Programming- MAB

Ajinkya Ambatwar
EE16B104
Dept. Of Electrical Engineering

March 3, 2019

Explanation and Plots

0.1 ϵ-greedy

For ϵ-greedy, the plot looks like this -

As we can see from the plots, the average reward for $\epsilon = 0$ is the saturated at a local optima which is a sub-optimal point. This is because this case lacks any exploration and quickly settles to an action with the highest instantaneous reward. This action might not be the optimal hence the optimal action percent is also bad for this case.

Comparing $\epsilon = 0.1$ and $\epsilon = 0.01$ cases, we can observe that for given number of iterations $\epsilon = 0.1$ stays at higher average reward and optimal action percentage. This is because for higher ϵ the algorithm will explore more and hence find the best action earlier but won't exploit it much because of more exploratory nature. Hence initial performance of $\epsilon = 0.1$ is better but in longer terms $\epsilon = 0.01$ will overtake it as this value of ϵ will make the algorithm to take the best action more often than the other actions.

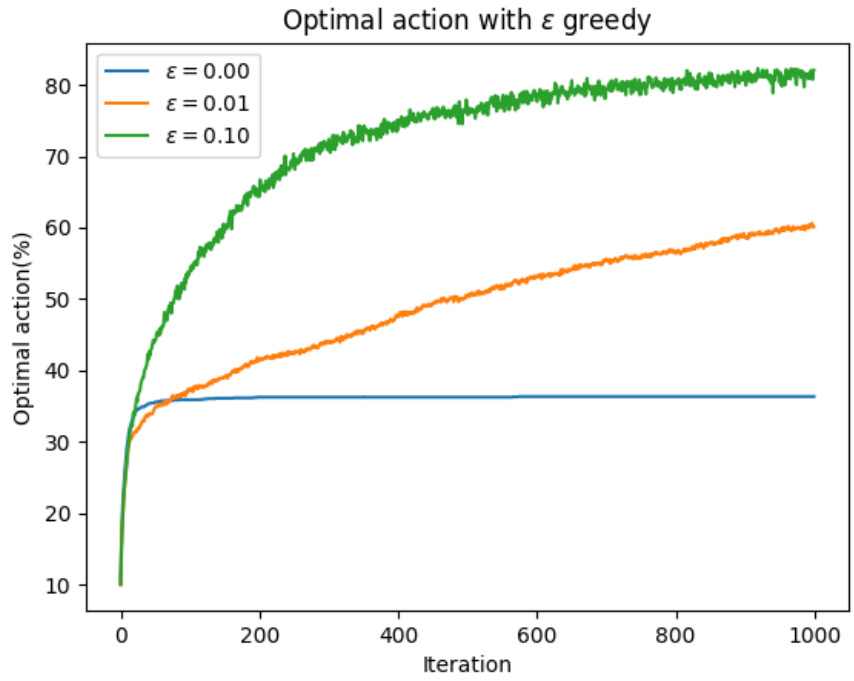


Figure 1: Plots for $\epsilon-greedy$ algorithm

0.2 Softmax

For Softmax action distribution, the plots look like this -

For softmax distribution, as the value of temperature increases the exploratory nature of the algorithm increases but it simultaneously reduces the exploitory nature. Hence for very high value of temp(Temp = 100), algorithm just goes on exploring and doesn't settle with a high rewarding arm. For smaller values of Temp, the algorithm gives enough weightage to exploration and exploitation and finds the right arm with higher reward.

For intermediate temperature value(Temp = 1), the algorithm will explore more but will exploits very less and hence reach a suboptimal point.

