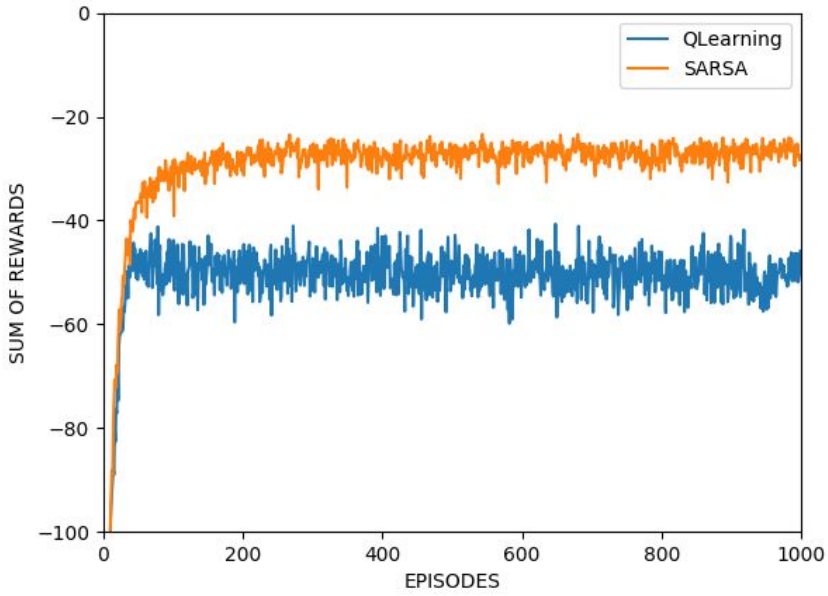
AI Assignment - 4

Case - 1 :

Parameters - ϒ = 1.0, ϵ = 0.1, □ = 0.5

Below are the plots for the sum of rewards in each episode vs episode number (For a total of 1000 episodes). To average out the results, we run the experiment 500 times (Otherwise the results will be having very high variance).



Average results on 500 runs for 1000 episodes

1. So from the above plot, we can clearly see that SARSA performs much better than Q-Learning on average. So, this happens because SARSA explores a lot in the starting whereas Q-Learning finds the shortest path to the Goal, due to which it falls off the cliff a lot of times which decreases the average performance of the Q-Learning.
2. SARSA explores a lot because of the ϵ greedy policy defined for it. SARSA first chooses its action with the help of ϵ greedy policy and then expect the future reward according to the Q-value of this state only. This policy helps SARSA to explore a lot and choose a safe path so that it does not fall off the cliff.

Below is the optimal policy for both methods-

a) Q-Learning

R R R R R D R R R R D D
R D D R D D D R R R R D
R R R R R R R R R R R D
U U U U U U U U U U U U

b) SARSA

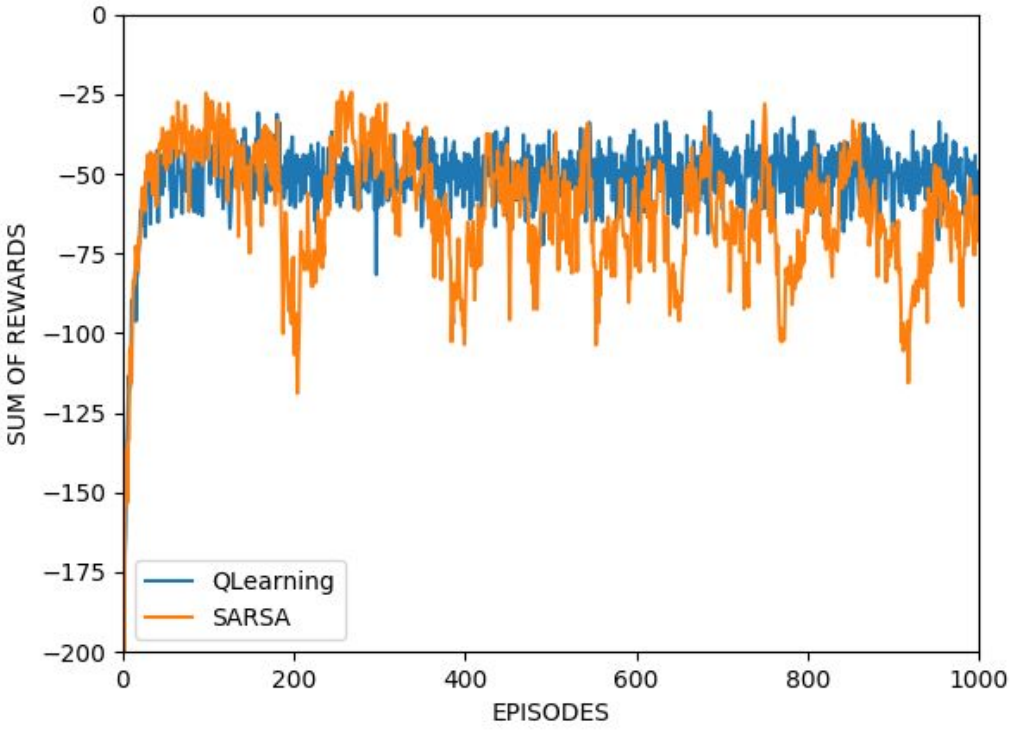
R R R R R R R R R R R D
U U U R R R U U U U R D
U U U U U U U U U U R D
U U U U U U U U U U U U

