Lab Report

Lab 2

1. Sourav Ganguly, 2. Gudepu Venkateswarlu, 3. Jyoti

19/01/2022

Reinforcement Learning(RL)

Solutions of the given problems

1. This task is to implement 5 bandit algorithms on a testbed, compare and rank their performance and come up with a report on the performance evaluations with a clear description of the settings considered. When you compare your algorithms, clearly write your observations. The test bed could be created along similar lines as in banditsComparison.pdf which is attached with the mail. The banditComparison.pdf assumes normal reward

   distribution for the arms. Recall in class we have discussed, Bernoulli reward distribution. For this assignment, you consider Bernoulli reward distributions and three

   different settings on the number of arms K=2,5 and 10. If any group is interested, they are free to explore normal reward distribution as well. In the case of normal reward distribution, for UCB algorithm refer to Figure 4 in auer.pdf with this mail and for Bayesian update in Thompson sampling refer to bayesNormal.pdf. Bonus points will be awarded if any group attempts normal reward distribution. For performance evaluation metric for comparison, refer banditsComparison.pdf. It uses three criterions. You can use the same criterions for performance comparisons and also any other reasonable performance metrices to compare different algorithms. The following are the five Bandit Algorithms whose performance needs to be compared.

   SOLUTION:

   The Bandit problem is a very well-known problem, which is used to make a decision best suited for making maximum gain. So, we can consider a series of slot machines, precisely K slot machines, that can be played upon to maximize the expected total reward. Saying that, we are trying to implement a series of algorithms with the general aim to solve the problem. Ultimate aim is to find the number of times optimal arm has been played and the regret produced at each time instant.

   Assumptions and Considerations:

   (a) The number of Steps considered is 20000 while finding the individual plots. Combinedly it was run for 20 runs in case of the bernoulli reward bandit. The same experiment was repeated for Gaussian distribution reward system where number of Steps considered is 30000 and number of runs =50. Here the mean = random number between 0 and 1 and standard deviation 1.

   (b) The distribution assumed for the reward is bernoulli distribution and gaussian distribution.

   (c) Standard packages like numpy, matplotlib and scipy was used only.

   Algorithms:

   Value based Approaches:

   (a) **Epsilon-Greedy (Fixed):** Epsilon Greedy technique is utilised to solve the Bandit problem. It is one of the simplest algorithm and works in the way that, we choose a particular ϵ value which is generally small. Now we generate some random choice for some $\epsilon * N$ times, when we actually explore the arms. Now in the beginning all the arms will be equi-probable. If we had the knowledge which arm is the optimal arm, then definitely, we would have pulled that arm. But since there is no way of knowing which the optimal arm, we have to try as many arms

as possible. Now after trying the arms randomly from sufficient amount of time, we now check our knowledge base as to which is the arm producing maximum reward and then we greedily choose that arm from that instant onwards. One check is that we need some policy to check which is the best arm possible. So, we calculate the estimated mean of the arm choosen. Ok now we talk about the choice of Epsilon.

i. If $\epsilon = 0$ then it similar to just greedy approach, i.e. we shall all the arms once and then based on that knowledge only select the optimal arm greedily.
ii. If ϵ is very small it means that we are exploring less. Hence the output may vary accordingly. (As shown below)
iii. If ϵ is very large, it means we are exploring more than required number of times. We then miss the goal of maxmizing the rewards.

At each time step estimates for each action are given by:

$$Q_t(a) = \frac{(\Sigma_{i=1}^{t-1} R_i.1_{(A_i=a)})}{\Sigma_{i=1}^{t-1} 1_{(A_i=a)}}$$

As denominator approaches infinity, the value approaches to true action value.

(b) **Epsilon Greedy(Variable Epsilon):** In the previous approach, we saw the cumulative regret curve is varying linearly. Or in simple words it is having linear time complexity (step complexity according to plot) when we vary the cumulative regret in regards with respect to time. Now in the variable epsilon greedy policy, we observe the step size to vary with respect to time. Hence, leading to variable Epsilon-Greedy policy. It produces logarithmic time complexity i.e. when we vary cumulative regret with respect to time, it varies logarithmically. Now suppose $\epsilon_t = \frac{c}{t}$
t varying as 1,2,3,...T

$$\begin{aligned} Total \;\; Regret &= \frac{c}{2} + \frac{c}{3} + \frac{c}{4} + \frac{c}{5} ... + \frac{c}{T} \\ &= c(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + ...\frac{1}{T} + 1 - 1) \\ &= c(\ln T - 1) \end{aligned}$$

(c) **Upper Confidence Bound(UCB):** Although above solutions are pretty impressive and do a good deal of work solving the Multi-arm Bandits problem. However, a small issue remains in the exploration and exploitation trade off. The problem is simple. What should be the value of ϵ. If it is large, then the agent spends more time exploring the different arms. If in case it is small then exploration is less and hence converging to the correct mean is somewhat ambiguous. So, we always are in a fix as to choose the optimal ϵ. Here, is when we decide to go for upper confidence bound. Now, the general question is why are we choosing the upper bound. So the choice of arm is taken by a simple formula (as given below).

