Reinforcement Learning

Assignment 3 Monte Carlo and TD Learning

Prashant Pathak (MT19051)

Introduction

This assignment deals with the RL algorithm where the MDP is not given. The agent tries to learn from its algorithm. Two important algorithms in this section are

1. Monte Carlo (MC) Algorithm: It is usually applied episodic task. In this, for updating the values of a state we wait for for the episode to end
2. Temporal Difference (TD) Algorithm: It can be applied to a continuous task. We can update the value of a state during the episode. It uses bootstrapping to update the values.

Question 1

KL - Assignment 3

Page No.
Date

Question 1: In ~~pseudocode~~ pseudocode of ~~for~~ Monte Carlo ES ~~of~~ for computing the estimate of $Q(S_t, A_t)$ we have maintained ~~maintained~~ a list of reward and then we compute the ~~average~~ average. But the average can be computed in a incremental method using below method.

In pseudocode of Monte Carlo ES for computing the estimate of $Q(S_t, A_t)$ we have maintained a list of reward and then we compute the average. The average can be computed in an incremental method also using below method.

$$Q_n(S_t, A_t) = \frac{1}{n} \sum_{i=1}^{n} G_i | S_t, A_t)$$

$$= \frac{1}{n} \left(G_n(S_t, A_t) + \sum_{i=1}^{n=n-1} G_i(S_t, A_t) \right)$$

$$= \frac{1}{n} \left(G_n(S_t, A_t) + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} G_i(S_t, A_t) \right)$$

$$= \frac{1}{n} \left(G_n(S_t, A_t) + (n-1) Q_{n-1}(S_t, A_t) \right)$$

$= \frac{1}{n}(G_n(S_t, A_t) + n Q_{n-1}(S_t, A_t) - Q_{n-1}(S_t, A_t))$

$= Q_{n+1}(S_t, A_t) + \frac{1}{n}(G_n(S_t, A_t) - Q_{n-1}(S_t, A_t))$

So we can update the Q values using above formula.

Also in Second code we have to maintain count & mean upto n-1 to compute estimate at n step.

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in A(s)$ (arbitrarily), for all $s \in S$

$Q(s,a) \in R$ (arbitrarily), for all $s \in S$, $a \in A(s)$

Returns $(s,a) \leftarrow$ empty list, for all $s \in S, a \in A(s)$

$N(s,a) \leftarrow 0$ (To store the count till Now)

Loop forever (for each episode):

① Choose $S_0 \in S$, $A_0 \in A(S_0)$ randomly such that all pairs have probability > 0

② Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

③ $G \leftarrow 0$

④ Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

⑤ $G \leftarrow \gamma G + R_{t+1}$

⑥ Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots S_{t-1}, A_{t-1}$:

⑦ Append G to Returns (S_t, A_t)

⑧ $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{n}(G - Q(S_t, A_t))$

// Calculating mean in incremental fashion

(9) $N(S,a) \mathrel{+}= 1$ // Increment the count by 1

(10) $\pi(S_t) \leftarrow \underset{a}{\text{argmax}}\ Q(S_t, a)$

Page No.
Date

→ Why both new pseudo code and old pseudo is is same is given by making proof.

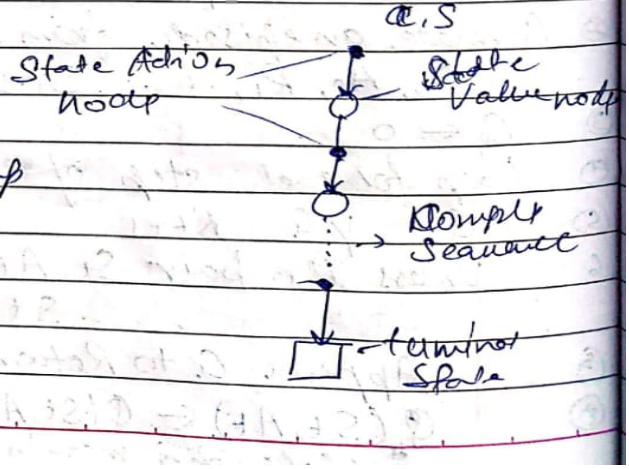
Question 2

Que 2 Draw the backup diagram for Monte Carlo q_π.

