CS 7641- Machine Learning

Project 4 – Markov Decision Processes

Madhu Mohan
MMOHAN8

Abstract

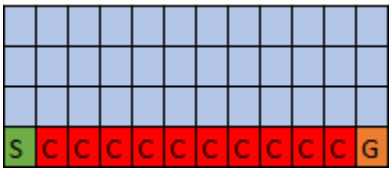
This report analyzes 3 reinforcement learning algorithms on 2 Markov Decision Processes. The 2 Markov Decision Processes used are the cliff walking problem and a 20X20 Frozen Lake Grid problem. The 3 RL algorithms analyzed are: Policy Iteration, Value Iteration and Q Learning. This report has 4 parts:

1. The Problems considered
2. The 3 Reinforcement Algorithms
3. Performance comparison of Policy Iteration and Value Iteration
4. Q Learning performance and comparison with Policy and Value Iteration algorithms

Part 1: The Problems

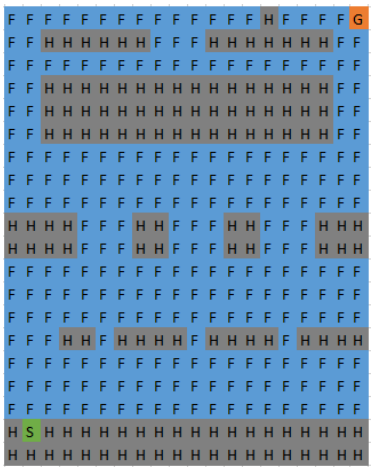
Cliff Walking:

This problem is supposed to be the easier of the 2 problems considered. In this problem, we are looking at a 4X12 grid where the bottom is the edge of a cliff and the challenge is to move from Start (Left Bottom) to the Goal (Right Bottom). The agent can move up, down, left or right. If the agent steps in to the cliff, the agent falls down the cliff and the game ends. It is also windy, so the agents moves may not land in the intended locations and can be swayed in to landing in neighboring locations. This problem has 48 states and 4 actions. Following is the diagram of the environment:



Frozen Lake:

This problem is the harder one. We have a 20X20 grid of the following structure. The challenge is to move from Starting Point to the Goal. This is a frozen lake, so when the agent moves, they might not make it to the intended goal and may land sideways. Also, there are holes in the grid, where the agent can fall into that ends the game. Agent can move up, down left or right. This problem has 400 states and 4 actions.



Part 2: The Reinforcement Algorithms

A Markov Decision Process (MDP) is an easy framework to model a fully observable environment, where an agent takes an action and the outcome of the action is, as in the real world, subject to some randomness. The most important part here is *The Markov Property*, that is "The future is independent of the past given the present". An MDP contains:

a. A set of possible states in the environment, S
b. A set of possible actions, a
c. A reward function, R(S,a)
d. A transition model T(S,a,S'), that represents probability of landing in state S', given that we are in state S and take action a.

In MDP, we try to find the optimal policy π* which maximizes the total reward.

Value Iteration:

The idea behind Value Iteration is to find the Utility of each state. Utility of a state is defined as the total reward if we are in the state and take optimal actions from that point on. Since we know only the immediate reward given by the reward function, we will have to find a way to compute the Utility of each state.

To solve this, we follow the following algorithm:

1. Start with an arbitrary utility for each state.
2. For each state, update utility based on utilities of neighbors.
3. For each state, calculate new utility as follows:

$$\widehat{U}(s)_{t+1} = R(s) + \gamma max_a \sum_{s'} T(s,a,s')\widehat{U}(s')_t$$

4. Repeat step 2 until convergence.

Policy Iteration:

In practice, we actually need to find the optimal policy. So rather than calculating utilities and converging, in policy iteration method, we iterate over policy and converge when policy does not change. This will reduce the number of iterations. The policy iteration algorithm is as follows:

1. Start with an arbitrary policy
2. Compute utility if each state
3. Update utility based on neighboring states
4. Select new optimal actions for each state
5. Repeat steps 2-5 till optimal actions converge.

