Yi Shuen Lim
ylim68 | 903647878
CS 7641: Machine Learning
Assignment 4

Markov Decision Processes

Introduction

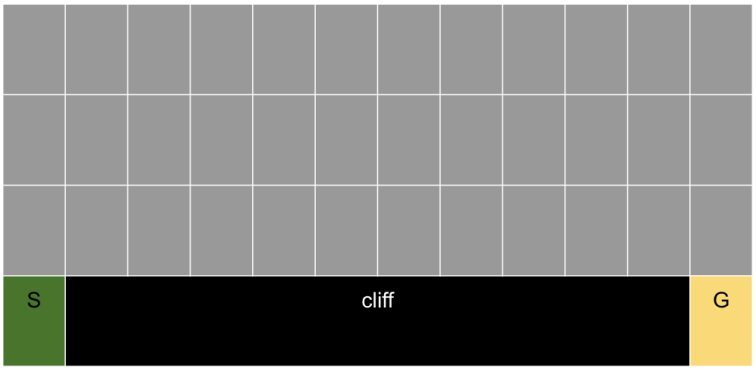
This report explores the relative performances of two Markov Decision Processes (MDPs) and a reinforcement learning method on two different MDP Problems -- the Cliff Walking Problem, and the Frozen Lake Problem. As we'll see in more detail in the next section, these two MDP Problems that are being evaluated are stochastic environments.

A Markov Decision Process models the decision making processes in stochastic, discrete and sequential environments. It looks at the environment as a series of states with associated transitions, allowed actions, and rewards to determine the best course of action that maximises reward in the long term -- the decision policy. We'll explore the MDP Learners Value Iteration and Policy Iteration, as well as the reinforcement learning technique Q-Learning, using their performance solving the MDP problems.

The 2 Markov Decision Process Problems

Cliff Walking Problem (4x12)

The Cliff Walking Problem involves an agent trying to cross the board without falling off the cliff. The environment we set up is 12 units wide and 4 units tall, and the agent needs to get from the bottom left corner to the bottom right corner. Below is our board:

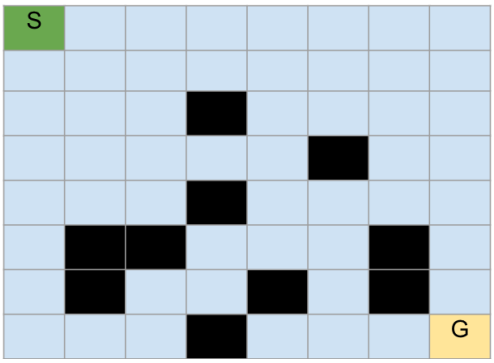


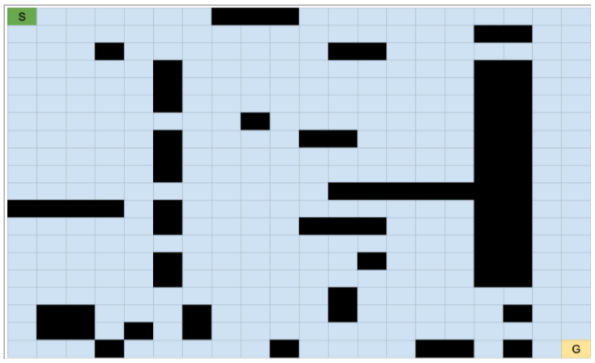
The agent can only move up, down, left or right from the start space (green) until they reach the goal space (yellow). When the agent makes it to the goal space, or if the agent moves into a cliff space (black), the try ends. The agent receives a reward of 100 upon reaching the goal space, -100 if they fall into a cliff space, and -1 for every step taken. Finally, there is a random probability of a downward wind that moves the agent down one space, towards the cliff, instead of whatever movement the agent chooses.

The Cliff Walking problem is simplicity coupled with a stochastic nature brought by the random wind element. It will be interesting to see whether the agent chooses to balance taking as few steps as possible due to the negative reward per step with higher risk of being blown off the cliff, or if they will choose to take the "safest" route far from the cliff at the expense of some extra steps. This can simulate navigation through obstacles and exogenous variables in the real world.

