REINFORCEMENT LEARNING

ASSIGNMENT 1

Samaksh Gupta

2019200

READ ME

Please refer to Github for all the code.

It is uploaded under assignment-1 in the repository.

You have been added as a collaborator.

The folder contains Jupyter Notebook, one for each question.

I have not attached certain plots here, which are unnecessary. However, you can view them by running the notebook file.

Flow of the Document: -

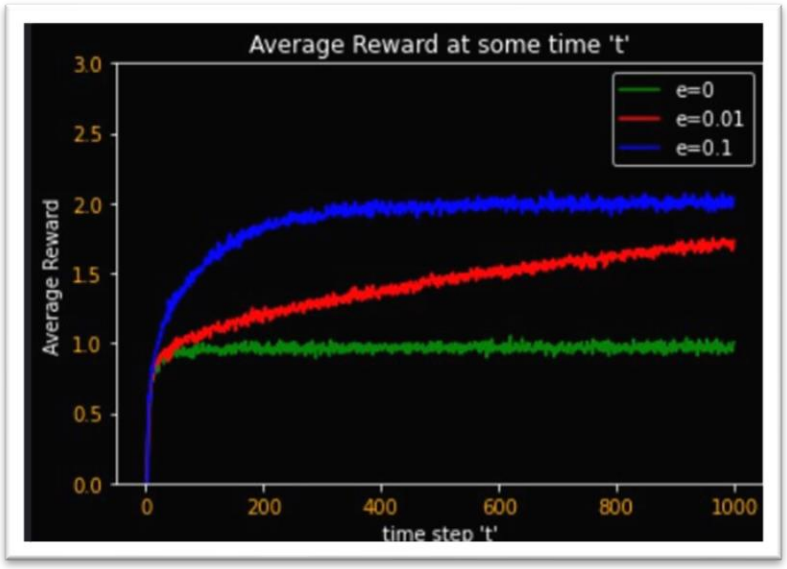
1. **Plots**
2. **Explanations**
3. **Theory Question**

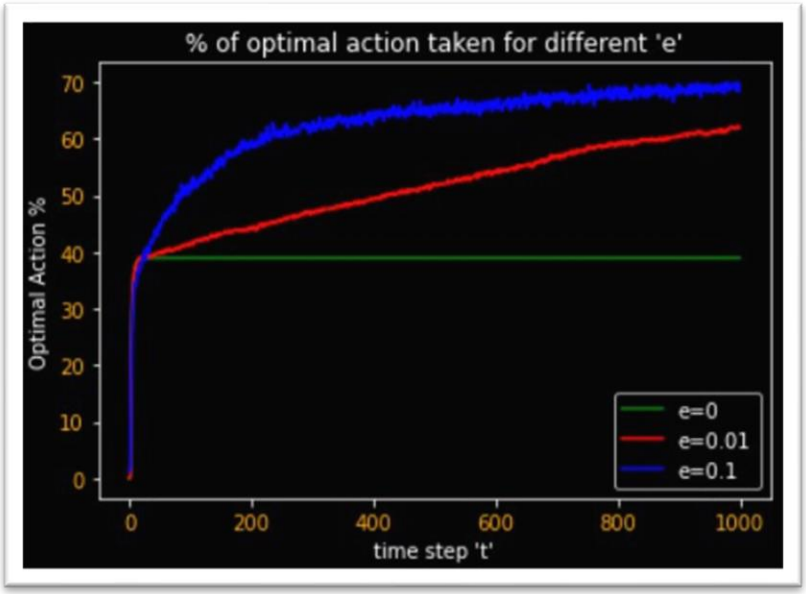
Please note: You might not get the same plot every time you run the Jupyter Notebook. It is merely because these plots will depend on the reward distribution too. It is likely that the distributions are a bit different and the plot doesn't come right. Majority of the times, the plot will be correct and similar to that of the book, but sometimes there might be some randomness.

https://github.com/Samaksh36/Reinforcement_Learning

Question 1. Read Section 2.3 of SB and generate the plots in Figure 2.2. Also, generate a plot that shows the average absolute error in the estimate for each action (arm) as a function of time steps. In addition to the in the book, generate the plots for when the changes with time in a matter such that the sequence of {(t), t ≥ 1} satisfies Equation (2.7).

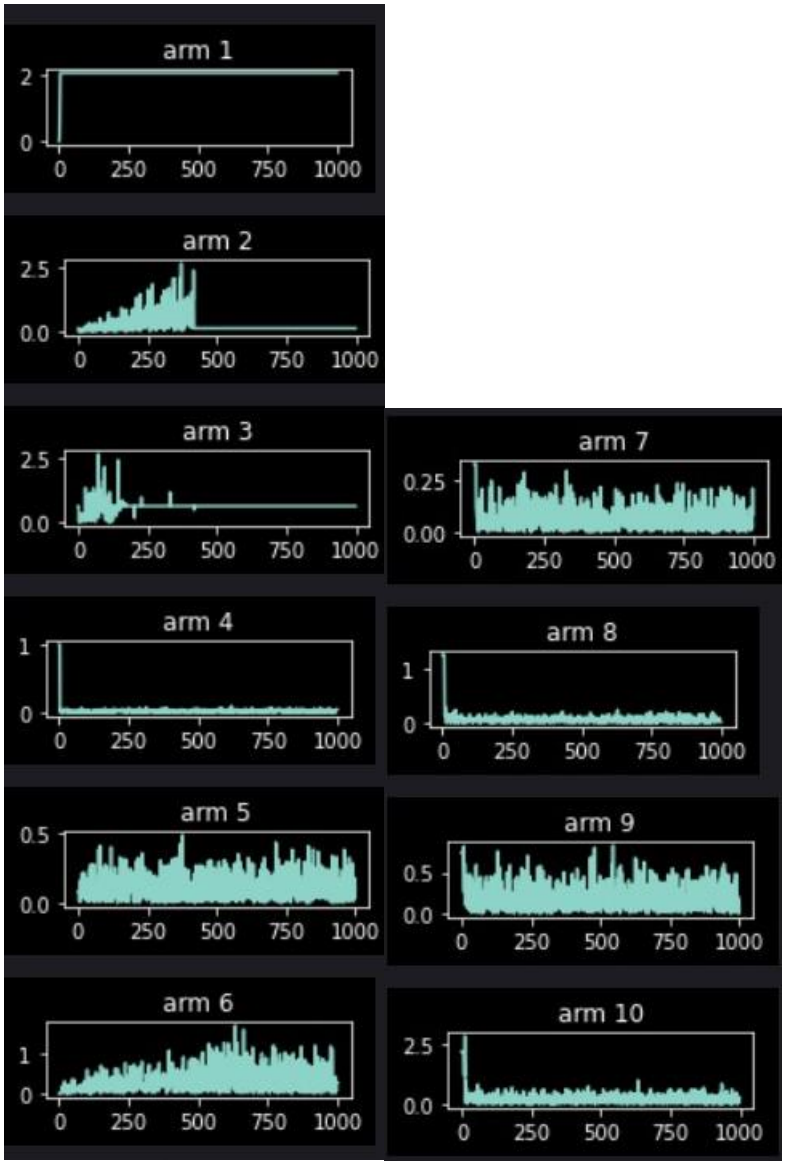
Answer 1.





