Reinforcement Learning Assignment-2
Multi-Armed Bandit Problem

Utkarsh Prakash
180030042

February 1, 2022

1 Problem Statement

A multi-armed Bandit is a set of distributions $\{\mathcal{R}_a | a \in \mathcal{A}\}$ where $\mathcal{A}$ is a known set of arms and $\mathcal{R}_a$ is the distribution of the rewards given the arm a. K represents the number of arms being considered in the problem. At each time step, we select an arm $A_t \in \mathcal{A}$ and get a reward $R_t \sim \mathcal{R}_{A_t}$. Let $q(a)$ denote the expected reward that can be obtained if the arm a is pulled, i.e. $q(a) = \mathbb{E}[R_t | A_t = a]$. Using, $q(a)$, we can define an optimal arm as the one having the highest value of $q(a)$. Let's denote the value $q(a)$ of this optimal arm as v_*. We can now define regret for an action a as $v_* - q(a)$.

The goal of the problem is to minimize the cummulative reward over fixed time (T), i.e.

$$\min \sum_{n=1}^{T} (v_* - q(A_n))$$

2 Experiment Setting

We consider Bernoulli and Normal Reward Distributions for our experiments. We sample the expected reward from a uniform distribution between [0, 1] for both the reward distributions for each of the arm. We assume that the variance (σ^2) of the Normal reward distribution is known and we experiment with $\sigma^2 = 0.1^2$ and $\sigma^2 = 1^2$.

For each experiment we run our algorithm for 1000 time steps. This is considered to be a run for an algorithm with a given Bandit problem. In order to compare the performance of the algorithms fairly, in each run we evaluate the performance of the algorithm on the same Bandit problem. We run our algorithms for 1000 runs with each run having a different Bandit problem. We use the following metrics for evaluating the performance of the algorithm:

- The total regret accumulated over time.
- The regret as a function of time.
- The percentage of plays in which the optimal arm is pulled.
- Average reward as a function of time.

Note that we average these metrics over 1000 runs of the algorithm.

3 Notations

$$x = \frac{\zeta}{t} \tag{1}$$

4 Bernoulli Reward Distribution

4.1 K=2 arm Problem

4.1.1 Greedy Algorithm

In this section, we compare the pure Greedy algorithm, ϵ-greedy with fixed $\epsilon = 0.1$ and $\epsilon = 0.01$ and variable ϵ with the following schedule:

$$\epsilon_t = min\{1, \frac{C}{t}\} \tag{2}$$

where $C = 10$ and t is the total number of plays. We observe the following graphs:

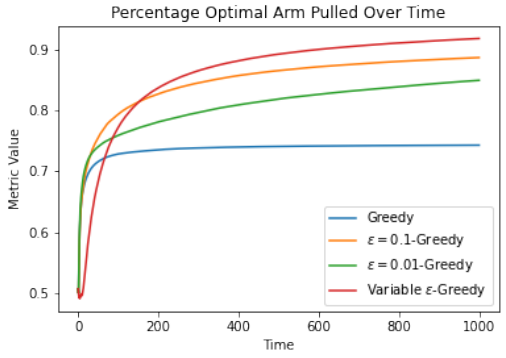

Figure 1: Percentage Optimal Arm Pulled Over Time

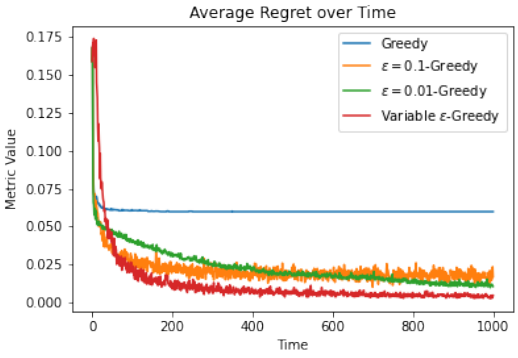

Figure 2: Average Regret over Time

Observations:

- The variable ϵ-Greedy tends to perform better than all of its counter-parts.
- In general, ϵ-Greedy performs better than pure Greedy because of its tendency to explore the arms along with exploting the current known knowledge.

