Evaluation of performance: multi-armed bandit vs. contextual bandit

by

Ranojoy Chatterjee

B.Tech., West Bengal University of Technology, 2017

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2019

Approved by:

Major Professor
Dr. William Hsu

Copyright

Abstract

This work compares two methods, the multi-armed bandit (MAB) and contextual multi-armed bandit (CMAB), for action recommendation in a sequential decision making domain. It empirically evaluates their effectiveness on a customer relationship management task. The goal of this project is to experiment using ε-greedy and random selection strategies to characterize the exploration vs. exploitation tradeoff , which manifests when trying to increase or maximize profit while gaining new information regarding the process. The first method under observation, the *multi-armed bandit* (MAB), is simpler to compute and scales better to larger amounts of data; it has a wide range of applicability, including website optimization, clinical trials, adaptive routing, and stock trading. The contextual multi-armed bandit (CMAB) is an advanced version of the multi-armed bandit which takes into consideration the user's past usage patterns, especially historical features of the user's search history; its training data incorporates this context, resulting in a model that is more accurate but also requires a lot of user data which incurs privacy liabilities, an adverse property. This study measures the difference in outcome if the MAB or CMAB have access to user data and assesses, for a real-world application domain, whether this trade-off is significant and worthwhile in the bigger prospective.

Table of Contents

List of Figures

Acknowledgements

Firstly, I am thankful to Dr. William Hsu for his academic support, time, and guidance towards completing these research project. I am also grateful to Dr. Hsu for his patience and allowing me to become independent towards undertaking my research works. I am also thankful to Dr. Mitchell Nielsen and Dr. Torben Amtoft for agreeing to be part of my committee and giving me great insights regarding my work.

I am also grateful to Matt Moody and the Bellwethr team for their patience and guidance and letting me work in numerous projects and teaching numerous coding hacks which were highly instrumental in this project. They were more a group of like-minded people rather than colleagues.

I am also indebted to my parents and my elder brother Dr. Suvo Chatterjee, for their patience, guidance, and unwavering trust in me, without which I would not be here. Dr. Chatterjee unconditional love and guidance was one of the pivotal reasons what made me who I am.

I would also like to thank my friends Parthapratim, Avishek, Abu Maroof for their constant love and support, they have been a family away from family.

Most importantly I would like to thank God, without his blessing nothing would have been possible.

Chapter 1- Introduction

1.1. Problem Definition

A massive amount of work is required for acquisition of customers in a company, including research, sales and marketing effort, and other costs – all of which are forfeited when a hard-earned customer leaves a company. It is more practical for the company to retain these existing customers, so as to increase the revenue of the company and direct this revenue to proper channels and make the best outcomes. Knowledge-based systems and collaborative filtering are the most prevalent methods used in industrial approaches to the task of recommendation, but each of these approaches has its own drawbacks. In knowledge-based filtering, the greatest flaw is that the knowledge base is static and for collaborative filtering the quality is dependent on large historical data; this knowledge base is also prone to large statistical anomalies and reacts slowly to drifts (1). Multi-armed bandit takes account the different strategies employed by the marketing team and presents the outgoing customer with a suitable option from those above strategies, which can lead to them not leaving.

According to studies done by Bain & Company, increasing customer retention by 5% can lead to an increase in profits of 25% – 95%, and the likelihood of converting an existing customer into a repeat customer is 60% – 70%, while the probability of converting a new lead is 5% – 20%, at best (2). Thus, we can see how important it is to use a retention engine to reduce the churn rate.

1.2. Objective

I chose the domain of retention engine within which to apply a multi-armed bandit, since there are any small to medium companies that don't have access to big resources to run a complete reinforced learning algorithm to get the best results and this in turn is making them loose many of their customer base to bigger multi-national companies. The results which will be produced from multi-armed bandits are significantly better than that of greedy and random recommendation. I will also use a version of contextual multi-armed bandit which is a reinforcement learning method and try to show, how much difference in result you get from multi-armed bandit and contextual multi-armed bandit.

1.3. Overview

In this project, I use a simulated data set which represents the choice of a customer given the various alternatives presented to them. The various choices are given a score which is binomially distributed according to a probability *p*, which is continuously adjusted by the steps taken by the agent. In the second part of the comparison I use a data set which is available online . This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule such as ”leaflets three, let it be” for poison oak and poison ivy [3]. The results are obtained using the simulated data set and the data set from UCI mushroom data set and compared using Thompson Sampling as the algorithm for both MAB and CMAB. Scikit-learn (Pedregosa, Varoquaux, Gramfort, & Michel, 2011) was instrumental in implementing the above said process. Visualization packages such as Matplotlib (Hunter, Dale, Dorettboom, & Team, 2012) and Seaborn (Waskom, 2012-2017) were used to compare the two process.

Chapter 2 – Background and Related Work

In this part of the chapter, I will introduce the MAB and CMAB used in this project. I will also describe some of the data preprocessing used for the CMAB.

2.1 Literature Survey

Multi-armed bandit problems have been introduced by Robbins (1952) and have since been used extensively to model the trade-offs faced by an automated agent which aims to gain new knowledge by exploring its environment and to exploit its current, reliable knowledge. Such problems arise frequently in practice, for example in the context of clinical trials or on-line advertising. The multi-armed bandit problem offers a very clean, simple theoretical formulation for analyzing trade-offs between exploration and exploitation. A comprehensive overview of bandit problems from a statistical perspective is given in Berry & Fristedt (1985).[2]

Internet search engines, such as Google, Yahoo! and Microsoft's Bing, receive revenue from advertisements shown to a user's query. Whenever a user decides to click on an ad displayed for a search query, the advertiser pays the search engine. Thus, part of the search engine's goal is to display ads that are most relevant to the user in the hopes of increasing the chance of a click, and possibly increasing its expected revenue. In order to achieve this, the search engine has to learn over time which ads are the most relevant to display for different queries. On the one hand, it is important to exploit currently relevant ads, and on the other hand, one should explore potentially relevant ads. This problem can be naturally posed as a multi-armed bandit problem with context. Here by context we mean a user's query. Each time a query x arrives and an ad y is displayed there is an (unknown) probability μ(x, y) that the user clicks on the ad.1 We call μ(x, y) the click-through rate (or CTR) of x and y. [3]