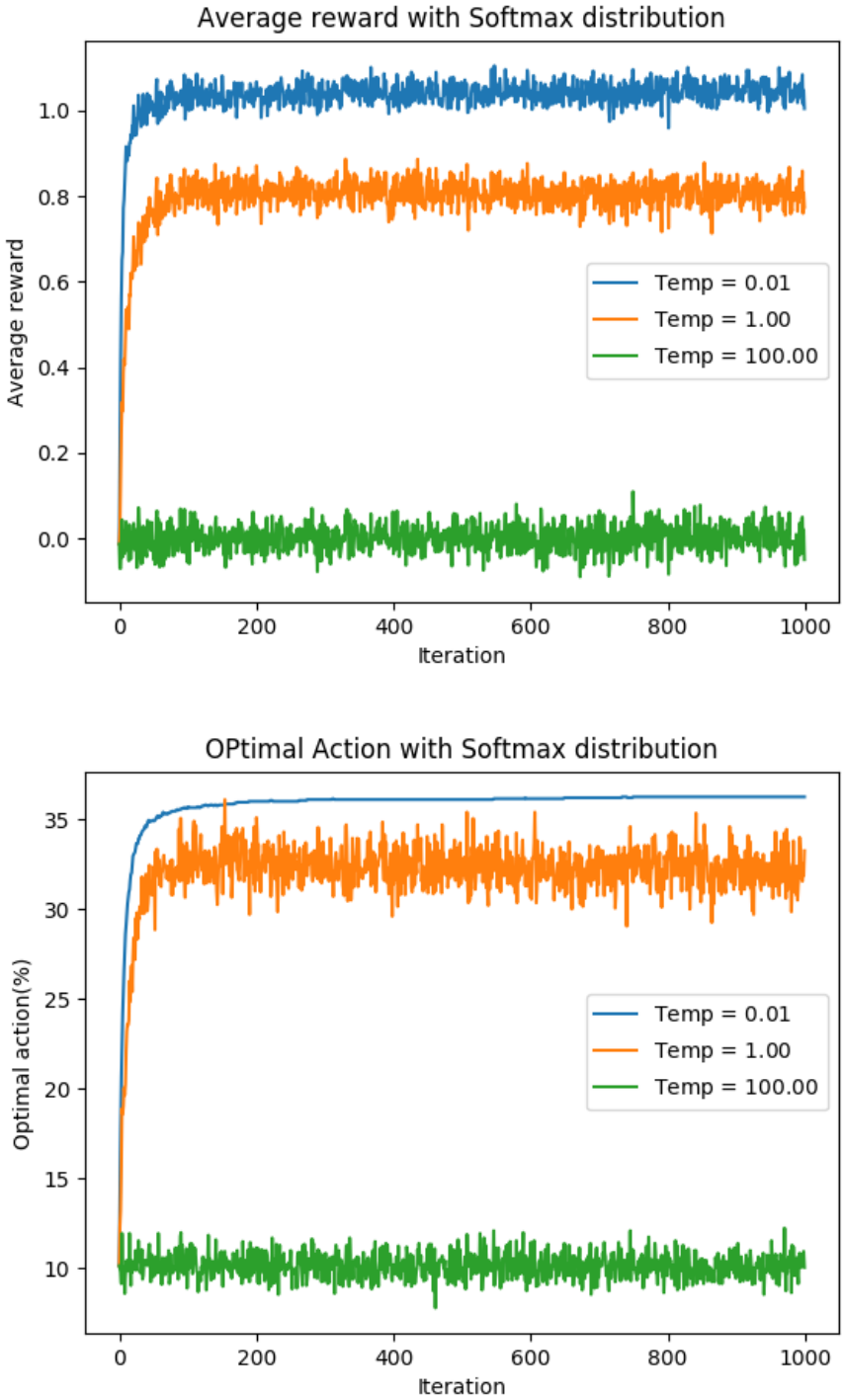


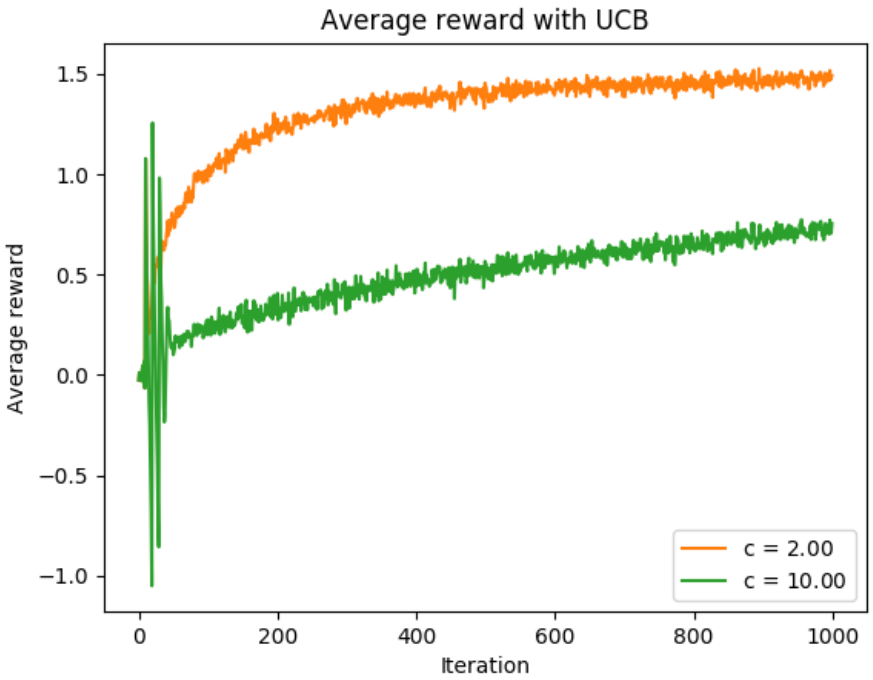
Figure 2: Plots for Softmax distribution

0.3 UCB

$$Q_{estimated}(a) = Q_t(a) + c\sqrt{\frac{ln(t)}{N_t(a)}}$$

For UCB, the plots look like this -