4.1.2 Softmax Policy

In this section, we compare the Softmax Policy which is defined as follows:

$$\pi_t(a) = \frac{e^{q_t(a)/\tau}}{\sum_{i=1}^{K} e^{q_t(a_i)/\tau}}$$

where τ is the temperature hyperparameter. We compare the performance of Softmax Policy for $\tau = 0.01$, $\tau = 10000$ and variable τ with schedule as defined in (1) where $\zeta = 10$. We reduce the τ

overtime to control exploration-exploitation tradeoff. The graph for different metrics are obtained as follows:

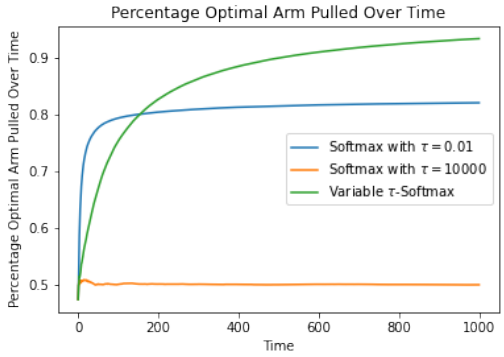

Figure 3: Percentage Optimal Arm Pulled Over Time

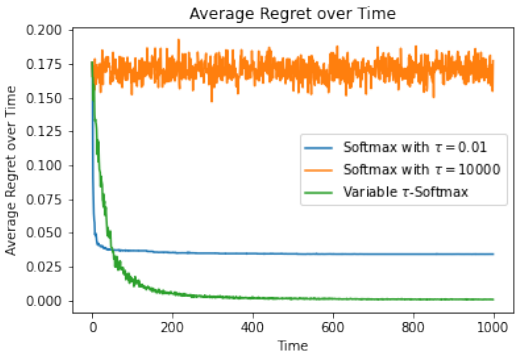

Figure 4: Average Regret over Time

Observations:

- For small values of $\tau = 0.01$, the algorithm performs poorly because in the limit when $\tau \to 0$, then the algorithm performs greedily.
- For large values of $\tau = 10,000$, the algorithm performs poorly because in the limit when $\tau \to \infty$, then the algorithm picks an arm uniformly at random. Therefore, for such large values of τ the algorithm explores at lot.
- The variable τ-Softmax tends to perform better than all of its counter-parts. This is because earlier when the value of τ is high, we tend to explore more, whereas as time progresses and we gather knowledge of different arms, we reduce the value of τ so as to exploit the knowledge that we have gathered (i.e., pull the arm which has highest estimated average reward with high probability).

4.1.3 UCB Algorithm

In this section we compare the UCB algorithm where we pick the arm which has the highest value of

$$\arg max_{a \in \mathcal{A}} \left(q_t(a) + C\sqrt{\frac{2 \ln t}{n_t(a)}} \right)$$

where C is a hyperparameter which controls exploration-exploitation tradeoff. We used three different values of C (1, 100 and variable). The graphs obtained are as follows:

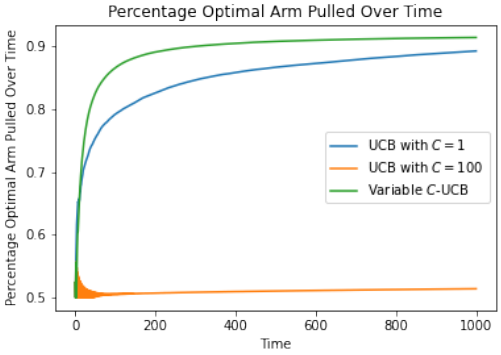

Figure 5: Percentage Optimal Arm Pulled Over Time

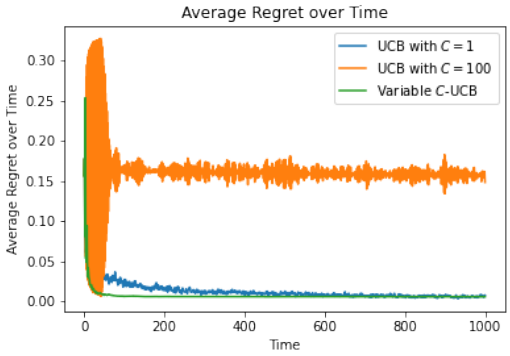

Figure 6: Average Regret over Time

Observations:

- We can see the variable-C-UCB algorithm outperforms its counter-parts.
- For large values of $C = 100$, the algorithm performs poorly because it explores too much.