So basically the major difference between the 2 methods is, Q-Learning chooses the shortest path despite the ϵ greedy policy whereas SARSA explores a little more and finds the more optimal path.
I would recommend using the SARSA algorithm because of its ability to find the best possible path on the average results. The downside of the SARSA algorithm is that it might take longer to find the best policy whereas the Q-Learning algorithm finds its best policy really quickly. If we just want to use the optimal policy then we can use Q-Learning otherwise SARSA is a better choice.

Case - 2 :

Parameters - γ = 0.9, ϵ = 0.1, □ = 0.5

Below are the plots for the sum of rewards in each episode vs episode number (For a total of 1000 episodes). To average out the results, we run the experiment 1000 times (Otherwise the results will be having very high variance).



Average performance for 100 Runs on 1000 episodes

1. If we see the average results then we can clearly see that average performance for Q-Learning is around -50 whereas it is much more unstable for the SARSA algorithm and most of the time result is below the Q-Learning Algorithm.
2. Decreasing the discount factor, γ decreases the pressure on the algorithm to reach the goal state quickly due to which the SARSA algorithm develops directions which make it move in loops and hence we get the unstable results. But for Q-Learning as we are choosing the maximum in the update equation it does not affect the learning procedure Q-Learning.

Below is the optimal policy for both methods-

a) Q-Learning

R R R L R R R R R R D D
R D D D D R R R D R R D
R R R R R R R R R R R D
U U U U U U U U U U U U

b) SARSA

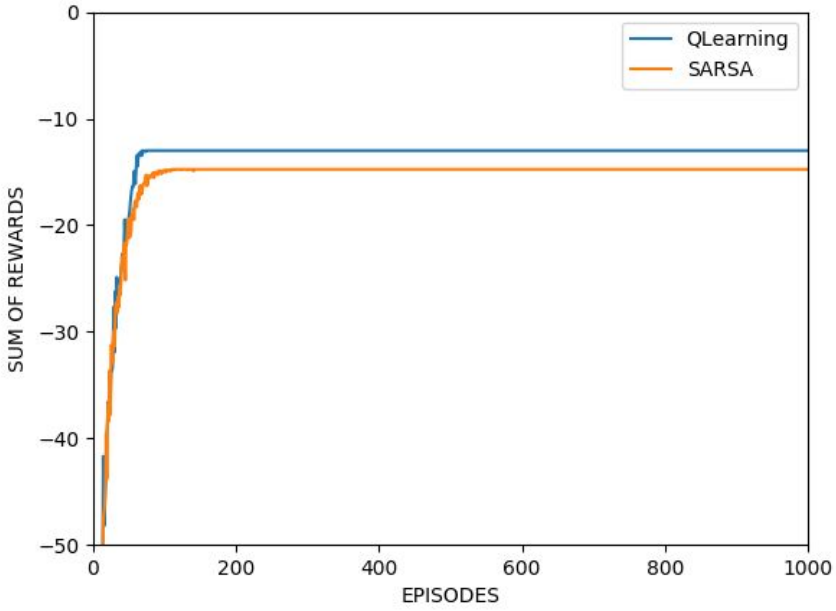
R R R R R R R R R R R D
U U U U U R U R U U D D
U L L U U U U U U U R D
U U U U U U U U U U U U

So the most prominent difference between Q-Learning and SARSA is that in Q-Learning we can assure a less variance in the average performance of the Algorithm but for the SARSA algorithm this approach creates Learning hard due to it explores more than needed and finally has a bad performance.
We also notice that time taken to run the SARSA algorithm is almost twice in this case because of more exploration.
In this case, I would recommend using the Q-Learning algorithm because it assures a better and stable average performance than the SARSA algorithm.