The parameter c indicates the degree the exploration. For c=10, the algorihtm will have more uncertainty in the estimation of true expected reward and will oscillate more. Because of more variance in the value it might pick an action with lower expected reward but larger value of $Q_{estimated}$ hence will get stuck an suboptimal arm as is seen from the graph.



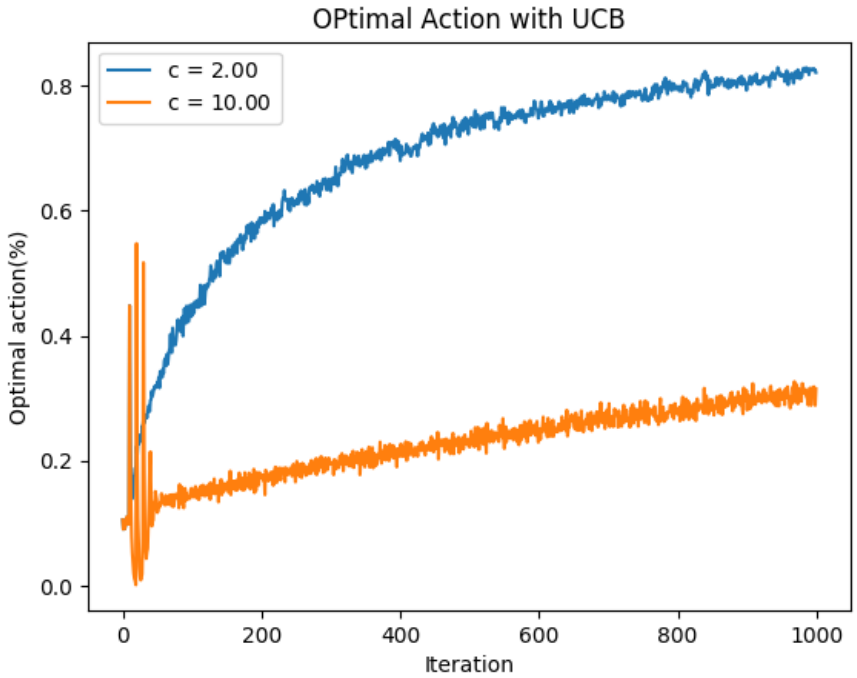
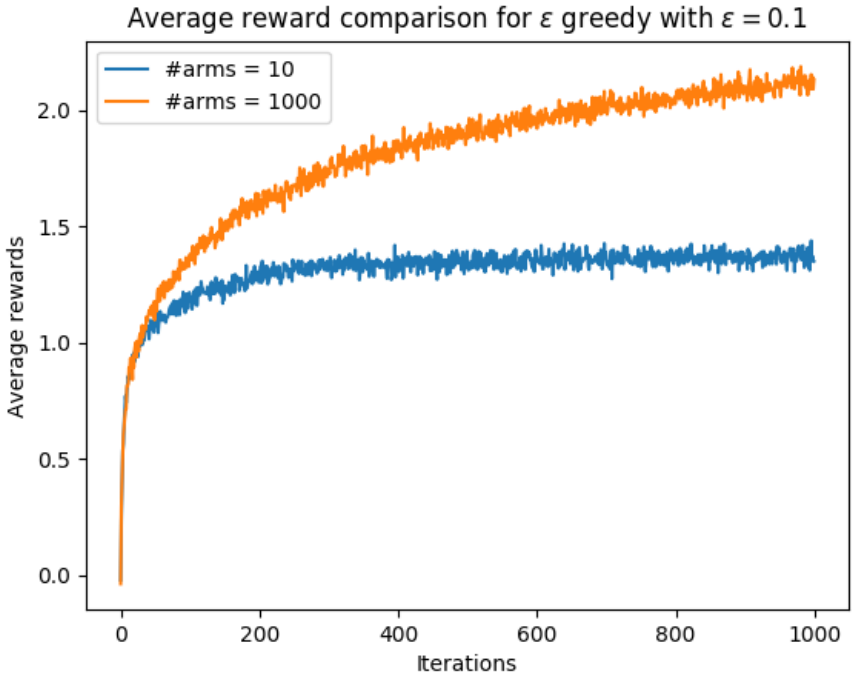


