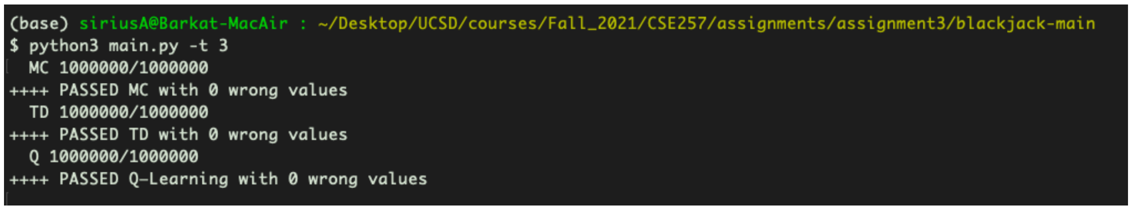
CSE257_A3_Blackjack

December 5, 2021

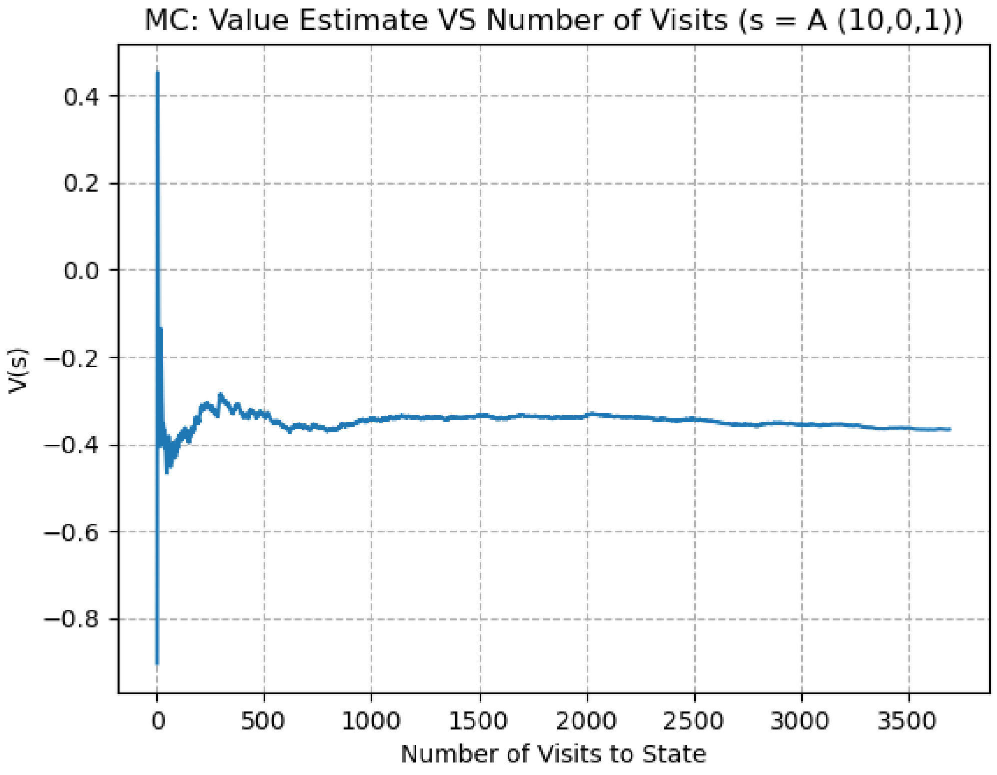
0.0.1 Question 8 (3 Extra Points):

Select two game states, A and B: In State A the player's sum of cards is 10, and in State B the sum of cards is 20. Plot how the value estimate of the each state changes over the number of visits to the state until the values convergence, under Monte Carlo policy evaluation and Temporal-Difference policy evaluation, respectively; so 4 plots in total.