Frozen Lake Problem (8x8, 20x20)

The Frozen Lake Problem involves an agent whose goal is to traverse from the top left to the bottom right of the "slippery" board while avoiding holes. To see the differences in the performance of Learners across different sized environments, we set up two environments for testing: two boards of size 8x8 units and 20x20 units. The two boards are shown as follows:





Similar to the Cliff Walking Problem, the agent can move up, down, left or right from the start space (green) to the goal space (yellow). The try ends upon reaching the goal space or if the agent falls into a hole (black space). The rewards are: +1 for reaching the goal space, -1 for falling into a hole space and -0.1 for each step taken. What makes this a stochastic space is that each non-hole space is also slippery -- meaning that there is a random probability that instead of moving in their intended direction, the agent might move in either direction perpendicular to what they intended. For example, if the agent wanted to move upwards, there is a random chance they might move left or right with equal probability.

The Frozen Lake Problem's hypothesis and solution spaces are much bigger compared to the Cliff Walking Problem's spaces due to how the holes are dispersed vs the cliff being on one end of the board as well as the bidirectional stochastic element (slippery in two directions vs wind only blowing south) resulting in more outcomes. Ideally, we'll be able to see how the learners deal with suboptimal policies in more detail. The Frozen Lake Problem in general is interesting as it can be a model for navigation algorithms, as the agent has to take into account the different risks and rewards of many different possible routes to the goal space.

The 3 Learning Algorithms

Value Iteration (VI)

The VI algorithm is an MDP that uses the Bellman equation to iteratively update utility values at each state. Starting with arbitrary utilities, these utility values are evaluated and updated based on subsequent states and their ability to maximize total reward. Directly influenced by the reward system that is given with each problem, the idea is to have these utility values converge to their true utility values, from which one can derive an optimal decision policy.

Policy Iteration (PI)

The PI algorithm is similar to VI in that it is also an MDP that uses the Bellman equation to optimize, but it iteratively improves upon an overall policy instead of changing utility values. Over time, policy improvement will plateau and eventually stop updating as an optimal policy is achieved. PI typically has faster convergence than VI as the policy strictly improves with each iteration, unless it is already optimal.

Q-Learning (QL)

QL is a reinforcement learning technique that unlike the aforementioned MDPs, does not need to know about the environment. At whatever state the agent is at, QL calculates the Q value, similar to an expected value, taking into account current and subsequent states. The agent then selects their next best action based on what maximizes their Q value. As the agent starts with no information about the environment, they start by randomly selecting subsequent states to explore and learn about the board. A random action rate *epsilon* is set with a decay rate such that over time, fewer random actions are taken as the agent has more information about the environment. A learning rate *alpha* is also set to control how much prior-learned Q values are affected by newly-calculated values, with larger alpha favoring updated Q values.

Experiment Set-Up

This entire experiment was implemented in Python using code adapted from *cmaron* on Github. The following table summarizes the environment parameters mentioned in the first section of this report:

	Cliff Walking	**Frozen Lake (8x8 and 20x20)**
Goal reward	100	1
Step reward	-1	-0.1
Cliff / Hole space reward	-100	-1
Wind / Slippery probability	10%	20% (10% left, 10% right)

A GridSearch-like process was implemented to test different hyperparameters for the three learners, all listed in the table below, to return the results yielded by the best combination:

	Value Iteration	**Policy Iteration**	**Q-Learning**
Max steps/trial	1000		1000
Max trials	100		1000
Theta	10^{-5}		
Convergence consecutive theta	10		
Discounts	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]		
Initial Q			[0, Random]
Alpha			[0.1, 0.5, 0.9]
Epsilon			[0.1, 0.3, 0.5]
Epsilon decay rate			0.00001

Convergence is defined by the convergence threshold *theta*, and if the policy does not change after 10 iterations, based on the *convergence consecutive theta* set. The discount rates being tested over the range of 0.1-0.9 allow us to see the effect of weighing the influence of a new reward against prior reward states. QL has several more parameters, including the initial Q values set for each state, the *alpha*, and the initial *epsilon*.

