AE 598: RL – HW1 Report

Mahshid Mansouri (mm64)

***Abstract*—The goal of this assignment was to implement five different RL algorithms, namely policy iteration (PI), value iteration (VI), SARSA, Q-Learning, and TD(0) for two different environments in a tabular setting: 1) A gridworld, and 2) a discrete inverted pendulum. This report demonstrates the results, discusses the observations from the results, and provides some details regarding the algorithms implementations (e.g., the choice of the hyperparameters, why one algorithm might be performing better, etc.).**

I. Results

A. Gridworld problem

- The learning curve plots for all algorithms are as below:

Policy iteration (PI)

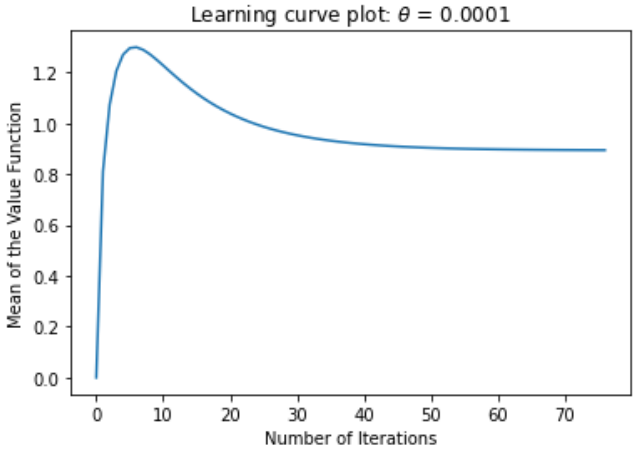

Fig.1. Learning curve plot for PI algorithm.

Value iteration (VI)

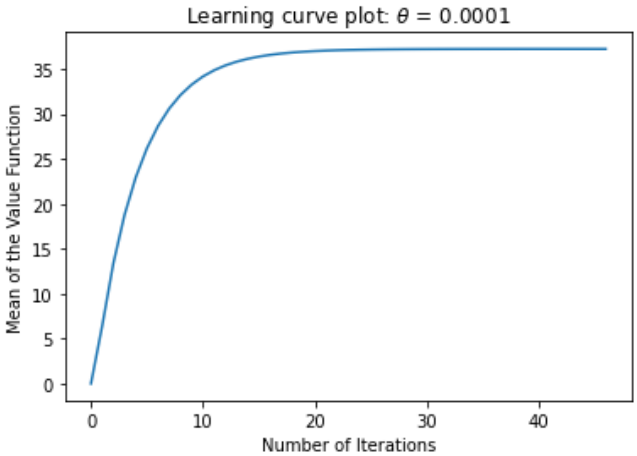

Fig.2. Learning curve plot for VI algorithm.

SARSA

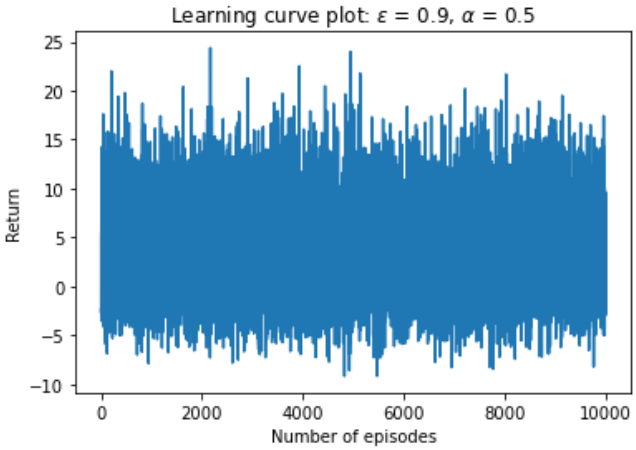

Fig.3. Learning curve plot for SARSA algorithm.

Q-Learning

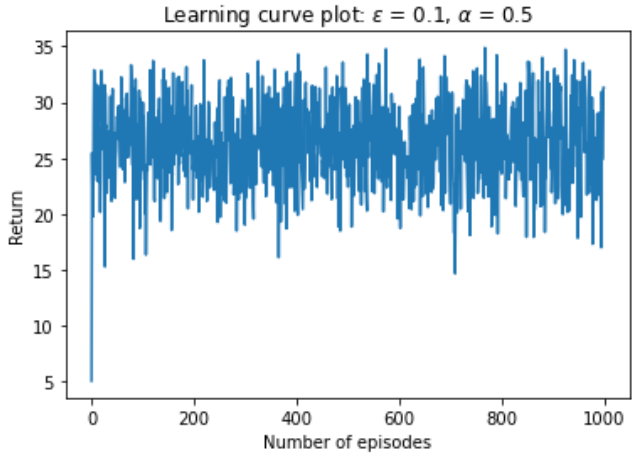

Fig.4. Learning curve plot for Q-Learning algorithm.