Q Learning:

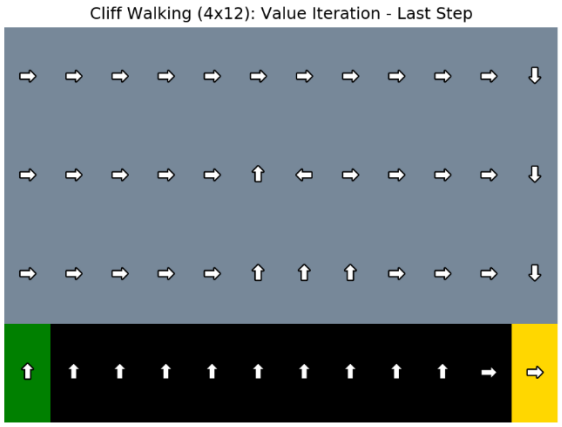
Q Learning refers to a family of model free reinforcement learning algorithms. In this, we try to build a Q table, that effectively gives the optimum action to take given a state. This Q table is built based on experience. When an action is taken from a state, we receive a reward which is then used to update the Q table for that state. In effect, Q Learning is similar to Value Iteration, in that it captures the value of an action a from state s assuming that we take optimal actions from that point on. But the difference is that Q Learning does not know the transition model and so tries to learn the optimum policy by experience.

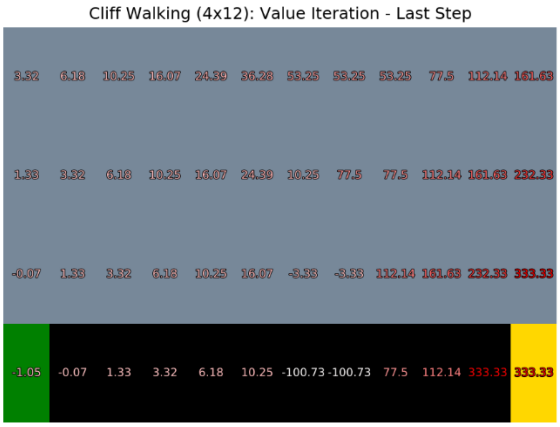
Part 3: Performance of Value and Policy Iteration

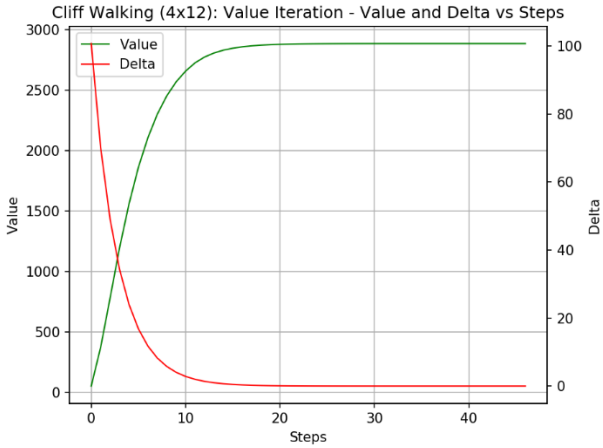
Cliff Walking:

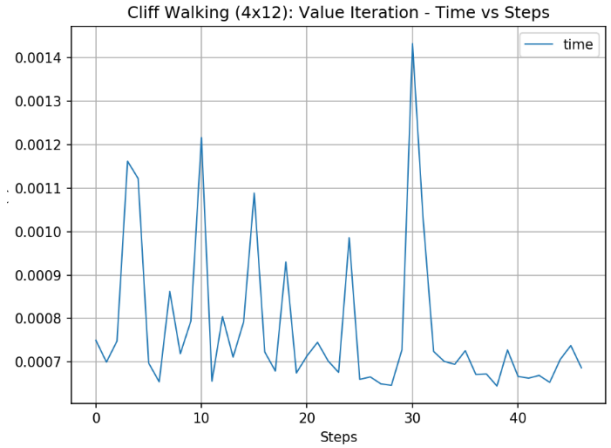
Following charts show the results obtained for the Cliff Walking problem for Value Iteration and Policy Iteration methods. I have shown the optimal policy along with Utility values computed, time taken and the total rewards obtained.