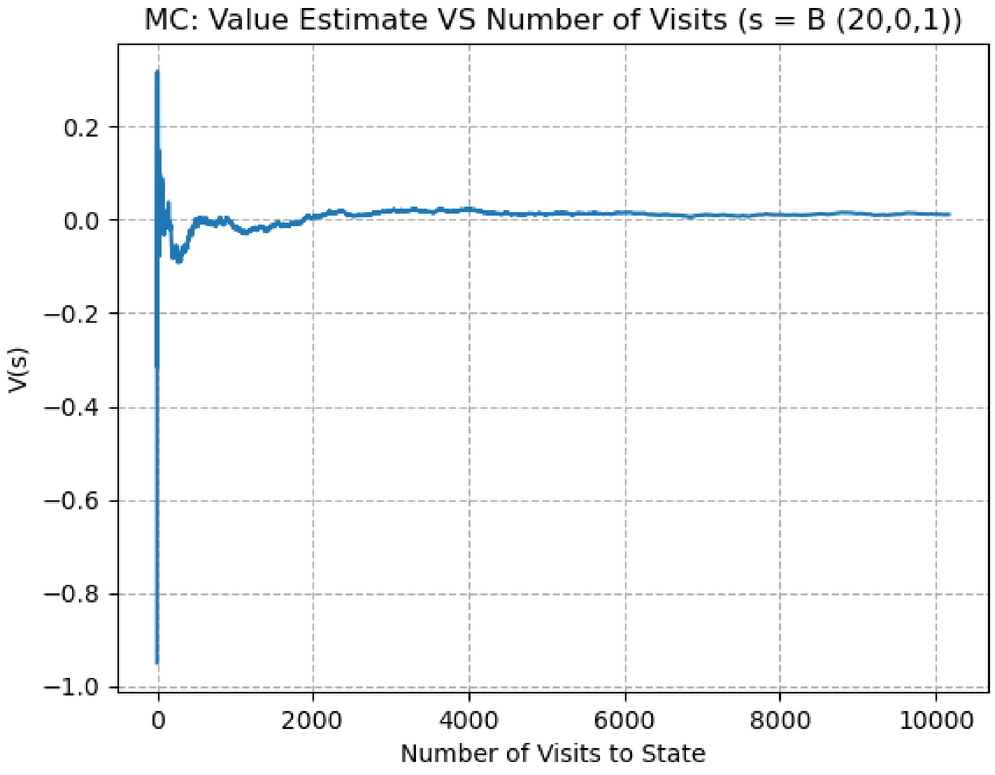


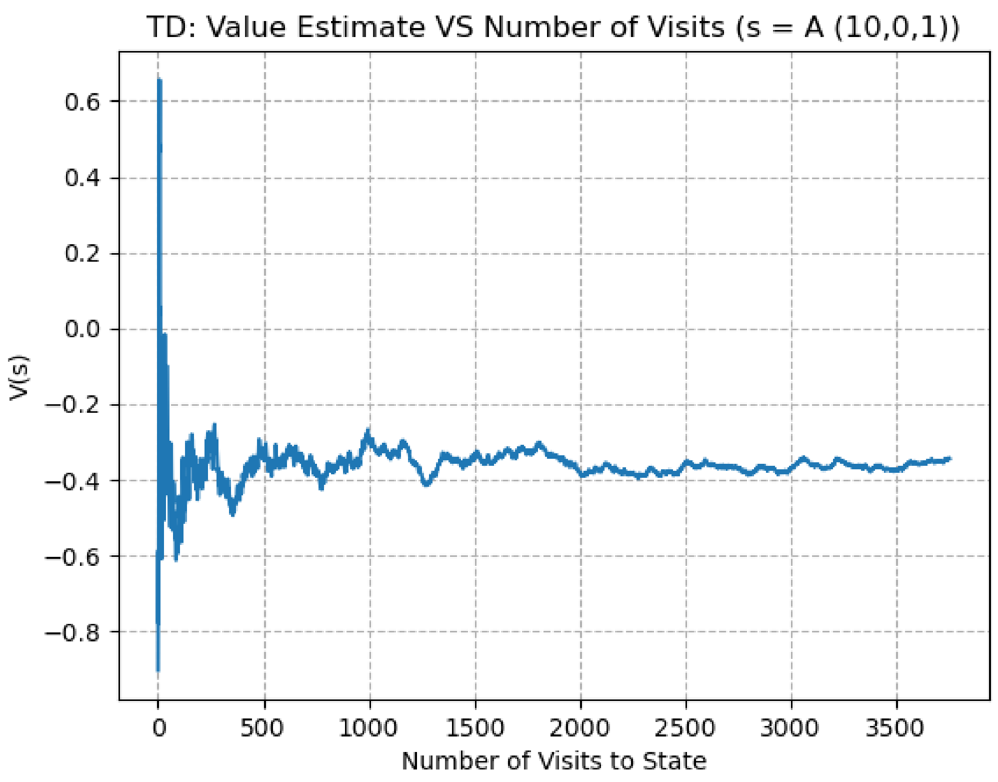
```
(base) siriusA@Barkat-MacAir : ~/Desktop/UCSD/courses/Fall_2021/CSE257/assignments/assignment3/blackjack-main
$ python3 main.py -t 3
  MC 1000000/1000000
++++ PASSED MC with 0 wrong values
  TD 1000000/1000000
++++ PASSED TD with 0 wrong values
  Q 1000000/1000000
++++ PASSED Q-Learning with 0 wrong values
```