2.2 Established Methods

2.2.1 Random Sampling

Random sampling refers to a variety of selection techniques in which sample members are selected by chance, but with a known probability of selection. Most social science, business, and agricultural surveys rely on random sampling techniques for the selection of survey participants or sample units, where the sample units may be persons, establishments, land points, or other units for analysis. Random sampling is a critical element to the overall survey research design. [4]

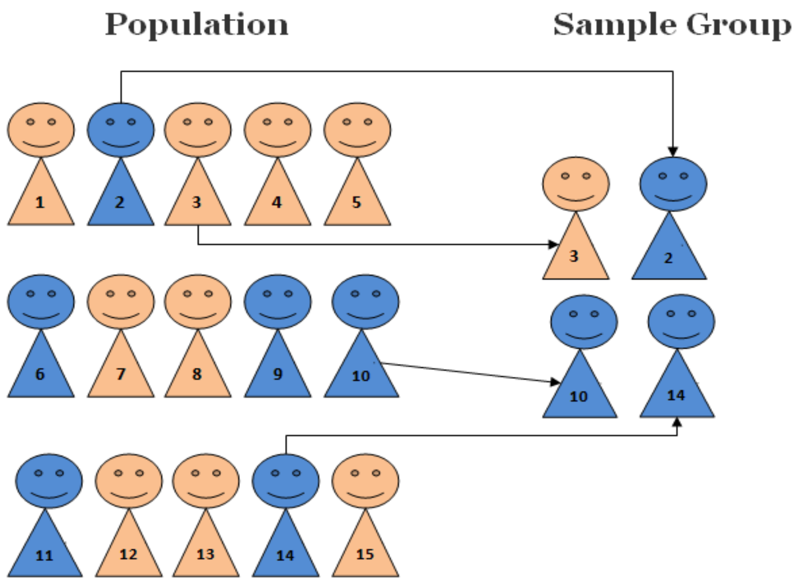


Figure 2.1 Random Sampling [5]

2.2.2 Epsilon Greedy

The ε-greedy algorithm is widely used because it is very simple and has obvious generalizations for sequential decision problems. At each round t = 1, 2, ... the algorithm selects the arm with the highest empirical mean with probability 1 − ε, and selects a random arm with probability ε. In other words, given initial empirical means μ1(0), ..., μK(0),

$$p_i(t+1) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{k}, & \text{if } i = \max\arg_{j=1,\dots,K} \mu j\,(t) \\ \frac{\varepsilon}{k}, & otherwise \end{cases}$$

otherwise. If ε is held constant, only a linear bound on the expected regret can be achieved. CesaBianchi and Fisher (1998) proved poly-logarithmic bounds for variants of the algorithm in which ε decreases with time. In an earlier empirical study, Vermorel and Mohri (2005) did not find any practical advantage to using these methods. Therefore, in our experiments, we will only consider fixed values of ε.[2]

2.2.3 Multi-armed bandit

In its simplest formulation (generally referred to as stochastic), a bandit problem consists of a set of K probability distributions [$D_1, \dots, D_{Ki}$] with associated expected values [$\mu_1, \dots, \mu_{Ki}$] and variances [$\sigma_1^2, \dots, \sigma_k^2$]. Initially, the Di are unknown to the player. In fact, these distributions are generally interpreted as corresponding to arms on a slot machine; the player is viewed as a gambler whose goal is to collect as much money as possible by pulling these arms over many turns. At each turn, t = 1, 2, ..., the player selects an arm, with index j(t), and receives a reward r(t) ∼ Dj(t). The player has a two-fold goal: on one hand, finding out which distribution has the highest expected value; on the other hand, gaining as much rewards as possible while playing. Bandit algorithms specify a strategy by which the player should choose an arm j(t) at each turn. The most popular performance measure for bandit algorithms is the total expected regret, defined for any fixed turn T as:

$$RT = T\,\mu^* - \sum_{t=1}^{T} uj(t)$$

where $\mu^* = \max\ i=1\ ,\ldots,\ k\ \mu_i$ is the expected reward from the best arm.

Alternatively, we can express the total expected regret as

$$RT = T\,\mu^* - \mu_{j(t)} \sum_{k=1}^{K} E\big(Tk(T)\big)$$

where $T_k(T)$ is a random variable denoting the number of plays of arm k during the first T turns. A classical result of Lai and Robbins (1985) states that for any suboptimal arm k,

$$E(Tk(T)) \geq \ln T / D(p_k \| p_*)$$

where $D(p_j \| p_*)$ is the Kullback-Leibler divergence between the reward density pk of the suboptimal arm and the reward density p^* of the optimal arm, defined formally as

$$D(p_k \| p_*) = \int pj = \ln\left(\frac{pj}{p} *\right)$$

Regret thus grows at least logarithmically, or more formally, $RT = \Omega(\log T)$. An algorithm is said to solve the multi-armed bandit problem if it can match this lower bound, that is if $RT = O(\log T)$.

