Reinforcement Learning

Homework 3 Report

Abhishek Agarwal
2016126

Ans 1)

*Attached below

Ans2)

*Attached below

Ans3)

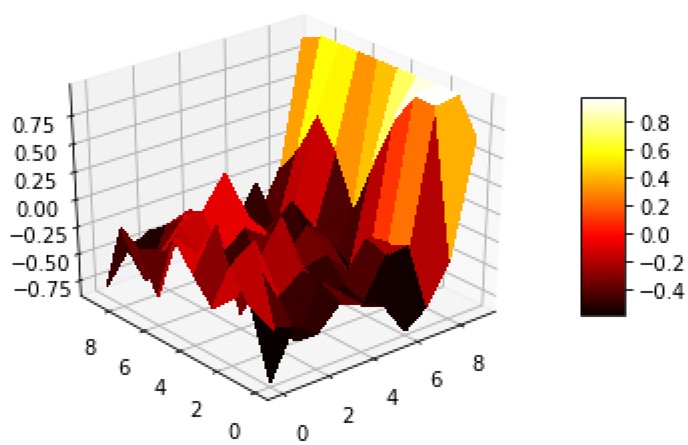
*Attached below

Ans4)

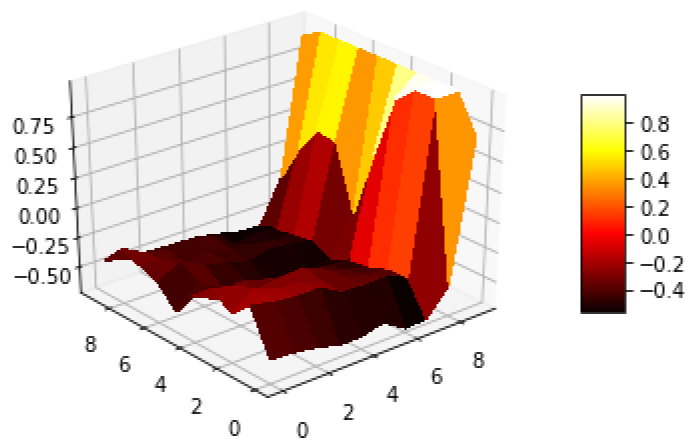
The plots generated are as follows:

All the graphs are shown in Jupyter Notebook.

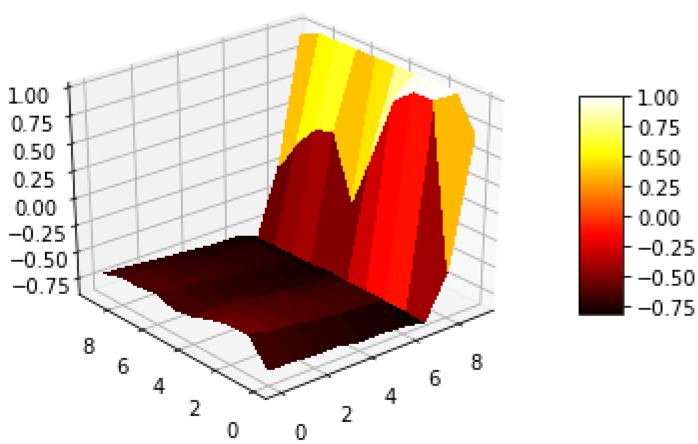
(a) 10k simulations with usable ace:



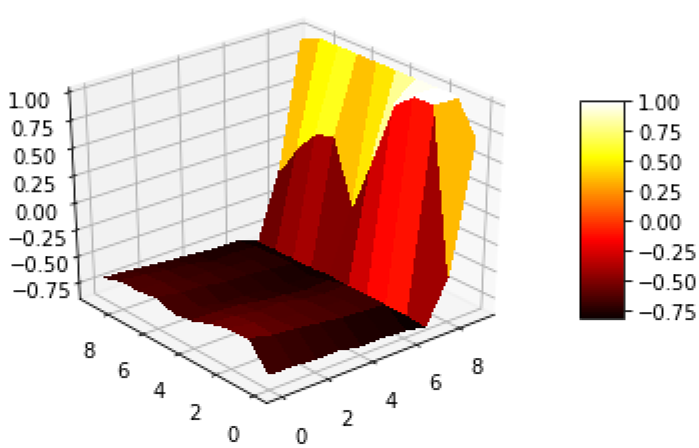
(b) 500k simulations with usable ace:



(c) 10k simulations with non usable ace:

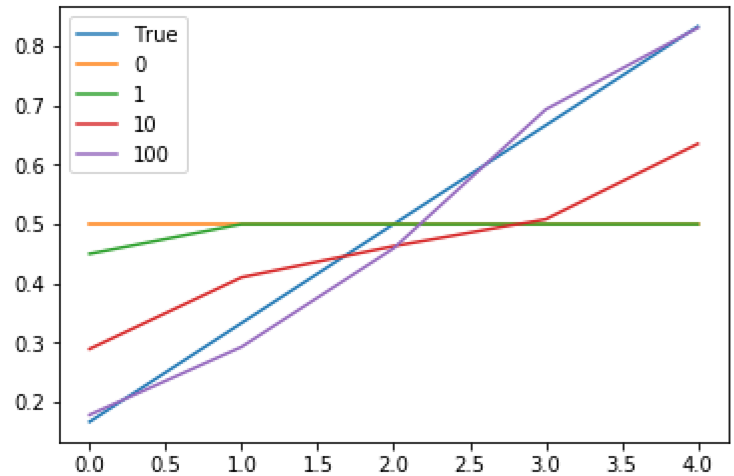


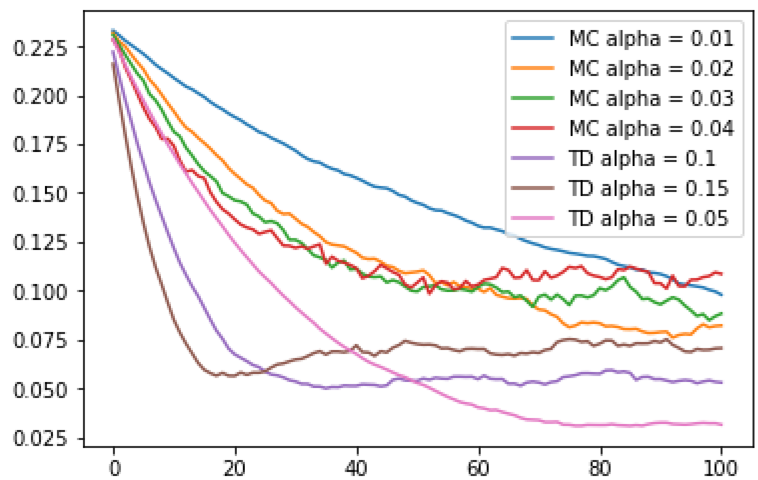
(d) 500k simulations with non usable ace:



Ans5)
*Attached below

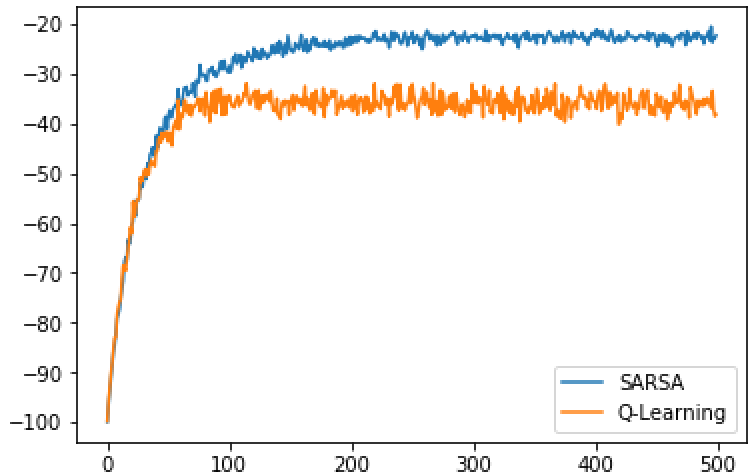
Ans6)
The generated plots are shown below:
The values on 100 episodes are very close to the true values.