Figure 3: Plots for UCB

0.4 Arms value comparison

The comparison plots for the algorithms for k_arms = 10 and k_arms = 1000 look like this

0.4.1 $\epsilon - greedy$



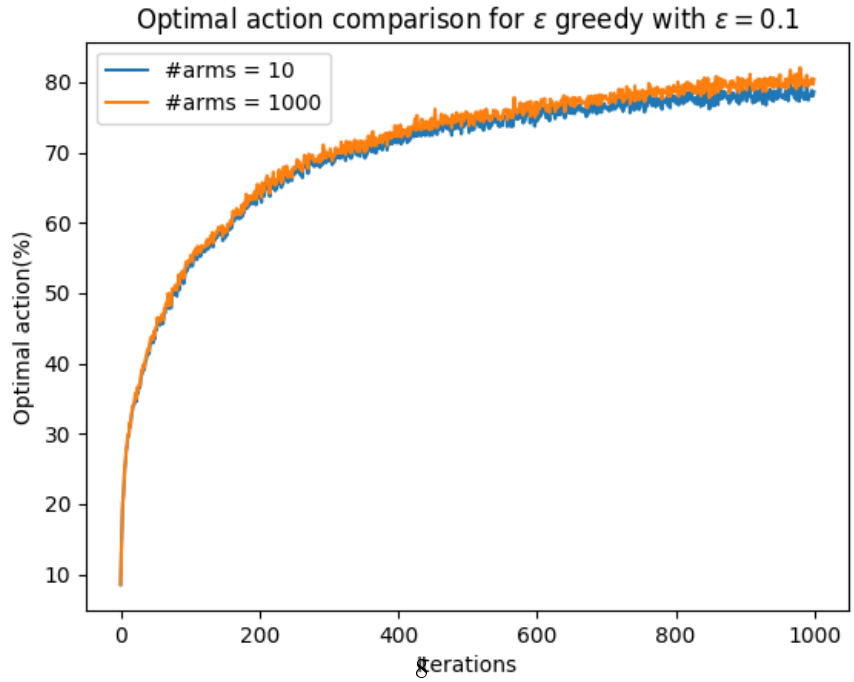
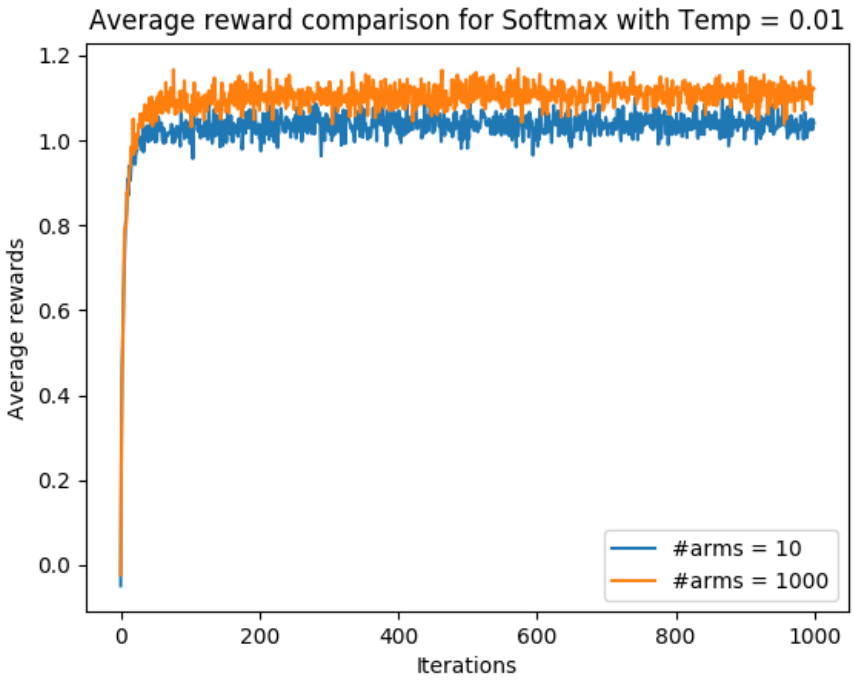


Figure 4: Comparison for $\epsilon - greedy$
As can be seen form the plot the average return for more number of arms is more as larger number of arms means more arms will have higher expected rewards.

