ASSIGNMENT 3

SAMAKSH GUPTA

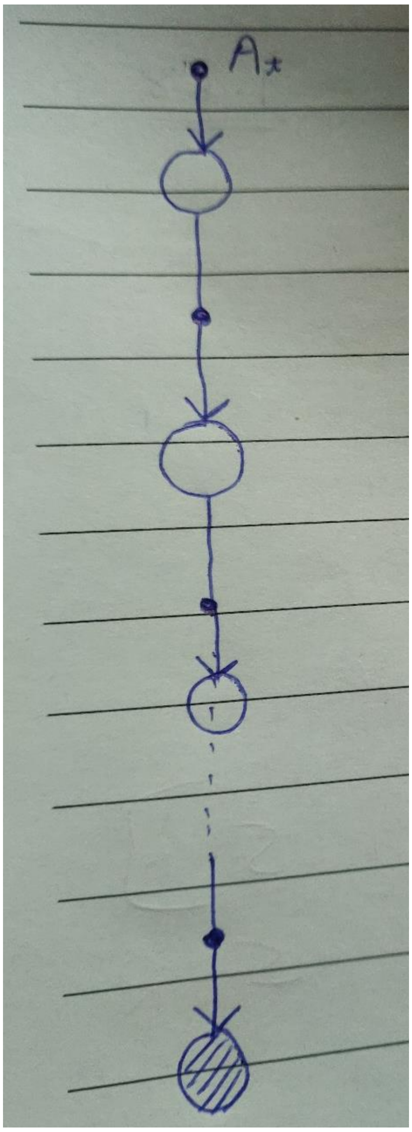
2019200

1)

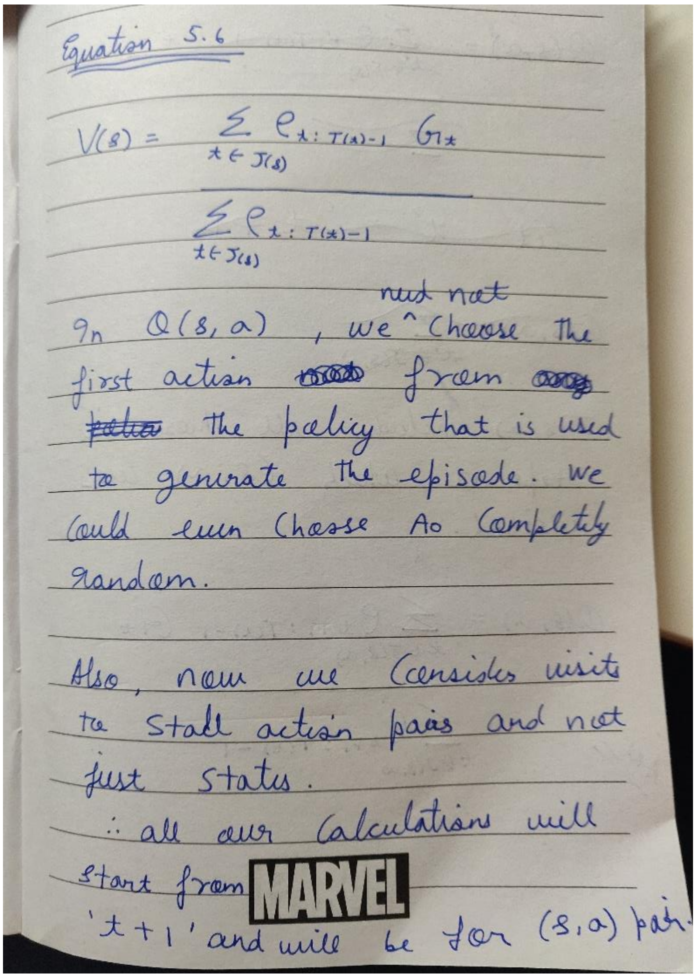
```
Initialize:
    pi(s)

    #Make 2d lists for every combination of state action pair. They are 2d as we have [state,action] as indices.
    Q[[]]       <--0
    SA_count[[]] <--0


loop forever (Generate infinite episodes):

    Choose a starting state, action pair (S0,A0)
    Using policy pi and starting state-action pair (S0,A0), generate an episode

    (S0,A0)-->(S1,A1,R1)-->(S2,A2,R2) ....... (S_(T-1), A_(T-1), R_T)

    G<--0
    loop for every length of episode (t= T-1, T-2,....2,1,0)#Loop backwards

        get St,At,R_(t+1) of the episode
        G <-- rG + R_(t+1)

        Q[St,At]<-- (G + (Q[St,At] * SA_count[St,At]))/(SA_count[St,At])
        SA_count[St,At] = SA_count[St,At] + 1

        pi_St= argmax(Q(St,a)) #In St row, check for all indices (actions)
```