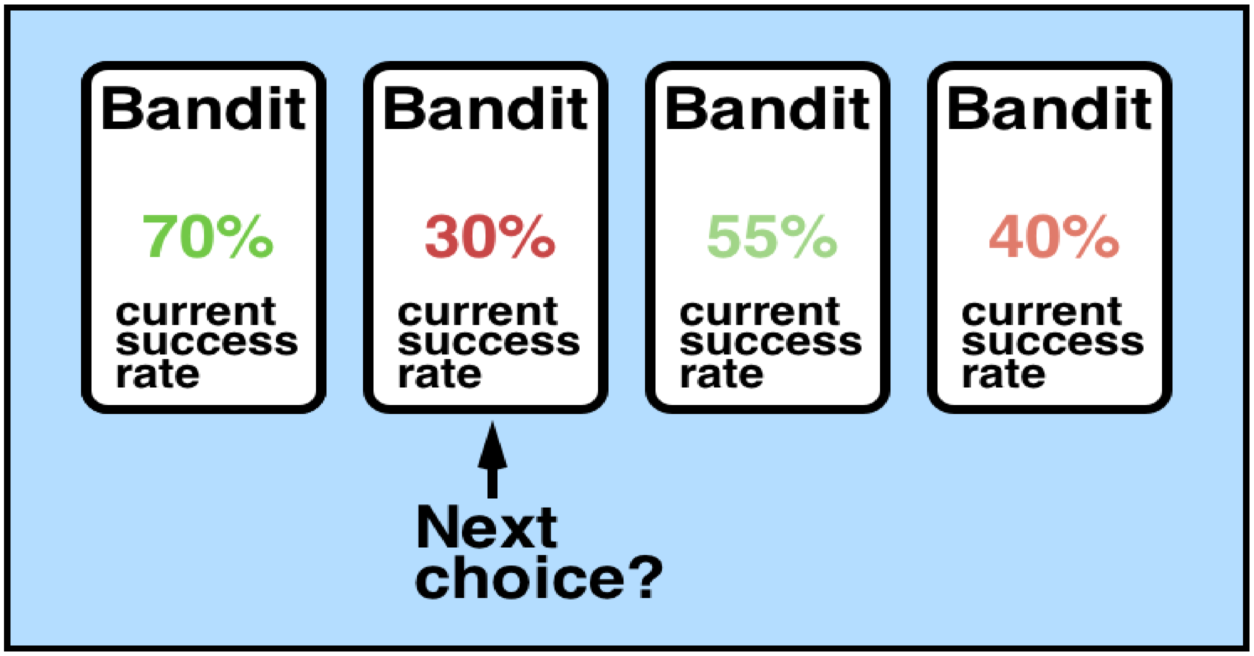


Figure 2.2 Multi-armed bandit[6]

2.2.4 Contextual Multi-armed bandit

In the contextual bandit problem, an agent collects rewards for actions taken over a sequence of rounds; in each round, the agent chooses an action to take on the basis of (i) context (or features) for the current round, as well as (ii) feedback, in the form of rewards, obtained in previous rounds. The feedback is incomplete: in any given round, the agent observes the reward only for the chosen action; the agent does not observe the reward for other actions. Contextual bandit problems are found in many important applications such as online recommendation and clinical trials and represent a natural half-way point between supervised learning and reinforcement learning. The use of features to encode context is inherited from supervised machine learning, while exploration is necessary for good performance as in reinforcement learning. [7]

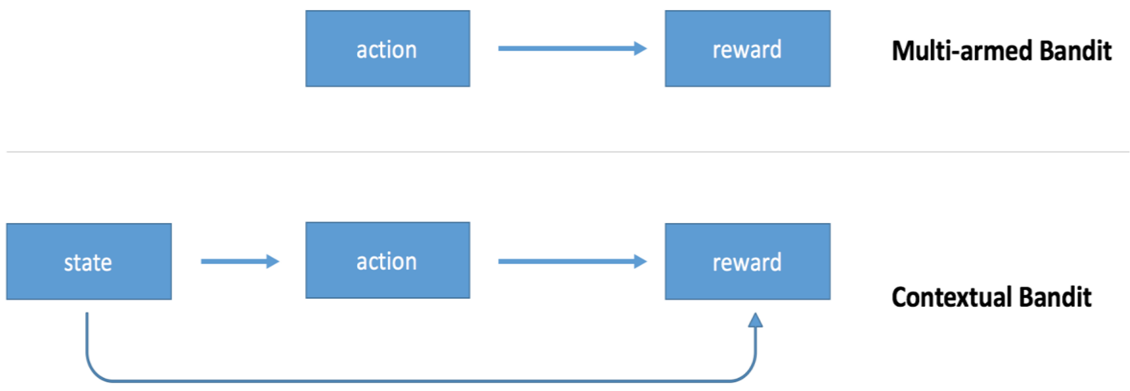


Figure 2.3 Multi-armed bandit vs Contextual Multi-armed bandit[1]

2.2.5 Thompson Sampling

Thompson Sampling (Posterior Sampling or Probability Matching) is an algorithm for choosing the actions that address the exploration-exploitation dilemma in 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 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 an 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 tendency to be more “search-oriented” when we have fewer data and less “search-oriented” when we have a lot of data.

Chapter 3 – Implementation

In this chapter I will discuss the implementation of the MAB and CMAB using Thompson Sampling.

3.1 Experiment Setup

Initially we define an environment, in which random sampling, ε-greedy and the MAB will run. The environment calls an agent , the agent takes the decision of which action to run and the environment executes it and then feeds the reward from the action taken to the agent so that the agent can updates itself. In the random sampling algorithm, the agent simply takes a decision randomly and then neither updates or learn from the decision taken, this method is simply put here for a baseline for the other algorithms defined here and see how better or worse the other defined algorithms work. In ε-greedy, the agent chooses a random (x) out of n-trials and on each trial estimates the payout for each choice and updates in the environment class, after n_k learning trials it then selects 1-ε% of time the *x* that has the highest reward rate and ε% of the time samples the variant randomly. In this approach the sampler explores the various variants and also exploits the variant which has the highest return rate at the same time and while exploration it keeps in track all the change in return rate for the various choices and chooses the optimal variant accordingly.

The MAB algorithm using the Thompson Sampler also works on the same principle as the ε-greedy but with some refinement.

- It is not greedy in nature;
- The exploration is done in more streamlined and refined manner;
- It is Bayesian in nature.