$$A^* = argmax_{A_i} \;\; Q_i + c.\sqrt{\frac{2.T_i}{n_i}}$$

Now according to this formula. As more and more arms are chosen, T_i increases. If at any instant 'i' arm 1 is left unexplored much, then the n_i value will be small

enough but T_i value will be of high numeric value. Ultimately the value of the bonus increases hence, increasing the value of the equation on the whole. Now using this metric we check for the arm it gives high value thus, resulting arm will be our next playing arm. Hence, we shall pull that arm for our next trial. Then we again calculate the updated empirical mean for that arm. Observe that the cumulative regret in the case of UCB algorithm is logarithmic. This is because the cumulative regret of UCB is of the order '$\ln t$'

(d) **Softmax:**In many cases we have observed that the true mean of pulling a certain arm may not necessarily sum upto 1. Supposingly the true mean of pulling arm 1 is given by μ_1^* and that of pulling arm 2 is given by μ_2^*. So both are quite close so, choosing any one is actually difficult. Thus, we used the softmax function to get a choice in these situations.

$$softmax(a_i) = \frac{\exp(a_i)}{\Sigma_{j=1}^{j=N} \exp a_j}$$

Now we use a temperature parameter to get a control over expectation and exploration. The Temp=0.05 for the below curve. Softmax is again a technique that uses that arm or plays that arm which has the highest following value.

$$softmax(\mu_i) = \frac{\exp(\frac{a_i}{\tau})}{\Sigma_{j=1}^{N} \exp(\frac{a_j}{\tau})}$$

where μ_i is the empirical mean of the arm 'i' at any instant 't'. τ is the temperature taken. Now Softmax also provides a logarithmic total regret value. Note when the temperature value is very high, the arms are equally likely(Exploration phase). When temperature is small, it takes greedy policy (Exploitation phase).

(e) **Thompson Sampling:**Thompson Sampling (Posterior Sampling or Probability Matching) is an algorithm for choosing the actions that address the exploration-exploitation dilemma in the multi-armed bandit problem. Actions are performed several times and are called exploration. It uses training information that evaluates the actions taken rather than instructs by giving correct actions. This is what creates the need for active exploration, for an explicit trial-and-error search for good behavior. Based on the results of those actions, rewards (1) or penalties (0) are given for that action to the machine. Further actions are performed in order to maximize the reward that may improve future performance. Suppose a robot has to pick several cans and put them in a container. Each time it puts the can to the container, it will memorize the steps followed and train itself to perform the task with better speed and precision (reward). If the Robot is not able to put the can in the container, it will not memorize that procedure (hence speed and performance will not improve) and will be considered as a penalty.

Thompson Sampling has the advantage of the tendency to decrease the search as we get more and more information, which mimics the desirable trade-off in the problem, where we want as much information as possible in fewer searches. Hence, this Algorithm has a tendency to be more "search-oriented" when we have fewer data and less "search-oriented" when we have a lot of data. So here we first take some value of $\alpha_{0,i} = 1 \;\; and \;\; \beta_{0,i} = 1$. Using these values of $\alpha_{t,i}$ and $\beta_{t,i}$, we sample from beta distribution as $\theta_i \;\; \forall \;\; i \in [1, K]$. Finally we select that arm which has highest sample

$$j^* = argmax_j \theta_j$$

So, we get the value of the suitable or optimal j. Now we sample our reward by using Bernoulli distribution or Gaussian distribution. Finally when we have got our reward, then verify if the reward is equal to 1 or not. If reward equals 1 then increase the value of $\alpha_{t,i}$ by 1. If the reward is 0 then increase the value of $\beta_{t,i}$ by 1, Again sample with modified α and β to get new samples.

Policy based Approaches:

(a) **Reinforce:** Reinforce is an algorithm that is focused on finding the optimal arm using the gradient ascent method. The algorithm progresses with talking an arbitrary policy. Here we have chosen each arm one after another generating some reward for the same. Now, based on the taken alpha and beta value, we change our beliefs in the following way. More formally, the algorithm maintains a set of preferences, $\pi_i(t)$, for each arm i. At each turn t = 1, 2, ..., the probability $p_i(t)$ is computed using a softmax function (as shown below). If j(t) which denotes that arm j is played at time interval 't'.

$$p_i(t) = \frac{\exp \pi_i(t)}{\Sigma_{j=1}^{k} \exp \pi_j(t)}$$

Now consider j(t) is the arm played at turn t and the reward received for this play is r(t), the preference $\pi_j(t)$ is changed as follows

$$\pi_{j(t)}(t+1) = \pi_{j(t)}(t) + \beta(r(t) - \hat{r}(t))$$

Also, at every turn, the rewards are updated using the below equation.

$$\hat{r}(t+1) = (1-\alpha)\hat{r}(t) + \alpha.r(t)$$

Here, α and β are learning rates between 0 and 1.

Total regret accumulated by various algorithms in tabular method

Table 1: 2 arm bandits with bernoulli reward distribution.