- The plots of the learning curve for several different values of ϵ for SARSA and Q-learning are as next page:

SARSA

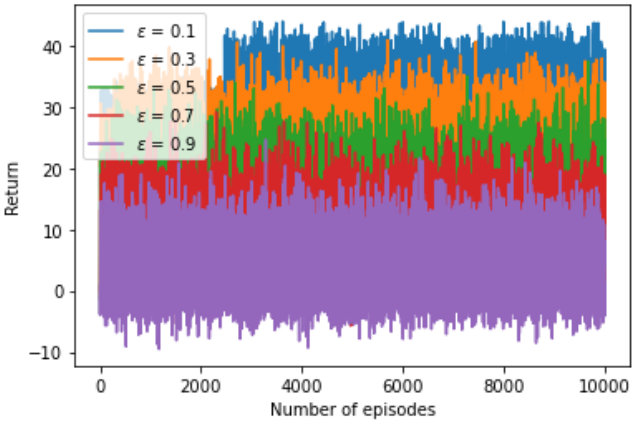

Fig.5. Learning curve plot for SARSA algorithm for varying ϵ and α = 0.5.

Q-Learning

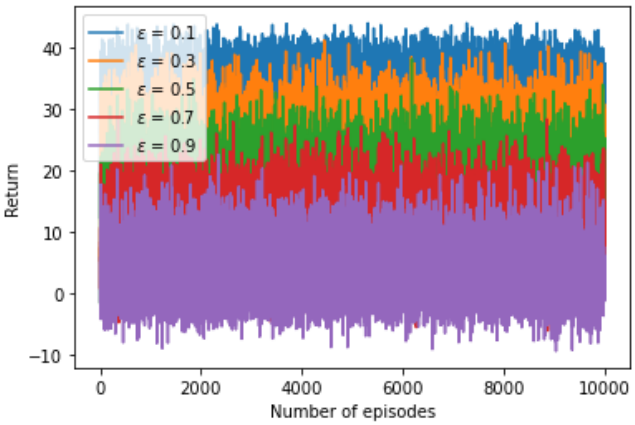

Fig.6. Learning curve plot for Q-Learning algorithm for varying ϵ and α = 0.5.

- The plots of the learning curve for several different values of α for SARSA and Q-learning are as below:

SARSA

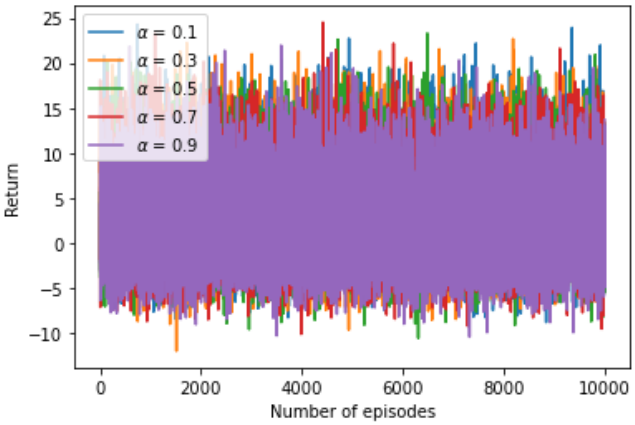

Fig.7. Learning curve plot for SARSA algorithm for varying α and ϵ = 0.9.

Q-Learning

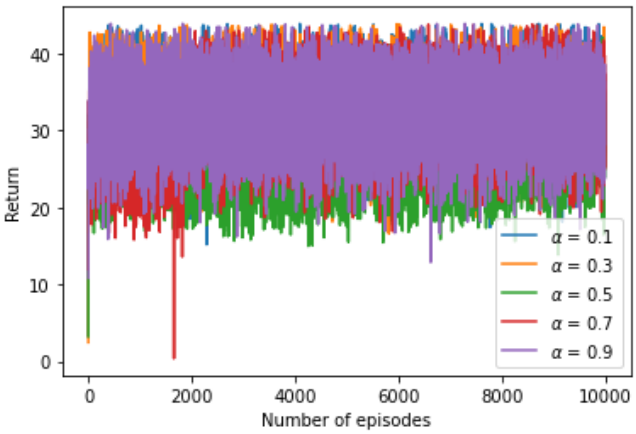

Fig.8. Learning curve plot for Q-Learning algorithm for varying α ϵ = 0.1.