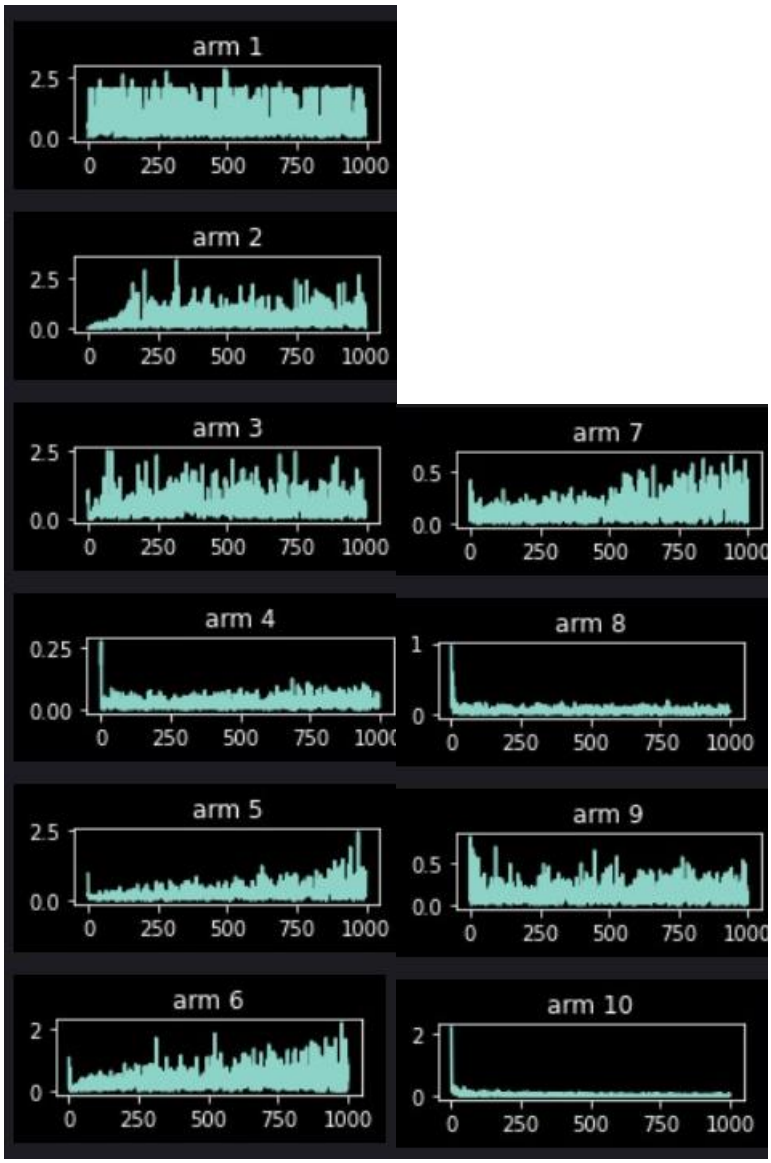
e=0 case

Absolute error for each arm between expected reward and true mean.



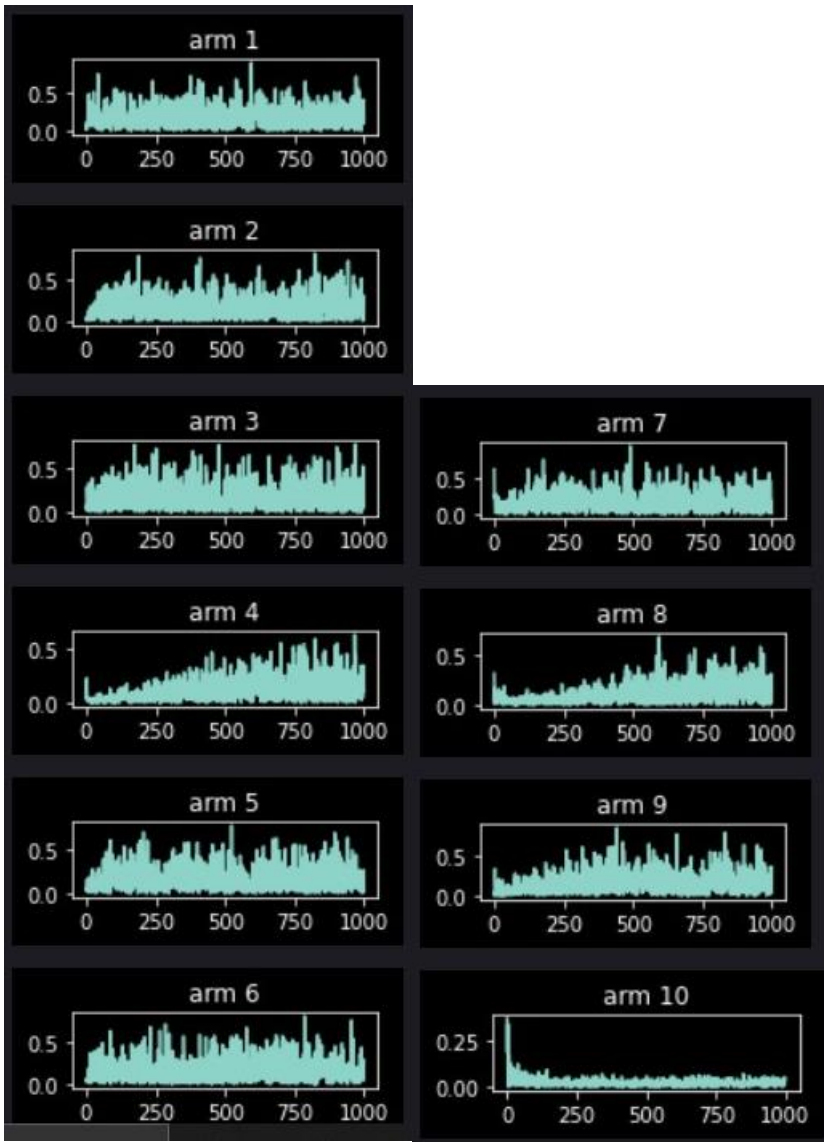
e=0.01

Absolute error for each arm between expected reward and true mean.