Algorithm	**Special condition**	**Regret value**
Epsilon Greedy	$\epsilon = 0.1$	290.698
Variable Epsilon-Greedy	c=1	134.116
Softmax	temperature = 0.05	101.076
Upper Confidence Bound	c=1	69.186
Thompson Sampling	None	13.405
Reinforce Algorithm(without baseline)	learning rate=0.1(lower values were too slow)	39.314
Reinforce Algorithm(with Baseline)	learning rate=0.1	55.209

The order of the plots below are first for 2 arms, followed by 5 arms and then 10 arms bandits. Each K arm test bed has three plots, one showing the cumulative regret, the second showing the optimal arm pulling percentage and regret per turn

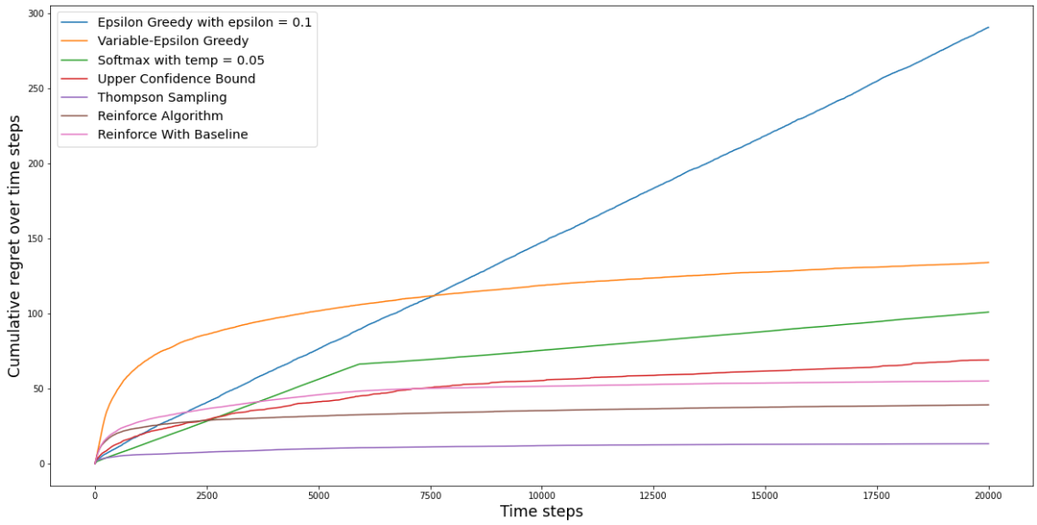

Figure 1: Cumulative regret of 2 arm bandit(Bernoulli reward)

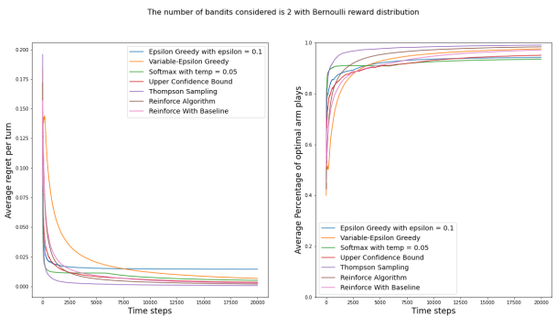

Figure 2: Optimal arm percentage and Regret per turn in 2 arm bandit(Bernoulli reward)

Table 2: 5 arm bandits with bernoulli reward distribution.

Algorithm	**Special condition**	**Regret value**
Epsilon Greedy	$\epsilon = 0.1$	654.094
Variable Epsilon-Greedy	c=1	641.759
Softmax	temperature = 0.05	2165.560
Upper Confidence Bound	c=1	216.680
Thompson Sampling	None	38.826
Reinforce Algorithm(without baseline)	learning rate=0.1(lower values were too slow)	109.249
Reinforce Algorithm(with Baseline)	learning rate=0.1	124.669

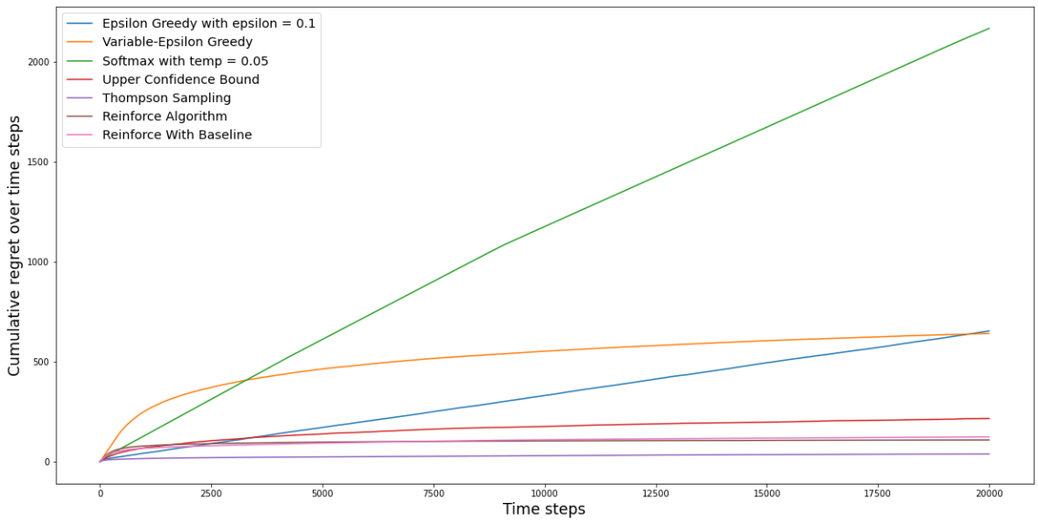

Figure 3: Cumulative regret of 5 arm bandit(Bernoulli reward)