4.1.4 Thompson Sampling

In Thompson Sampling, we maintain a Beta distribution prior over $q_t(a)$. We sample a value $Q_t(a)$ from this Beta distribution for each arm, i.e. $q_t(a) \sim Q_t(a)$. Now, we select the arm which has the highest value of $Q_t(a)$. The results obtained for this algorithm is as follows:

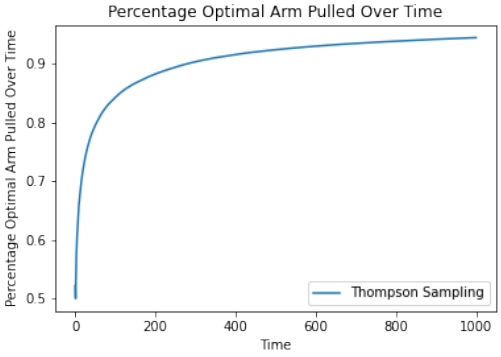

Figure 7: Percentage Optimal Arm Pulled Over Time

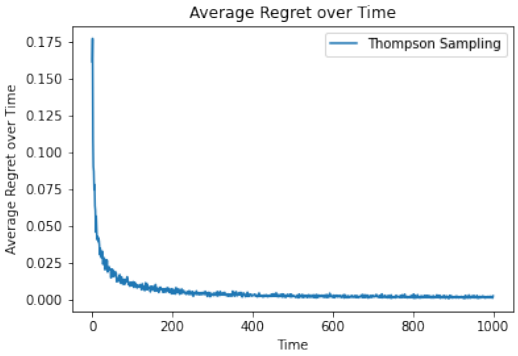

Figure 8: Average Regret over Time

4.1.5 Reinforce Algorithm

The reinforce algorithm follows the following policy:

$$\pi_t(a) = \frac{e^{\theta_t(a)}}{\sum_{i=1}^{K} e^{\theta_t(a_i)}}$$

where $\theta_t(a)$ are the learnable parameters of the algorithm. These parameters can be learned using Stochastic Gradient Ascent with the following update rule:

$$\theta_t(a) = \begin{cases} \theta_t(A_t) + \alpha(R_t - b)(1 - \pi_t(A_t)) & if a = A_t \\ \theta_t(a) + \alpha(R_t - b)\pi_t(a) & if a \neq A_t \end{cases}$$

where b is a baseline and is generally chosen to be the average of all the rewards up through and including time t. The following graphs were obtained with and without baselines:

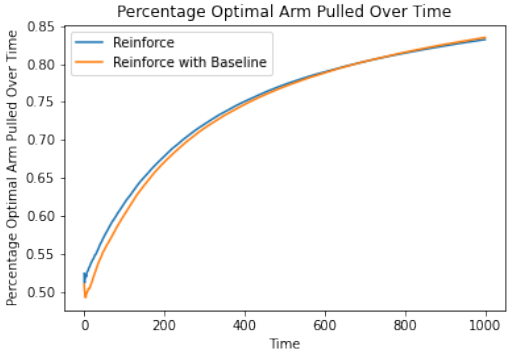

Figure 9: Percentage Optimal Arm Pulled Over Time

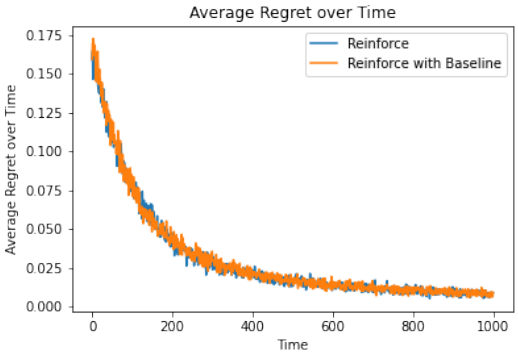

Figure 10: Average Regret over Time

Observations:

- We really don't see any difference in performance between the algorithm with and without baselines.