Ans7)
In the cliff example, we can see the difference in Q Learning and SARSA as shown below. The Q learning method chooses the greedy path which is difficult as it is risky,

whereas the SARSA method explores the different paths. The path selected by Q learning is near the cliff and hence can receive high negative reward on taking one wrong action.



Ans8)
*Attached below

Ans 6.3)
Since the rewards are 0 when we go to any non terminal state and we have initialized all the values of non terminal states as 0.5.
When we end up in the leftmost terminal state, only the value of state A is updated. The new value will be 0.45 i.e, there is a change of 0.05.

Question 1)

In the step, $Q(S_t, A_t) \leftarrow$ Average (Returns (S_t, A_t)) we are taking average every time and thus it involves a lot of calculations.

We can avoid this by just storing the value of G and the number of times the particular state action pair has been encountered.

i.e.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{\text{count}[s][a]} \{G - Q(S_t, A_t)\}$$

Pseudo code:

Initialize:

$\pi(s) \in A(s)$

$Q(s,a) \in \mathbb{R}$

Count $(s,a) = 0 \quad \forall s, a$

Loop forever (for each episode) ~~...~~ :

~~...~~ Choose $S_0 \in S$, $A_0 \in A(S_0)$

Generate an episode from S_0, A_0, following π:

$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Count $[S_t][A_t]$ += 1

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, \ldots, S_{t-1}, A_{t-1}$:

~~Append G to Returns~~

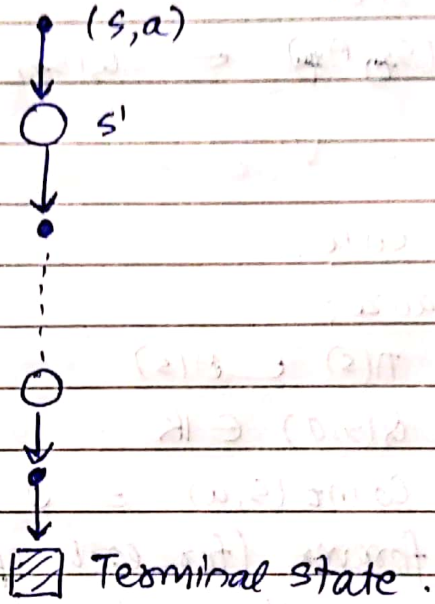
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{\text{count}(S_t, A_t)} \times [G - Q(S_t, A_t)]$$

$$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$$

Question 2:

In Monte Carlo ES, if we use $q(s,a)$ instead of $v(s)$, we generate a state action pair at the start of episode and continue till we reach a terminal state.

This can be shown in the form of backup diagram as follows:



Question 3

We need to maintain an array of time steps for the pair (s,a), which will store the time instants when that pair is visited for that episode.

Updated formula in terms of $Q(s,a)$ can be written as:

$$Q(s,a) = \frac{\sum_{t \in \mathcal{T}(s,a)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s,a)} \rho_{t:T(t)-1}}$$

where,
$T(t) \rightarrow$ first time of termination following time t
$G_t \rightarrow$ return after t up through $T(t)$

Question5

TD would work better in an example in which we move to a new building and a new parking lot, which is near the same highway and we have previous experience of driving to the old building.
This is because: In this case, only a part of the driving route has changed and some of the states (like highway) still remain the same.
Now, in the TD method, the state values are updated on the fly i.e. without the need to generate the full episode. Thus since the starting values for these states are already close to true values, it will lead to faster convergence.
Yes, same thing will happen in original scenario because of the above reason if initial state values estimate is close to true values.

Question8: In case of greedy action selection, Q-Learning will not be the same as SARSA.
This is because the way we update Q in both the methods differs. In Q-learning the q values are updated irrespective of the action taken and the action is then chosen as per the updated Q value. Whereas, in SARSA next action is chosen according to the current Q value and then Q is updated.
So, since the action ~~are~~ chosen might be different in both cases, both the methods perform differently.