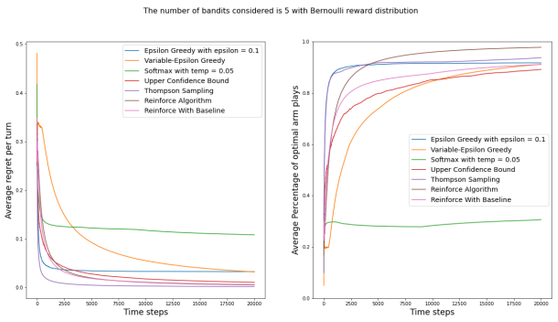

Figure 4: Optimal arm percentage and Regret per turn in 5 arm bandit(Bernoulli reward)

Table 3: 10 arm bandits with bernoulli reward distribution.

Algorithm	**Special condition**	**Regret value**
Epsilon Greedy	$\epsilon = 0.1$	888.366
Variable Epsilon-Greedy	c=1	1383.007
Softmax	temperature = 0.05	2184.919
Upper Confidence Bound	c=1	443.435
Thompson Sampling	None	43.148
Reinforce Algorithm(without baseline)	learning rate=0.1(lower values were too slow)	298.605
Reinforce Algorithm(with Baseline)	learning rate=0.1	274.121

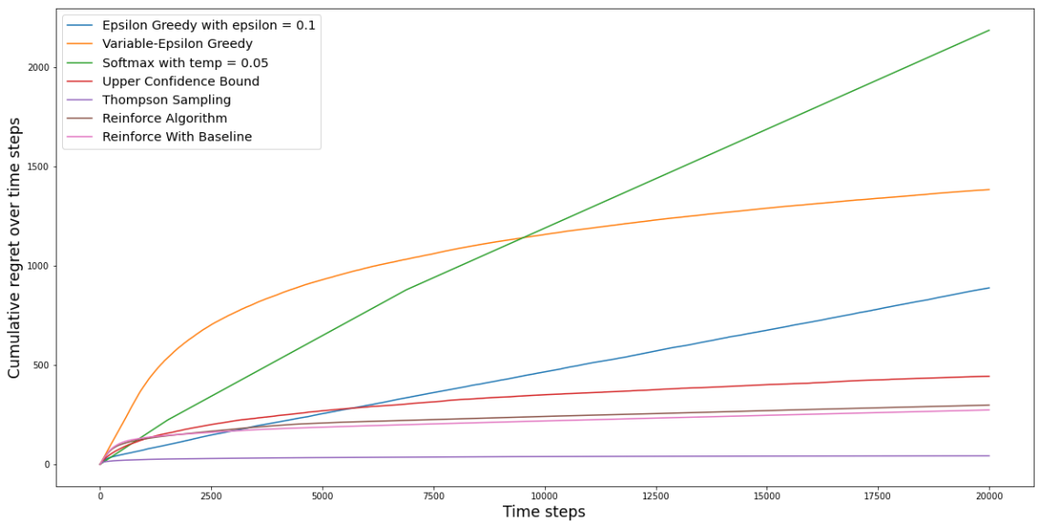

Figure 5: Cumulative regret of 10 arm bandit(Bernoulli reward)

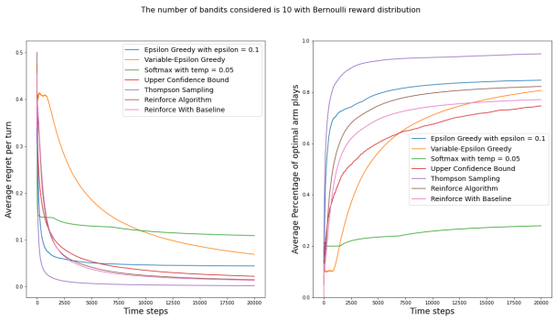

Figure 6: Optimal arm percentage and Regret per turn in 10 arm bandit(Bernoulli reward)

Table 4: 2 arm bandits with Gaussian reward distribution with mean=random(0,1) and standard deviation = 0.5.

Algorithm	**Special condition**	**Regret value**
Epsilon Greedy	$\epsilon = 0.1$	527.509
Variable Epsilon-Greedy	c=1	180.278
Softmax	temperature = 0.05	714.153
Upper Confidence Bound	c=1	75.003
Thompson Sampling	None	177.763
Reinforce Algorithm(without baseline)	learning rate=0.1(lower values were too slow)	53.258
Reinforce Algorithm(with Baseline)	learning rate=0.1	47.934

