Multi Armed Bandit Problem with different datasets

Report submitted in fulfillment of the requirements for the Exploratory Project of

Second Year B.Tech.

by

Satya Prakash and Nikita

Under the guidance of

Dr. Lakshmanan Kailasam

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY (BHU), Varanasi
Varanasi 221005, India

May 2021

Dedicated to

Our parents , professors, supervisor and mentor, family, friends and all those who made this project possible

DECLARATION

We certify that

1. The work contained in this report is original and have been done by ourselves and the general supervision of my supervisor.

2. The work has not been submitted for any project.

3. Whenever we have used materials (data, theoretical analysis, results) from other sources, we have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever we have quoted written materials from other sources, we have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Satya Prakash and Nikita

B.Tech Students

Department of Computer Science and Engineering

Indian Institute of Technology (BHU) Varanasi

Varanasi 221005, India

CERTIFICATE

*This is to certify that the work contained in this report entitled "**Multi Armed Bandit Problem with different datasets**" being submitted by **Satya Prakash and Nikita (Roll No. 19075066 and 19075048)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Dr Lakshmanan Kailasam
Department of Computer Science and Engineering
Indian Institute of Technology (BHU) Varanasi
Varanasi 221005, India

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who helped us throughout the project and made this possible. We are grateful to our project guide Dr. Lakshmanan Kailasam for providing us an opportunity to implement our project entitled "Multi Armed Bandit Problem with different datasets" and also for his constant encouragement and support throughout the course of this project. We would also like to thank our parents and friends for their support. They encouraged us to finalize our project and were there with us whenever we needed them.

Satya Prakash and Nikita

ABSTRACT

In this report, we have discussed and analysed performance of various algorithms for multi-armed bandit problem and studied their behaviour on normal and heavy-tailed distribution.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Multi armed bandit problem is a classical reinforcement learning example. To understand the k-armed bandit problem, let us consider a situation where we have to select an action out of the k possibilities repeatedly, which have random reward value associated to the actions which depends on a static probability distribution which is unknown. Our goal is to maximize the expected total reward obtained over a time period.

Nowadays, the digital platforms want to offer personalized product recommendation to their users. It is similar to the multi armed bandit problem like when a new user signs in, what recommendations are to be shown to a user out of the available options, how the user responds i.e. whether the user decides to watch/ buy/ play the recommended option or not, and then using the users response to improve the recommendations.

There are many other practical applications of Multi armed bandit problem like A/B testing of online advertisements, sequential portfolio selection and clinical trials to test the effect of various medical treatments while reducing patient's damage to body/death.

1.2 Dilemma of Exploitation vs Exploration

At each time step, we have an option to "exploit" or to "explore".

1. **Exploitation**: Select a greedy arm. It allows us to exploit our present knowledge of the action's values.

2. **Exploration**: Select a non-greedy arm. It allows us to better our estimate of the non-greedy action's values.

Exploitation might be the right thing to do, when we want to maximise the estimated reward on one step, but exploration produces better results in the long run, as it allows us to explore the non-greedy arms, find a better arm and to exploit it.

Chapter 2

Notations and Terminology

2.1 Terminology

1. Suppose we are given k arms with Probability distribution P_k, k = 1, 2..., K. Consider the bounded random variable $X_{j,t} \in [0, 1]$ associated with arm j for $1 \leq\ j \leq$ k and time index $t \geq$1.

2. **Action** : In RL, selecting an option out of the available ones, is termed as taking an action

   A : set of all the possible actions

   A_t : action taken at time step 't'

   .

3. **Reward** : It is a random value returned when an action is performed. The reward obtained for the j^{th} arm is drawn from a corresponding probability distribution whose expected value μ_j for $1 \leq j \leq$K is unknown .

   X_t : reward obtained at time 't' after taking action A_t.

   $X_{j,t}$: reward obtained when j^{th} arm is pulled the t^{th} time.

4. **Value of an Action** : Expected reward of an action. Basically, it is the mean reward value obtained when j_{th} arm is selected many times. We assume that

we do not know the action values with conviction, although we may know the estimate value. If an action 'a$_j$' is performed infinitely large number of times, and average of reward values is calculated , we will eventually obtain the true value of $\overline{X}_j$, mean reward for 'a$_j$'.

Value of action a : $q_*(a) = \mathbf{E}[X_t \mid A_t = a]$

Mean reward obtained when jth arm has been pulled n times : $\overline{X}_{j,n} = \frac{\sum_{j=1}^{K} X_{j,t}}{n}$

$Q_t(a)$: estimated value of action a at time step t.