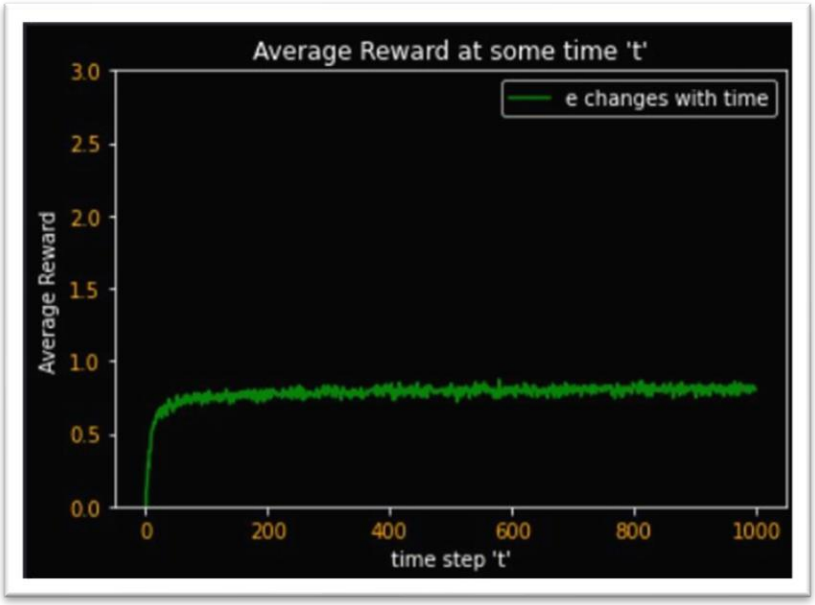
e=0.1

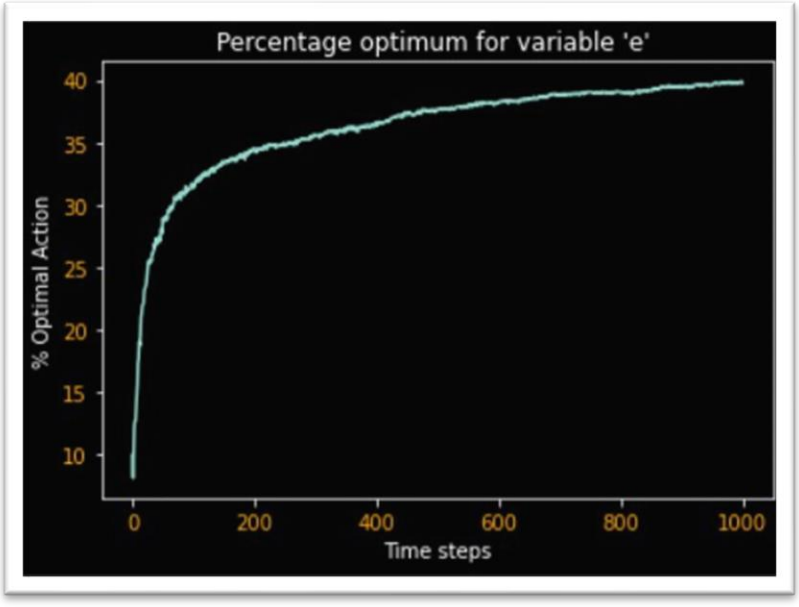
Absolute error for each arm between expected reward and true mean.



For variable ‘e’ following eq 2.7

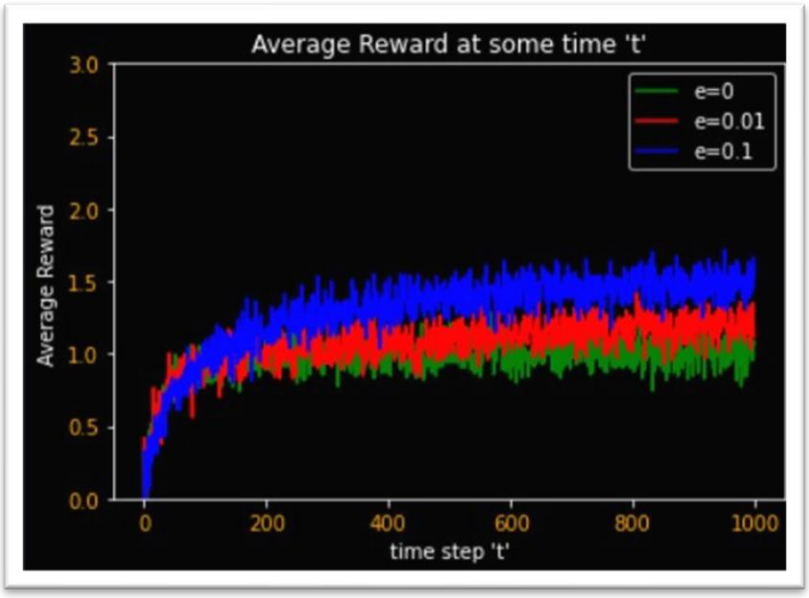
e=1/(t+2) is the update rule followed

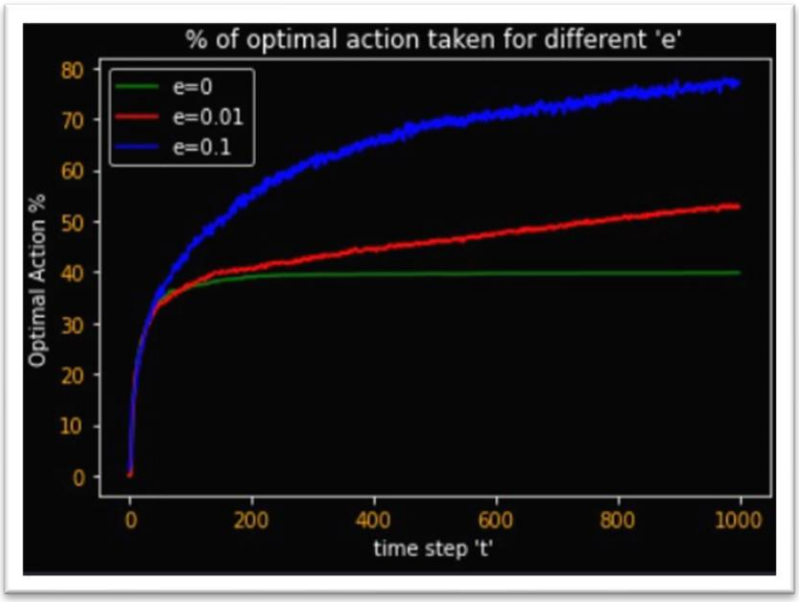




Question 2. Repeat Question 1 for when the variance corresponding to each arm is 4 instead of 1.