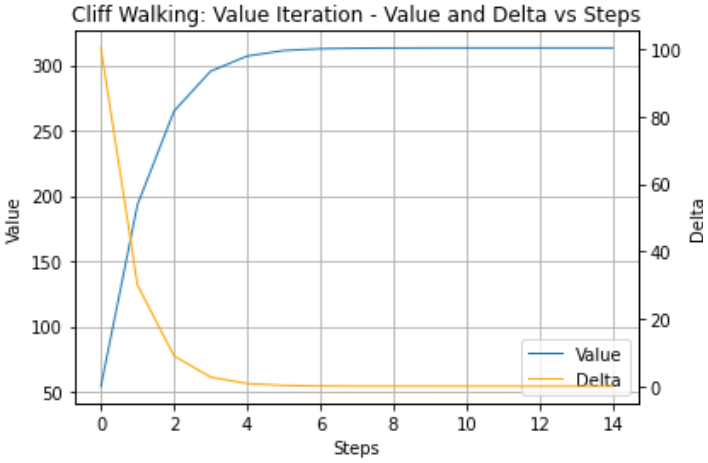
Results and Analysis

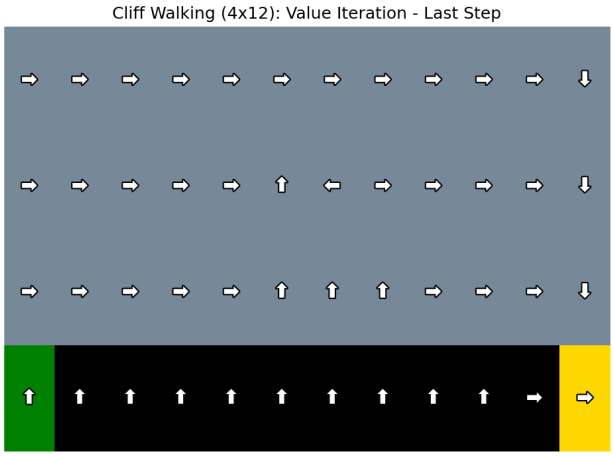
In the following sections per MDP problem, the optimal decision policies shown are the ones yielded from the optimal combination of hyperparameters.

Cliff Walking Results

The optimal policy obtained through VI, with its associated utility values and deltas over steps, is shown below. The best experiment was the one with a discount factor of **0.3**. We see that convergence was obtained after14 iterations or steps, though the reward values and deltas began to plateau around 6-8 steps. This optimal decision policy has the agent mostly walking across the cliff, but towards the middle of the board, playing it safe and moving upwards in case of the downwind. This means that with the 10% chance of a downwind, the agent sees that the risk of falling off the cliff is not enough to warrant taking the penalty of extra steps in the beginning. Midway across the cliff, perhaps as the probability of having no downwind throughout the entire try decreases, the agent takes a couple steps up before continuing across.

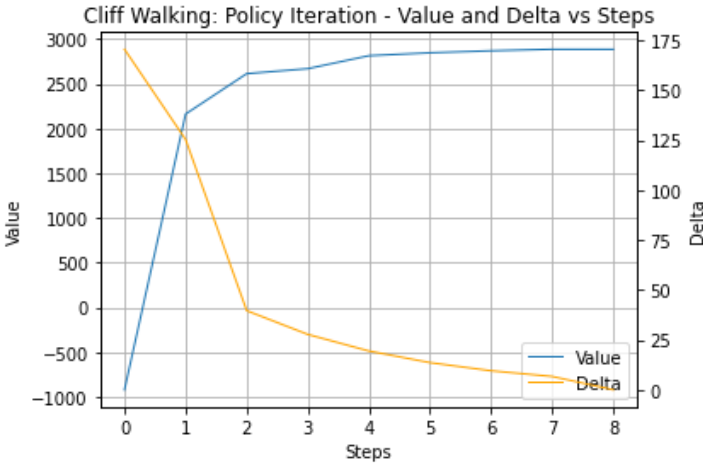
Likely, if the step penalty was lower or if the cliff penalty were higher, we might have seen the agent take the safest possible route, going all the way up and across before heading down towards the goal space. Conversely, a higher step penalty and lower cliff penalty might have the agent taking the shortest possible route right above the cliff.