We would like $Q_t(a)$ to be close to $q_*(a)$

5. **New estimate in terms of the last estimate and the new reward** : Incremental Update method

   $\overline{X}_{n+1} = (1 - \frac{1}{n})\overline{X}_n + \frac{1}{n} r_n$

6. **Arm with the largest mean reward** : It is represented by j*. It is the most optimal arm and theoretically represents the case when the best arm is selected every time. It has zero regret value.

   $\mu_{j^*} = \max_{1 \le i \le K} \mu_i$

7. **Regret** : It is the difference between the reward sum for the optimal strategy (theoretically - selecting the best arm every time) and the reward sum for the algorithm that we are applying. Basically, it is the loss that we suffer for the time spent while learning. Our goal while trying out different algorithms is to minimize the expected cumulative regret.

   $\mathbf{E}[R_n] = n\mu_{j^*}$ - $\mu_j \sum_{j=1}^{K} \mathbf{E}[T_j(n)]$

   Another useful formula to represent regret : $R_n = \sum_{k=1}^{K} \mathbf{E}[T_k(n)]\Delta_k$

   where $T_k(n)$ is the number of times the kth arm is selected until 't' time steps and $\Delta_k = \mu_* - \mu_k$ which denotes the difference between the expected reward of the optimal arm and the suboptiomal arm k

Chapter 3

Different Algorithms

3.1 Overview

We will now discuss some algorithms to solve the multi armed bandit problem. These algorithms are based on how we do exploration.

1. **No exploration**: It is the most naive algorithm based on greedy approach. Select the arm that has the highest mean reward in every iteration without any further exploration. If it selects a sub-optimal arm, then it will remain glued to it as it won't ever explore any other arm. This is not a good strategy as it does not explore the other arms and there is a high chance of incurring regret at every time step. In this case, regret grows polynomially with time which makes it a poor strategy to opt for.

2. **Random exploration**: Try out different arms, and then exploit the one that has the highest mean reward till that time step. While exploiting that arm, once in a while, with a small probability, select an arm randomly from all the possible arms, independent of the mean of action's value. We will discuss ε-greedy algorithm in detail.

3. **Exploration favouring uncertainty**: In the above approach, the exploration factor was constant. But, now we will discuss algorithms that take into account the fact that some arms are more greedy than other ones and some arms are particularly uncertain. We will discuss the UCB - Upper Confidence Bound Algorithms in detail.

3.2 ε-Greedy Algorithm

It is the most simple and probably the most used algorithm to solve multi armed bandit problem. It involves selecting the arm that has highest mean reward - greedy action with a probability of $(1 - \varepsilon)$ - which is most of the time, and selecting any random arm with a probability of ε out of all the possible arm with equal probability.

3.2.1 Advantages

- It is the most basic and easy to implement algorithm. It is easy to optimize
- It yield better results as compared to greedy approach.

3.2.2 Disadvantages

- It randomly selects an action to explore, and does not take into account the factor that some arms are more favourable than other ones.
- It does not consider confidence intervals i.e. if an action has been explored enough number of times, we don't need to explore it more.

3.3 UCB - Upper Bound Algorithm

1. As we have seen, in case of ε-greedy algorithm, all the arms have equal probability of being selected when exploration will be done. But we know, that all arms are not same and maybe we might end up exploring a bad arm, which we had previously confirmed. To avoid such careless exploration, we have two options i.e. either to reduce ε over time or to explore those arms for which we don't have a confident estimated value until now.

2. UCB family of algorithms is a simple, more elegant implementation of the idea of optimism in the face of uncertainty.[1] It means that it is more important to explore the arms that we are more uncertain about. This uncertainty is caused by variance in the data.

3. UCB algorithm tries to define the upper bound of the mean reward value, $\hat{U}_t$ so that the actual value of the mean reward would always be less than the defined upper bound, $Q(a) \leq \widehat{Q}_t(a) + \widehat{U}_t(a)$ with a high probability.[2]
   $\widehat{U}_t(a)$: It is function of $N_t(a)$, which is the number of times an action a has been selected upto time step 't'.