For simplicity of discussion, we first provide the details of Thompson Sampling algorithm for the Bernoulli bandit problem, i.e. when the rewards are either 0 or 1, and for arm i the probability of success (reward =1) is μi. This description of Thompson Sampling follows closely that of Chapelle and Li (2011). Next, we propose a simple new extension of this algorithm to general reward distributions with support [0, 1], which will allow us to seamlessly extend our analysis for Bernoulli bandits to general stochastic bandit problem. The algorithm for Bernoulli bandits maintains Bayesian priors on the Bernoulli means μ_I's. Beta distribution turns out to be a very convenient choice of priors for Bernoulli rewards. Let us briefly recall that beta distributions form a family of continuous probability distributions on the interval (0, 1). The Probability Distribution Function (pdf) of Beta(α, β), the beta distribution with parameters α > 0, β > 0, is given by

$$\mathrm{f}(\mathrm{x};\ \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

The mean of Beta(α, β) is α/(α + β); and as is apparent from the pdf, higher the α, β, tighter is the concentration of Beta(α, β) around the mean. Beta distribution is useful for Bernoulli rewards because if the prior is a Beta(α, β) distribution, then after observing a Bernoulli trial, the posterior distribution is simply Beta(α+1, β) or Beta(α, β+1), depending on whether the trial resulted in a success or failure, respectively. The Thompson Sampling algorithm initially assumes arm i to have prior Beta(1, 1) on μi, which is natural because Beta(1, 1) is the uniform distribution on (0, 1). At time t, having observed Si(t) successes (reward = 1) and Fi(t) failures (reward = 0) in ki(t) = Si(t) + Fi(t) plays of arm i, the algorithm updates the distribution on μi as Beta(Si(t) + 1, Fi(t) + 1). The algorithm then samples

from these posterior distributions of the μi's, and plays an arm according to the probability of its mean being the largest. We summarize the Thompson Sampling algorithm below.
We adapt the Bernoulli Thompson sampling algorithm to the general stochastic bandits case, i.e. when the rewards for arm i are generated from an arbitrary unknown distribution with support [0, 1] and mean μi, in a way that allows us to reuse our analysis of the Bernoulli case. To our knowledge, this adaptation is new. We modify TS so that after observing the reward $r\tilde{}t \in [0, 1]$ at time t, it performs a Bernoulli trial with success probability $r\tilde{}t$. Let random variable rt denote the outcome of this Bernoulli trial, and let {Si(t), Fi(t)} denote the number of successes and failures in the Bernoulli trials until time t. The remaining algorithm is the same as for Bernoulli bandits.

Thus, the probability of observing $rt = 1$ is same and Si(t), Fi(t) evolve exactly in the same way as in the case of Bernoulli bandits with mean μi. Therefore, the analysis of TS for Bernoulli setting is applicable to this modified TS for the general setting. This allows us to replace, for the purpose of analysis, the problem with general stochastic bandits with Bernoulli bandits with the same means. We remark that instead of using rt, we could consider more direct and natural updates of type $Beta(\alpha i, \beta i)$ to $Beta(\alpha i + \tilde{}rt, \beta i + 1 - rt)$. (Agrawal & Goyal, 2012)

Thompson Sampling (Posterior Sampling or Probability Matching) is an algorithm for choosing the actions that address the exploration-exploitation dilemma in 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 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.

The CMAB was implemented using the python package c*ontextual bandit* which is based "*Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling"* . Thompson Sampling is a meta-algorithm that chooses an action for the contextual bandit in a statistically efficient manner, simultaneously finding the best arm while attempting to incur low cost. Informally speaking, we assume the expected reward is given by some function E[rt | Xt, at] = f(Xt, at). Unfortunately, function f is unknown, as otherwise we could just choose the action with highest expected value: at* = arg maxi f(Xt, at).

The idea behind Thompson Sampling is based on keeping a posterior distribution πt over functions in some family f $\in$ F after observing the first t-1 datapoints. Then, at time t, we sample one potential explanation of the underlying process: ft $\sim$ πt, and act optimally (i.e., greedily) according to ft. In other words, we choose at = arg maxi ft(Xt, ai). Finally, we update our posterior distribution with the new collected datapoint (Xt, at, rt).

The main issue is that keeping an updated posterior πt (or, even, sampling from it) is often intractable for highly parameterized models like deep neural networks. [8]

3.2 Data set

Mushroom Data set contain 8124 attributes and 22 features collected from 23 species of gilled mushrooms in the Agaricus and Lepiota family, each species has been defined as edible or inedible.

Chapter 4 - Experimental Results

4.1 Regret

Regret is calculated by subtracting the reward of the choice by the maximum available reward in a given turn.

$Reg_{i} = \max(R_i) - R_i$, where R_i = reward chosen in the ith iteration

4.2 Cumulative Regret

The sum of the regret generated over one simulation.

$$R_{total=} \sum_{i=1}^{n} regret$$

4.3 Variant Selection

A plot which shows how different variants are sampled in random, ε-greedy and in MAB.

4.4 Score

Total reward gained by an agent per simulation. It is the cumulative valuation of the total reward received per trial.