4.1.6 Comparison of all Algorithms

In this section we compare the performance of all the algorithms for the 2 arm problem. We pick the best performing variation of the algorithm as found in the above experiments for comparison i.e. the following algorithms were used comparison:

- Variable ϵ-Greedy where ϵ decreases according to the schedule defined in (2) with $C = 10$.
- Variable τ-Softmax where τ decreases according to the schedule defined in (1) with $\zeta = 10$.
- Variable C-UCB where C decreases according to the schedule defined in (1) with $\zeta = 10$.
- Thompson Sampling Algorithm
- Reinforce Algorithm without baseline. This was chosen since we didn't find any considerable improvement in the performance by using baseline.

The results obtained are as follows:

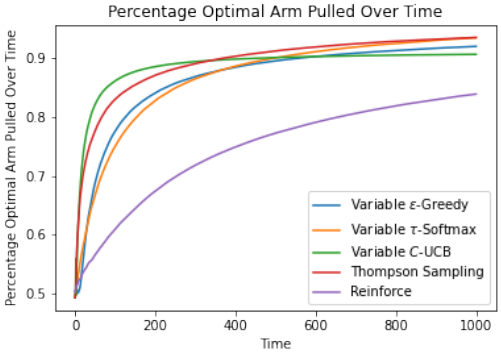

Figure 11: Percentage Optimal Arm Pulled Over Time

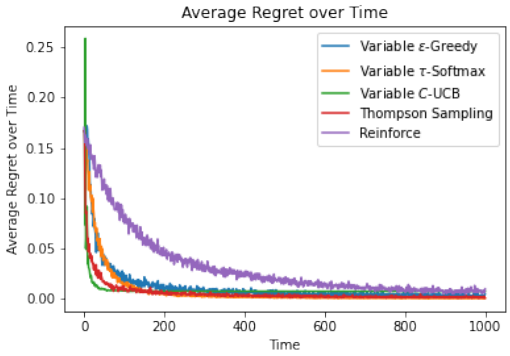

Figure 12: Average Regret over Time

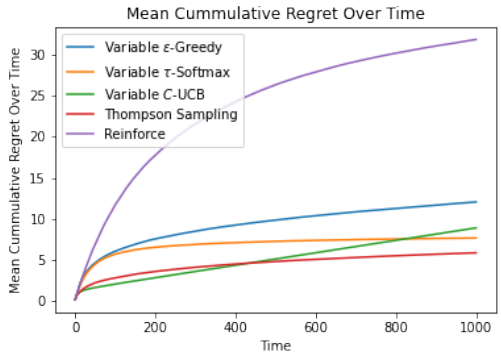

Figure 13: Percentage Optimal Arm Pulled Over Time

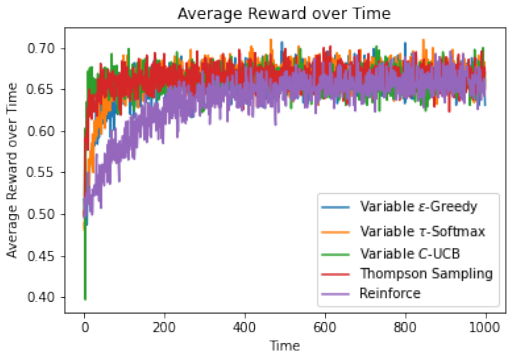

Figure 14: Average Regret over Time

Observations:

- Variable ϵ-Greedy, Variable τ-Softmax, Variable C-UCB and Thompson Sampling algorithm seems to perform equally well. This means that any algorithm which optimally trades-off between exploration and exploitation, should perform well i.e. have logarithmic cummalative regret over time. This is an interesting observation as simple algorithms with simple heuristics tend to perform at par with other algorithms with more sophisticated heuristics.
- Reinforce algorithm doesn't tend to perform that well when compared to its counter-parts. If we look at the plot for the average regret over time for Reinforce algorithm, we find that it intially decreases very slowly. However, later it reaches the same level as that of other algorithms. This nature can be attributed to the fact that the parameters of the algorithm are learned slowly. Thereby, by increasing the learning rate for the stochastic gradient ascent may fix the problem and the performance can become at par with other algorithms.