4. In UCB algorithm, we always select the greediest action to maximize the upper confidence bound: [2]

$$a_t^{UCB} = \arg\max_{a \in A}(\widehat{Q}_t(a) + \widehat{U}_t(a))$$

3.4 UCB1

- In this algorithm, the value of $\widehat{U}_t(a) = \sqrt{\frac{2ln(t)}{N_t(a)}}$.
- $A_t = \arg\max_a \left[(Q_t(a) + \sqrt{\frac{2ln(t)}{N_t(a)}} \right]$ [3]

- $Q_t(a)$: Exploitation Factor i.e. it selects the arm that has the highest mean reward.
- $\sqrt{\frac{2ln(t)}{N_t(a)}}$: Exploration Factor i.e it provides an estimate of the action's rewards.
- If an action has been tried few times, then $N_t(a)$ will be small . It means that the uncertainty term will be high, increasing the chances of this action being selected. If an action has been tried many times, then $N_t(a)$ will be larger, meaning that the uncertainty term will be lower, making it less likely that this action will be selected.
- As time steps increases, the uncertainty term for an action not being selected often, will increase because of the $ln(t)$ term in the numerator.
- As time steps increases, the exploration term tends to zero, that is eventually, actions would be selected on the basis of the exploitation factor.

3.4.1 Advantages

- The exploration factor is not entirely random but it is dependent upon number of times an arm has been selected ($N_t(a)$) and time steps ($ln(t)$).
- It has lower level of regret than that obtained in ε-greedy algorithm. In this case, expected cumulative regret grows logarithmically with time steps. Most of the regret occurs during the initial phase of exploration and after some time, it becomes nearly flat.
- The above feature helps in getting the algorithm better as we want an unexplored arm to get explore more and an explored arm to get explore less.

3.4.2 Disadvantages

- The exploration factor doesn't get affected by the quality of rewards that arm receive.
- UCB1 makes an assumption that the rewards are stochastic and independent for every round which is not always the case. Payoffs could be adversarial as well, which is not considered by UCB1 algorithm.

3.5 UCB3

- It is one of the simplest bandit algorithm. It is a modified ε-greedy algorithm. So, it selects an arm with highest mean reward with probability of (1 - $\sum_{j=1}^{K} \varepsilon_j^n$) and selects any random arm with probability ε_j^n. The difference here is that, the exploration factor depends on the arm, unlike ε-greedy, where every arm had equal probability of being selected during exploration.
- ε_j^n : $\min(1/K, c/d^2 n)$
- UCB3 has two parameters:
  1. **d** : such that $0 < d < \Delta_k(\ \mu_* - \mu_k > 0)$
  2. **c** : such that $c > 0$
  3. These two parameters always occur together as c/d^2

3.5.1 Advantages

- It is an algorithm that takes the good attributes of both ε-greedy and UCB algorithms.
- Regret grows logaritmically with time steps rather than polynomially, as in the case of ε-greedy algorithm.

3.5.2 Disadvantage

- If the parameters are not tuned properly, the performance of the algorithm reduces rapidly.

3.6 UCBV

This algorithm takes into account variance of the rewards in the exploration factor. At any time t plays the arm j that maximizes, [3]

$$\bar{X}_{j,t} + \sqrt{\frac{2\theta V_{j,n_j} log(t)}{N_t(a)}} + \frac{3\theta log(t)}{N_t(a)}$$

3.6.1 Advantages

- Exploration depends on the quality of reward i.e. if the variance of the rewards of that arm is high, it means that we don't have a good idea of that arm and need to explore it more.
- It also has all the good attributes of UCB1.
- An advantage of UCBV over other algorithms that do not involve sample variance is that the regret bound involves σ_j^2/Δ_j instead of $1/\Delta_j$. The value σ_j^2 can be very small in comparison to 1.

Chapter 4

Experiment I

4.1 Dataset

For this project, we have taken a 10 arm bandit problem.
The true value of mean reward i.e.$q_*(A_t)$ for each of the arm was selected according to a normal distribution with mean equal to zero and variance equal to one. If the algorithm selects an arm A_t it gives a reward R_t which was selected from a normal distribution with a mean $q_*(A_t)$.

4.2 Experiment

We analysed performance of different algorithms on this data set for 1000 time steps. We then repeated the same experiment 2000 times, each time with a different bandit problem. This helped us to understand the average behaviour of all algorithms involved. We calculated average reward at each time step over 2000 runs. We also calculated average percentage for number of times best arm was selected by the algorithm.[4]

4.3 Pseudocode

```
rewards[1000] : records the reward at each step
optimal_action[1000] : records if optimal action is taken at this step
for i in range(2000):
  for j in range(1000):
    A = arm_choosen(as_proposed_by_the_algorithm)
    reward[j] += get_reward(A)
    if A == best_arm
    optimal_action[j] += 1

rewards[1000]/2000                    #average over 2000 times
optimal_action[1000]/2000             #average over 2000 times
optimal_action[1000]*100              #percenatage for optimal action
```

Figure 4.1 Pseudocode

Code link

4.4 Results

We measured the performance of these algorithms by calculating two parameters that are **average reward** and **optimal action percentage**. It was found that higher the average reward obtained, better the algorithm performed. Similarly, higher the percentage of optimal actions, better the algorithm.