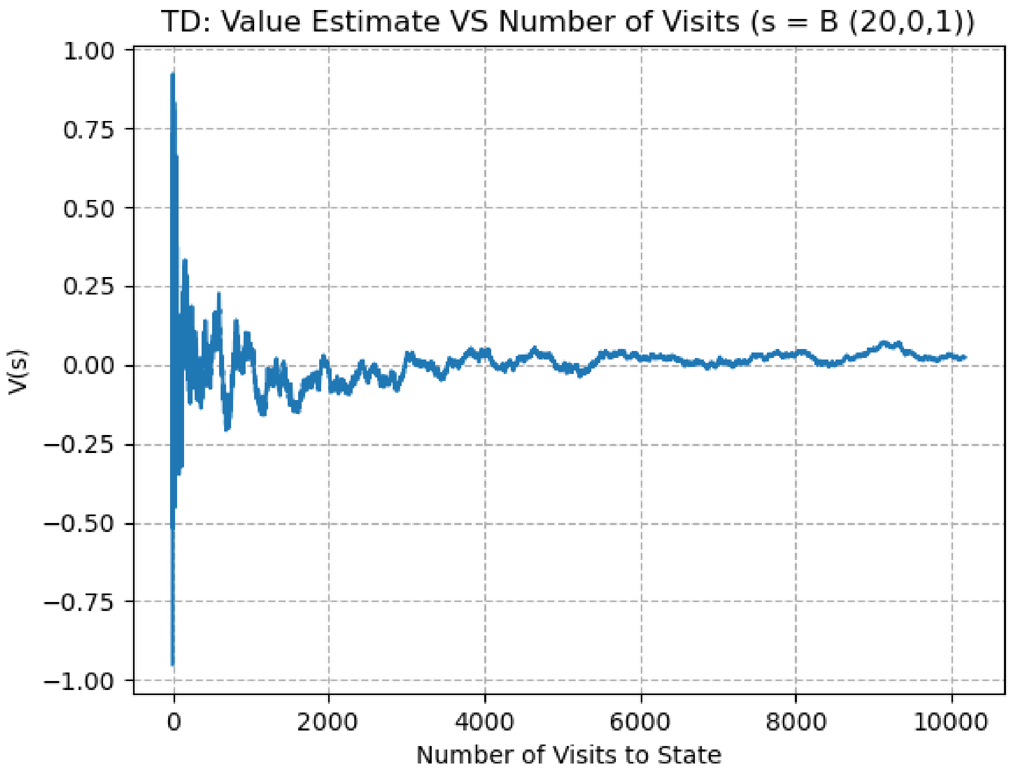
The Monte-Carlo values over the number of visits are stable whereas the Temporal-Difference state values are more fluctuating as compared to Monte-Carlo. This is because, in Monte-Carlo, each update to the state values happens after taking the expectation over large number of simulation sequences. On the other hand, in Temporal difference, each update happens with every incoming


MC: Value Estimate VS Number of Visits (s = A (10,0,1))
V(s)
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
500
1000
1500
2000
2500
3000
3500
Number of Visits to State

sample.