- The plots of an example trajectory for each trained agent for all algorithms are as below:

Policy iteration (PI)

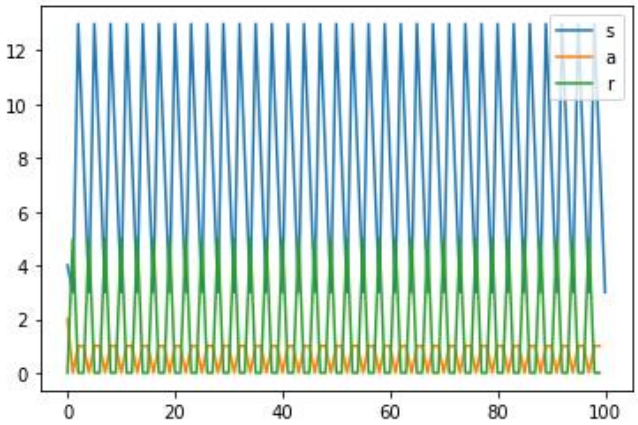

Fig.9. Example trajectory plot for PI algorithm. θ = 0.0001

Value iteration (VI)

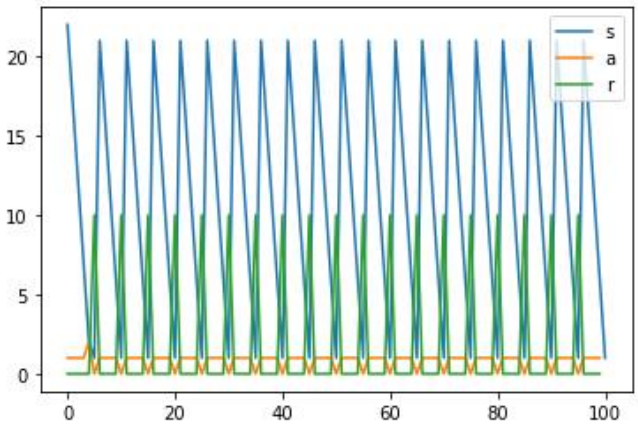

Fig.10. Example trajectory plot for VI algorithm. θ = 0.0001

SARSA

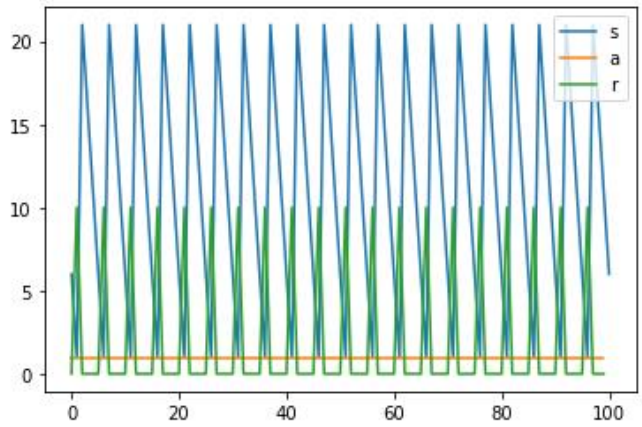

Fig.11. Example trajectory plot for SARSA algorithm. ϵ = 0.9 and α = 0.5

Q-Learning

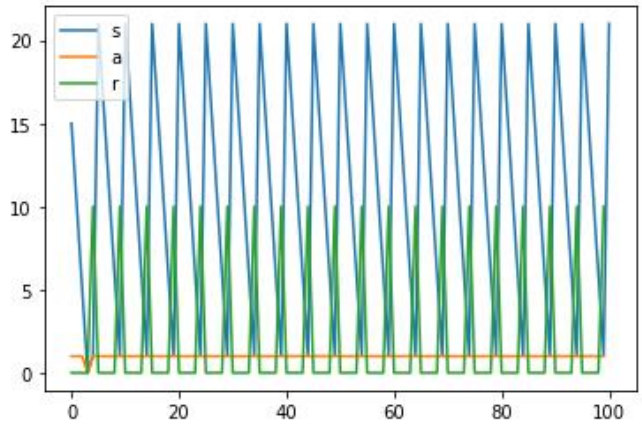

Fig.12. Example trajectory plot for Q-Learning algorithm. ϵ = 0.1 and α = 0.5

- The plots of the policy that corresponds to each trained agent for all algorithms are:

Policy iteration (PI)

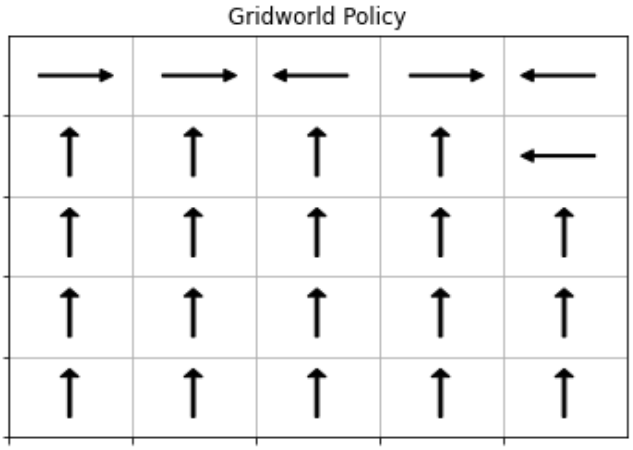

Fig.13. Learned policy for the PI algorithm.