→ When we are estimating q_π i.e. state-action value when following policy π, we sample a episode from the policy π. Sampling gives us a a sequence of $S_0, A_0, R_1, S_1, A_1, R_2 \ldots$

Now in backup diagram we show that for up dating the root what are values do we need.
So for updating $q_\pi(S_0, A_0)$ we need one complete sequence, so backup diag is

We can see that it is similar to State value backup diagram.



Question 3

Exercise 5.6

Page No.
Date

Equation Analogous to 5.6 for action values $Q(s,a)$ instead of $V(s)$

Eq. 5.6 $$V(s) = \frac{\sum_{t \in T(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in T(s)} \rho_{t:T(t)-1}}$$

In an off policy method we have two policy behaviour policy & target policy. We want to find the state values or state action values of target policy following a behavious policy.

Suppose we want to estimate $Q_\pi(s,a)$ following π but what we have is the return from behaviour policy b.

We have $Q_b(s,a) = E_b[G_t \mid S_t = s, a_t = a]$

i.

With the help of important sampling we have to change this expection with expected return which is for b to expected return for target policy.

So change the expection we have to multiply it be important sampling ration.

$$\rho_{t:T-1} = \prod_{k=t}^{T=1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

This is for $V(s)$

This ratio doesn't depend on MDP it depend on policy & sequence

~~So $V(s)$ can be~~ eq 5.6 ~~can be easily change for $q_\pi(s,a)$~~

~~$q_\pi(s,a) = \sum_{t} t+1(s,a) \rho_t$~~

So for Q we have been given sequence as initially we have S_t, A_t

Then sequence is $S_{t+1}, A_{t+1}, \ldots$

$$\Pr\{S_{t+1}, A_{t+1} \ldots S_T | S_t, A_t, A_{t+1}, T-1 \sim \pi\}$$

$$= p(S_{t+1}|S_t, A_t)\, \pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1})$$

$$= \left(\prod_{k=t+1}^{T-1} \pi(A_k|S_k)\, p(S_{k+1}|S_k, A_k)\right) p(S_{t+1}|S_t, A_t)$$

So relative probability.

$$\rho_{t:T-1} = \frac{\left(\prod_{k=t+1}^{T-1} \pi(A_k|S_k)\, p(S_{k+1}|S_k A_k)\right) p(S_{t+1}|S_t, A_t)}{\left(\prod_{k=t+1}^{T-1} b(A_k|S_k)\, p(S_{k+1}|S_k A_k)\right) p(S_{t+1}|S_t, A_t)}$$