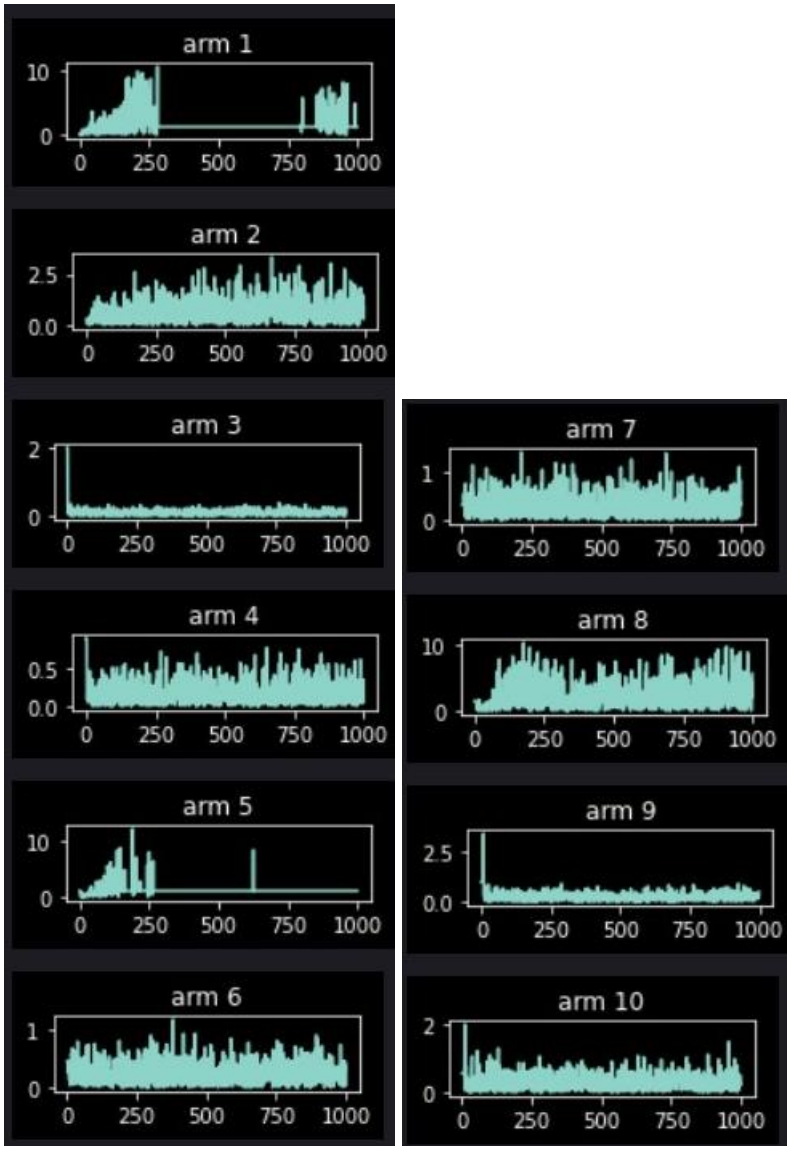
Answer 2.





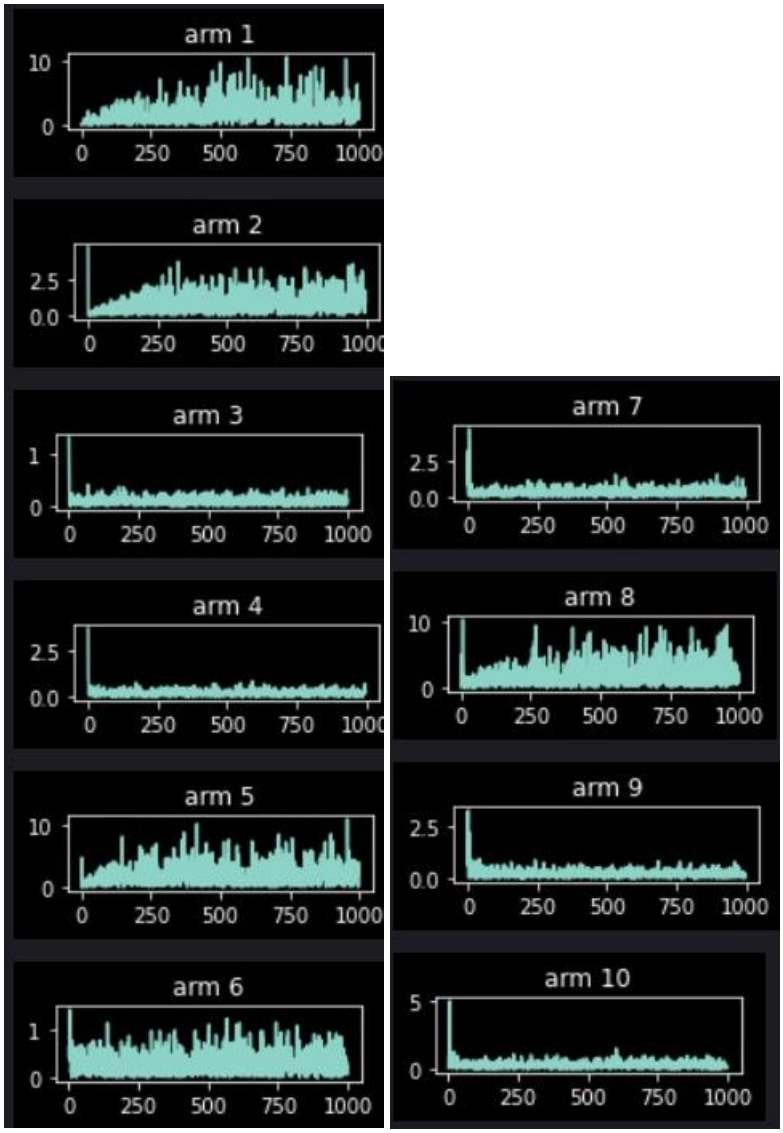
e=0

Absolute error for each arm between expected reward and true mean.