4.2 K=5 arm Problem

We repeated similar experiments for each of the algorithm as we did for $K = 2$ arm case to find the best performing variant of an algorithm. We found the graphs to be following a similar trend as for $K = 2$ case, therefore, we do not include them in our report. We then compared each of these best performing algorithms i.e. we compared the performance of the following algorithms for $K = 5$ arms:

- Variable ϵ-Greedy where ϵ decreases according to the schedule defined in (2) with $C = 10$.
- Variable τ-Softmax where τ decreases according to the schedule defined in (1) with $\zeta = 10$.
- Variable C-UCB where C decreases according to the schedule defined in (1) with $\zeta = 10$.
- Thompson Sampling Algorithm
- Reinforce Algorithm without baseline. This was chosen since we didn't find any considerable improvement in the performance by using baseline.

The results obtained are as follows:

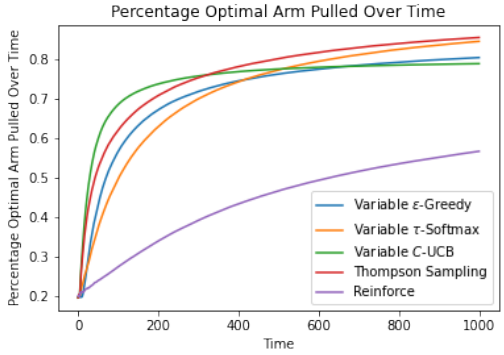

Figure 15: Percentage Optimal Arm Pulled Over Time

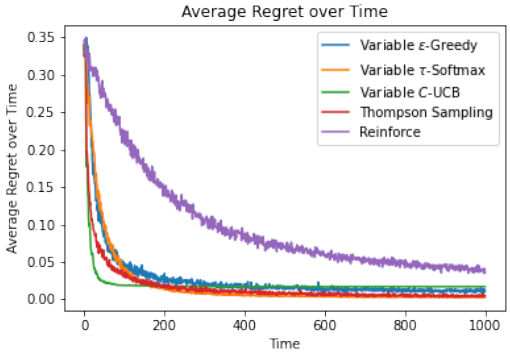

Figure 16: Average Regret over Time

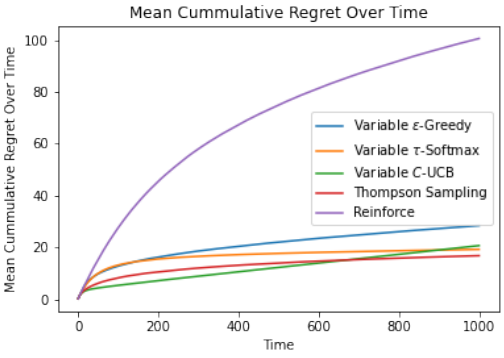

Figure 17: Percentage Optimal Arm Pulled Over Time

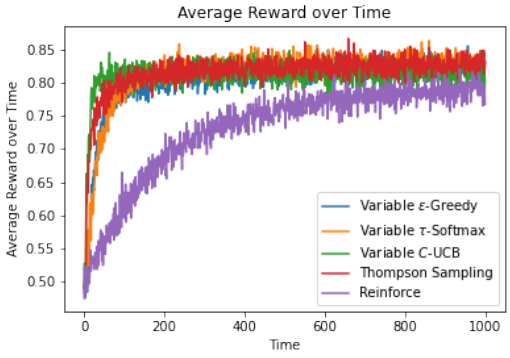

Figure 18: Average Regret over Time

Since, the trends in the graphs are similar to $K = 2$ case, we defer the further discussion on the results to the section 4.4 where we compare the relative performance of the algorithm for different number of arms.

4.3 K=10 arm Problem

We repeated similar experiments for each of the algorithm as we did for $K = 2$ arm case to find the best performing variant of an algorithm. We found the graphs to be following a similar trend as for $K = 2$ case, therefore, we do not include them in our report. We then compared each of

these best performing algorithms i.e. we compared the performance of the following algorithms for $K = 10$ arms:

- Variable ϵ-Greedy where ϵ decreases according to the schedule defined in (2) with $C = 10$.
- Variable τ-Softmax where τ decreases according to the schedule defined in (1) with $\zeta = 10$.
- Variable C-UCB where C decreases according to the schedule defined in (1) with $\zeta = 10$.
- Thompson Sampling Algorithm
- Reinforce Algorithm without baseline. This was chosen since we didn't find any considerable improvement in the performance by using baseline.