2)

Action → State → Action …. Terminal State



3)

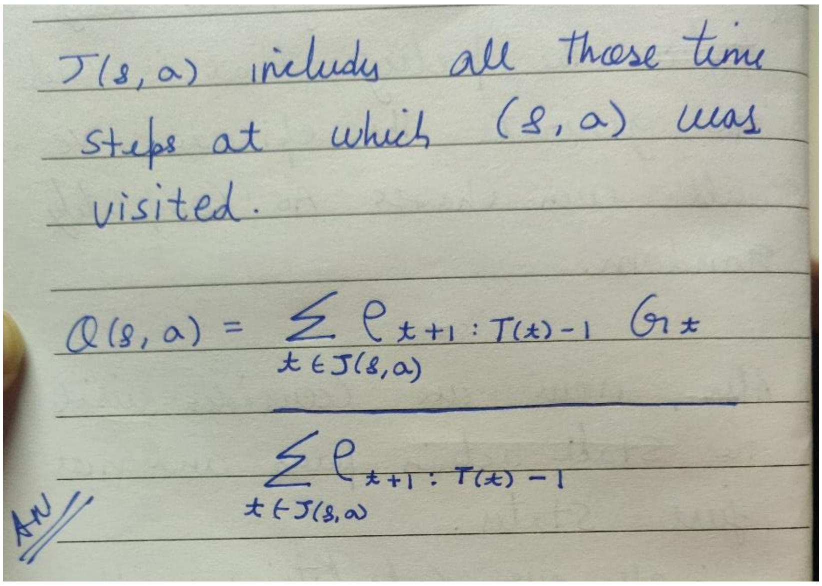


Equation 5.6

$$V(s) = \frac{\sum_{t \in \mathcal{J}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{J}(s)} \rho_{t:T(t)-1}}$$

In Q(s, a), we need not choose the first action from the policy that is used to generate the episode. We could even choose A_0 completely random.

Also, now we consider visits to state action pairs and not just states.
∴ all our calculations will start from 't+1' and will be for (s, a) pair.

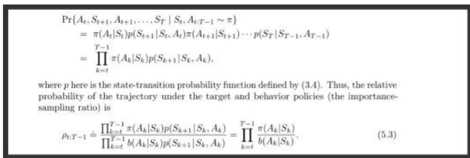


$\mathcal{J}(s, a)$ includes all those time steps at which (s, a) was visited.

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

AN//

MATHEMATICALLY



$$\begin{aligned}\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\ = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\ = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),\end{aligned}$$

where p here is the state-transition probability function defined by (3.4). Thus, the relative probability of the trajectory under the target and behavior policies (the importance-sampling ratio) is

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}. \quad (5.3)$$

Start in time 't'?