0.4.2 Softmax



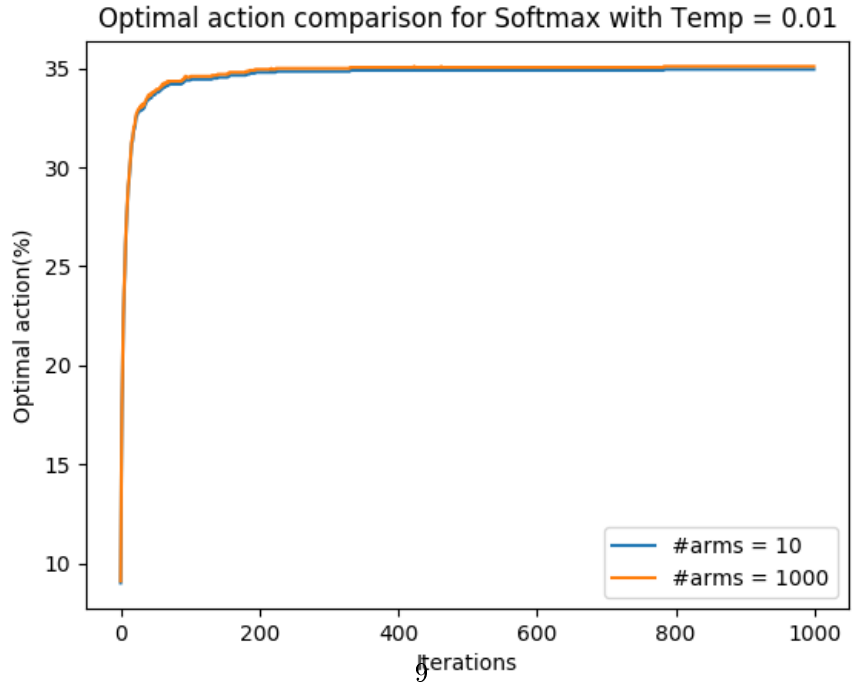
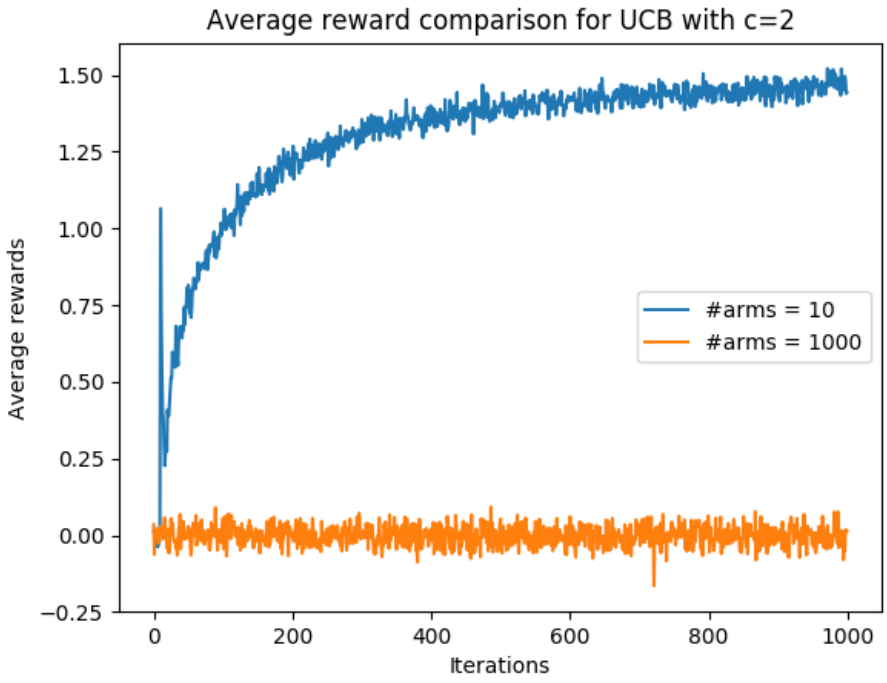
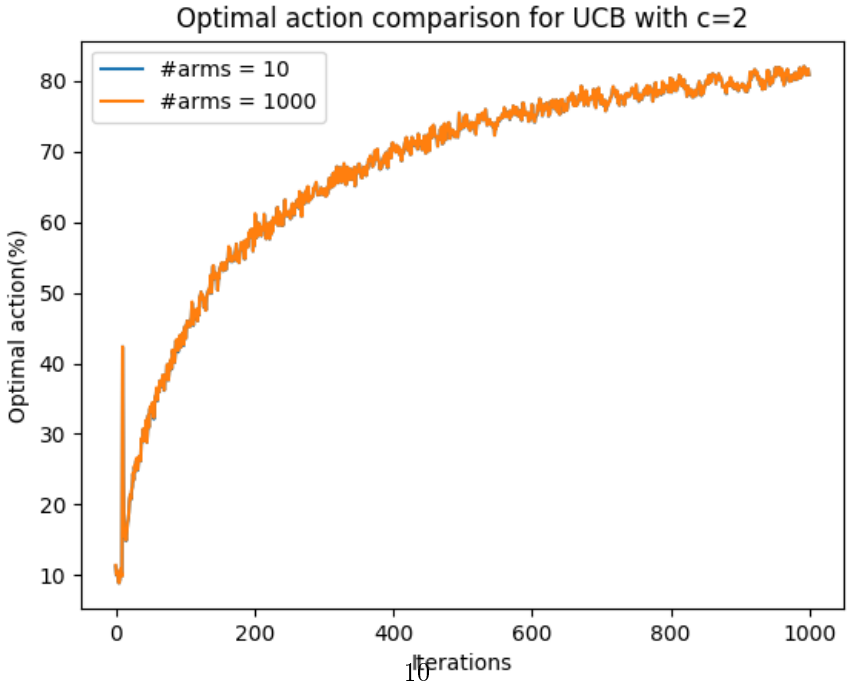


Figure 5: Comparison for Softmax
For softmax the effect of arms is very less as the probility is a function of $exp(Q_t)$ and is divided by the sum of it. Hence the probilit values will be more of less same independent of number of arms.