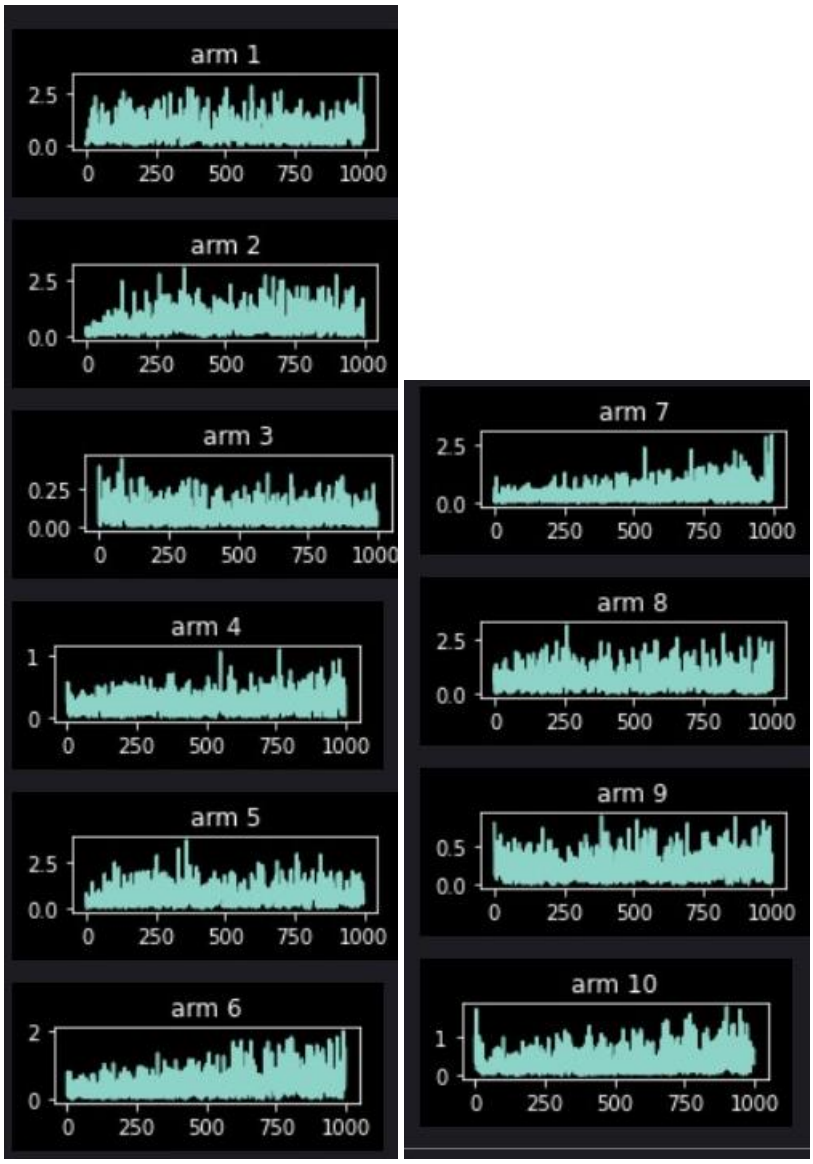
e=0.01

Absolute error for each arm between expected reward and true mean.



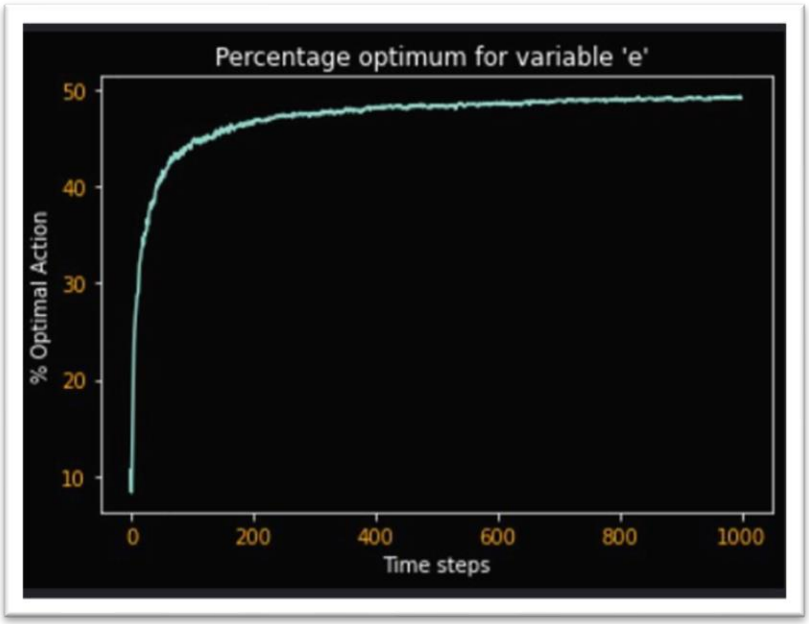
e=0.1

Absolute error for each arm between expected reward and true mean.