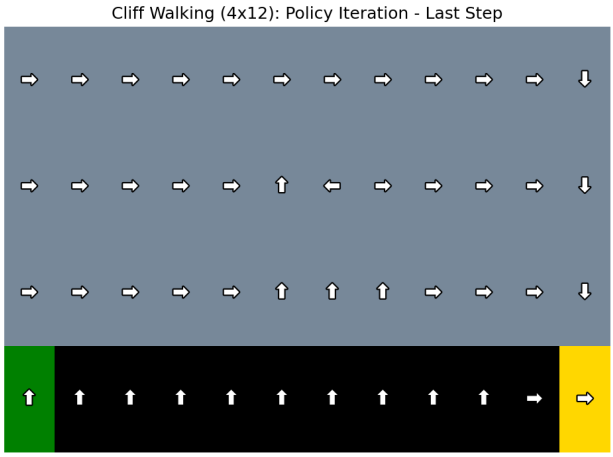




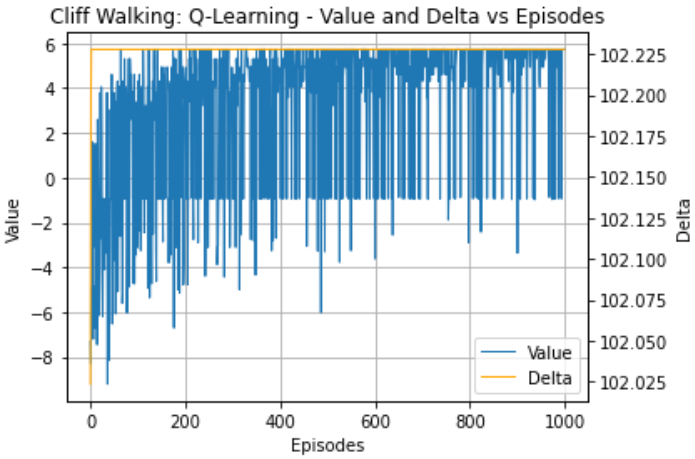
Next, we see the results of PI below. The best experiment for this one had a discount factor of **0.7**. Here we see that convergence was achieved in just 8 steps, and the converged decision policy was the same as the one obtained by VI. Because both VI and PI learn via the Bellman equation, the matching optimal policies is not surprising. The slightly quicker convergence of learning via PI in this problem confirms that updating and improving upon policies is more efficient overall compared to updating individual utility values.

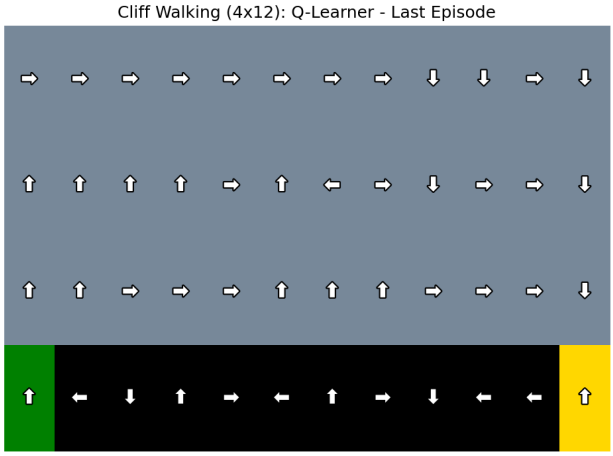
In examining the converged policy maps of the other VI and PI experiments at other discount factors, they all match the optimal policies returned here. Therefore, it is safe to assume that this is an optimal policy for the Cliff Walking experiment.





Finally, we see the results obtained by QL. These are the results with a discount factor of **0.7**, **random** initial values of Q, a starting epsilon of **0.5** and an alpha of **0.9**. From these results, we see that QL converged to a non-optimal policy, especially as we don't expect to see any down or left movements prior to the rightmost column. It has done some learning however, because other than these three down movements towards the top right and a single left movement close to the middle of the board, all other states have the agent directed either right or upwards. The seemingly random sprinkling of the ups and rights are due to the QL algorithm having the agent *explore* random movements throughout the board. In future experiments, perhaps raising the number of episodes or the number of steps per episode might give the learner more leeway to learn more about the space.