Value Iteration (VI)

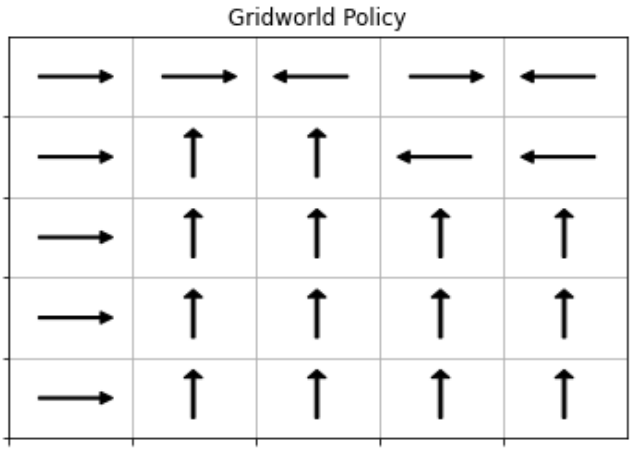

Fig.14. Learned policy for the VI algorithm.

SARSA

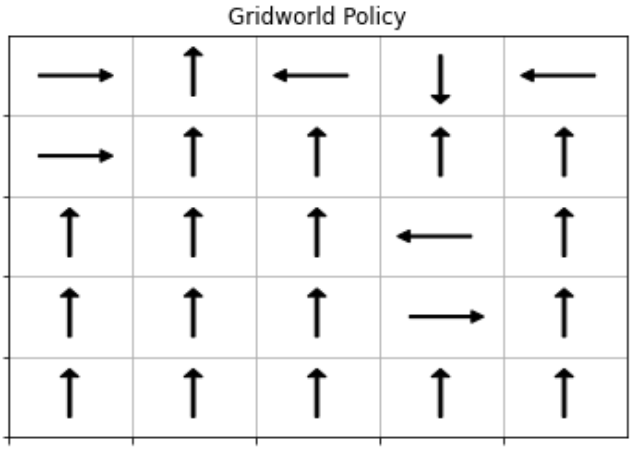

Fig.15. Learned policy for the SARSA algorithm.

Q-Learning

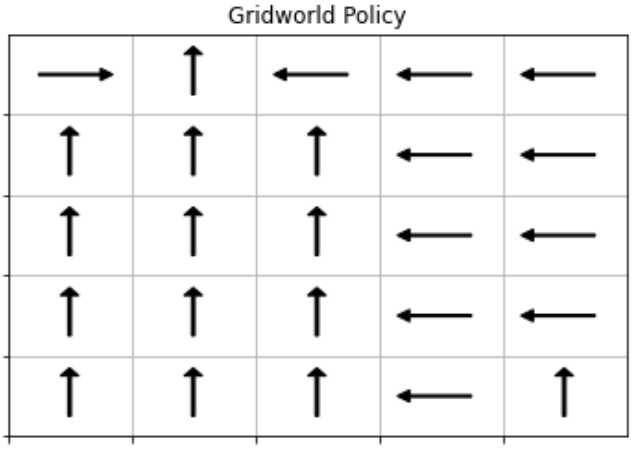

Fig.16. Learned policy for the Q-Learning algorithm.

- The plots of the state-value function that corresponds to each trained agent for algorithms are as below:

Policy iteration (PI)

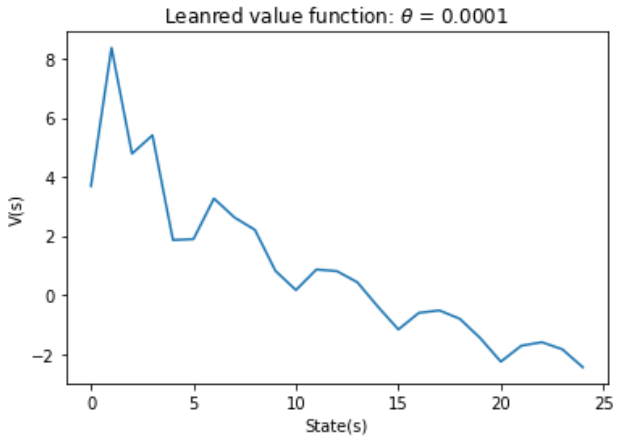

Fig.17. Learned state value function for the PI algorithm.

Value iteration (VI)

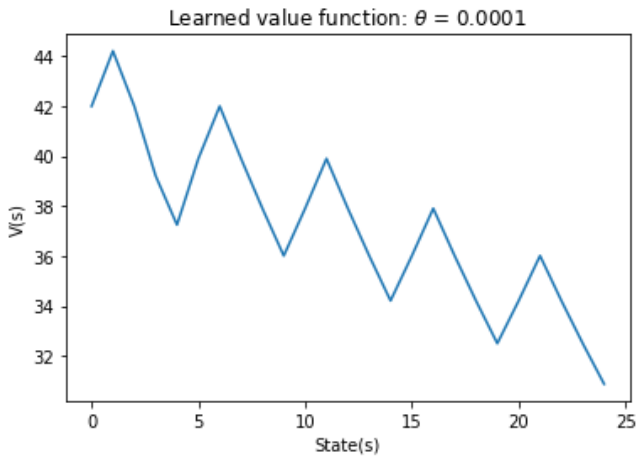

Fig.18. Learned state value function for the VI algorithm.