For variable ‘e’ following eq 2.7

e=1/(t+2) is the update rule followed

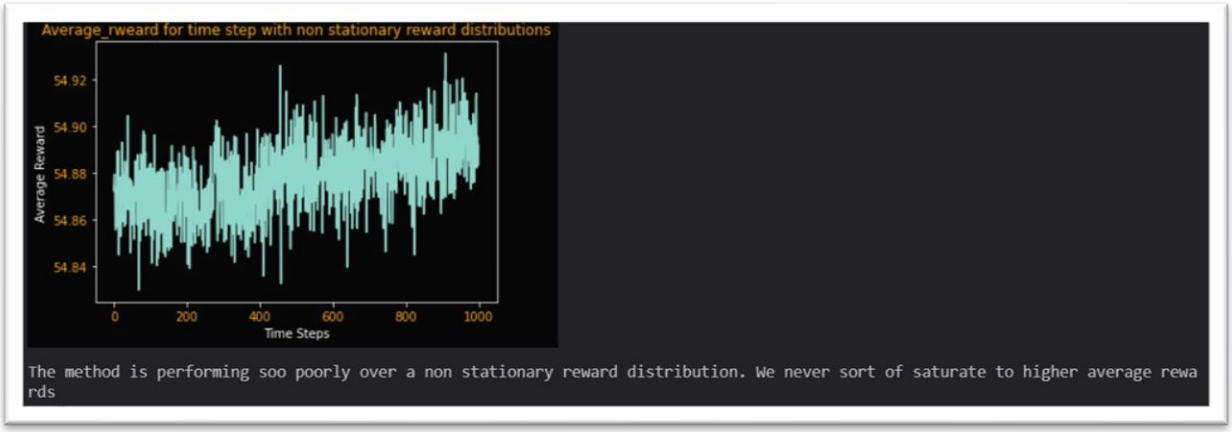


QUESTION 3 and 4 are theoretical and are towards the end of this pdf

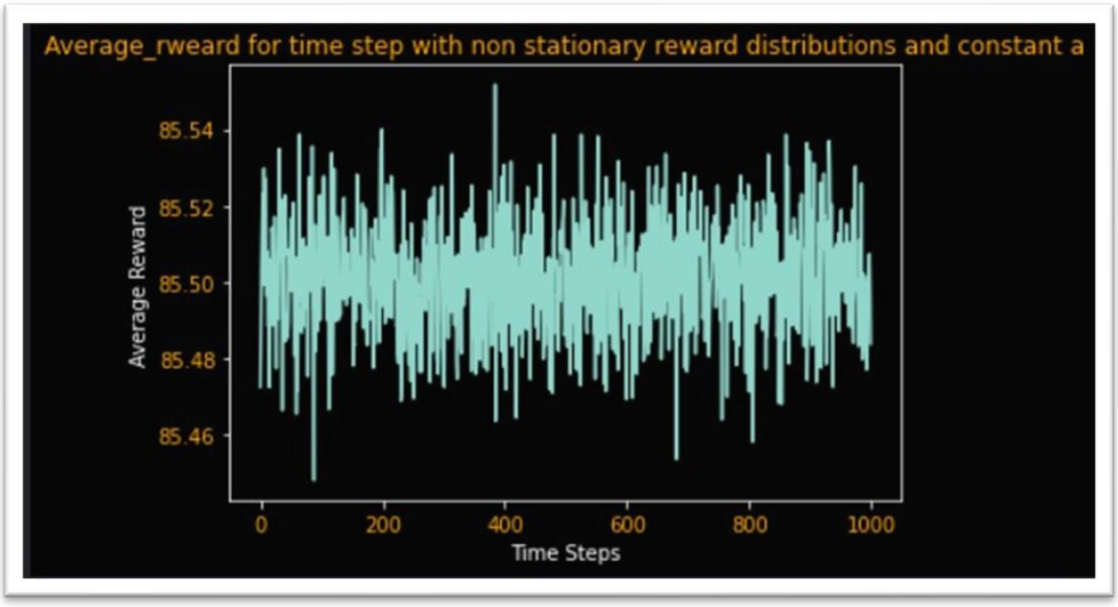
Question 5. Do Exercise 2.5.

Answer 5.

Action value method using sample averages, incrementally calculated



With Constant Step size parameter, a=0.1

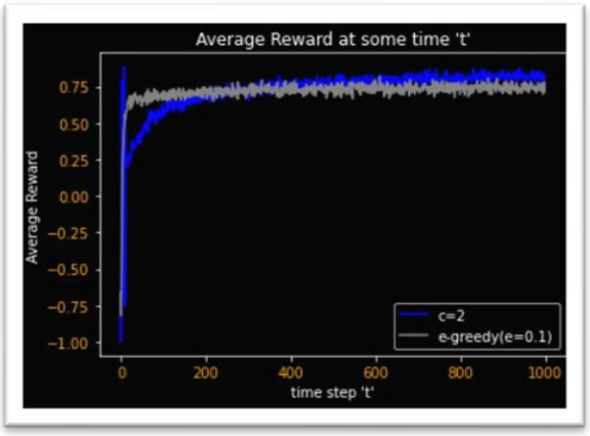


For constant alpha our expected rewards are much higher than that of incremental. Also, the values don't tend to saturate. The problem lies in the fact that true mean of each arm changes at every time step. So, we can never be sure about an expected reward even after infinite trials. For alpha as constant, we get better expected rewards. Please see the scale on y axis carefully!

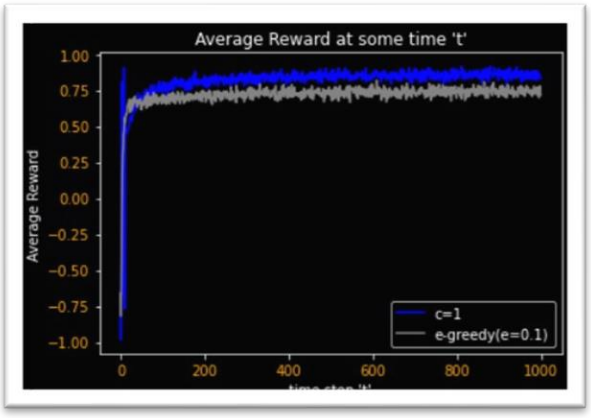
Question 6. Generate Figure 2.4 and solve exercise 2.8. Repeat generation of the figure and solve the exercise for c = 1 and c = 4 too.

Answer 6.

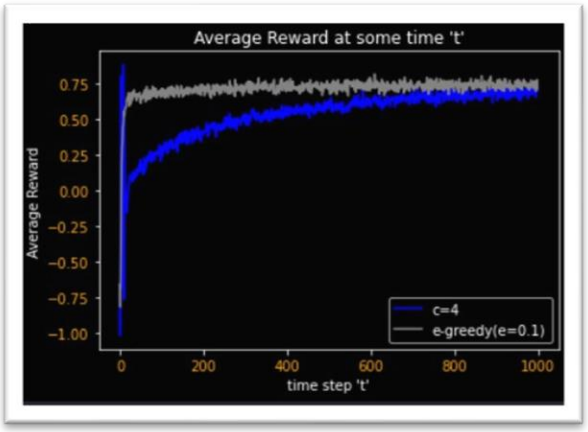
C=2



C=1

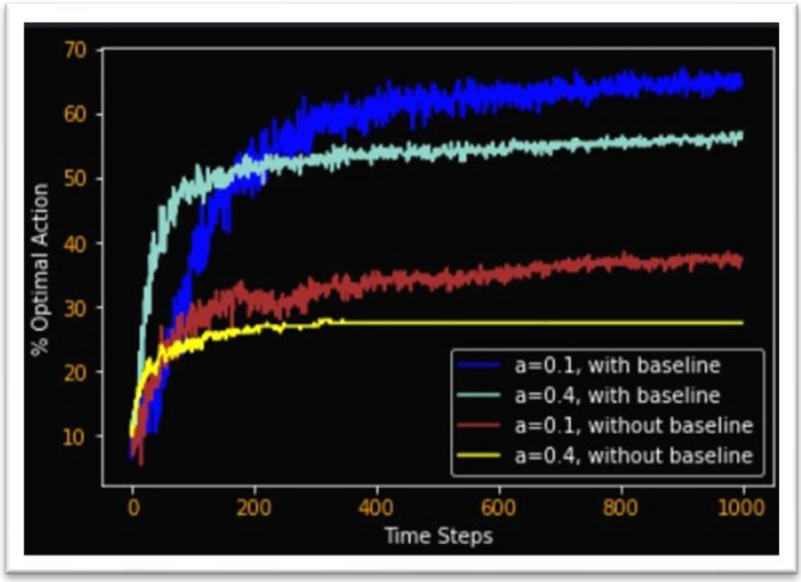


C=4



Question 7. Generate Figure 2.5.

Answer 7.



The following code is computationally expensive and thus has been run for 200 trials and not 2000 trials. Time steps are 1000 only in each trial. Thus, the data is more distorted in comparison to that of the book. However, the plot follows the same trend as that of the figure.