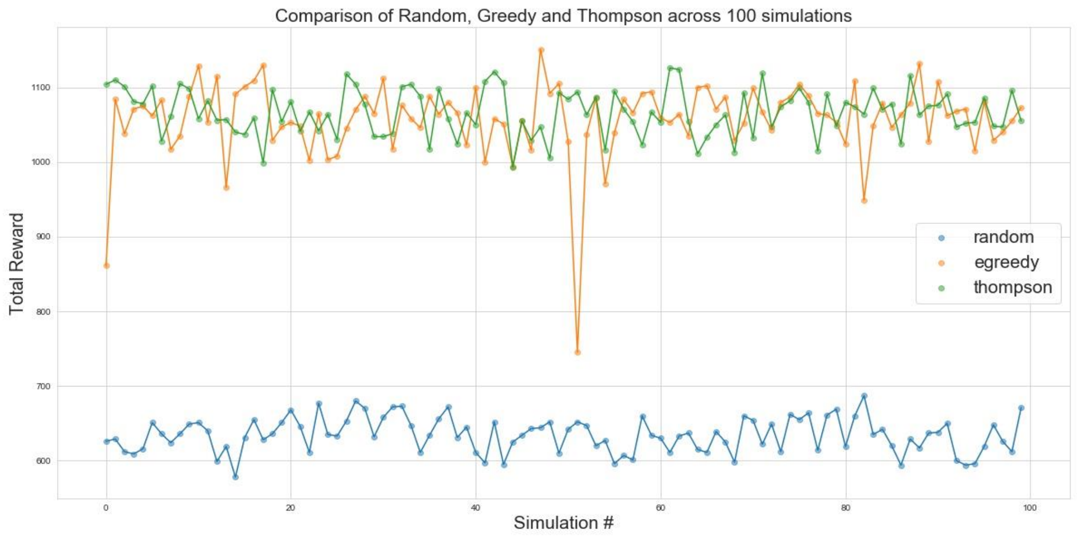


Figure 4.1 Comparison of random, ε-greedy and MAB score

In the above figure, we can clearly see the different score of random, epsilon-greedy and MAB. MAB outperforms random sampling but is near to epsilon-greedy apart from some sub-optimal points where epsilon greedy gets stuck and produces a bad score. Using seaborn[9] and matplotlib[10] we plot these graphs.

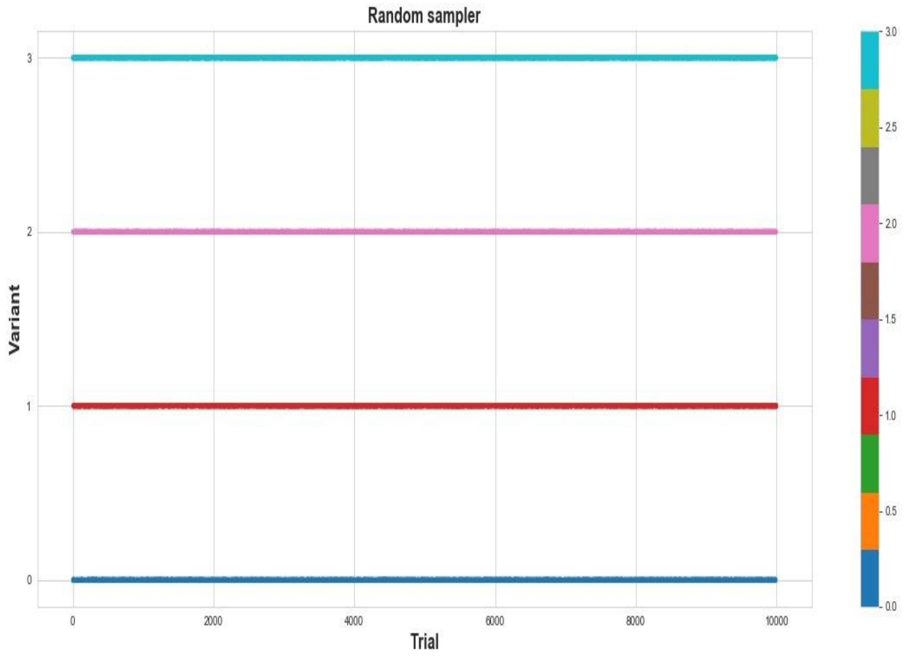


Figure 4.2 Variant Sampling in Random Sampler

The figure above shows, how the variant(arms) are selected in a random sampler which clearly shows that it randomly selects all the arms. The concept of random sampling is total exploration and no exploitation. The color coding for 0, 1, 2 and 3 were only used since we are dealing with only 4 arms.

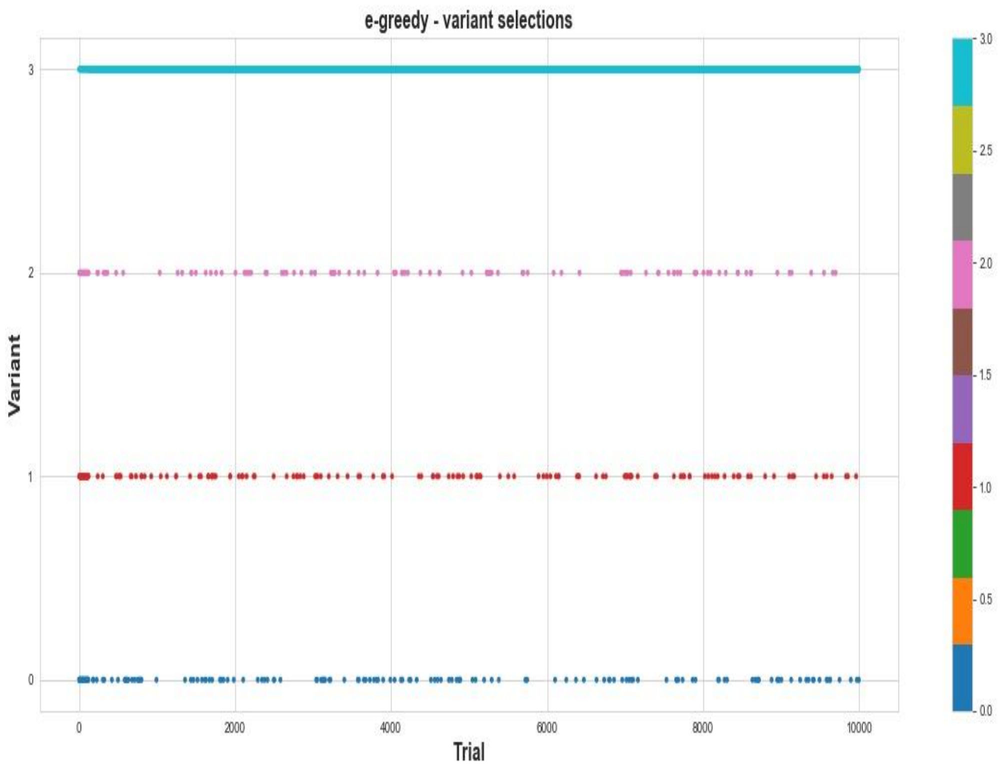


Figure 4.3 Variant Sampling in ε-greedy

The figure clearly depicts of total exploitation of the best arm and random exploration in ε% times, so the rest of the arms are randomly picked, and no information is collected from the exploration. The color coding for 0, 1, 2 and 3 were only used since we are dealing with only 4 arms.