$$\prod_{k=t+1}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

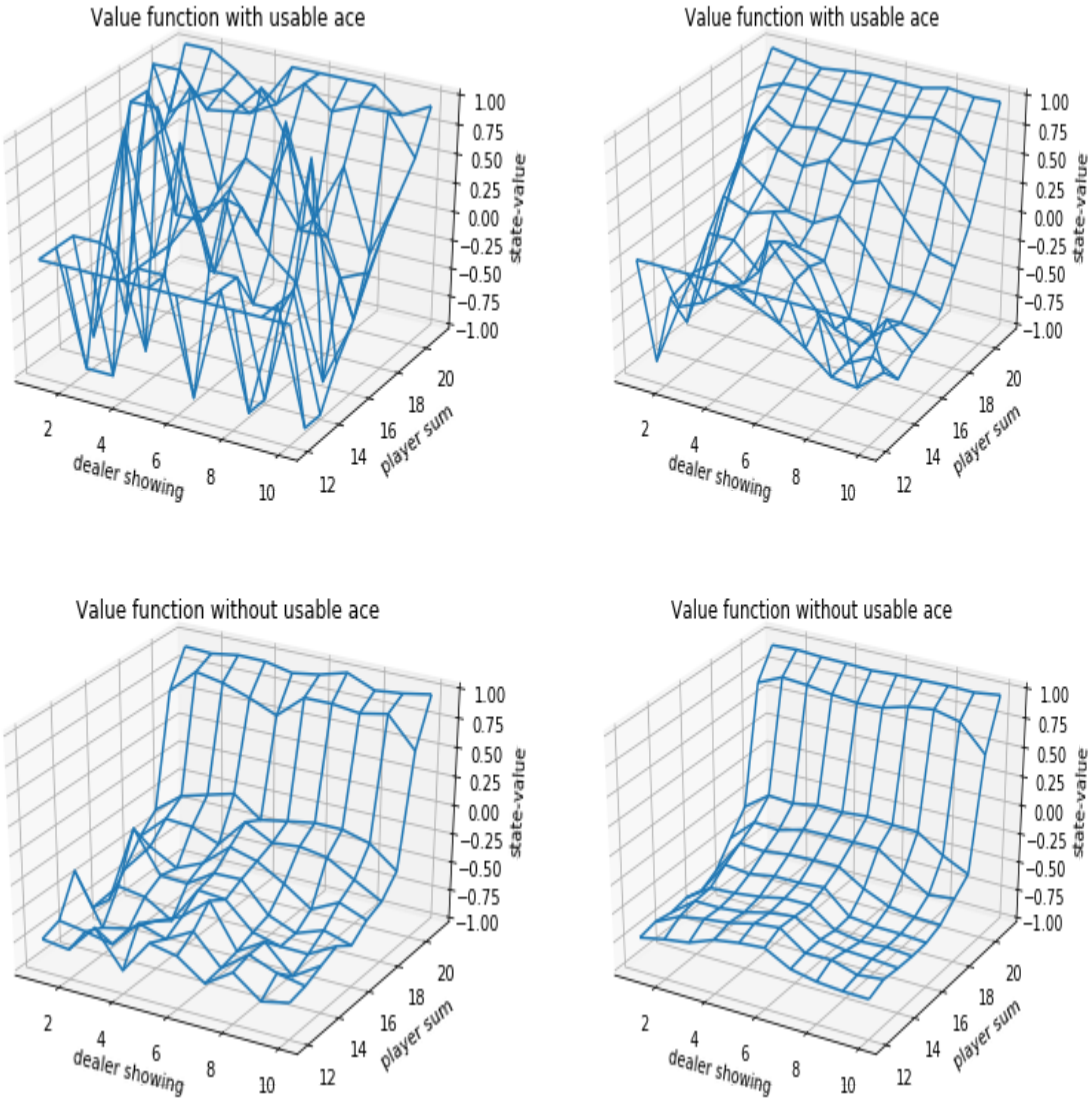
So our final equation become

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

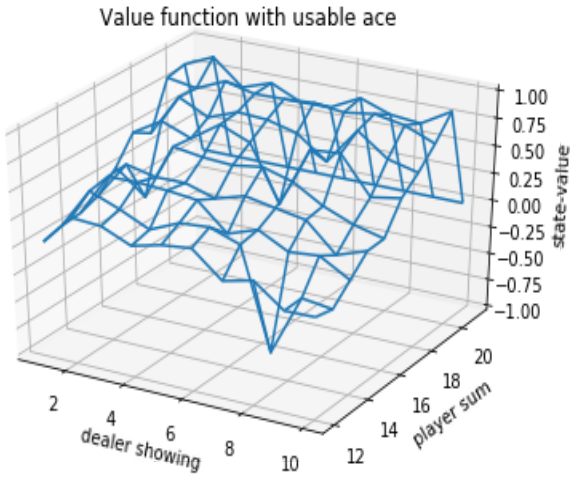
Question 4

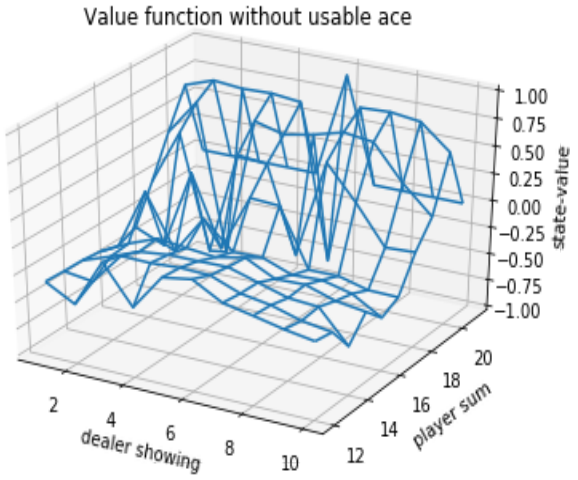
Code for Blackjack can be found at code/Blackjack_stateValue.ipynb and code/Blackjack_Blackjack_QValue.ipynb

