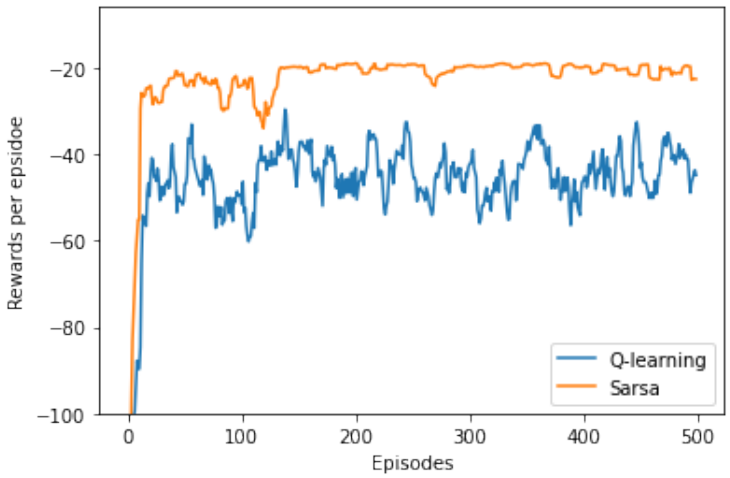
Assignment 3

March 7, 2022

1. **Recreate the figure below in the text using the cliffwalking task in OpenGym.**

   I implement the Q-learning agent and Sarsa agent using learning rate = 0.3, $\epsilon = 0.1$ and $\gamma = 1$.

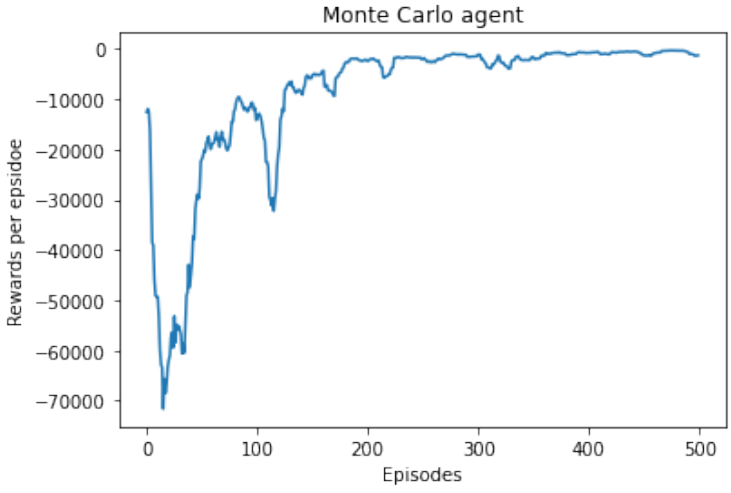
   The main thing I struggle with is understanding the meaning of "smooth", I tried different hyper-parameter to make my graph closer to the graph in PDF, which takes me lots of time.



2. **Add on-policy Monte Carlo and plot it again. How does it compare?**

   I implement the first visit Monte Carlo agent with ϵ-greedy policy and apply ϵ decay on it. The discount factor is 0.9 and the ϵ decay rate is 1 - 4e-3.

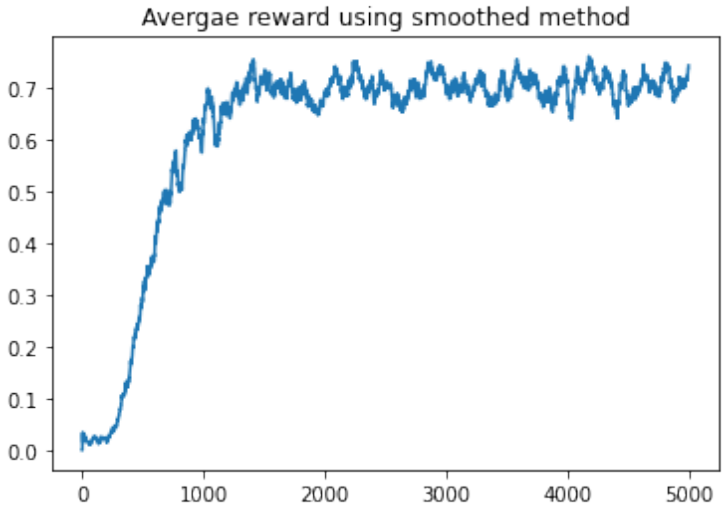
   I tried to use same hyper-parameter as above but it seems that the Monte Carlo agent will be very inefficiency at the first few episodes. The best average return per episode would be around -2000 in a single run. Therefore I applied ϵ decay on it to see if it can gain a better performance. Even if I tried lost of hyper-parameter such as the learning rate and the decay rate. The Monte Carlo agent converge much slower than Q-learning agent and Sarsa agent. And it cannot get as many rewards as Q-learning agent and Sarsa agent. I think if we increase the episodes number and let Monte Carlo agent did more on exploration, then it can better calculate the average value of each state.