EXPLANATIONS

Question 1)

From the plots we can conclude that for a lesser epsilon, as the number of steps increase, we get more closer to the higher expected award or we would pick thee optimum action more often. Detail solution of this problem statement is in question-3 (refer to the theoretical section).

Question 2)

Follows a similar trend except for the fact that the plots are noisier. Obviously, they will be noisier as the distribution of rewards for each arm has 4 times more variance, so the rewards that you get will have a thicker distribution in comparison to question 1. However, the trend of picking the optimum arm is maintained and e=0.01 wins.

Question 5)

When the distributions are non-stationary, constant parameter alpha gives way expected reward. This is because, when distribution is changing with time, then constant alpha tends to give more preference or weightage to recent rewards; rather than the rewards that were received way back in time. However, incremental method gives equal weight to every reward received till now. Our distributions follow a trend, where they are incrementing at every ‘t’, so it makes sense to give higher preference to rewards that were receiver sooner, because their true mean and their reward distribution is more likely to be closer to that of the current one. So, the information that the rewards we received sooner will provide is better than the information earlier rewards would give.

Question 6) (Solution of 2.8 included)

In UCB we use our first 10-time steps to initialise every arm. Since, n=0 for each arm in the beginning has the ln(t)/n term blows up and we have to pick each arm once in the beginning. After that on the 11^{th} step the story is different. Here we would pick a reward of the arm which gave the highest reward in first ten tries (each arm has been picked once when we reach 11^{th} step). This time for all 2000 trials we will be sort of in sync. Since every trial would pick the arm that gave the highest reward in the 11^{th} step. So, all agents would in sync pick a very good arm (not certainly optimum, but good) and hence the average reward over the 2000 trials for that time step would go up. But for the subsequent steps it would be layer as this sync would end up. However, later when the expected values will start to converge, we would go above the spike. The spike is merely because of the design of UCB.

Please note that it is not guaranteed that the spike would be the maximum at the 11^{th} step. The probability of that happening isn't one. It is very well possible that at time step 7, we pick arm 7 and somehow, in all 2000 trials we got a very good value for arm 7 reward. And then on the 11^{th} step we picked arm 7 again, because it has the highest Q value so far, but say the reward we got was -ve in majority of trials. In this case the spike would be on step 7 and not 11. Hence, it is not guaranteed that the spike will occur on step 11 only, but it is just more likely to happen.

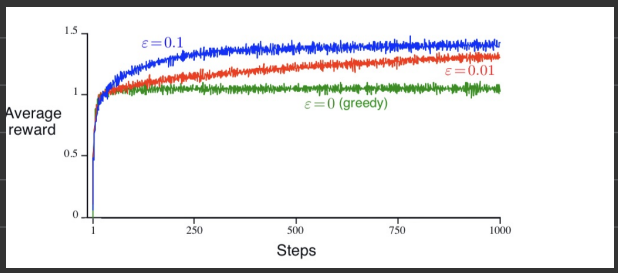
As we increase c to 4, the UCB algorithm performs worse than e-greedy. Because having such a high constant value, will sort of make the other part (Qt(a)) less relevant, so the algorithm performs poorly for a very high c value.

Question 7)

First of all, this algo takes time to run for 2000 trials with 1000 steps each. So, I ran it for 200 trials. I got the plots as shown in the figure, but they are noisier because of lesser trials.

We can clearly observe that with a baseline we perform better, in comparison to no baseline. Having a smaller alpha gave a better result, which basically meant that we stepped in the direction of the grad ascend slowly. Hence, in the beginning higher alpha was performing better as we were taking larger steps towards the predicted direction. But later, after more steps when the means start to converge, taking smaller steps is better, as we re run our calculations after every step. If we take a larger step we might cross over and have to come back and sort of solve the problem in stochastic manner.

Que - 3)



From the figure we can clearly observe, that as t increases the red curve approaches the blue curve & eventually, it will go above it.

This makes sense as well.

Let i^* denote the true optimum arm.

Let $i_\circ$ denote the greedy arm of the trial at time step 't'.

We know,

$$P[i_\circ] = \frac{\epsilon}{K} + (1 - \epsilon)$$

→ Probability of choosing the greedy arm

$$P[i] = \frac{\epsilon}{K}, \quad i \neq i_\circ$$

→ Probability of choosing any arm, except the greedy arm.

In the long run, we can be sure about the fact, that, we have learned the True mean of the reward distribution of every arm.

∴ In a long run, the greedy arm would infact be the optimum arm.

$i_o = i^*$ in the long run.

In order to maximize our reward we would want to pick the Truely optimum arm i^*.

$$P[i^o] = P[i^*] = \frac{\epsilon}{k} + (1 - \epsilon)$$

In the long run.

So, if we can minimize 'ϵ' then we can maximize the P[i*] & hence we will get best expected rewards.

However, ϵ ≠ 0. If ϵ is zero then we cannot be sure about the fact that we have learned the true mean of the reward distribution of every arm. If we never explored then the distribution of greedy arms will decide whether we will ever pick a new arm or not. In this case our estimates will be noisy

even when $t \to \infty$.

we need the following 4 conditions to maximize our expected Reward in the long run.

- $i^{\circ} = i^{*}$
- $\max\left(P[i^{*}]\right) = \max\left(\frac{\epsilon}{\kappa} + (1-\epsilon)\right)$
- $\min(\epsilon)$
- $\epsilon \neq 0$

$\therefore$ In the long run $\epsilon = 0.01$ will perform the best

In the long run :-

* for $\epsilon = 0.01$

$$P[i^*] = \frac{1}{100K} + 99 \qquad \text{(Case 1)}$$

* for $\epsilon = 0.1$

$$P[i^*] = \frac{10}{100K} + 90 \qquad \text{(Case 2)}$$

for $\epsilon = 0$ (Case 3)

We can't be sure about what the truly optimum arm is. So we can't say $i = i^*$ & hence $\epsilon = 0$ is the worst.

On comparing Case 1 & 2 we can see that the $P[i^*]$ in Case 1 is $\left(9 - \frac{9}{100K}\right)$ more as compared to Case 2.