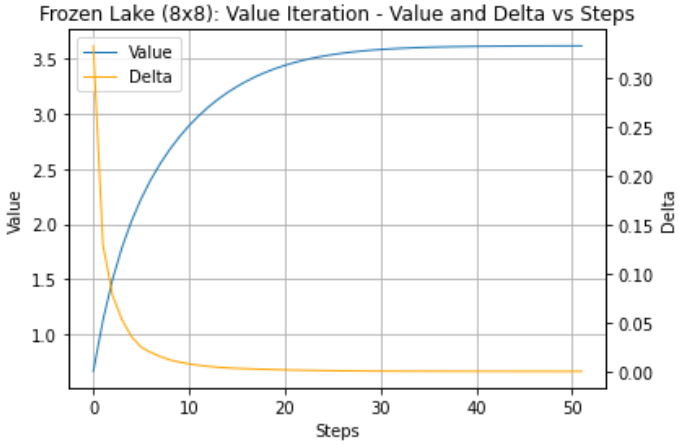


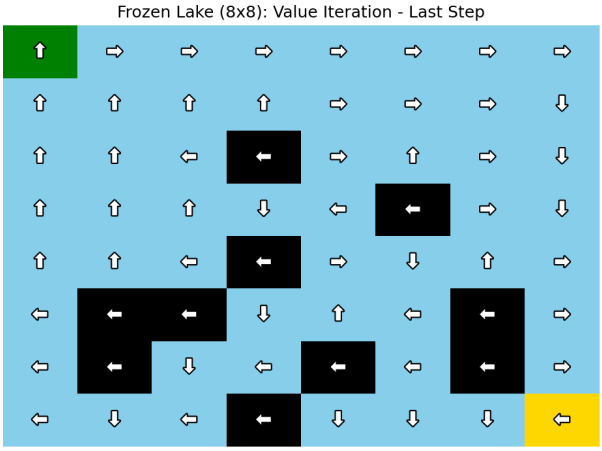


Frozen Lake 8x8 Results

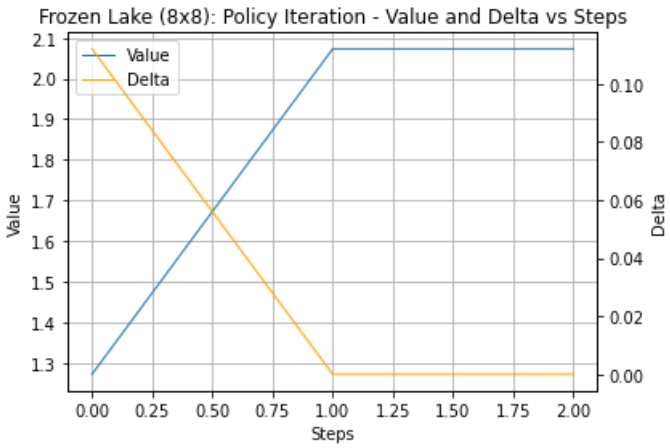
Below, we see the optimal converged policy obtained by VI, this with a discount factor of **0.3**, the same optimal discount factor as was obtained in the Cliff Walking Problem. This policy starts with having the agent start by moving upwards, hitting the edge of the board. This is likely because there are no holes in the first two rows of the board, and directing the agent upwards leaves zero probability for the agent moving down towards the holes. Upon *slipping* to the right, the agent is then directed to move all the way across to the right of the board, then down towards the goal. As there are a couple holes in the second-to-right

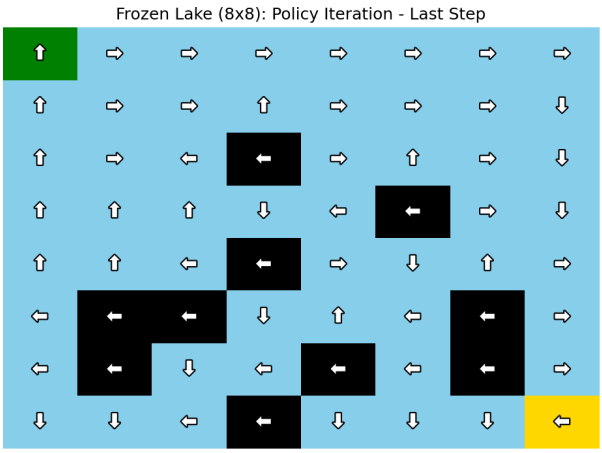
column, we see that adjacent to those spaces, the agent is directed to move to the right, hitting the right edge of the board, with the potential of slipping only upwards and downwards, not falling into a hole. This policy shows that the agent would rather take step penalties over falling into holes. We see this same pattern occurring with almost every space adjacent to a hole. As we'll see later, this is very similar to the optimal policy obtained by PI. VI achieved convergence after 52 iterations, though it started to plateau around the 25th-30th iteration.