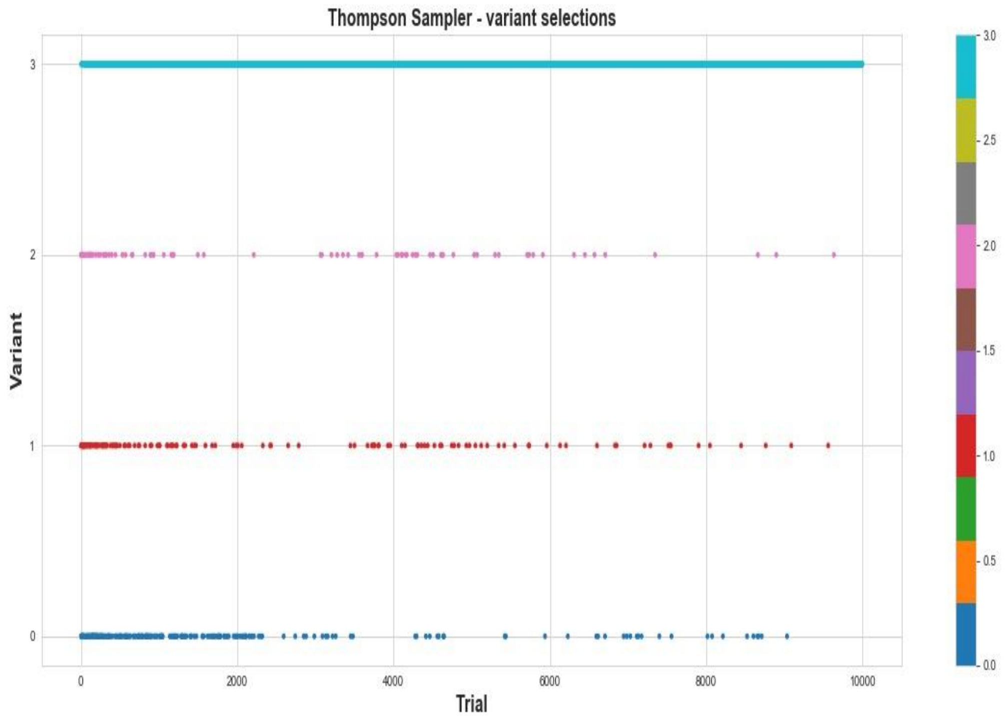


Figure 4.4 Variant Sampling in Thompson Sampler

It can be clearly seen that the Thompson takes a more refined and selection, because it exploits the best option which is arm 3 but it also constantly visits arm 0 and arm 1, because they have a clear posterior probability distribution which is better than arm 2. The color coding for 0, 1, 2 and 3 were only used since we are dealing with only 4 arms.

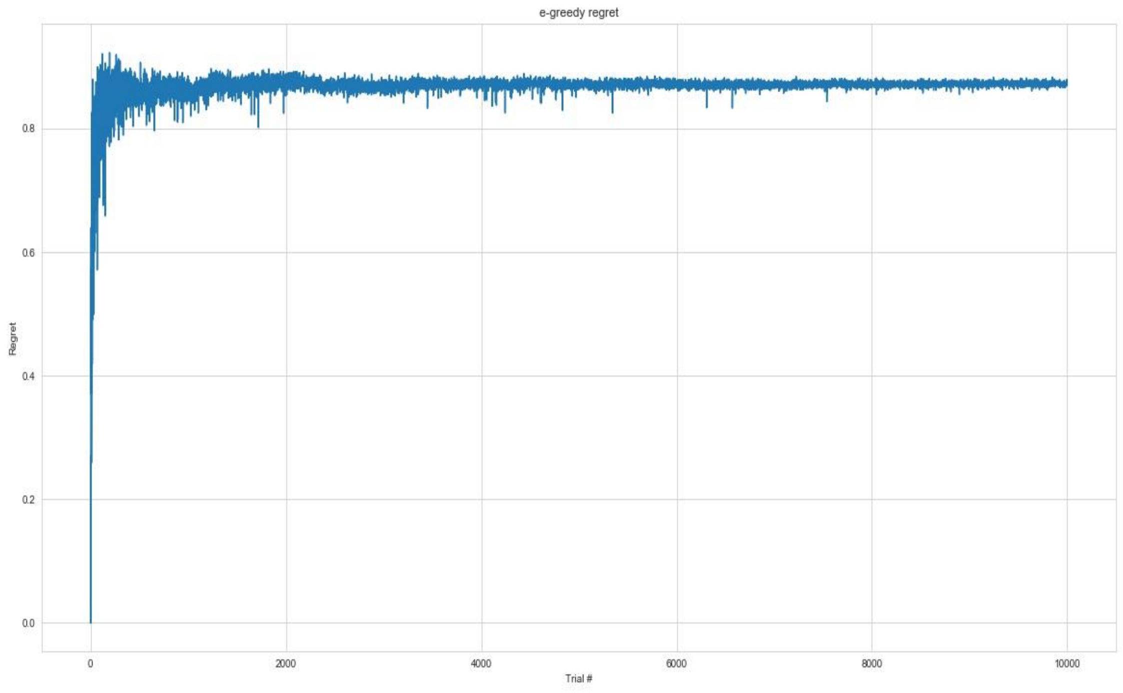


Figure 4.5 Regret for ε-greedy

The graph shows clearly the regret of the epsilon greedy process over iterations. We can clearly see that; the regret has a steep incline but after enough iteration it seems to bottom out and converge.