In other words,

We are more likely to pick the optimum arm for $\epsilon = 0.01$ as compared to $\epsilon = 0.1$ by $\left(9 - \frac{9}{100K}\right)$, where K is the number of arms.

$$E[R_t] = \phi^*(i^*)\, P(i^*) + \phi^*(i)\, P(i)$$

As $t \to \infty$

$\phi^*(i^*)$: True mean of the optimum arm

$\phi^*(i)$: True mean of all other arms

Let $E[R_{t_1}]$ denote for $\epsilon = 0.01$

$E[R_{t_2}]$ denote for $\epsilon = 0.1$

as $t \to \infty$

$$E[R_{*1}] = \phi^*_{(i^*)}\left(99 + \frac{1}{100n}\right) + \phi^*_{(i)}\left(\frac{1}{100n}\right)$$

$$E[R_{*2}] = \phi^*_{(i^*)}\left(90 + \frac{10}{100n}\right) + \phi^*_{(i)}\left(\frac{10}{100n}\right)$$

$$E[R_{*1}] - E[R_{*2}] = \phi^*_{(i^*)}\left(9 - \frac{9}{100n}\right) + \phi^*_{(i)}\left(-\frac{9}{100n}\right)$$

$$\boxed{\phi^*_{(i^*)}\left(9 - \frac{9}{100n}\right) - \phi^*_{(i)}\left(\frac{9}{100n}\right)}$$

$\phi^*_{(i^*)}$ is greater than any other $\phi(i)$ as i^* is the true optimum arm.

$$\& \quad 9 - \frac{9}{100N} > \frac{9}{100N}$$

$$\therefore$$

$$E[R_{*1}] - E[R_{*2}] > 0$$

& thus we can conclude that our Average reward is maximized by

$$\phi^*_{(i*)}\left(9 - \frac{9}{100N}\right) - \phi^*_{(i)}\left(\frac{9}{100N}\right)$$

in the long run if we choose $\epsilon = 0.01$ instead of $\epsilon = 0.1$

End of Question 3.

Question 4 from next page!

Iterative Calculations

Let's define a new average value, $Q_n(i)$. Here 'n-1' represents the no. of time arm 'i' has been picked while $Q_t(i)$, where 't' is the time step.

$R_n \rightarrow$ Reward of picking an arm n^{th} time.

$n \in [1, 2, \ldots N]$ $\quad Q_n \rightarrow$ Average of picking the arms n-1 arms.

Initialize $Q_1(i) = 0$;

Now let's say whenever we pick arm 'i' for the first time we get Reward: R_1. Similarly when we pick it for $(n-1)^{th}$ time we get R_{n-1}.

$\because n-1=1$ ↖ This is the first time we picked this arm. Now, we can define $Q_n(i)$ for $n=2$ as arm 'i' has been picked once.

$$Q_2(i) = R_1$$

& for rest :-

$$Q_3(i) = \frac{R_1 + R_2}{2}$$ → Reward we get when we pick arm 'i' the 2nd time.

$$Q_4(i) = \frac{R_1 + R_2 + R_3}{3}$$ & so on...

$$Q_n(i) = \frac{R_1 + R_2 + R_3 + \cdots R_{n-1}}{n-1}$$

n^{th} update in $Q_n(i)$ depends on $(n-1)$ rewards from that arm. Or n^{th} Update in $Q_n(i)$ depends on arm 'i' being selected $(n-1)$ times.

$$Q_n(i) = \left(\frac{R_1 + R_2 + \cdots R_{n-2}}{n-1}\right) + \frac{R_{n-1}}{n-1}$$

$$Q_n(i) = \left(\frac{R_1 + R_2 + \cdots R_{n-2}}{n-1}\right)\frac{n-2}{n-2} + \frac{R_{n-1}}{n-1}$$

$$Q_n(i) = \left(Q_{n-1}(i)\right)\left(\frac{n-2}{n-1}\right) + \frac{R_{n-1}}{n-1}$$

Equal

$$Q_n(i) = Q_{n-1}(i) + \frac{1}{n-1}\left(R_{n-1}(i) - Q_{n-1}(i)\right)$$

sample mean

$$Q_n(i) = Q_{n-1}(i) + \frac{1}{n-1}\left(R_{n-1}(i) - Q_{n-1}(i)\right)$$

sample mean

Let's put $n = 2$ in the equation and Check.

$$Q_2(i) = Q_1 + \frac{1}{2-1}\left(R_1(i) - Q_1(i)\right)$$

$$Q_2(i) = \cancel{Q_1(i)} + R_1(i) - \cancel{Q_1(i)} \Rightarrow Q_2(i) = R_1(i)$$

$$Q_3(i) = Q_2(i) + \frac{1}{3-1}\left(R_2(i) - Q_2(i)\right)$$

$$Q_3(i) = R_1(i) + \frac{1}{2}R_2(i) - \frac{1}{2}R_1(i)$$

$$Q_3(i) = \frac{R_1 + R_2}{2}$$

We can clearly see that neither Q_3 nor Q_2 depend on Q_1. We have a recursive relation $\therefore$ all $n > 3$ will depend on Q_2 eventually.

But Q_2 doesn't depend on Q_1, so neither will any other values.

So, we can safely conclude that sample mean is independent from the choice of initial Q values.

Now, let's use a constant step size 'α' instead of $\frac{1}{n-1}$

$$Q_n(i) = \left(Q_{n-1}(i)\right) + \alpha\left(R_{n-1} - Q_{n-1}(i)\right)$$

$\alpha \in [0,1]$

$$Q_n(i) = \left[(1-\alpha)\, Q_{n-1}(i)\right] + \alpha\left[R_{n-1}(i)\right]$$

Exponentially weighted Average

This way we are updating the average of reward from chosing an arm 'i' n-1 times.

Default value

Reward when we picked arm i the first time

$$Q_2(i) = (1-\alpha)\, Q_1(i) + \alpha\, R_1(i)$$

Reward of picking arm i for the 2nd time

$$Q_3(i) = (1-\alpha)\, Q_2(i) + \alpha\, R_2(i)$$

$$Q_3(i) = (1-\alpha)\left[(1-\alpha)(Q_1(i)) + \alpha\, R_1(i)\right] + \alpha R_2(i)$$

$$Q_3(i) = \left((1-\alpha)^2\, Q_1(i)\right) + \alpha\left((1-\alpha)\, R_1(i) + R_2(i)\right)$$

In General

$$Q_n(i) = (1-\alpha)^{n-1}\, Q_1(i) + \alpha \sum_{j=1}^{n-1} (1-\alpha)^{n-1-j}\, R_j(i)$$

Exp Weighted Average

We can clearly see that the choice of initial Q value affects the Q values in the future as whichever next an arm will be selected.

$$Q_n(i) = (1-\alpha)^{n-1} Q_1(i) + \alpha \sum_{j=1}^{n-1} (1-\alpha)^{n-1-j} R_j(i)$$

Exp Weighted Average

If the value of α is small, then the Co-efficient of $Q_1(i)$ will be larger.

As $(1-\alpha)$ will be more for smaller 'α'. ∴ The effect of Initial bias will increase if we reduce 'α'.

If we need to have Constant 'α' & no dependence on Initial bias, then we need to increase 'α' as much as possible.

$\alpha \neq 1$ as then nothing is updated but $\lim_{\alpha} \alpha \longrightarrow 1$ is what we need.