Assume

(S_t, A_t) is known to us $\Rightarrow P[A_t|S_t] = 1$ as A_t is known

$\therefore$ We can modify our Probability Expression.

$$\Pr\{S_{t+1}, A_{t+1}, S_{t+2} \ldots S_T \mid S_t, A_t, (A_{t+1:T} \in \pi)\}$$

$$= \left(\pi(A_{t+1}|S_{t+1}) P[S_{t+1}|A_t, S_t]\right)$$

$$\left(\pi(A_{t+2}|S_{t+2}) P[S_{t+2}|A_{t+1}, S_{t+1}]\right)$$

$$\ldots P[S_T|S_{T-1}, A_{T-1}]$$

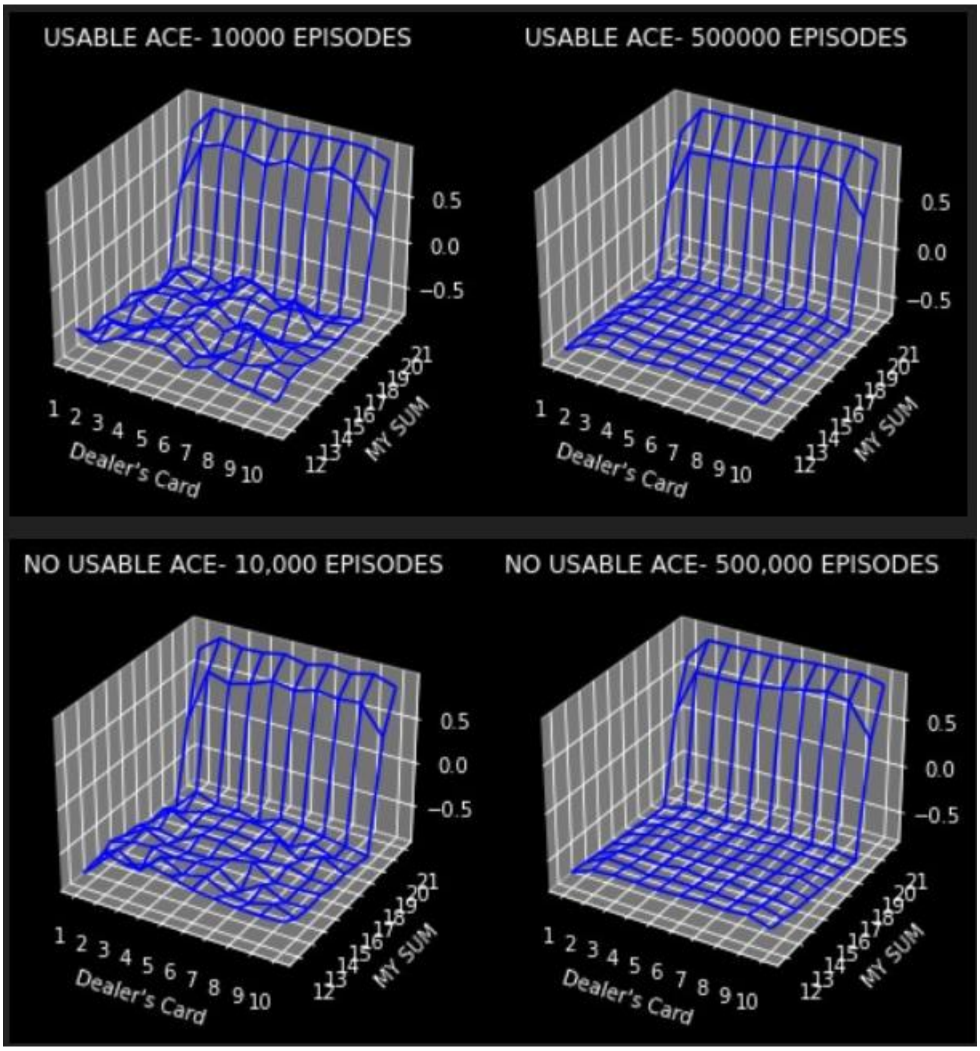
$$= \prod_{k=t+1}^{T-1} \pi(A_k|S_k) P[S_{k+1}|S_k, A_k]$$

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

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

4)

Figure 5.1



Obviously as we increase the number of episodes, the noise decreases and our state values sort of converge near the truth. Clearly Usable ace case is more noisy than usable No Usable ace case in the book. This has been successfully achieved.