SARSA

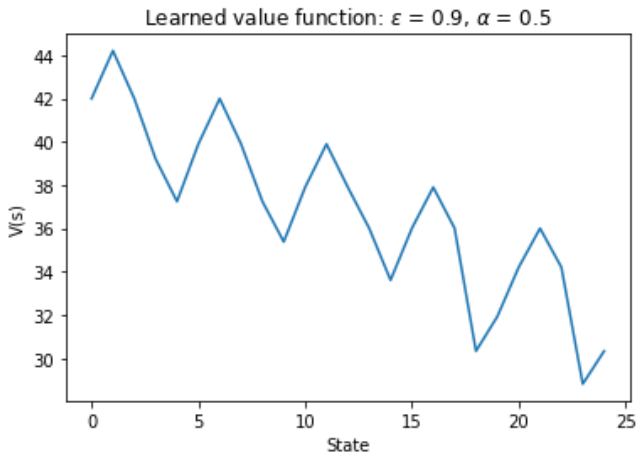

Fig.19. Learned state value function for the SARSA algorithm.

Q-Learning

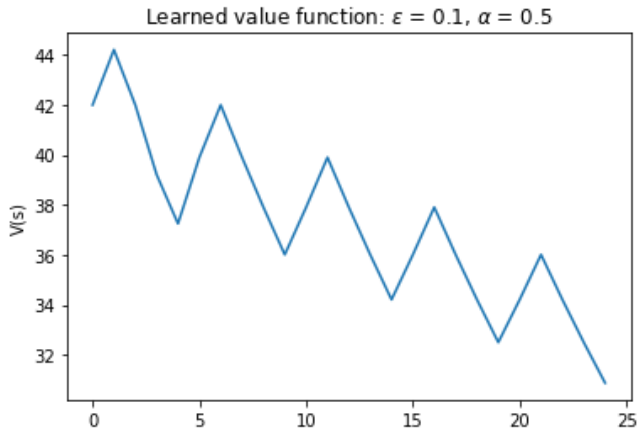

Fig.20. Learned state value function for the Q-Learning algorithm.

B. Discret pendulum problem

The learning curve plots for SARSA and Q-Learning algorithms are as below. Note that to speed up the computation, I changed the default n_theta, n_theta_dot, and n_tau from (31, 31, 31) to (20, 20, 20). Additionally, the return plots have been plotted on log scale for better visualization purposes for the inverted pendulum.

SARSA

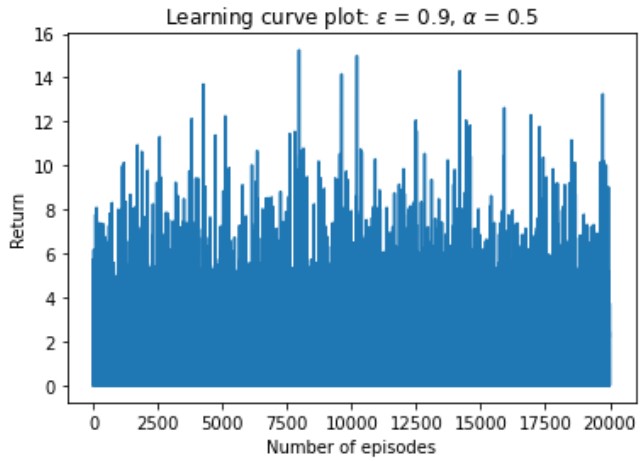

Fig.21. Learning curve plot for SARSA algorithm.

Q-Learning

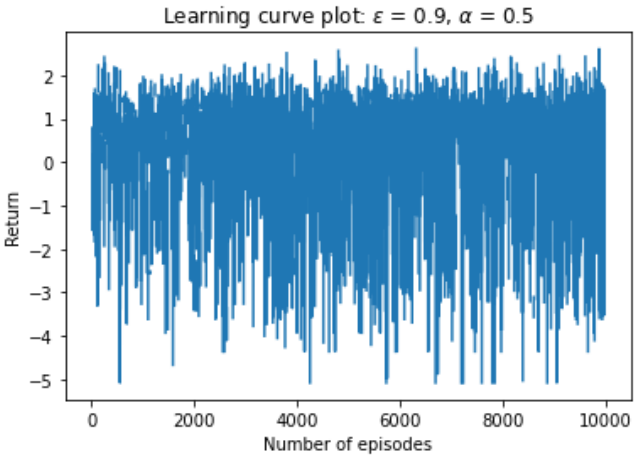

Fig.22. Learning curve plot for SARSA algorithm.

- The plots of the learning curve for several different values of ϵ are as below. Note that due to the computational power limitations, the algorithms were only run for 3000 episodes for plotting purposes.