<u>Value Iteration:</u>

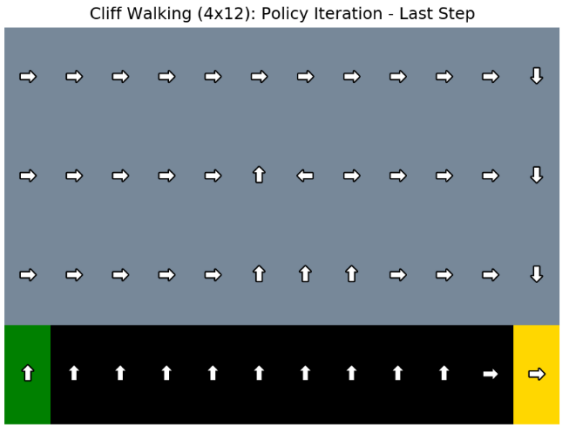


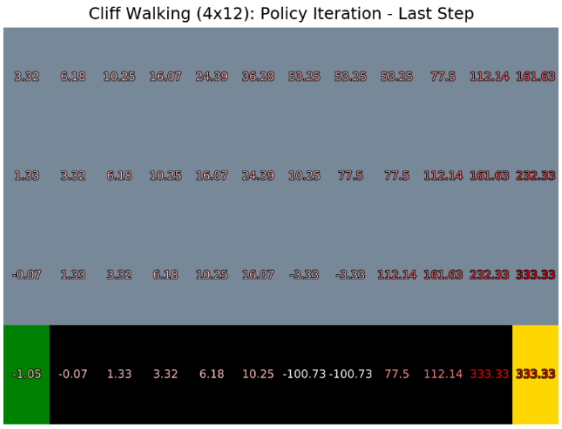


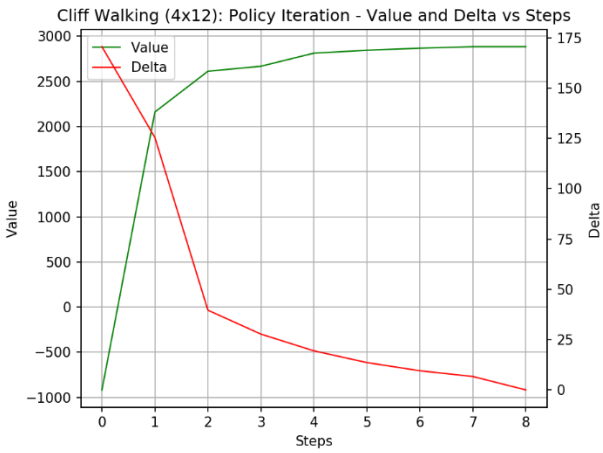


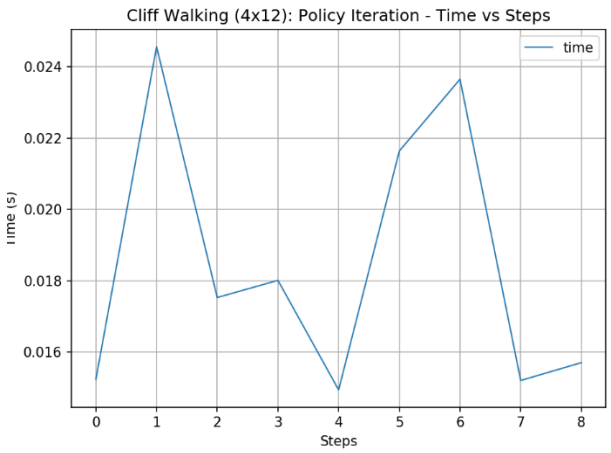


Policy Iteration:









Points of interest:

1. The optimum policy and the utility values are identical.
2. Policy Iteration took less number of iterations to convergence. I found that Policy Iteration took an average of 3 iterations while Value Iterations took up to 45 iteration and the number

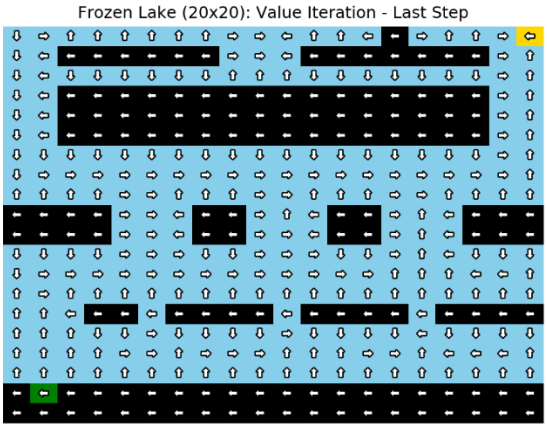
iterations increased with learning rate. This was as expected since policy will converge faster than individual utility values.