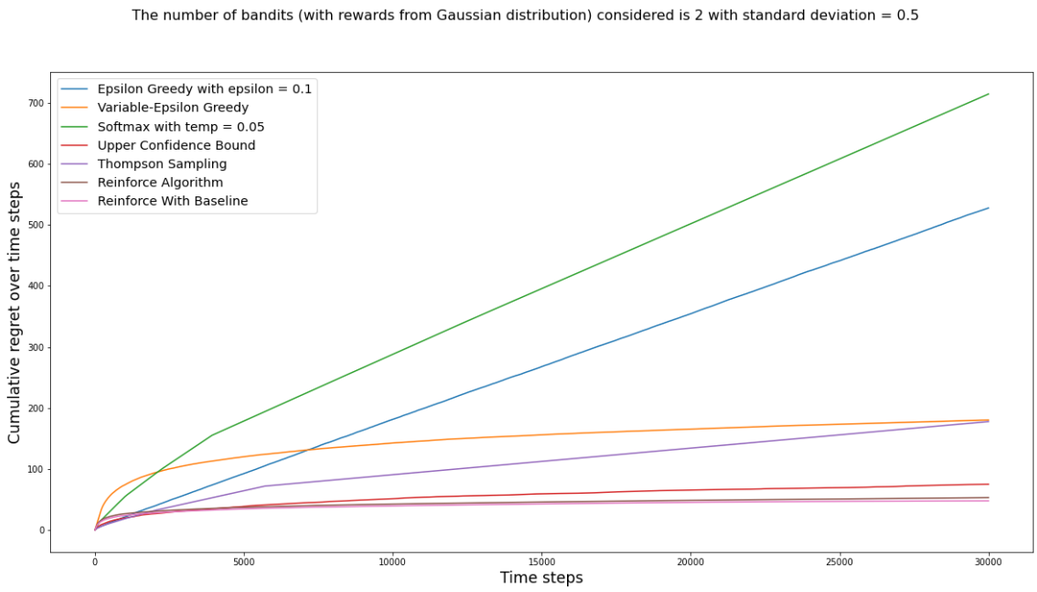

Figure 7: Cumulative regret of 2 arm bandit(Gaussian reward std = 0.5)

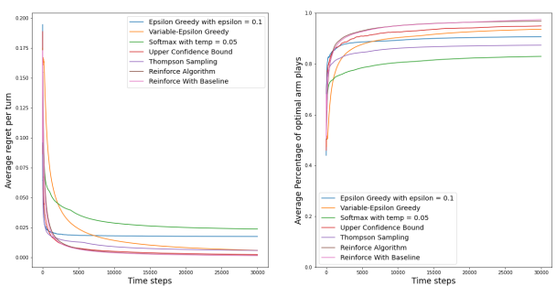

Figure 8: Optimal arm percentage and Regret per turn in 2 arm bandit(Gaussian reward std = 0.5)

Table 5: 5 arm bandits with Gaussian reward distribution with mean=random(0,1) and standard deviation = 0.5.

Algorithm	**Special condition**	**Regret value**
Epsilon Greedy	$\epsilon = 0.1$	984.749
Variable Epsilon-Greedy	c=1	665.664
Softmax	temperature = 0.05	2631.349
Upper Confidence Bound	c=1	259.146
Thompson Sampling	None	516.599
Reinforce Algorithm(without baseline)	learning rate=0.1(lower values were too slow)	219.685
Reinforce Algorithm(with Baseline)	learning rate=0.1	119.601

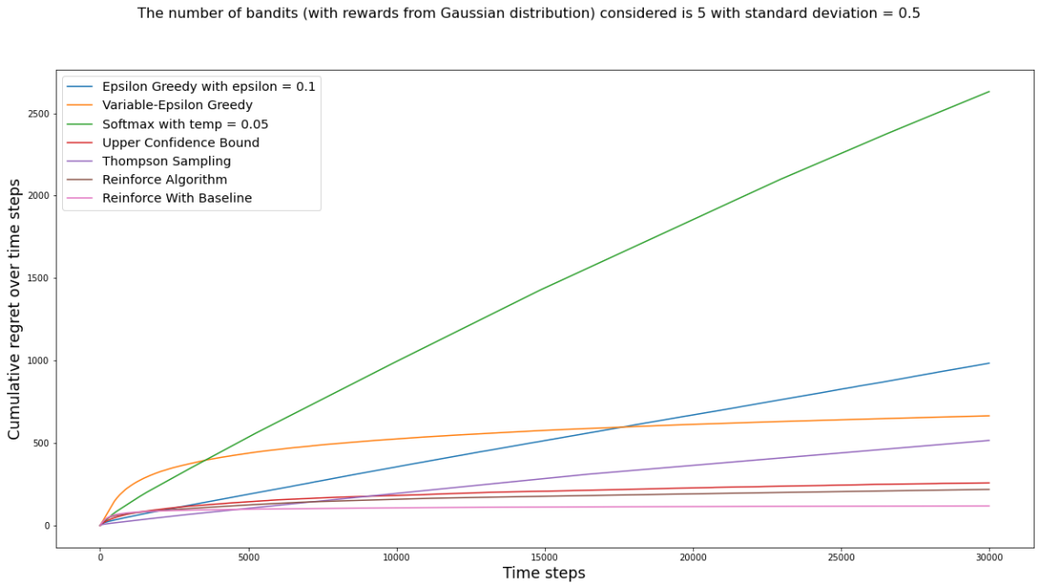

Figure 9: Cumulative regret of 5 arm bandit(Gaussian reward std = 0.5)

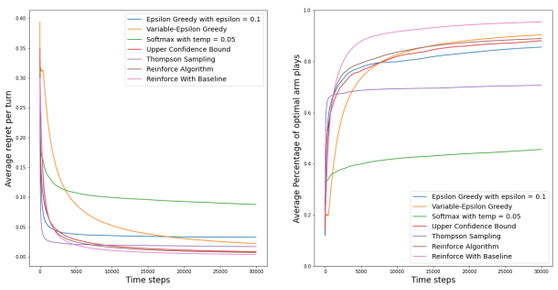

Figure 10: Optimal arm percentage and Regret per turn in 5 arm bandit(Gaussian reward std = 0.5)

Table 6: 10 arm bandits with Gaussian reward distribution with mean=random(0,1) and standard deviation = 0.5.