There is not really any sort of explanation. The figure attached merely gives the state value of the policy we used.

One interesting thing is that EVERY VISIT and FIRST VISIT would perform equally well. As we would never be able to encounter the same state twice in one episode, as we cannot draw a card of zero value. SO, essentially every visit is first visit only in this setting.

Rest figures would be uploaded on Github.

Also, In those figures we find the optimal policy using monte carlo control.

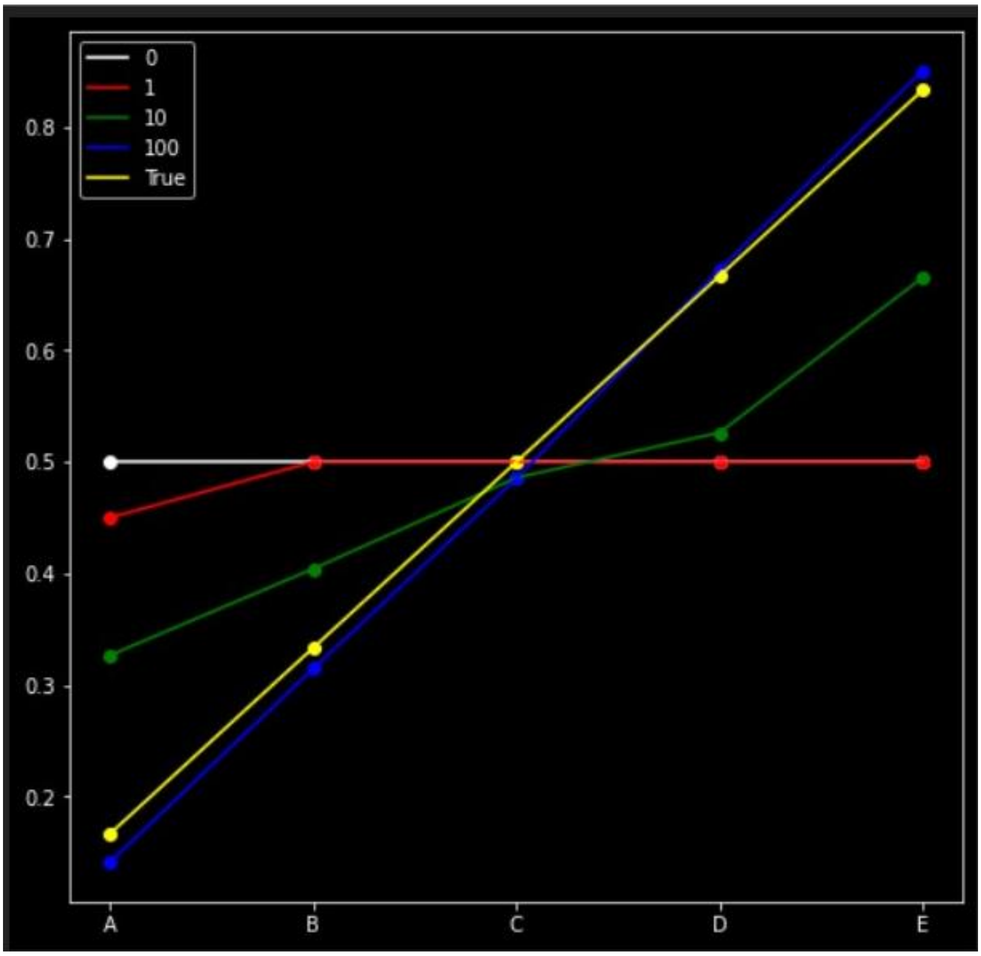
5) Excercie: 6.2

TD estimates do not wait for the episode to end. The estimate while episode is running. In case of the scenario mentioned in the book, TD methods are more efficient, because we know certain states already (highway). We already have its estimates from prior examples and we can boot strap new building and new parking space to highway states (whose estimates are less noisy) while we are en route. However, for MC method we would take no advantage from the fact that we have less noisy estimates of highway because we do not boot strap to future states in MC.