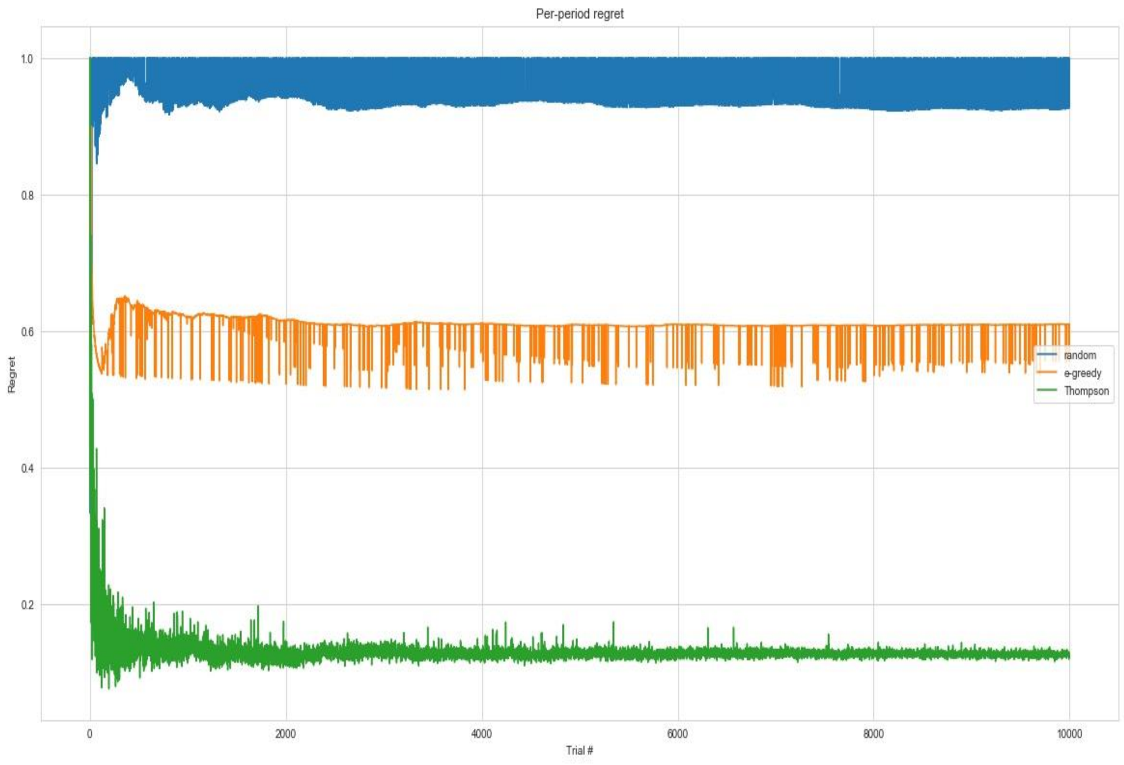


Figure 4.6 Per-period regret for random, ε-greedy and MAB

This plot shows the per trial regret. It can be clearly seen that Thompson sampling converge very quickly to a point of less regret, than that of ε-greedy but random sampling has no convergence at all.

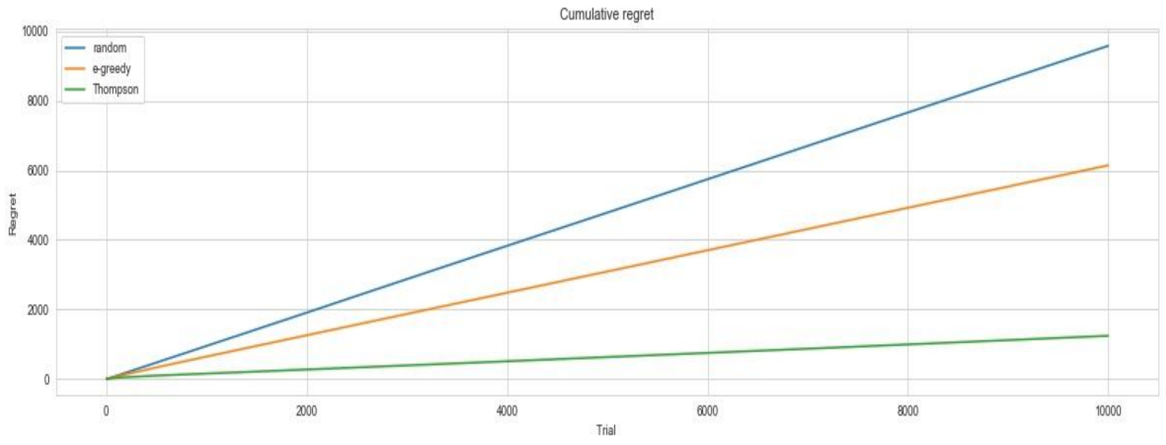


Figure 4.7 Cumulative regret of random sampling, ε-greedy and MAB

We can clearly see that Thompson Sampling converges better and reduces the regret better. The slope of ε- greedy is much steeper than the Thompson Sampling. Random Sampling is the worst of the three, but that was expected.

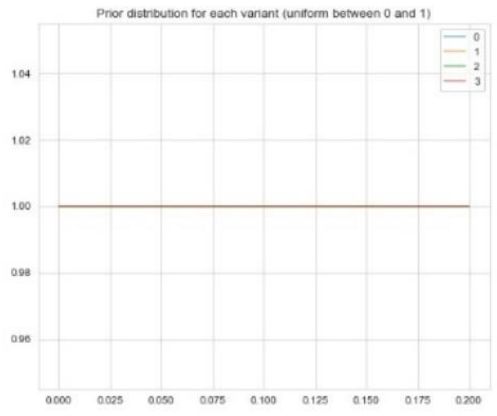


Figure 4.8 Beta distribution of 4 machines initially

In the beginning the posterior distribution of all the arms are set at zero, since we got no information regarding the success and failures of the respective arms.