Algorithm	**Special condition**	**Regret value**
Epsilon Greedy	$\epsilon = 0.1$	1343.664
Variable Epsilon-Greedy	c=1	1563.707
Softmax	temperature = 0.05	3349.213
Upper Confidence Bound	c=1	481.97
Thompson Sampling	None	501.555
Reinforce Algorithm(without baseline)	learning rate=0.1(lower values were too slow)	669.371
Reinforce Algorithm(with Baseline)	learning rate=0.1	324.465

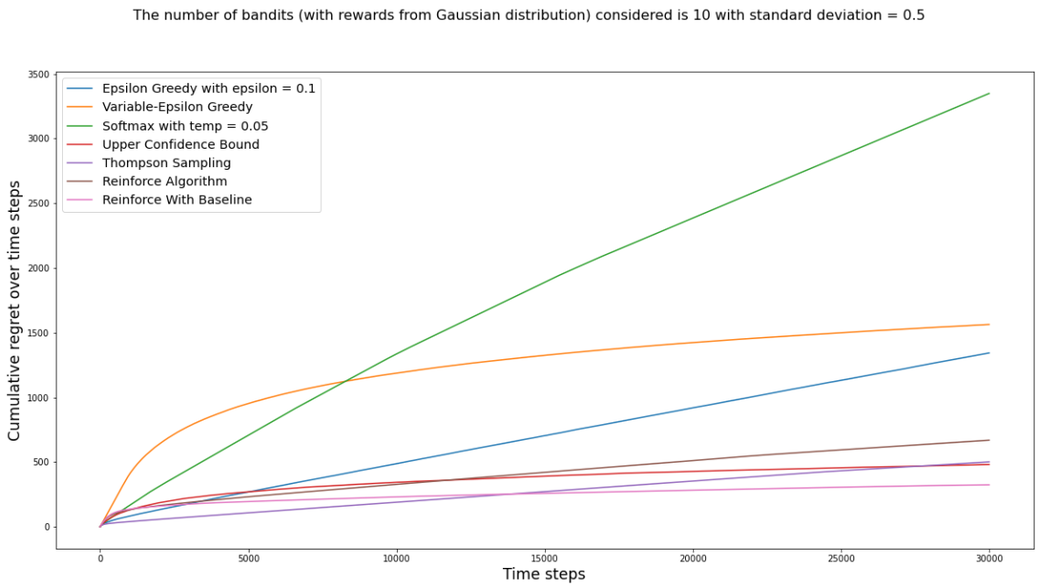

Figure 11: Cumulative regret of 10 arm bandit(Gaussian reward std = 0.5)

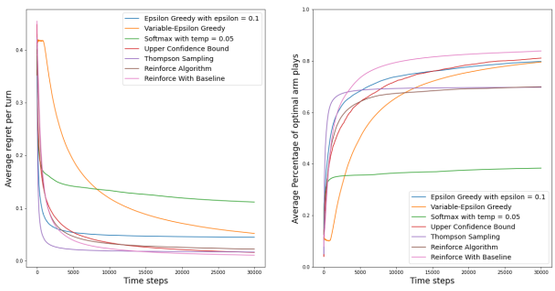

Figure 12: Optimal arm percentage and Regret per turn in 10 arm bandit(Gaussian reward std = 0.5)

Table 7: 2 arm bandits with Gaussian reward distribution with mean=random(0,1) and standard deviation = 1.

Algorithm	**Special condition**	**Regret value**
Epsilon Greedy	$\epsilon = 0.1$	489.025
Variable Epsilon-Greedy	c=1	172.066
Softmax	temperature = 0.05	2104.024
Upper Confidence Bound	c=1	72.316
Thompson Sampling	None	568.697
Reinforce Algorithm(without baseline)	learning rate=0.1(lower values were too slow)	42.779
Reinforce Algorithm(with Baseline)	learning rate=0.1	63.698