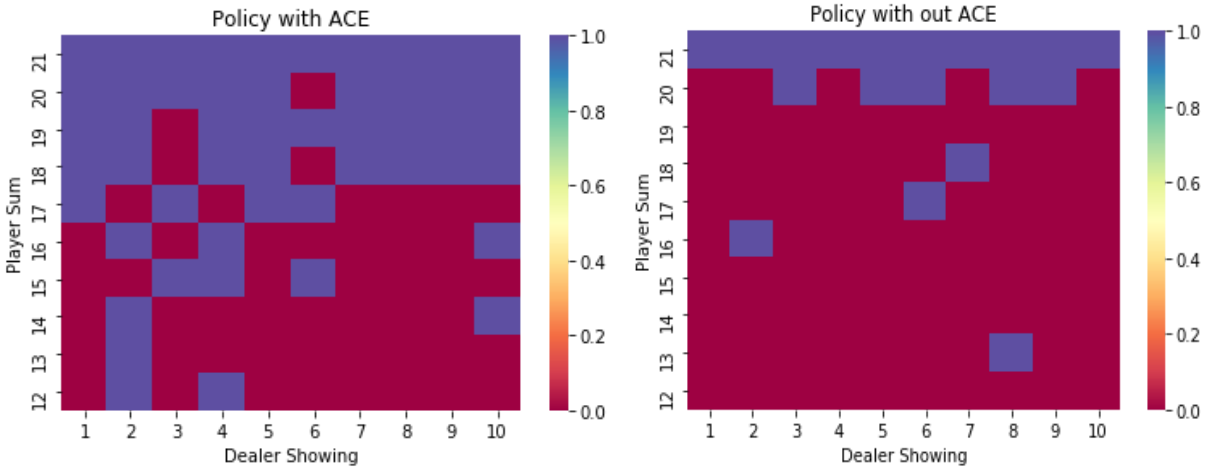
Below figure is for state value when we are following the policy to Stick when the player sum is 20 above, The graph on the left-hand side is when iteration is 10000 and one the right side is then iteration is 500000.



