Mountain Car

Batuhan ERDEN

Introduction

In this assignment, I have implemented One-step semi-gradient Double Q-Learning, N-step semi-gradient SARSA algorithms and tried to implement N-step semi-gradient Expected SARSA. I have used to the information in Sutton's Book. Lastly, I tested my algorithms' performances using the graphs plotted using **mathplotlib**.

One-step semi-gradient Double Q-Learning vs. N-step semi-gradient SARSA

To correctly implement these algorithms, I firstly decided what my layers, learning rate, discount factor and other parameters would be. I used batch training to train my model. Double Q-Learning is an off-policy and SARSA is an on-policy TD control algorithm. Both algorithms choose which action to take by state S using the policy derived from the models trained (***epsilon-greedy***). Double Q-Learning has an advantage of using two different policies while SARSA has an advantage of obtaining the n-step return. Since I have a one-page limitation on the report, I will describe the results now.

As can be seen in the *Figure 1* and *Figure 2*, number of episodes is 100. In the *Figure 1*, it is obvious to see that Double Q-Learning has done its job and learned through time. As the number of episodes increased, the algorithm found the solution faster. However, in *Figure 2*, the result of the SARSA should have been better than Double Q-Learning. SARSA has also learned well in time but when I looked down the graphs in *Richard Sutton's Book*, the output of SARSA looked more like a constantly decreasing graph. It could be because that graph is based on the data of 100 runs averaged while mine only depends on a single run. Yet, I still think that there is a minor bug in my code that prevents my algorithm to learn even better.

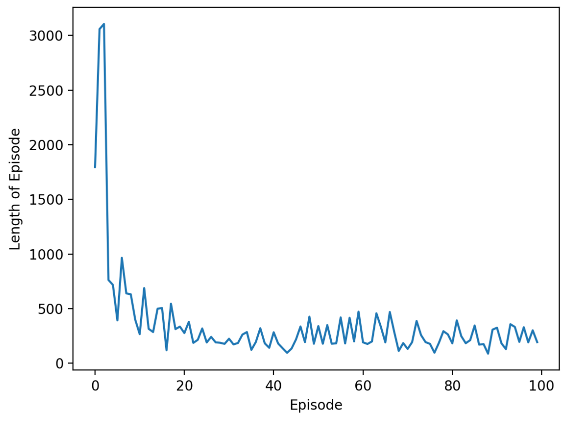

Figure 1: One-step semi-gradient Double Q-Learning

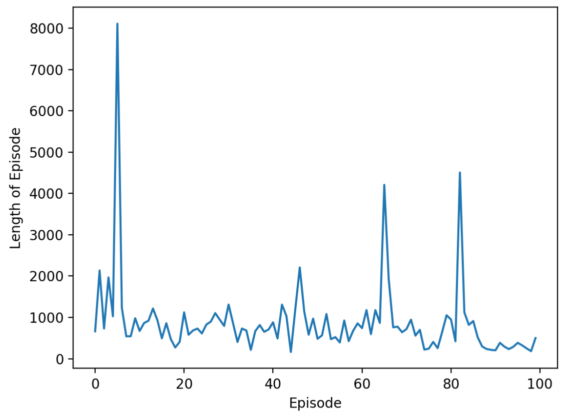

Figure 2: N-step semi-gradient SARSA

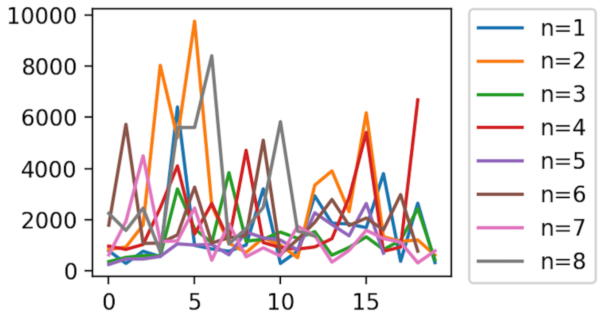

Figure 3: N-step semi-gradient SARSA with 8 different n values

I also checked the results of SARSA with 8 different n values. The results can be seen in the *Figure 3*. I realized that it worked the best for **n=7** and some episodes took more time in some n values. The graph demonstrates some defects that might be caused by the bugs in the code or the wrong parameterization in building the model. Also, this graph does not correlate with the graph on *Richard Sutton's* book that increases the possibility of implementation bug in the algorithm that I have created. Yet, the results were encouraging because both algorithms learned well in the episodes.

Conclusion

Overall, this assignment has been a great assignment for me to understand the logic between Deep Neural Network and Reinforcement Learning. I had some difficulties in implementing N-step semi-gradient Expected SARSA. I have done some programming but it was not working as I expected. So, I have decided not to include it in my submission report. However, the code can be found in my submission folder.