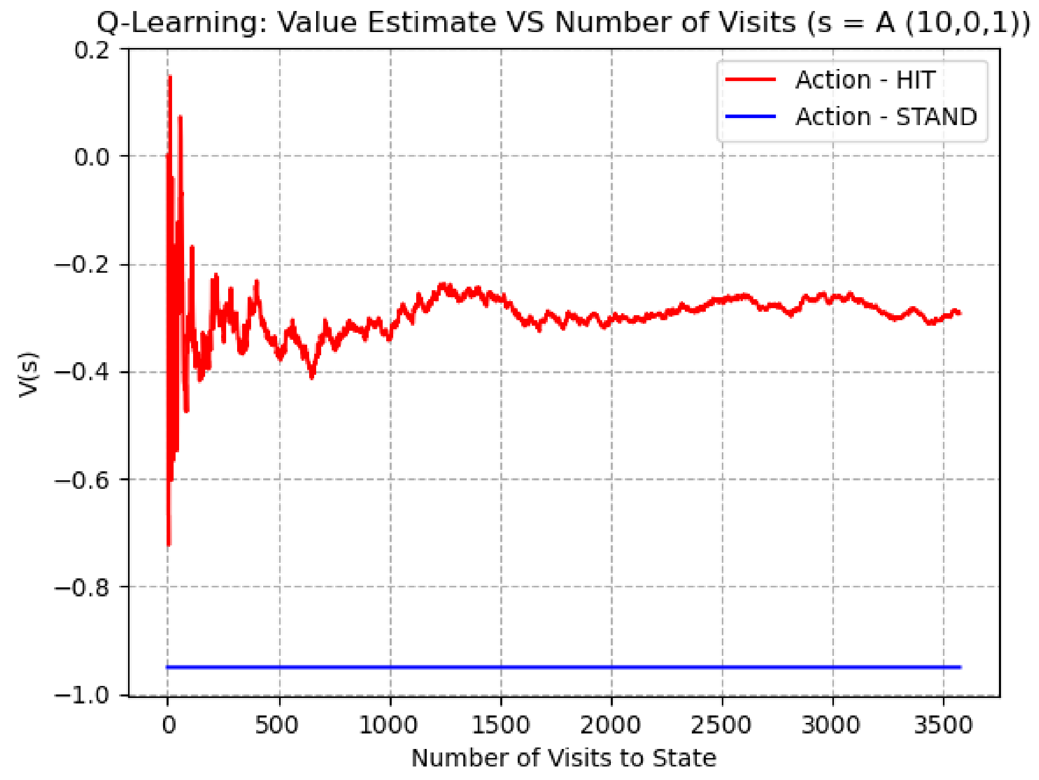

MC: Value Estimate VS Number of Visits (s = B (20,0,1))
V(s)
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
2000
4000
6000
8000
10000
Number of Visits to State

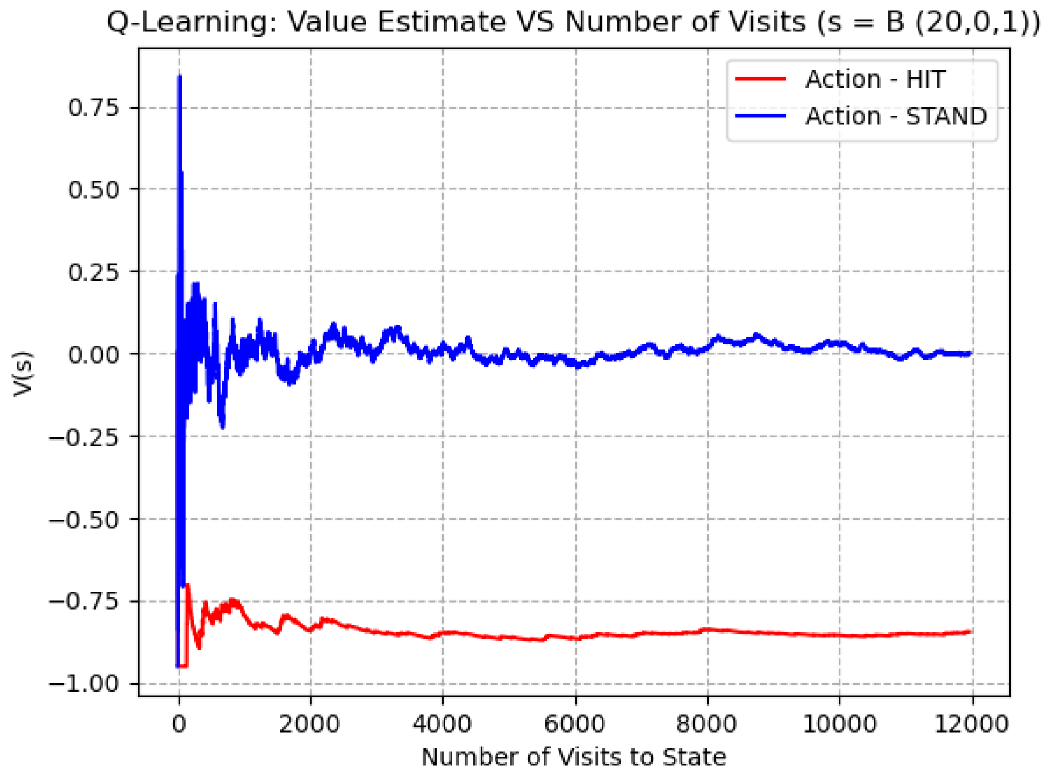

TD: Value Estimate VS Number of Visits (s = A (10,0,1))
V(s)
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
500
1000
1500
2000
2500
3000
3500
Number of Visits to State