We can see from the graphs below (figure 4.2 and figure 4.3) that UCBV performed better than UCB1 which performed better than ε-greedy algorithm.

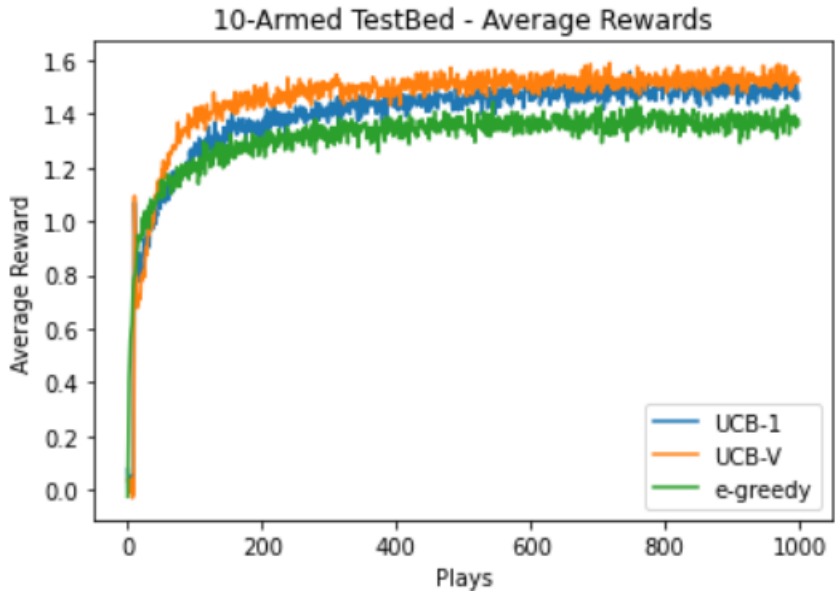

Figure 4.2 Average reward for normal distribution

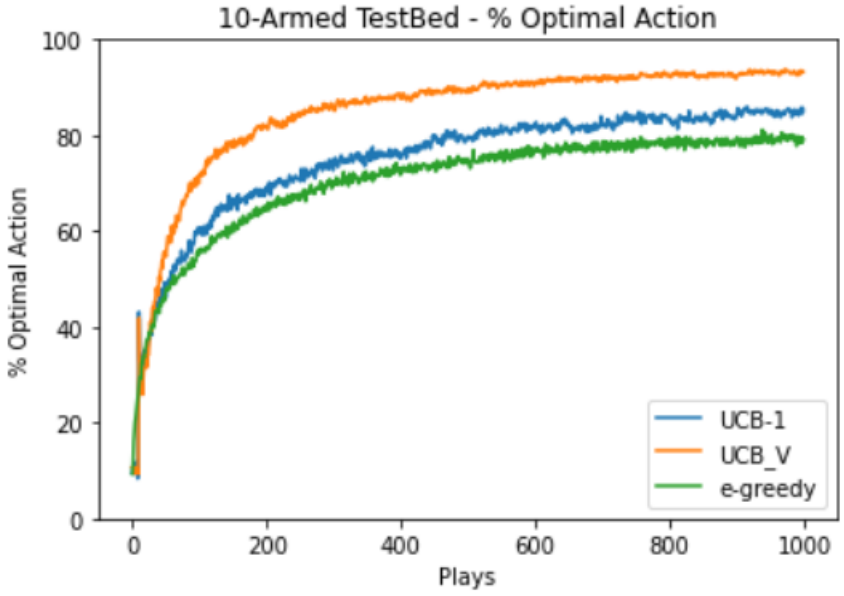

Figure 4.3 Percenatge optimal action for normal distribution

Chapter 5

Experiment II

5.1 Dataset

For this project, we have taken a 10 arm bandit problem.
The true value of mean reward i.e. $q_*(A_t)$ for each of the arm was selected according to a heavy-tailed distribution with mean equal to zero. If the algorithm selects an arm A_t, it gives a reward R_t which was selected from a heavy-tailed distribution with a mean $q_*(A_t)$.
Procedure and **Pseudo code** is same to the ones mentioned in the previous experiment.