SARSA

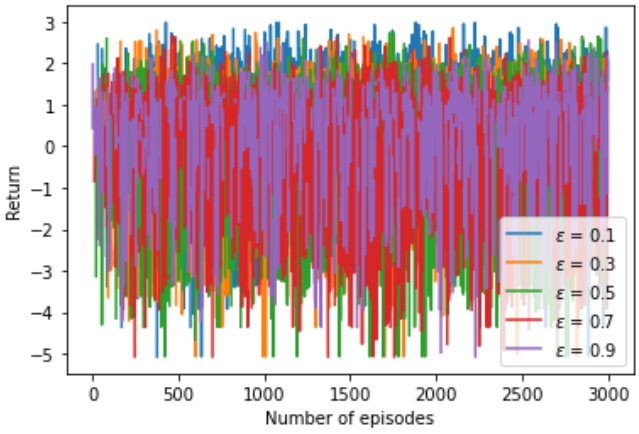

Figure 23. Learning curve plot for SARSA for varying ϵ and α = 0.5

Q-Learning

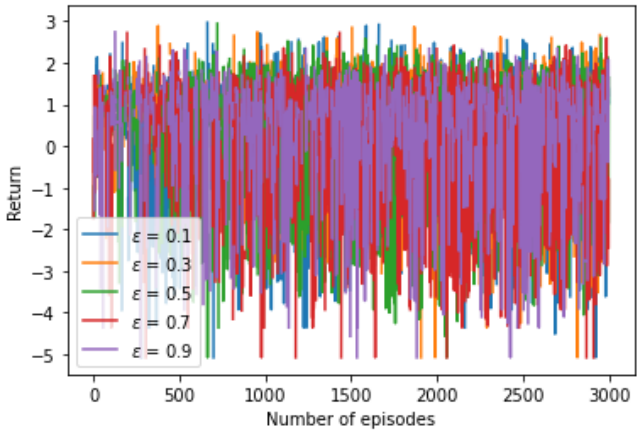

Fig.24. Learning curve plot for Q-Learning algorithm for varying ϵ and α = 0.5.

- The plots of the learning curve for several different values of α are as below:

SARSA

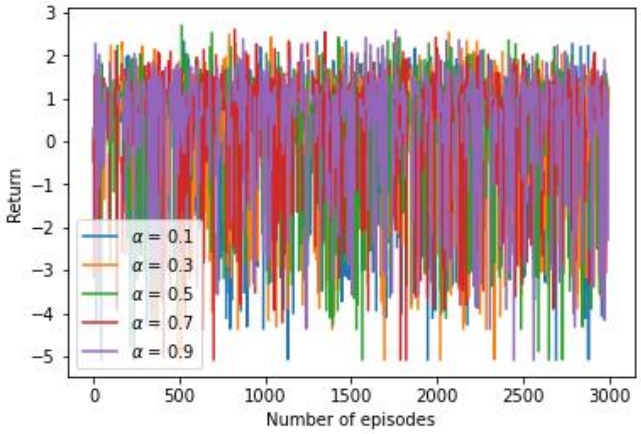

Fig.25. Learning curve plot for SARSA algorithm for varying α and ϵ = 0.9.

Q-Learning

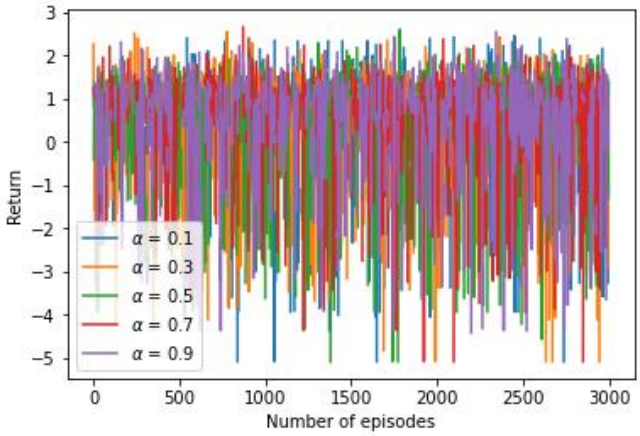

Fig.26. Learning curve plot for Q-Learning algorithm for varying α and ϵ = 0.9.

- The plots of an example trajectory for each trained agent are as below:

SARSA

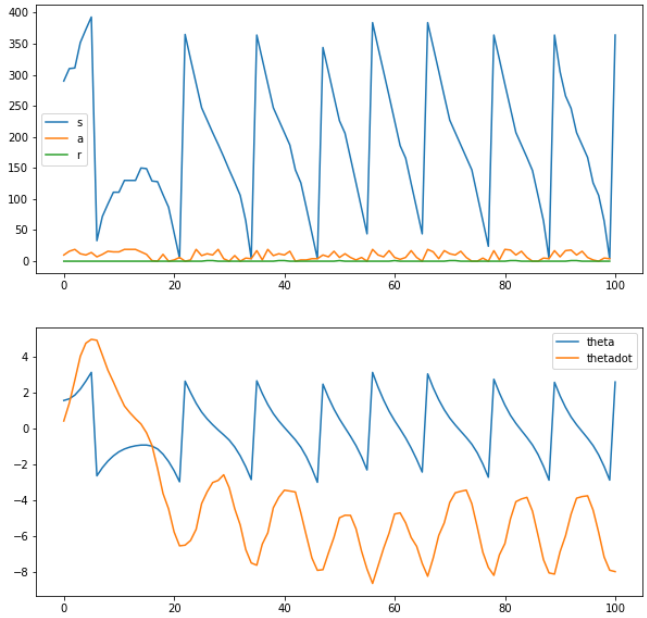

Fig. 27. Example trajectory plot for SARSA algorithm

Q-Learning

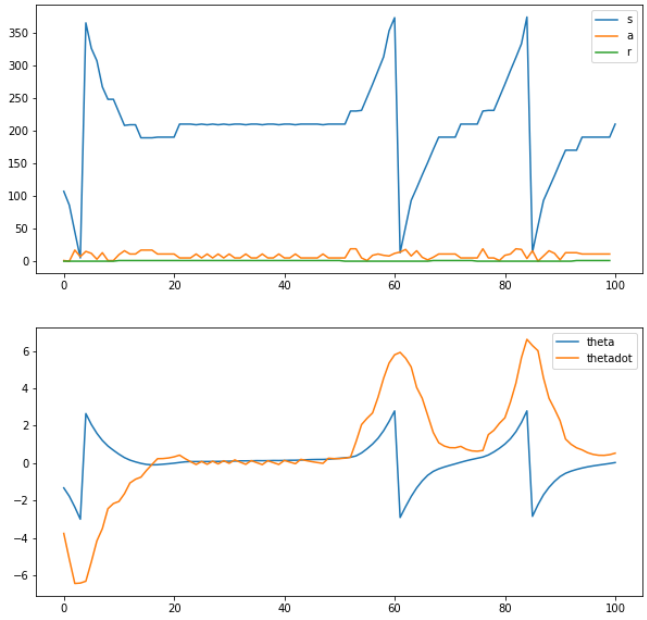

Fig. 28. Example trajectory plot for Q-Learning algorithm

- The plots of the policy that corresponds to each trained agent for all algorithms:

Note that for the optimal policy plot for the pendulum problem, it was hard to visualize the optimal policy and therefore, we can use the trajectory plots (Fig. 27-28) to see how the action and state values are changing over time.

- The plots of the state-value function that corresponds to each trained agent for algorithms are as below:

SARSA

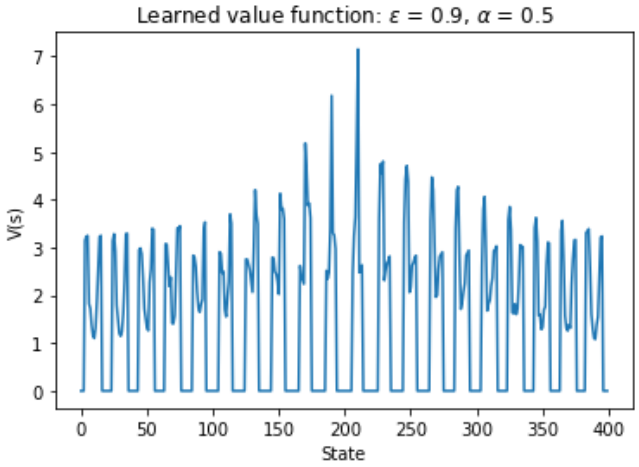

Fig.29. Learned state value function for the SARSA algorithm.

Q-Learning

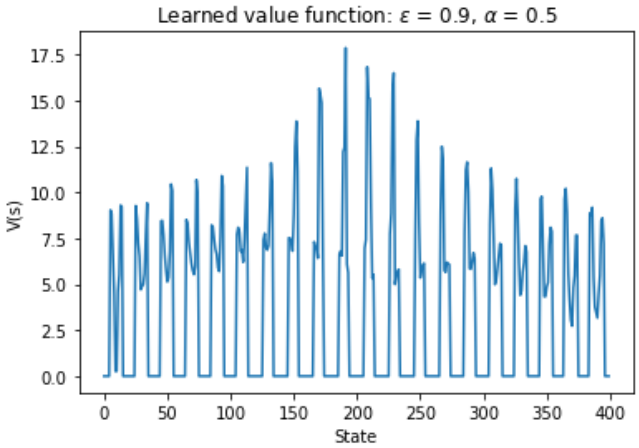

Fig.30. Learned state value function for the Q-Learning algorithm.

II. Discussion

A. *Gridworld problem*

1) Learning curve plots

The learning curve plots for PI and VI algorithms have leveled off after enough training (Fig. 1-2). For the PI algorithm, the values of the mean value function seem to be pretty low compared to VI, which I'm not sure what's causing the issue. I expect to see a similar curve as VI for PI. For SARSA and Q-Learning, the final value of the expected return stays around a fixed value with slight oscillations around the fixed point, meaning that the algorithms have converged to their optimal values (Fig. 3-4).