Now Monte Carlo Control

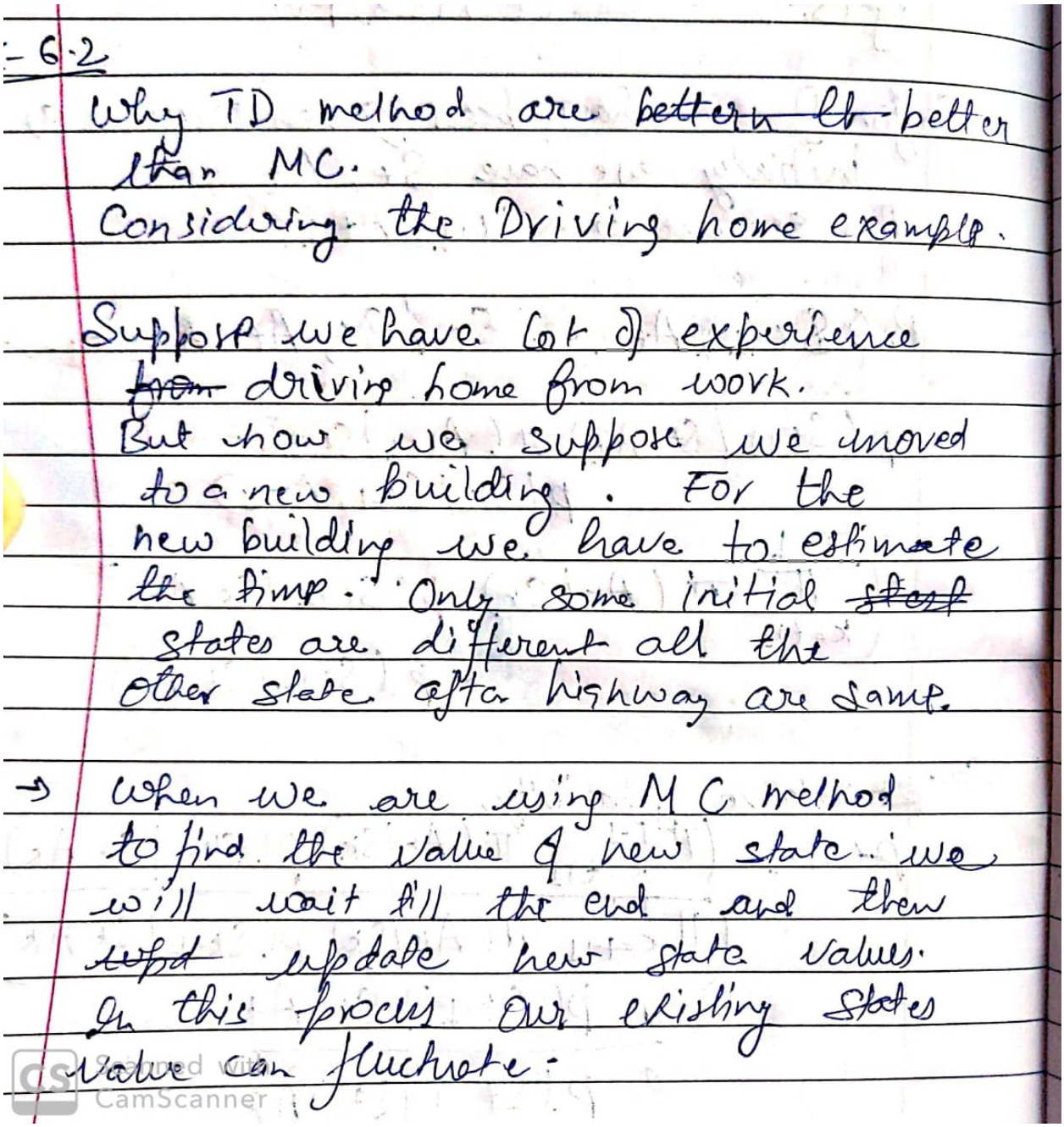






There is a convergence problem that is why our policy is not correct.

Question 5



6.2

Why TD method are ~~better~~ ~~th~~ better than MC.

Considering the Driving home example.

Suppose we have lot of experience ~~from~~ driving home from work.

But now we suppose we moved to a new building. For the new building we have to estimate the time. Only some initial ~~start~~ states are different all the other state after highway are same.

→ When we are using MC method to find the value of new state. we will wait till the end and then ~~upd~~ update new state values.

In this process our existing States value can fluctuate.

→ When using a TD learning as we update in an online fashion

Page No.
Date

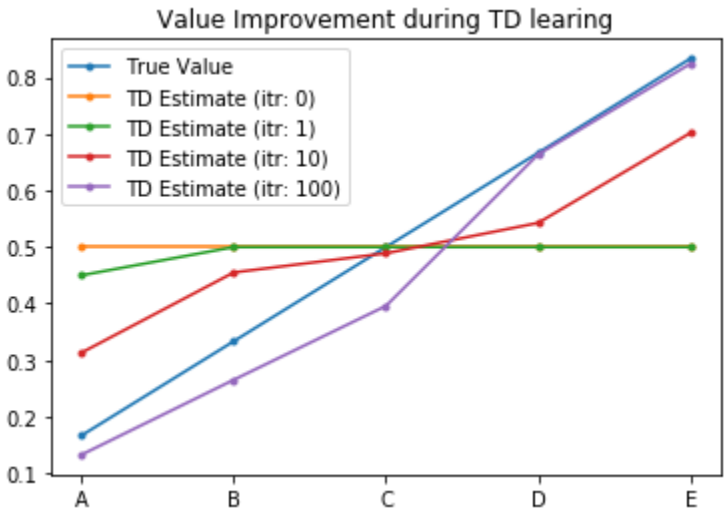
and use next state estimate to update new state, our update will be faster because our old state already have good estimate because of experience. TD update will also be faster.

→ Another scenario where TD method work better is when we have an infinite episode. MC method will ignore state to find values but TD method can work faster here.