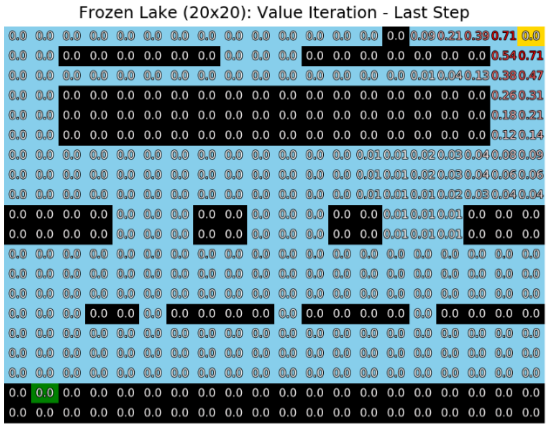
3. The actual time taken for Value iteration was comparatively same as the time taken for Policy iteration despite the far fewer iterations required. This was expected also, since the over head of calculating the policy for each iteration took more processing power than quickly updating utilities in each iteration. On average, Policy Iteration took 1.60 seconds to converge. Whereas, Value Iteration took about 1.78 seconds to converge.

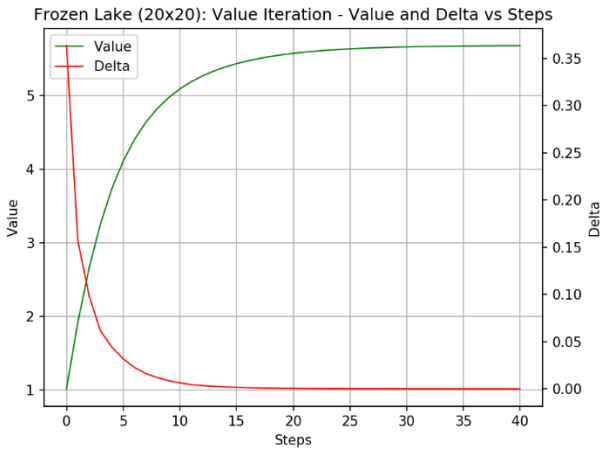
Next, we will see the results obtained for the Frozen Lake problem which has more states and is more complex than this problem.

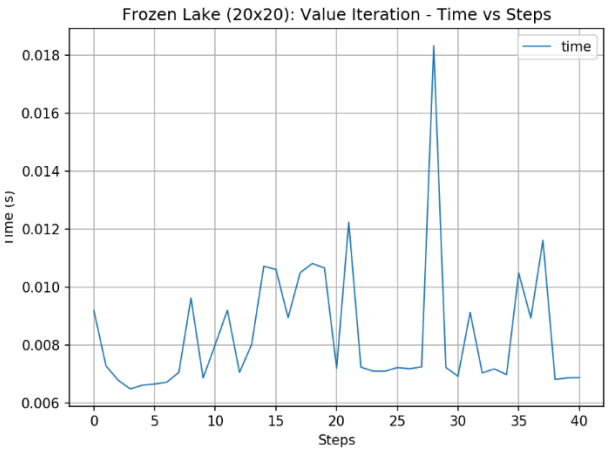
Frozen Lake:

Value Iteration:

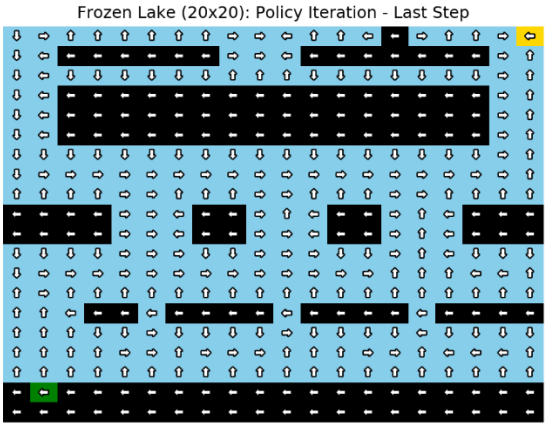


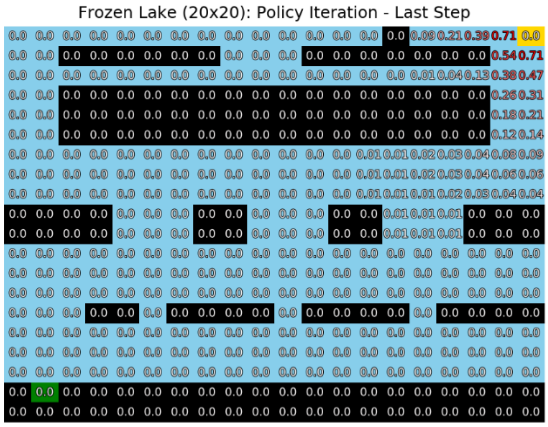


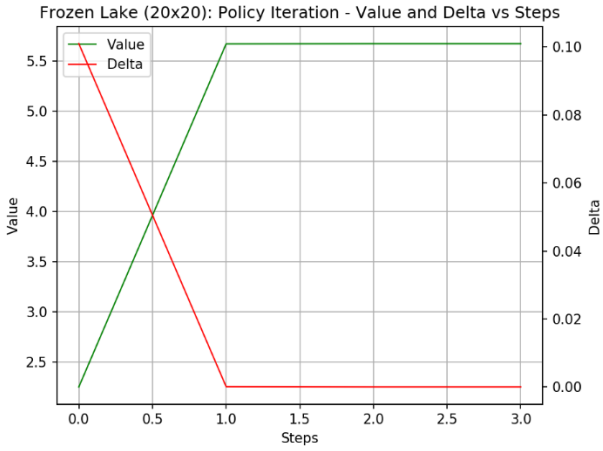


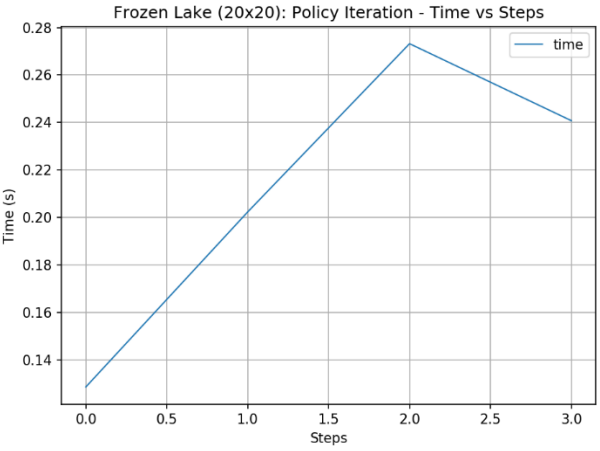


Policy Iteration:









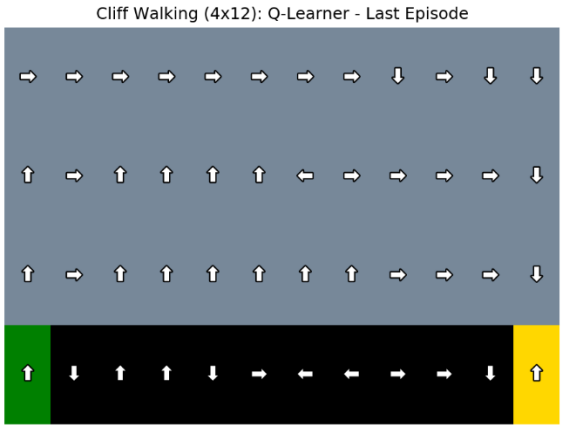
Points of interest:

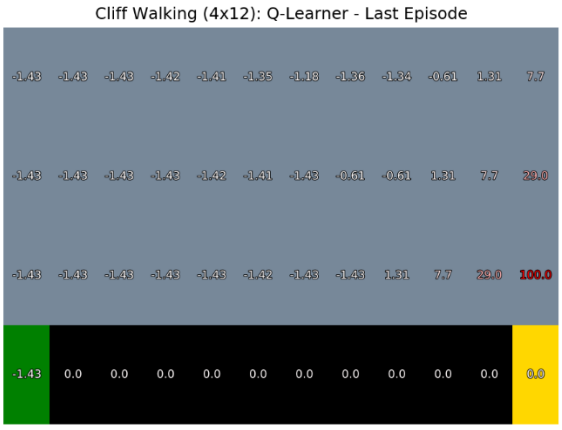
1. As the previous problem: the optimum policy and the utility values are identical. (There were difference in other policies)
2. As expected, Policy Iteration took less number of iterations to convergence. I found that Policy Iteration took an average of 3 iterations while Value Iterations took up to 40 iterations and the number iterations increased with learning rate.
3. The actual time taken for Value iteration was less than the time taken for Policy iteration. This difference was even more significant in this harder problem. On average, Policy Iteration took 4.15 seconds to converge. Whereas, Value Iteration took about 3.93 seconds to converge.

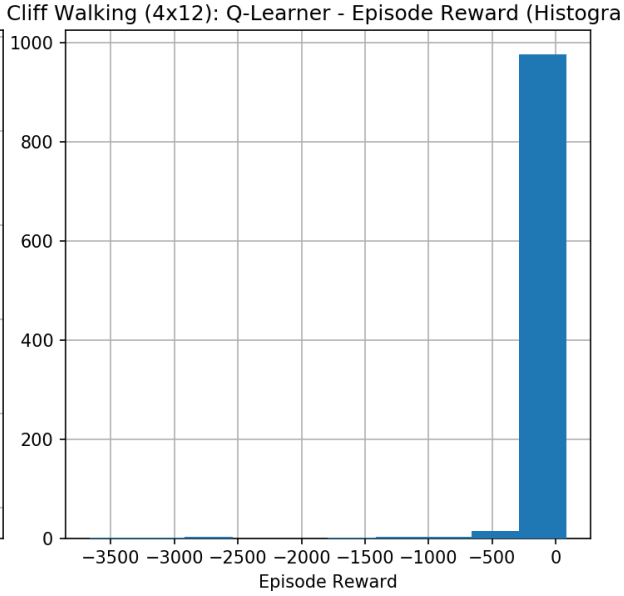
Part 4: Performance of Q Learning Algorithm

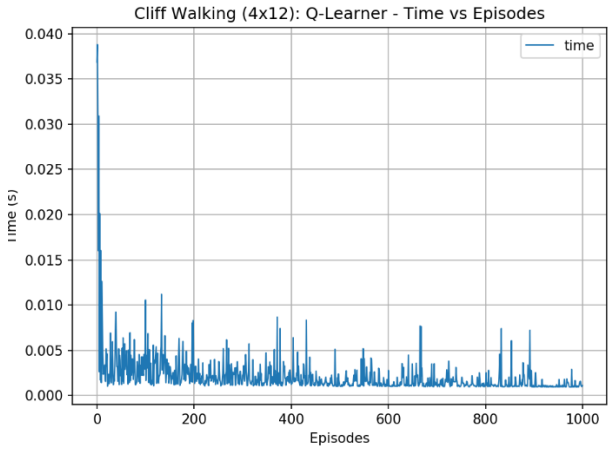
Following are the results obtained when running Q Learning algorithm for both problems. I have listed the optimum policy obtained, rewards obtained and time taken charts. I tried different params for Q Learning, by varying alpha, Initial Q, epsilon, epsilon decay and discount factors.