3. **Implement one of the following to learn the Frozen Lake task in OpenGym. Don't worry if you can't solve it right away. Frozen Lake is a "toy" task, but solving it without changing the reward function is non-trivial.**

   I implemented an expected Sarsa agent with discount factor = 0.98, $\epsilon = 0.9$, learning rate = 0.3 and apply ϵ decay and learning rate decay on it.

   The main challenge I met here is how to get a better performance. Firstly i used a constant ϵ and learning rate. The return was very low. Even if the question told me solving this is non-trivial, but I'm still not sure if my implement is correct since the result of using constant ϵ seems not converge. Then I tried fine-tune the model to get a better result and finally I got the result below with average return converge to around 0.7.

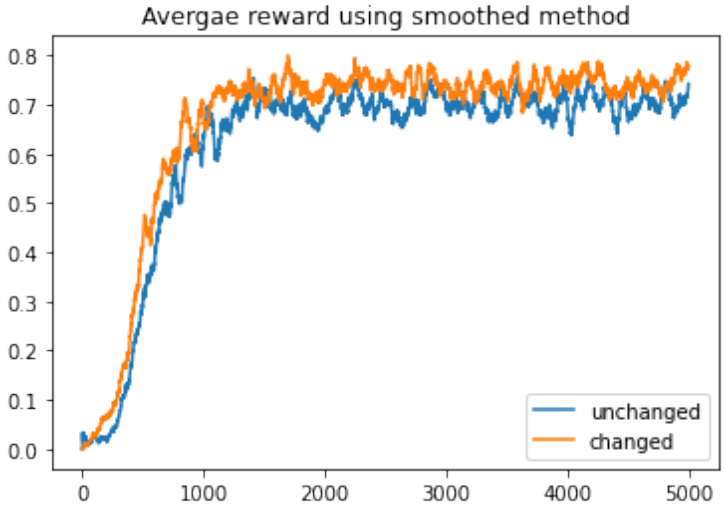


4. **What is the reward function for the Frozen Lake task? How does this affect your agent's ability to learn? How might you make things easier for your agent? Modify the reward your agent receives from its actions. Can you achieve better performance?**

   The reward function for Frozen Lake task is reaching the goal = 1 and otherwise = 0. This reward does not punish on step in a hole. Therefore I changed the reward function to reaching the goal = 100 step in a hole = -100 and otherwise = 0. This actually have a better performance.

   I have tried different reward function such as reaching the goal = 100 step in a hole = -100 and otherwise = -1. However, this does not have a better performance. I'm guessing this is because the state space is low and we have 200 steps to reach the goal state. Therefore punish on each step is not necessary.

   Another challenge is since the reward function was changed, it is unfair to compare them as total reward gained. So I used the new reward function to update the agent and use the old reward function as returns to generate the graph.



5. Implement tabular Q-Learning to solve the Mountain Car task in OpenGym.

   I implemented a tabular Q-Learning agent with discount factor = 0.98, $\epsilon = 0.9$, learning rate = 0.3 and apply ϵ decay and learning rate decay on it.

   The first graph is returns without smooth method and the yellow line is the average returns every 50 episodes.

   The second graph is rewards using some smoothed method.

   The main challenge I met here is to calculate the discrete space and how to transverse a observation state to my state. The second thing is how to improve the performance, at the very beginning I used discretize each state to 10 states and run 2000 episode, but the average return is around -190 and it was very unstable. Then I thought it was because the way I transverse the states was wrong. Finally I tried 10000 episode then I found it's just need time to converge, but it took me lots of time to realize it.