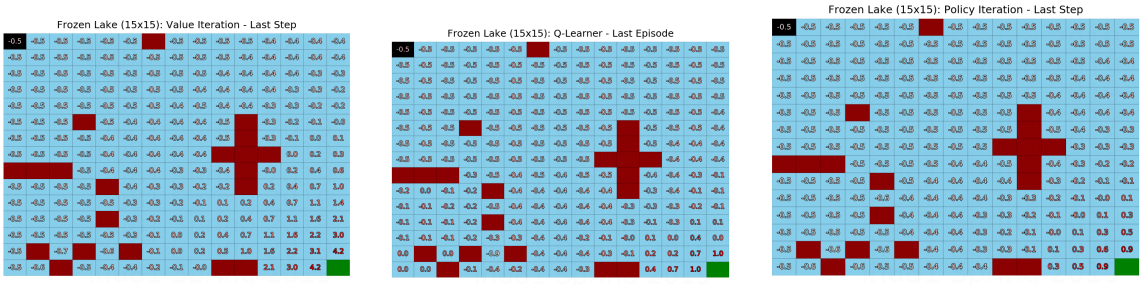
Final Step Policy Comparison – Cliff Walking: Figure 13 (left); Figure 14 (middle); Figure 15 (right)



Final Step Value Comparison – Cliff Walking: Figure 16 (left); Figure 17 (middle); Figure 18 (right)



Final Step Policy Comparison – Frozen Lake: Figure 19 (left); Figure 20 (middle); Figure 21 (right)

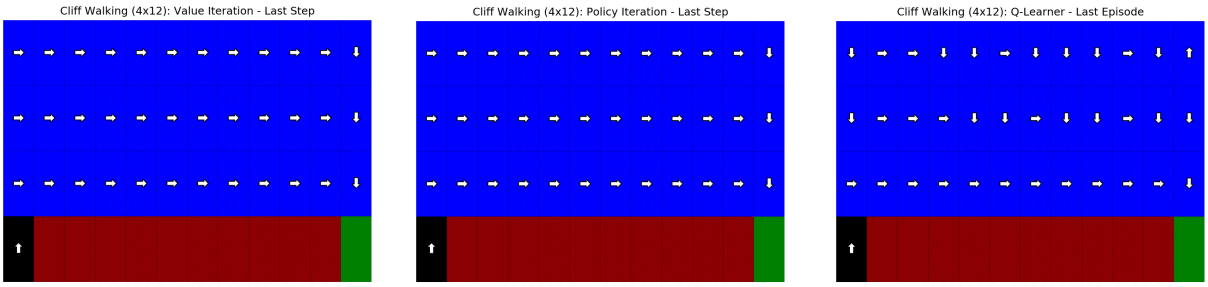


Final Step Value Comparison – Frozen Lake: Figure 22 (left); Figure 23 (middle); Figure 24 (right)

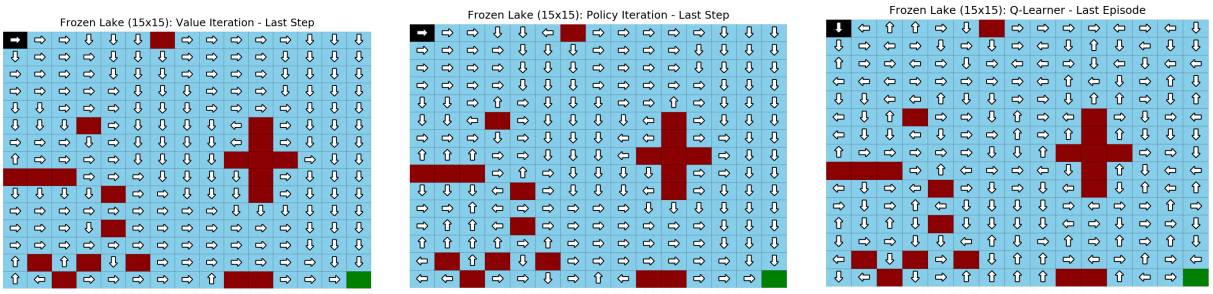
See Figure 25 – 30 for output to the deterministic environments. By comparing the deterministic final policy with the non-deterministic final policy for Cliff Walking MPD, I noticed that deterministic envoronment outputed the same final policy for all three algorithms, which is just following the shortest path between the start state and the end state, right beside the cliff, since there is no possibility of falling in to the cliff due to wind. This compares with the non-deterministic final policy for Cliff Walking, where some states close to the cliff had a final policy pointing upwards, in an attempt to avoid mistakenly falling into the cliff by wind.

I further compared the final policy for Frozen Lake, an environment with larger number of states. For the deterministic version of the MDP, the final policy for value iteration and policy iteration are fairly similar, with a few differences that will not change the final reward. Therefore, this difference in optimal policy is due to there being more than one optimal policy. I made similar observations when looking at final policy from the non-deterministic environment. In fact, the few places where the optimal policies differ are not paths that will impact the final reward.

The last episode for Frozen Lake for Q-Learner however, shows a very random policy, this makes sense because of the final episode of Q-learner is not necessarily the optimal policy. In fact, Q-learning uses random q initiation some times, resulting in exploring new directions.



Deterministic Cliff Walking Comparison – Cliff Walking: Figure 25 (left); Figure 26 (middle); Figure 27 (right)



Deterministic Cliff Walking Comparison –Frozen Lake: Figure 28 (left); Figure 29 (middle); Figure 30 (right)

It is interesting that Cliff Walking actually had fairly different final policies between deterministic and non-deterministic environment. This could be due to the fact that the probability of wind (ending up

somewhere unintended) is fairly high at 50%, versus 20% chance of slipping for the Frozen Lake problem.

6. Conclusion

In this assignment, I explored two different MDPs – Frozen Lake with larger number of possible states, and Cliff Walking with smaller number of possible states. I applied three different methods to find the optimal policy, value iteration, policy iteration and Q-learning. Value iteration and policy iteration are both based on the Bellman Equation and therefore are fairly similar in terms of the math behind them. Q-learning is a bit different in which it is able to estimate optimal policy without knowing the reward and transition functions. Because of the differences between each algorithm, they had performance differences. Policy iteration seems to be the most time intensive algorithm to use to converge, because even though it generally takes less iterations to converge, each iteration takes significantly longer, especially for more complicated cases. However, it is very useful in the real world when iterations are very expensive. Value iteration and Q-learning both converge in about the same amount of time, for environments with both small and large number of possible states. However, the main drawback of Q-learning is that it takes a significantly larger number of iterations to converge, with the difference getting larger as the number of states (complexity) increases. Value iteration strikes a balance between Q-learning and policy iteration as it requires fewer iterations than Q-learning and less time than policy iteration to converge.

The choice to choose one algorithm over the others will depend largely on the actual problem at hand, considering factors such as whether or not we have full knowledge of the full environment, and how expensive are iterations.

Reference:

https://github.com/cmaron/CS-7641-assignments/tree/master/assignment4

https://gym.openai.com/envs/FrozenLake-v0/

https://www.quora.com/How-is-policy-iteration-different-from-value-iteration

https://github.gatech.edu/mmallo3/CS7641_Project4

Code: https://github.com/huhu42/assignment4