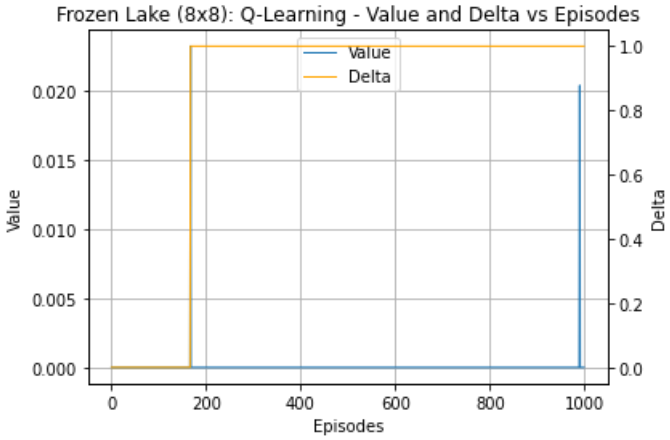
Below is the optimal policy obtained by PI with a discount factor of **0.8**. As mentioned earlier, the policy is very similar to the one obtained by VI, with the exception of a couple spaces (bottom left corner, a few in the top left corner). These minor differences seem to direct the agent more towards the goal, rather than moving the agent away from the holes. Furthermore, while VI took over 50 iterations to converge, PI only took 2 iterations to reach its optimal policy. Similar to what we observed with the Cliff Walking Problem, this once again confirms that PI converges over fewer iterations due to it updating overall policies over individual utility values.

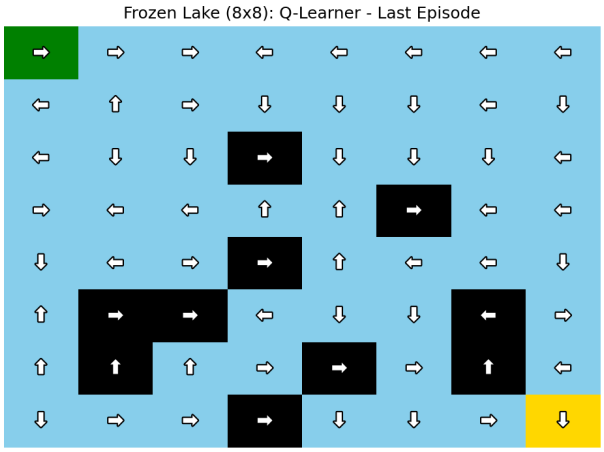




Finally, looking at QL below, we got optimal parameters of: a discount factor of **0.7**, **random** initial values of Q, a starting epsilon of **0.5** and an alpha of **0.9**. We see that even in this optimal setting, convergence didn't occur, with values staying at 0 and spiking to 0.02 at a single episode. In this inferred policy, we see a lot of randomness. It can be concluded that the learner had not fully learned the environment, especially since there are some parts of the policy that push the agent into a terminal state, i.e. towards a hole. For example, on the right side of the fourth row from the top, and the space right above the goal space has the agent going into a hole instead of towards the goal. QL already took much longer to run in terms of the number of

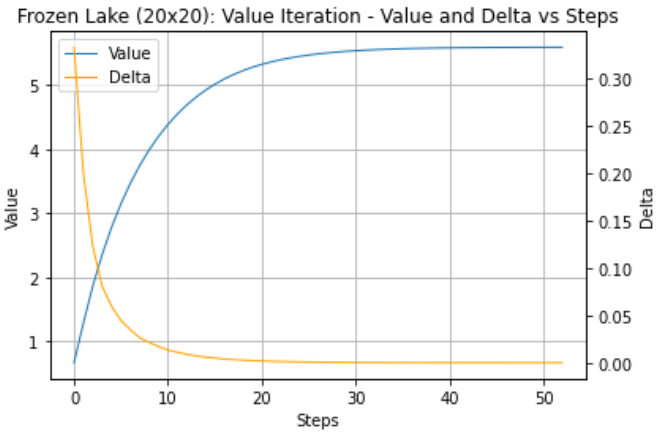
seconds per experiment, but it would be interesting as future work to see when learning picks up if allowed to go for more iterations and more episodes.