5.2 Result

We measured the performance of these algorithms by calculating the same as calculated in the above experiment, on heavy-tailed distribution.
We can see from the graphs below (figure 5.1 and figure 5.2) that UCBV performed better than UCB1 which performed better than ε-greedy algorithm.

$$UCBV > UCB1 > \varepsilon - greedy$$

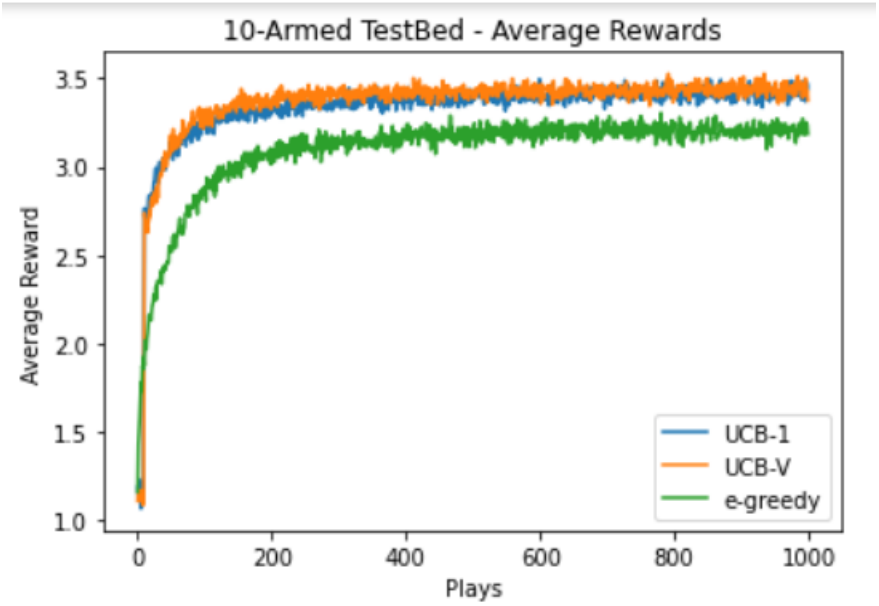

Figure 5.1 Average reward for heavy-tailed distribution

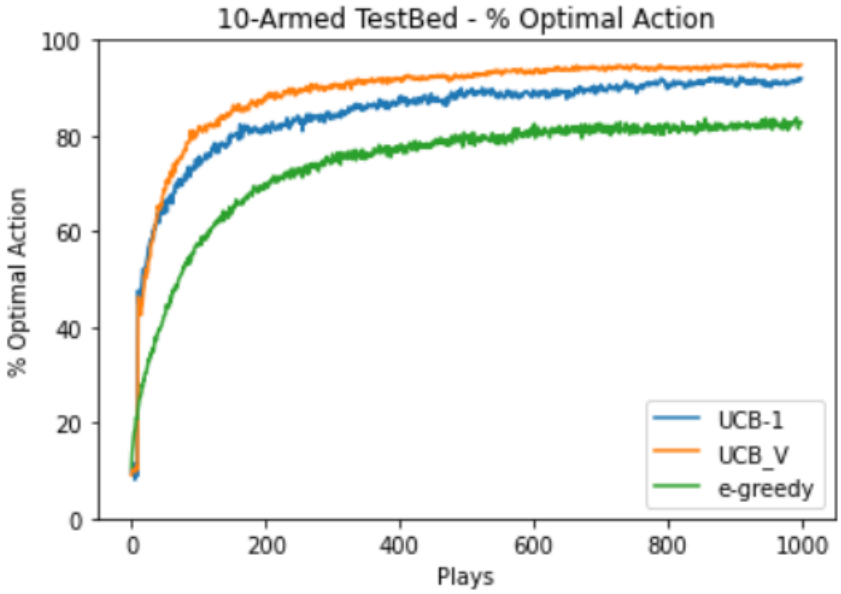

Figure 5.2 Percenatge optimal action for heavy-tailed distribution

Chapter 6

Conclusions

The report has clearly shown that the performance of these algorithms is similar on normal and heavy-tailed distribution and does not depend on the true mean reward values of their arms.

So, we can clearly say that the relative behaviour of these algorithms will not change with change in the reward distribution of the data.

Bibliography

[1] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *arXiv preprint arXiv:1402.6028*, 2014.

[2] L. Weng, "The multi-armed bandit problem and its solutions."

[3] P. Shivaswamy and T. Joachims, "Multi-armed bandit problems with history," in *Artificial Intelligence and Statistics*. PMLR, 2012, pp. 1046–1054.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.