The results obtained are as follows:

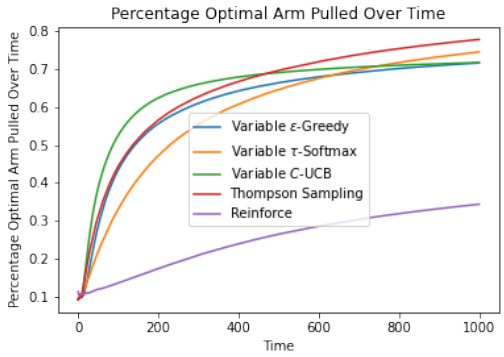

Figure 19: Percentage Optimal Arm Pulled Over Time

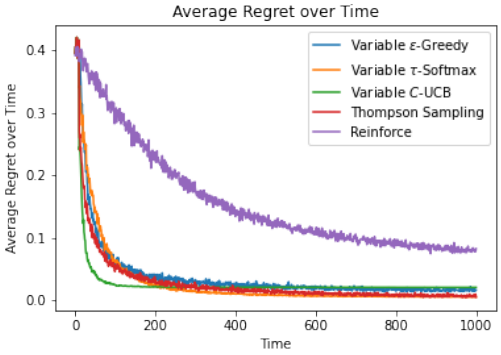

Figure 20: Average Regret over Time

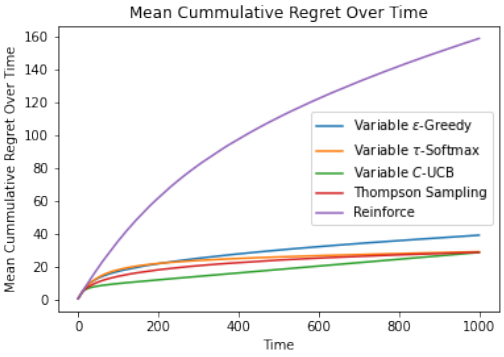

Figure 21: Percentage Optimal Arm Pulled Over Time

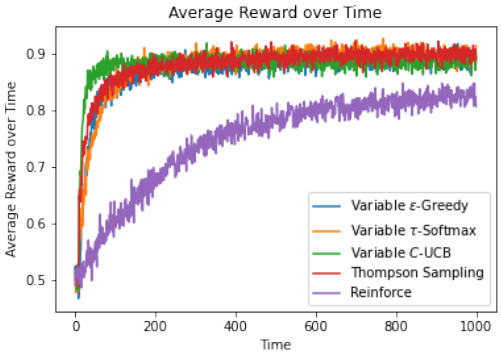

Figure 22: Average Regret over Time

Since, the trends in the graphs are similar to $K = 2$ case, we defer the further discussion on the results to the section 4.4 where we compare the relative performance of the algorithm for different number of arms.

4.4 Comparison of algorithms for different number of arms

The following table lists the cummalative regret at the end of 1000 plays averaged over 1000 runs:

Algorithms	**K=2 Arms**	**K=5 Arms**	**K=10 Arms**
Variable ϵ-Greedy	12.07	28.48	39.04
Variable τ-Softmax	7.68	19.32	28.97
Variable C-UCB	8.91	20.77	28.52
Thompson Sampling	5.87	16.89	28.58
Reinforce Algorithm (without baseline)	31.88	100.61	158.87

Table 1: Comparison of cummalative regret of different algorithms for different number of arms for 1000 plays averaged over 1000 runs.

One thing to note is that we used the same hyperparameters for different number of arms in order to facilitate fair comparison between the performance of the algorithms. The above table and the plots of section 4.1.6, 4.2 and 4.3 highlights that the performance of all the algorithms decreases considerably as the number of arms are increased. However, the relative order of performance between algorithms remain as it is. We observe similar trends for other metrics as well.

•