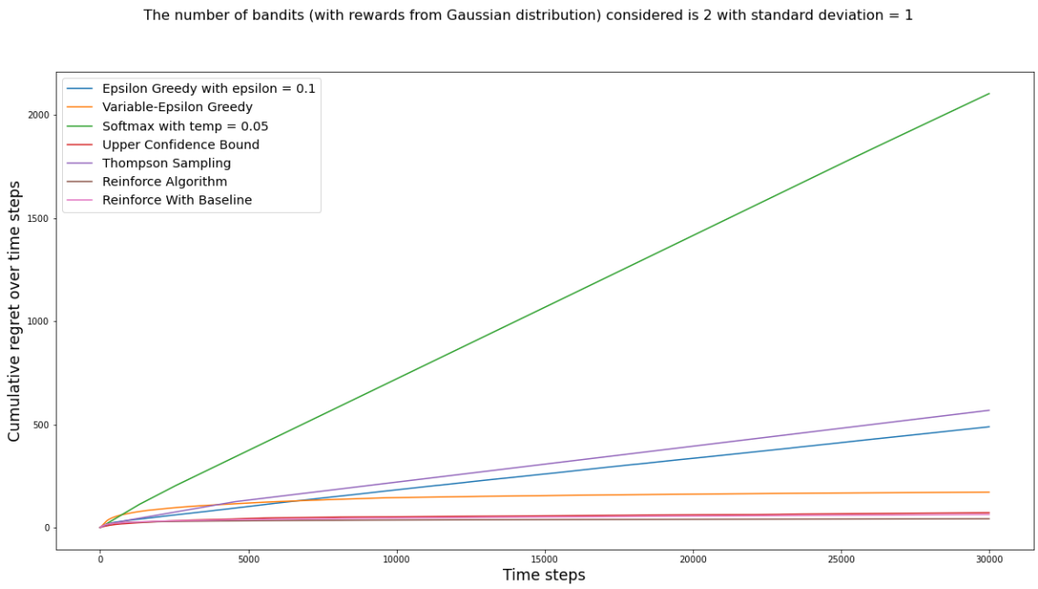

Figure 13: Cumulative regret of 2 arm bandit(Gaussian reward std = 1)

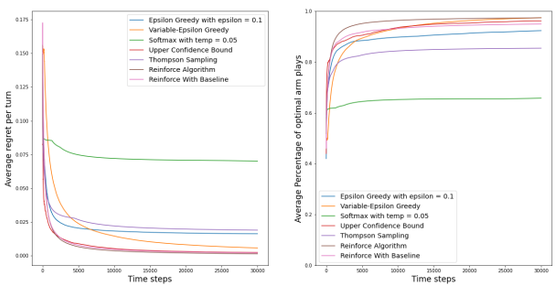

Figure 14: Optimal arm percentage and Regret per turn in 2 arm bandit(Gaussian reward std = 1)

Table 8: 5 arm bandits with Gaussian reward distribution with mean=random(0,1) and standard deviation = 1.

Algorithm	**Special condition**	**Regret value**
Epsilon Greedy	$\epsilon = 0.1$	1204.590
Variable Epsilon-Greedy	c=1	908.447
Softmax	temperature = 0.05	2345.889
Upper Confidence Bound	c=1	232.862
Thompson Sampling	None	476.513
Reinforce Algorithm(without baseline)	learning rate=0.1(lower values were too slow)	388.697
Reinforce Algorithm(with Baseline)	learning rate=0.1	169.249

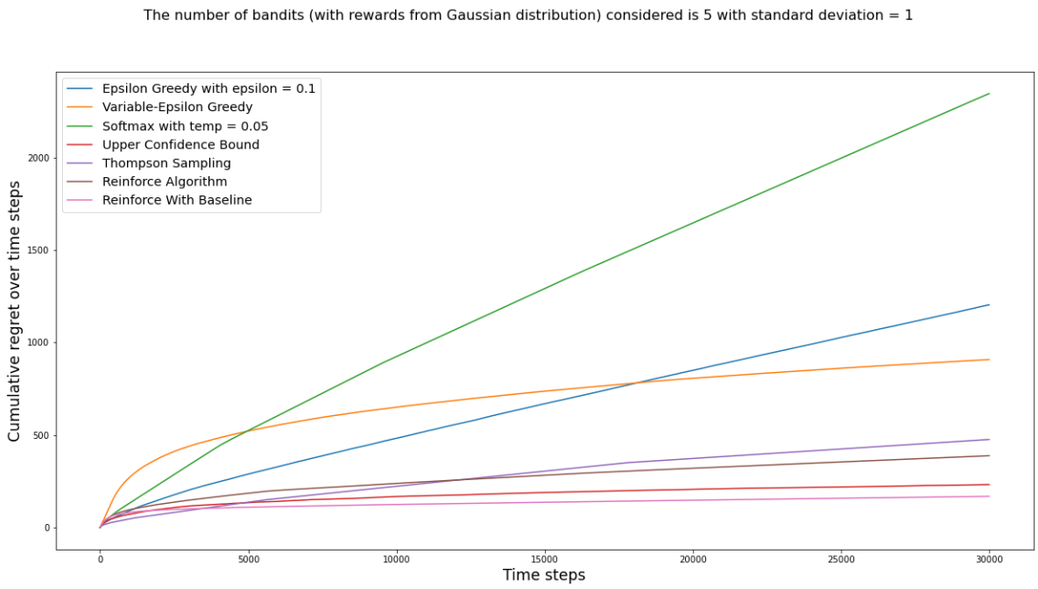

Figure 15: Cumulative regret of 5 arm bandit(Gaussian reward std = 1)

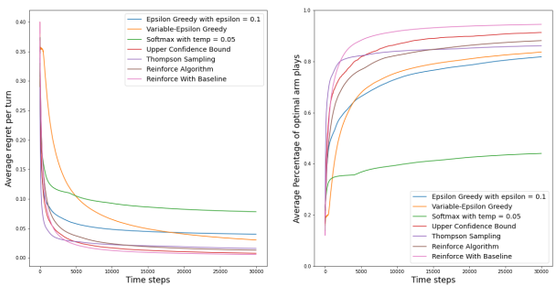

Figure 16: Optimal arm percentage and Regret per turn in 5 arm bandit(Gaussian reward std = 1)

Table 9: 10 arm bandits with Gaussian reward distribution with mean=random(0,1) and standard deviation = 1.