Cliff Walking:





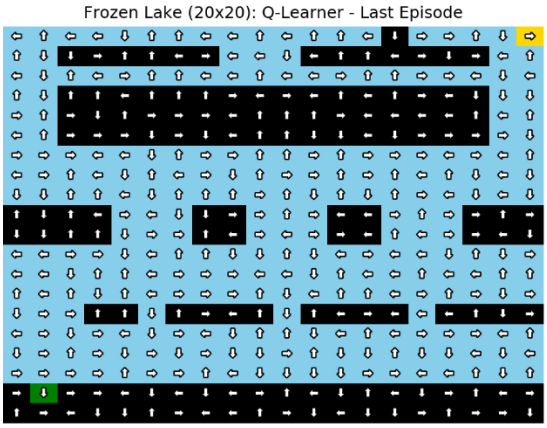


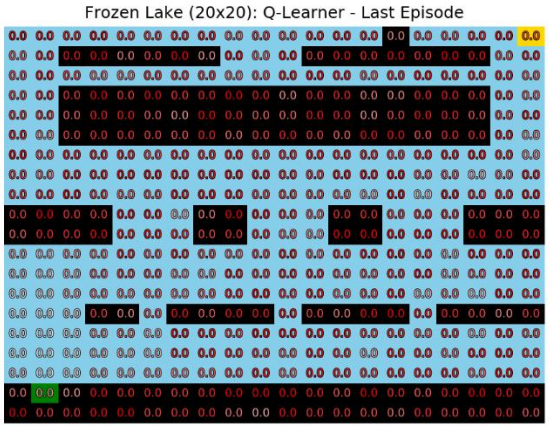


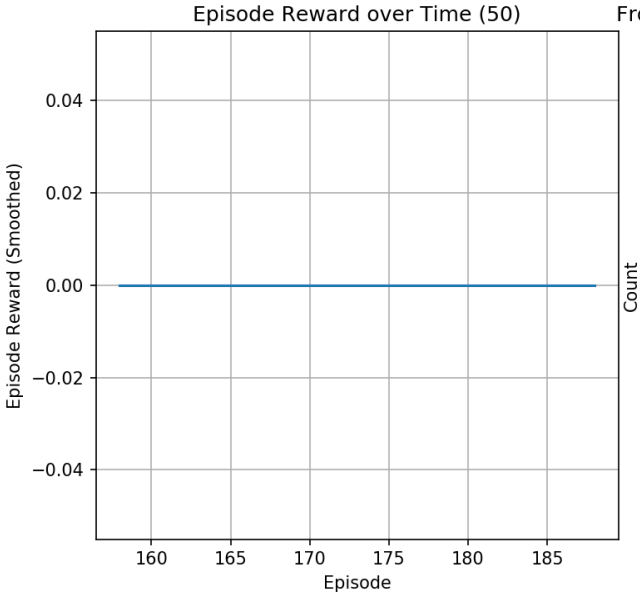
Points of Interest:

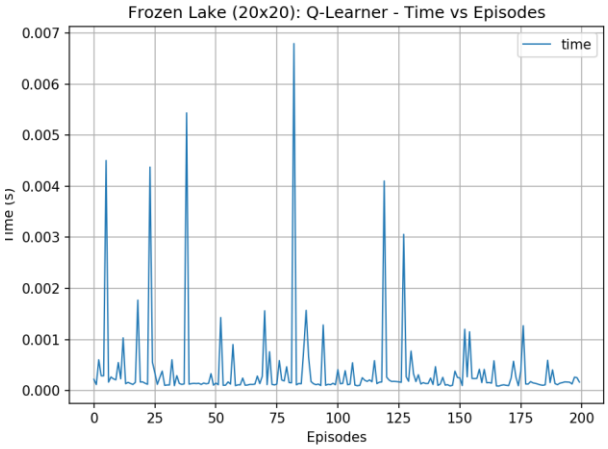
1. The optimum policy obtained was different from the one obtained from PI and VI. But the policy obtained was close to optimum.
2. The time taken to converge was about 4 seconds in this case, which is more than double the time required for PI and VI.
3. The best performance was obtained with following parameters: {"alpha": 0.9, "q_init": "random", "epsilon": 0.5, "epsilon_decay": 0.0001, "discount_factor": 0.3}

Frozen Lake:









Points of Interest:

1. Q Learning did not perform well at all compared to PI and VI. This was expected as PI and VI knew the transition model whereas Q Learning had to learn the model based on experience.
2. The time taken to converge was about 4 seconds in this case, which is more than double the time required for PI and VI. But this was same as the simpler problem. So for Q Learning complexity does not seem to impact performance.