0.4.3 UCB





With very large number of arms and all other parameters kept same, the exploratory nature of the algorithm will overcome the exploitation and the rewards will decrease significantly. Regarding the optimal action, it may be independent of the number of action. Due to some unknown error, the optimal action plot for 10 arms is not plotted here.

Figure 6: UCB comparison

0.5 Advertise suggestion

From the data given at every iteration -

Algorithm 1 Advertisement recommendation

1: W(a) = [w1 w2 w3 w4]
2: $\pi_t(a) = Softmax(W(a) * X^T)$
3: Pick an action with probability $\pi_t(a)$
4: Get a reward
5: Do the reinforcement update on W
6: Test the updated W on 450 iterations
7: Get the average reward for 450 iteration

The reinforcemnt update for W is given by

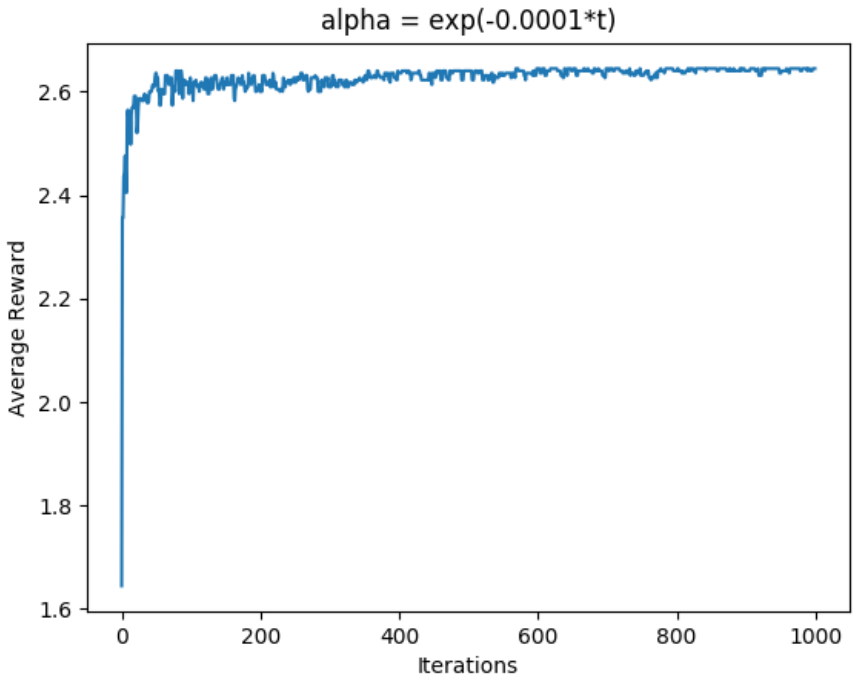
$$W_{t+1}(a) = W_t(a) + \alpha(R_t - b_t)(1_{a=A_t} - \pi_t(a))X_t$$

Where X_t is the feature vecture for the sample/user picked up at time t and b_t is the baseline set at the avaerage total reward till time t. $1_{a=A_t}$is one only when action picked up at time t is a else 0.

The parameters chosen for simulation are -

1. TRAIN_STEPS = 10000
   (a) Increasing it further to 20000 didn't given much improvement for performnace
2. LEARNING_RATE α
   (a) $\alpha$is made a function of time step as
   $$\alpha_t = exp(-0.0001t)$$
      i. This makes sure that when initially α is more the updates will be made faster
      ii. As time progresses α becomes smaller making the further updates smaller and improving convergence rate as well as stability

The average action plot looks like this -



The reward got saturated at around 2.64 and didn't show much improvement with change in parametes.