Hence, atleast initially when we have highly noisy estimates for new parking space and building, MC methods would perform poorly. Also, since they wait for the episode to end, they'll be slower as well.

6)

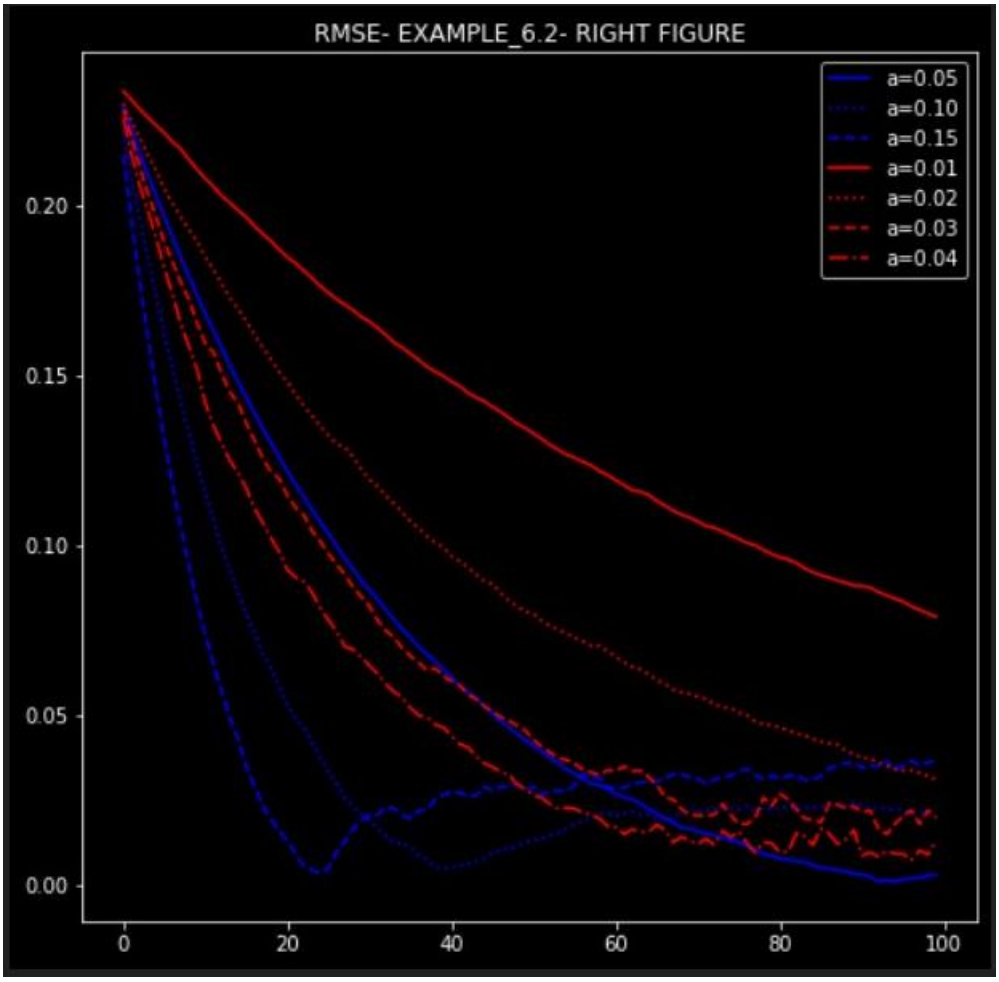
EXAMPLE 6.2 – LEFT FIGURE



In 1 episode only state A changed ➔ episode terminated in T1 (left). Detailed reason on page 8.

We converge to the Truth after about 100 episodes.

EXAMPLE 6.2- RIGHT FIGURE



The least rmse error is given by TD method with alpha=0.05(lowest among TD). However, in MC lowest error is given when the value of alpha is higher. This means that taking larger steps is better in case of MC. Since, we don't bootstrap in MC and we estimate the values of state according to previous values, we can take larger steps specially later in time, when state values are less noisy. However, in TD larger alpha is performing poorly, it is better that we take smaller steps.