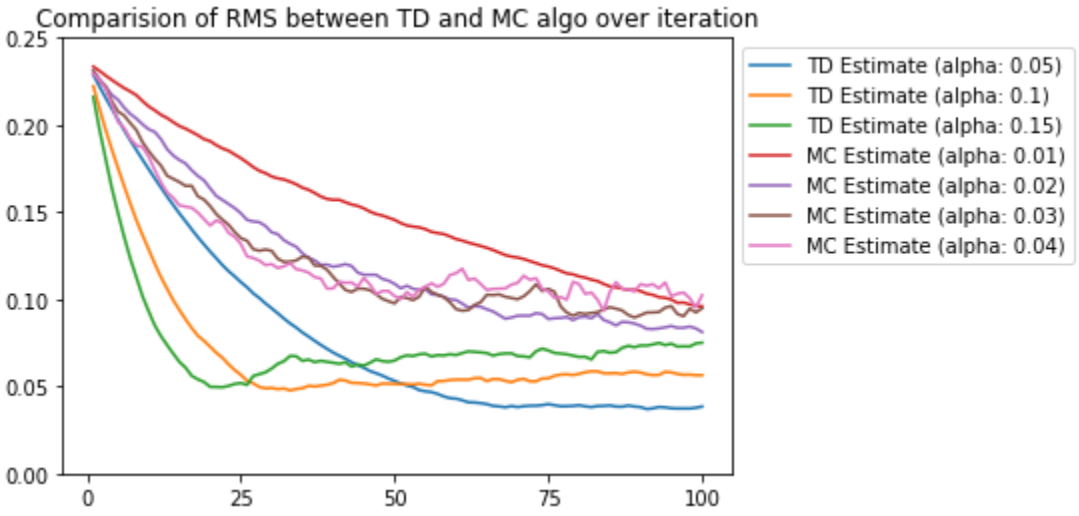
Question 6

TD vs MC

We have taken a Random walk environment to compare the algorithm. For detain see code/RandomWalk_TDvsMC.ipynb.



From the graph, we can see that the estimate values in by TD are actually closer to True values.



From the RMS plot, we can see that TD(0) method is able to reach closer to the true values compared to the MC method.

Q6 6.5

→ From the graph after first iteration we can tell that our episode ended at 'A'.

⇒ We cannonot tell about the immediate states because all intermediate reward is 0 & Episode is undiscounted.

→ V(A) is updated by following formula.

$$V(A) = V(A) + \alpha(R + \gamma V(S') - V(A))$$

Next state (Terminal state)

$$= 0.5 + 0.1(0 + 1 \cdot 0 - 0.5)$$

$$= 0.5 + (0.1 * -0.5)$$

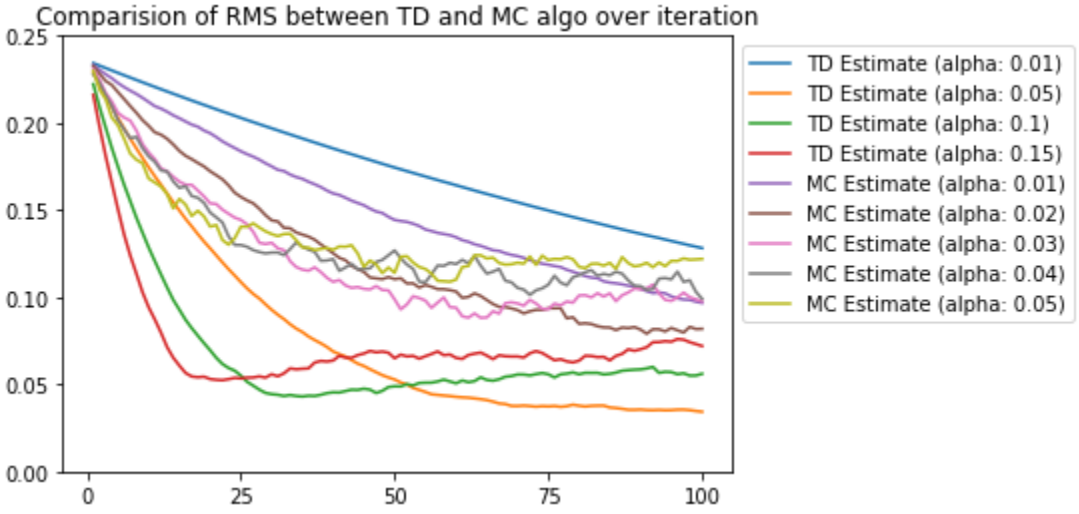
$$= 0.5 - 0.05 = 0.45$$

6.4 Basically what we are doing when doing calculating the estimate is using exponentially moving average.

α parameter is telling us how much of the past we must remember.