Case - 3 :

Parameters - γ = 0.9, ϵ = 0.1 to 0.0 with time, □ = 0.5

In this case, we decrease the ϵ linearly with episodes.
Below are the plots for the sum of rewards in each episode vs episode number (For a total of 1000 episodes). To average out the results, we run the experiment 1000 times (Otherwise the results will be having very high variance).



Average performance for 100 Runs on 1000 episodes

1. In this case, we see that the Q-Learning algorithm performance is optimal and the SARSA algorithm gets stuck in sub-optimal performance. This is because of the SARSA algorithm exploration with the initial epsilon. It discovers that the performance can be improved by taking the safer path for this world but after some time we change the rules of taking an action due to which it gets stuck in the local maxima.
2. The final reward for the Q-Learning algorithm is -13 whereas for SARSA it is just -15 because of the extra up step.

Below is the optimal policy for both methods-

c) Q-Learning

D R R U D R U R R R L D
R D L R R D L R R D D D
R R R R R R R R R R R D
U U U U U U U U U U U U

d) SARSA

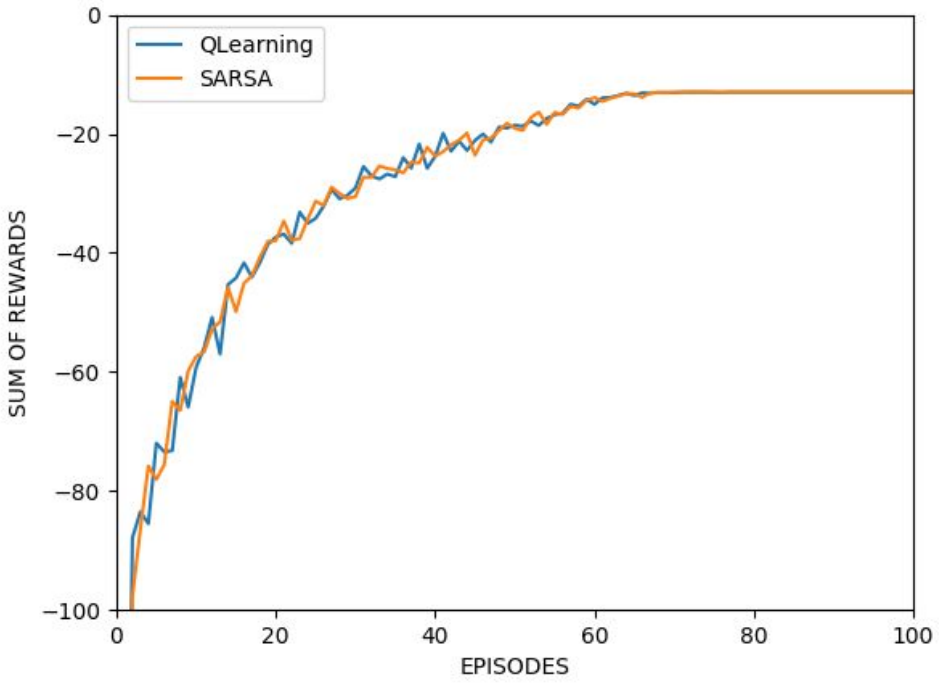
R L R R R D R R U R L D
R R R R R R R R R R D D
U U U U R R R R R U R D
U U U U U U U U U U U U

We can see that in this case, the SARSA algorithm goes just 1 step above the optimal solution this is because after some time our policy becomes greedy and we now don't need to maintain a safe distance from the cliff.
In this case, I would recommend the Q-Learning algorithm because it shows the results for the optimal policy. It even learns quicker than the SARSA algorithm.

Q-4 Suppose action selection is greedy. Is Q-learning then exactly the same algorithm as SARSA? Will they make exactly the same action selections and weight updates? Supplement your answer with results from your implementation.

Yes, if we make the epsilon equal to 0 then both algorithms will be the same because we will always be choosing the max Q value operation according to our policy.

Below are the results for 500 episodes averaged over 50 Runs



So we see that finally, both the algorithms converge to the same optimal reward. The little difference between the 2 algorithms is because we break the ties randomly and so we get to update the values differently every run. If we would break ties exactly the same then these curves would overlap.

Below is the optimal policy for both methods-

e) Q-Learning

U R R D R R R R R R R D
U U R R R R D D D U R D
R R R R R R R R R R R D
U U U U U U U U U U U U

f) SARSA

R L R R L L R D R R D D
U R R R R R R D R D D D
R R R R R R R R R R R D
U U U U U U U U U U U U

The changes in some of the states are because of the randomness of the implementation. As we break ties randomly both algorithm updates the values differently and hence we reach to a little different final policy.