TD with alpha =0.05 performs the best out of everyone.

WHY A SLIGHT DIFFERENCE IN THE FIGURE?

I KNOW THE FIGURE IS NOT EXACT AS GIVEN IN BOOK. HOWEVER, THE GENERAL TREND IS FOLLOWED. IT IS POSSIBLE THAT 'MC' WOULD PERFORM BETTER THAN TD UNIKE IN BOOK IN WHICH MC IS HIGHER. YOU CAN CHECK MY ALGORITHM TO SEE, IF I HAVE MADE A MISTAKE THERE; BUT TO HAVE A SLIGHTLY DIFFERENT FIGURE IS ABSOLUTELY POSSIBLE.

Also, both TD and MC methods are not operating under the same episodes. I am generating different sets of episodes for every run, to have more randomness and verify the trend of the result. This is also a reason as to why MC performed better than TD with lower alpha.

EXERCISE 6.3)

From the results shown in the left graph of the random walk example it appears that the first episode results in a change in only V (A). What does this tell you about what happened on the first episode? Why was only the estimate for this one state changed? By exactly how much was it changed?

ANS) The first episode can only change the state value for 'A' or 'E'. This is because as per the update rule: v(s)= v(s) + a(v(s')+r-v(s)), for all states other than terminal states v(s) is 1/2. This means V(s') and V(s) cancel with each other inside the alpha bracket. V(s)= V(s) + ar is left. Since, all rewards in the middle are zero ==> v(s)= v(s) and there is no change for the middle states.

However, for 'A' and 'E', this cancelation won't take place if s' is 'T1' or 'T2'; because the value of terminal states is zero. Since, in episode 1 the value of state 'A' changed this means that in episode 1, we reached state 'A' and Terminated on the left 'T1'. V(A)= V(A) + a(0 + V(T1) - V(A)) V(A)= 0.5 + a(0) + a(0) - a(1/2)

V(A)= 0.5 - a(1/2) v(A)= 0.5- a/2

and hence, the value of state A went down in the first episode. The value of state 'A' reduced by a/2. Where a is alpha parameter. For alpha=0.1... reduction is 0.05 Hence, the new value os state 'A' is 0.45

EXERCISE 6.4)

The specific results shown in the right graph of the random walk example are dependent on the value of the step-size parameter, ↲. Do you think the conclusions about which algorithm is better would be a↲ected if a wider range of ↲ values were used? Is there a di↲erent, fixed value of ↲ at which either algorithm would have performed significantly better than shown? Why or why not?

ANS) No, there cannot be any special a for which one method would perform. First of all alpha must be small for convergence to occur in both Monte carlo and TD. So, we cannot use a very wide range, we would still want alpha to be sufficiently small to ensure convergence. Moreover in all the graphs shown, the learning curve has saturated in every case of alpha, so there cannot be any meaningful conclusion drawn from taking some other alphas. However, having alpha dependent on time or the states might be a better option and help in better estimates.

EXERCISE 6.5)

In the right graph of the random walk example, the RMS error of the TD method seems to go down and then up again, particularly at high ↲'s. What could have caused this? Do you think this always occurs, or might it be a function of how the approximate value function was initialized?

Taking larger steps in higher alphas in case of TD method is turning out to rise the errors later in the state. Maybe since, we initialised every state value to the same numerical number, this results in cancelling out of v[s] and v[s'] a lot of times. However, later in the episode this cancellation would not occur and we might make our estimates of current states noisy due to noise of other states. Since, state 'C' is initialized to its actual True value, it will move away from it and hence RMSE would increase.

7)

ON GITHUB

8)

No, they will still not be the same in a very specific case. Consider when we transition from s=S to s’=S. That is we remain in the same state. In this case SARSA will use previous A’ generated from S’ to generate the next state action pair. However, Q learning would take A’ after the update and then generate the next state action pair.