TD: Value Estimate VS Number of Visits (s = B (20,0,1))
V(s)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2000
4000
6000
8000
10000
Number of Visits to State

0.0.2 Question 9 (3 Extra Points):

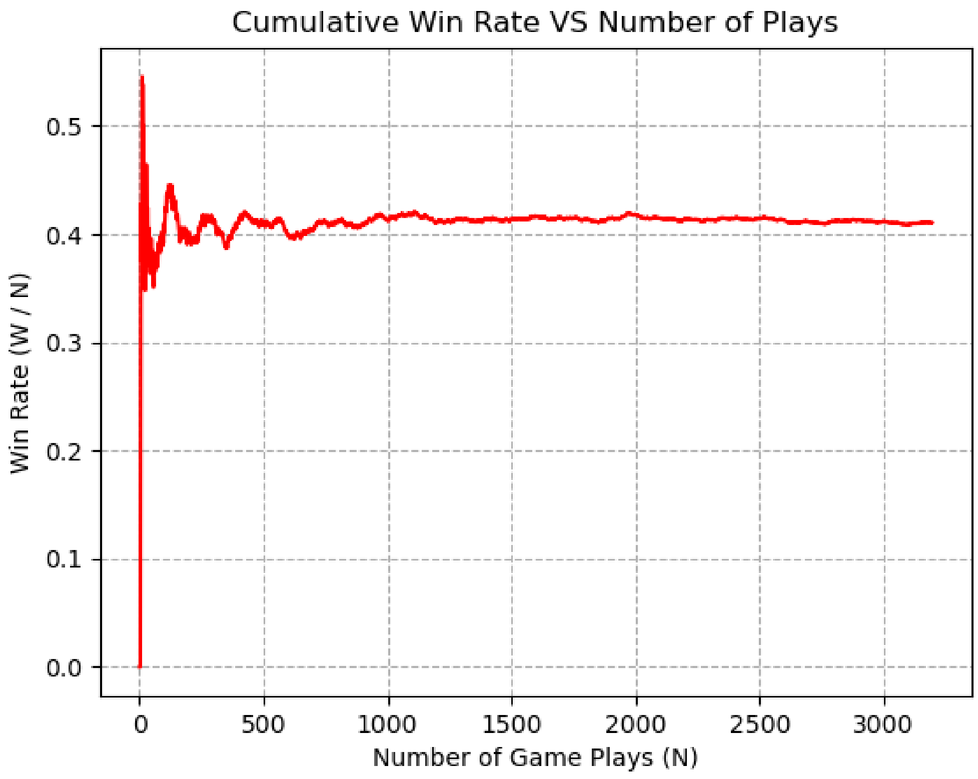
Perform Q-learning and plot how the Q-value changes over the number of visits to each action for the same two game states you selected above, until you have run Q-learning for long enough so that the Q-value converges at least on some action for each state (note: a very bad action may receive a small number of visits, so this requirement is saying you only need to wait till the better action has been visited enough times so that the Q-value of it stabilizes).


Q-Learning: Value Estimate VS Number of Visits (s = A (10,0,1))
V(s)
Number of Visits to State
Action - HIT
Action - STAND
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
500
1000
1500
2000
2500
3000
3500


Q-Learning: Value Estimate VS Number of Visits (s = B (20,0,1))
V(s)
Number of Visits to State
Action - HIT
Action - STAND
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2000
4000
6000
8000
10000
12000

Also plot the cumulative winning rate over the number of plays in the game: for every n number of plays (x-axis), show the ratio w/n (y-axis) where w is the total number of wins out of the n plays.

For plotting the cumulative win-rate vs number of game plays, I have set the `self.autoQL = True` and `self.autoPlay = True`. Then, I am running sufficiently large number of iterations (=5000). In each iteration, the Q-Learning algorithm (50 simulations) runs, which updates the Q-values of the states followed by the game playing function. The cumulative winning rate stabilizes at around ~ 41 %.


Cumulative Win Rate VS Number of Plays
Win Rate (W / N)
Number of Game Plays (N)