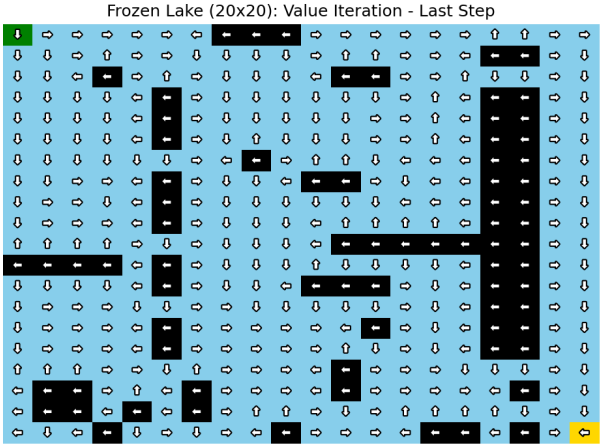




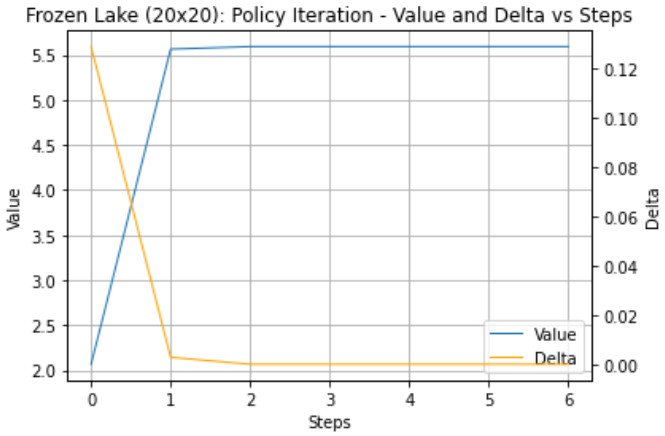
Frozen Lake 20x20 Results

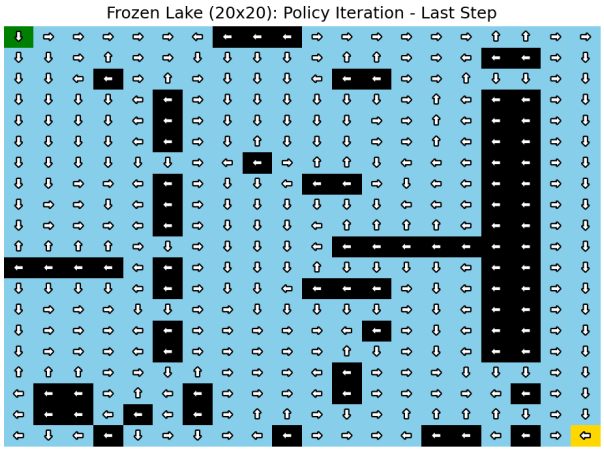
Finally, we examine the results of our learners on the larger Frozen Lake Problem. First we see the results from VI with a discount factor of **0.9** that took 53 iterations to converge. This was one more iteration compared to the smaller Frozen Lake we had. In the results, we see that there is a consensus here between the results of 8x8 and the results of 20x20 -- mainly that we want to avoid the holes as much as possible. On top of that, if there is a space flanked by two hole spaces, the agent will be directed towards the hole that brings it closer to the goal space.