Both TD & MC method requires a small α to converge. & So trying on a wide variety of α will not change the conclusion.

We have performed experiment on different values of alpha but we did not find any specific α that perform significantly better than the others.



6.5

The RMS first decreased and then increases because of the following reasons

1) The value of alpha is very large
2) Because of the initialization. State ‘C’ is initialized with 0.5 which is its true value. So initially the error decrease. As the error is propagated from terminal state to C. The value of ‘C’ fluctuates and moves away from 0.5 thus can cause thus RMS to increase.

Question 7

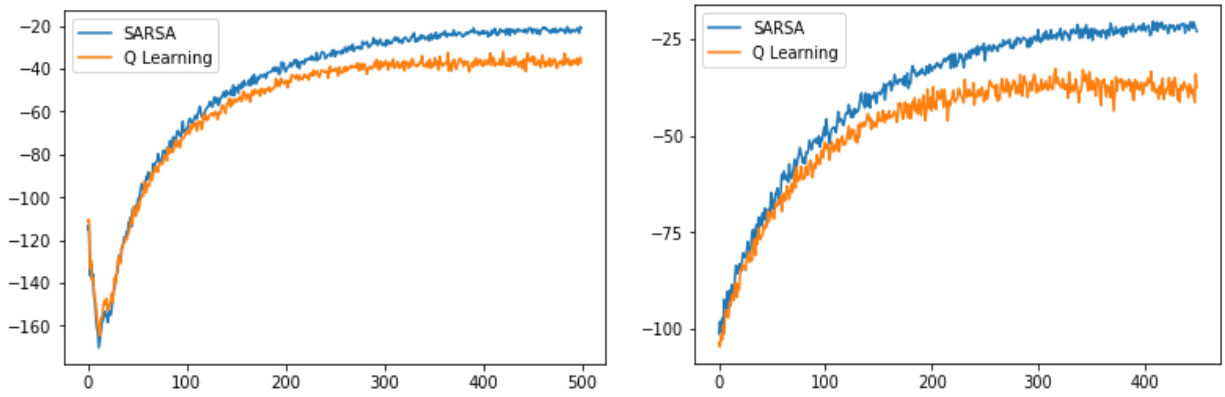
In this, we have to Compare SARSA and Q-learning Algorithm.

The environment is Cliff Walking. For detail on how the plot is generated see the notebook in code/QLearningVs SARSA.ipynb

Policy after 500 iteration

```
SARSA Grid
****************************************
 |U |R |R |R |R |R |R |R |R |R |R |D|
 |R |R |R |R |R |R |R |R |R |R |R |D|
 |U |U |U |U |L |U |U |R |L |R |R |D|
 |S |C |C |C |C |C |C |C |C |C |C |G|
****************************************


Q Learning Grid
****************************************
 |L |R |R |D |R |U |R |U |U |U |R |D|
 |R |R |R |R |R |R |R |R |D |R |R |D|
 |R |R |R |R |R |R |R |R |R |R |R |D|
 |S |C |C |C |C |C |C |C |C |C |C |G|
****************************************
```



- Graph plot Iteration in x and sum of reward in y. In left, we have plotted all the iterations. In right we by skipping some of the initial values.
- In the left graph, we can see that in starting agent is taking some time to figure the correct path. But in the end, Q-learning performs better than SARSA fro this particular Epsilon values.
- As Q-learning tries to be greedy is takes a path closer to that of the cliff.
- In SARSA we explore a little so Agents takes the safer path away from the Cliff.

Question 8

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?

No, the Q-learning and SARSA Algorithm are not the same. SARSA is an on-Policy Algorithm while Q-learning os an Off-Policy Algorithm.

Even if we make the behaviour policy of Q-learning greedy it will not is same as SARSA and the reason is visible if we see the Psedu code of both the code

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

In SARSA we choose A' from the not updated state Q values. I.e we choose A' prior to updating the Q values but in Q-leaning we choose A' from the updated Q values.