Algorithm	**Special condition**	**Regret value**
Epsilon Greedy	$\epsilon = 0.1$	1499.945
Variable Epsilon-Greedy	c=1	1547.792
Softmax	temperature = 0.05	3471.440
Upper Confidence Bound	c=1	535.339
Thompson Sampling	None	1246.568
Reinforce Algorithm(without baseline)	learning rate=0.1(lower values were too slow)	1064.088
Reinforce Algorithm(with Baseline)	learning rate=0.1	478.231

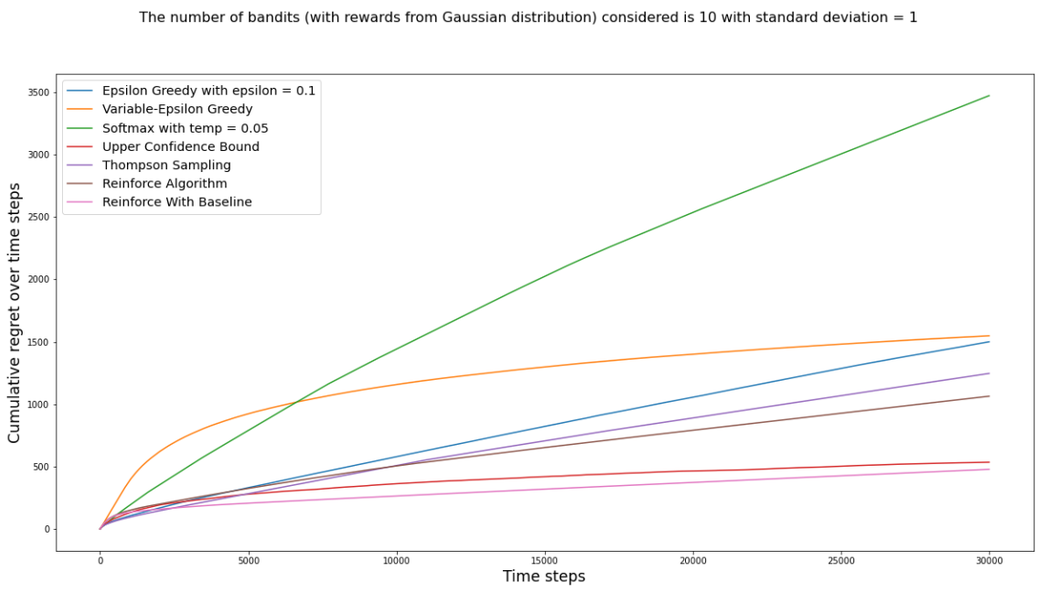

Figure 17: Cumulative regret of 10 arm bandit(Gaussian reward std = 1)

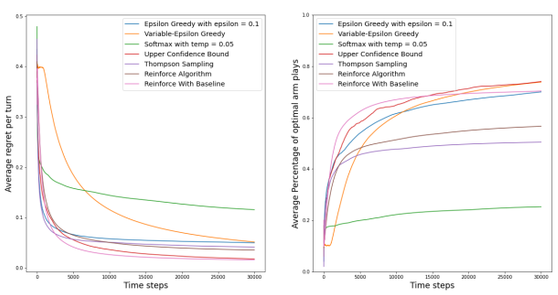

Figure 18: Optimal arm percentage and Regret per turn in 5 arm bandit(Gaussian reward std = 0.5)

OBSERVATIONS:

As per the results obtained, we can clearly see Epsilon Greedy with fixed epsilon has a linear cumulative regret whereas the others have logarithmic cumulative regret.. Some things to note are as follows

1. Thompson Sampling seems to have the least regret in most of the cases.
2. With increase in the number of arms, the regret for the algorithms seem to increase. This is where it shows the large number of choices require large exploration time to converge to optimal arm.
3. Softmax algorithm seems to give the highest regret in all cases.
4. Reinforce with and without baseline have close regret.
5. 0.1 learning rate takes the learning of the gradient descent algorithm very aggressively, hence, in all the graphs fast convergence to optimal can be seen. Other learning rates seem to be slow.
6. For Gaussian distribution of rewards, increase of variance or standard deviation introduces increases regret metric.
7. None of the algorithms cross each other in terms of complete convergence to optimal arm percentage.

Conclusions:

Various Bandit algorithms i.e. Value based approaches and Policy Based Approaches were tested across different test beds. The test beds are 2-arm test bed, 5-arm test bed, 10-arm test bed with Bernoulli reward and Gaussian Reward. The order of performance can be shown as below: Thompson Sampling $\geq$ Reinforce(with baseline) $\geq$ UCB $>$ Reinforce(without baseline) $>$ Variable-Epsilon Greedy $\geq$ Softmax $>$ Fixed Epsilon Greedy