2) Learning curve plots for several different values of ϵ for SARSA and Q-learning

Higher values of ϵ result in a lower return (Fig. 5-6). This is because when ϵ is increased, the probability of choosing the

next action greedily decreases, meaning that the agent will most likely choose a random action which might not necessarily result in the highest return at the end given that other hyperparameters of the problem are fixed.

3) Learning curve plots for several different values of α for SARSA and Q-learning
Qualitatively speaking, changing α values does not change the expected return significantly, despite what we observed with changes of ϵ (Fig. 7-8). Therefore, it seems like both SARSA, and Q-Learning algorithms are more sensitive to the changes in ϵ compared to α.

4) *Example trajectory plots for each trained agent*
The trajectory plots for all four algorithms show that starting from state 1(i.e., state A), no matter what action the agent takes, it will get to state 21 to achieve the highest reward of r = 10, as expected (Fig. 9-12)

5) Policy plots corresponding to the trained agent
The PI algorithm was able to learn the optimal policy well as stated in [1], however, the predicted action for state = 8 is not true and is probably due to either mistake in the algorithm implementation which I was not able to identify. I also tried to change the value of θ, but that didn't help. Both VI and Q learning algorithms have resulted in a possible optimal policy as stated in Figure 3.5 in [1] (Fig. 13-16). On the other hand, the optimal policy obtained from SARSA is not quite like what we expect to see based on Figure 3.5 in [1] (Fig. 15). Note that during the training of all four algorithms, we tried to compare the learned state value function and stop further training the algorithm once the value functions roughly looked similar. Therefore, this might explain why SARSA did not give us our expected optimal policy. This could be achieved by further training and tweaking of the hyperparameters such as ϵ, α, the number of episodes, and the number of maximum steps. However, due to the time limit, we were not able to show the perfect result in this report.

6) *Learned state value function plots*

As explained in the previous section, the learned state value function from all algorithms except for PI roughly look similar meaning that they were able to train the agent and learn the optimal value function (Fig. 17-20).

B. Discret pendulum problem

1) Learning curve plots
Looking at the mean of the return plot, it seems like the Q-Leaning mean return is closer to 0 whereas SARSA has a higher mean value (Fig. 21-22). It is expected that if the pendulum is perfectly kept upright, it achieves a very small reward of close to zero. As can be seen, the current trained Q-Learning algorithm is doing a better job compared to SARSA, and SARSA needs further tweaking which is out of the time limit and computational power of this work.

2) Learning curve plots for several different values of ϵ
3) Learning curve plots for several different values of α
Compared to the gridworld problem, it seems like the expected return in the inverted pendulum is less sensitive to the choice of α and ϵ, however, the choice of these parameters did have a significant effect on the obtained optimal policy trajectory during the training the algorithms which are not shown in this report (Fig. 23-26).

4) *Example trajectory plots for each trained agent*
We observe that the Q-Learning algorithm was able to stabilize the pendulum in the upright position for only a certain amount of time (around t = 20~50 s) where θ and $\dot{\theta}$ values were close to 0 (Fig.28). However, the results look good enough for the purpose of the class assignment. On the other hand, the current trained SARSA algorithm did worse than the Q-Learning in terms of stabilizing the pendulum, where it was able to balance it for a shorter period of time (Fig. 27). In general, there are a few hyperparameters that can change the performance of SARSA and Q-Learning algorithms, namely the choice of the learning rate α, ϵ, the number of episodes, and the number of iteration steps that we are iterating in each episode. Specifically, the choice of the first three hyperparameters can influence the performance significantly and getting the right values to get reasonable results needs a lot of iterations. Additionally, there's a tradeoff between the algorithm performance, the computational cost, and time limit. Increasing the number of episodes and playing around with the α and ϵ values can help with improving the performance of both algorithms.

General observations and conclusions
Compared to the gridworld problem, it was trickier to get reasonable results from the Q-Learning and SARSA algorithms given the nature of the problem which was model free and the fact that the state-action space size was much larger. In addition, for the pendulum problem itself, it was harder to tune the hyperparameters to get the SARSA algorithm to work compared to Q-Learning. In particular, a greater number of episodes were needed for SARSA.

Additionally, Q-Learning seems to be performing faster compared to SARSA for the pendulum problem. This is because Q-learning directly learns the optimal policy because it maximizes the reward with a greedy action selection strategy. This removes the chance that the agent uses an exploration step from the second step in the update function. However, SARSA uses an exploration step in the second step, because it keeps following the ε-greedy strategy. Therefore, in general, Q-learning will converge faster to an optimal policy than SARSA.

One last suggestion that might help with faster convergence of both algorithms is to re-use the TD(0)-learned value function as an initial guess for V(s) for both SARSA and Q-Learning every certain number of iterations.

REFERENCES

[1] "Reinforcement Learning: An Introduction" by Sutton and Barto (MIT Press, 2018