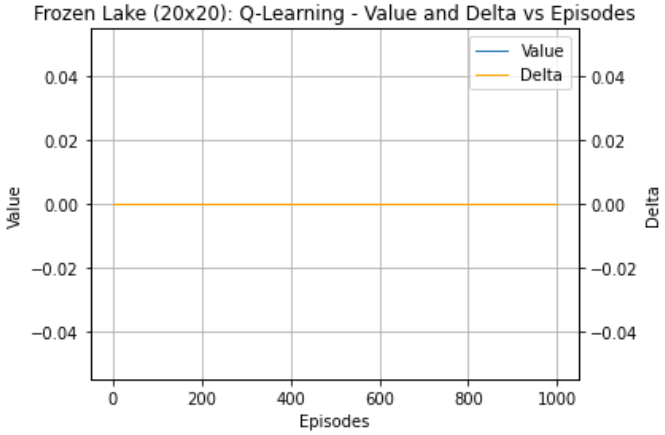


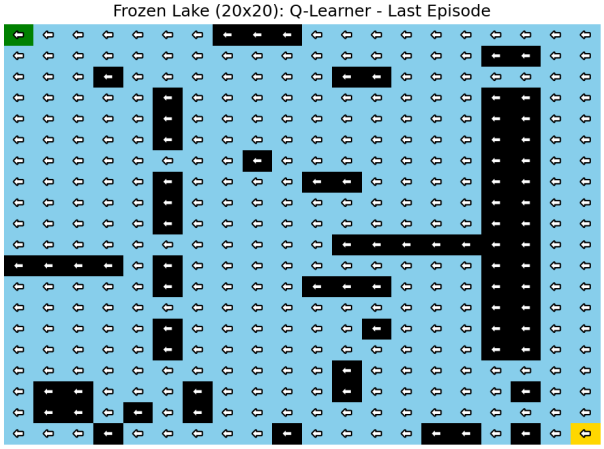
Next, we see the results from PI with an optimal discount factor of **0.9**. The overall decision policy turns out to be identical to that of the one obtained by VI, and this one converged after 6 iterations. Because the two learning methods yielded the same policy, it's safe to assume that this is the optimal policy or an optimal policy.





Finally, we look at the performance of QL on the 20x20 Frozen Lake Problem. None of the experiments tested were able to yield any learning. Going off of what we saw with the 8x8 problem, this is not surprising, as the inability to converge on a much smaller space would mean that performance would be worse on this space. Similar to what was concluded in for the 8x8 problem, perhaps allowing it to run for more steps per episode as well as more episodes in total will allow for some learning. We see from the "optimal" decision policy as well that it could not find a better policy than the initial one of always staying left, never reaching the goal.





In Summary

Below, I summarize findings with some key points.

VI and PI on Different-Sized Problem Spaces

Comparing the performance of the two MDP Learners across the different sized problem spaces, it is interesting that VI only took a single iteration more to converge with a much larger problem space. Likely, it is due to the higher number of possible actions taken, thus *relatively*, the number of actions tested versus the number of states per iteration was higher, leading to better inference of policy per iteration. For PI, although the number of iterations were much lower for both problem spaces, the 20x20 space took three times as many iterations to converge. Because the space is so much bigger, when converging on policies instead of

values, there are many more policies to be tested, so it took that many more iterations to find a good policy to converge on.

Algorithm Runtimes

We took the solve time per iteration only for the 8x8 Frozen Lake Problem. Out of the three learners, Value Iteration performed the quickest at just a couple seconds per experiment, as the algorithm only had to converge on individual utility values. Policy Iteration performed slightly slower, ranging from 8-10 seconds per experiment. This is likely because the solving of the Bellman equation for an entire policy to converge is slightly more computationally intensive. However, it is worth noting that for the 8x8 Frozen Lake Problem, VI ran for over 50 iterations while PI ran for only 2. Therefore, overall time per iteration was less for PI, but convergence still happened quicker overall for VI. Finally, Q-Learning took the longest to run, taking roughly 10 minutes per experiment, though it is worth noting that we had each QL experiment running for 1000 episodes compared to the 100 max trials set for VI and PI. QL needed to have more episodes because of its lack of knowledge of the environment.

Learnings on Q-Learning

In all three problems, Q-Learning was unable to really learn, let alone converge. Because the Q-Learning algorithm slowly learns about the environment through random trials, the complexity of the spaces did not allow for enough learning in the allotted number of iterations and episodes. This is why we saw some learning in the much simpler Cliff Walking Problem compared to the higher-dimensionality Frozen Lake Problems. However, this does not mean that QL is not *as good as* the MDP learning algorithms. QL has the benefit of not needing to know how the environment and its reward system is set up. In a much more complex and stochastic environment, it is possible that QL performs better relative to VI or PI.

References

- Maron, Chad. ChedCode: CS 7641 Assignments (2018), GitHub repository, https://github.com/cmaron/CS-7641-assignments