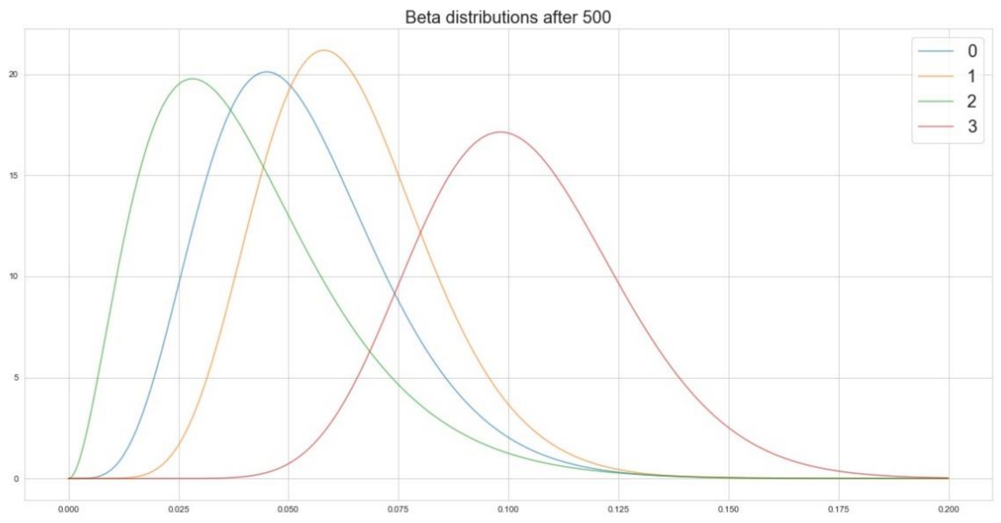


Figure 4.9 Beta distribution of 4 machines after 500 runs

After 500 iteration we can see how the posterior distribution starts to change and we can clearly see that arm 3 has the best posterior distribution but arm 2 and arm 1 are very alike each other.

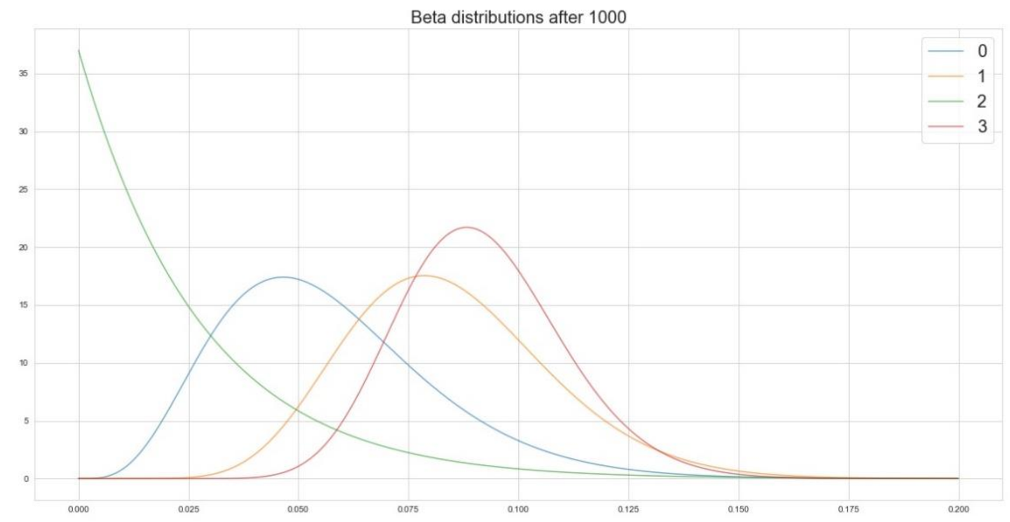


Figure 4.10 Beta distribution of 4 machines after 1000 runs

After 1000 iteration we still can see that arm 4 wins over the rest of the alternatives, but the posterior distribution of the arm 1 gets better than arm 1, but still inconclusive to separate from each other.

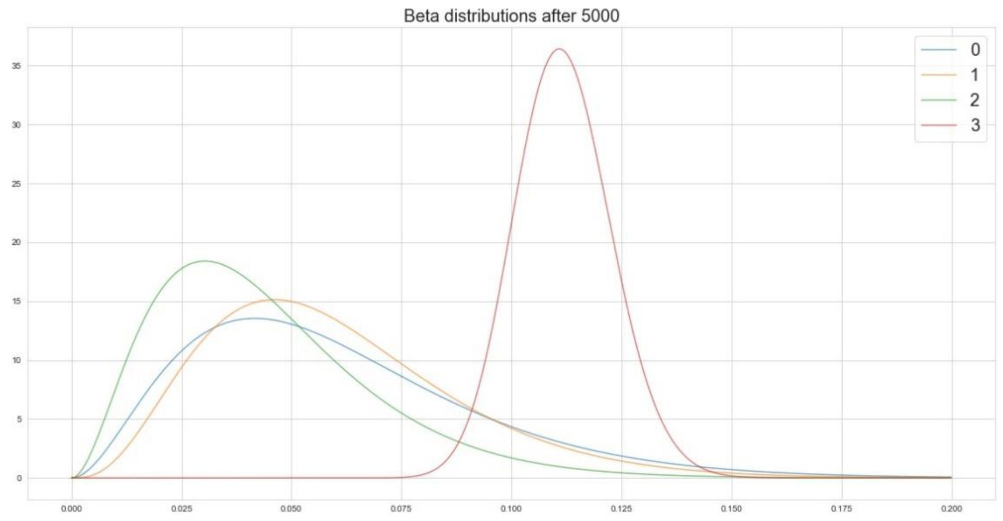


Figure 4.11 Beta distribution of 4 machines after 5000 runs

In the figure we see a stark difference in the posterior distribution of the arms where clearly arm 3 clearly but we see the difference in the posterior probability in arm 1 and arm 2. This clearly suggests that the arm 2 has more success in relation to arm 1.

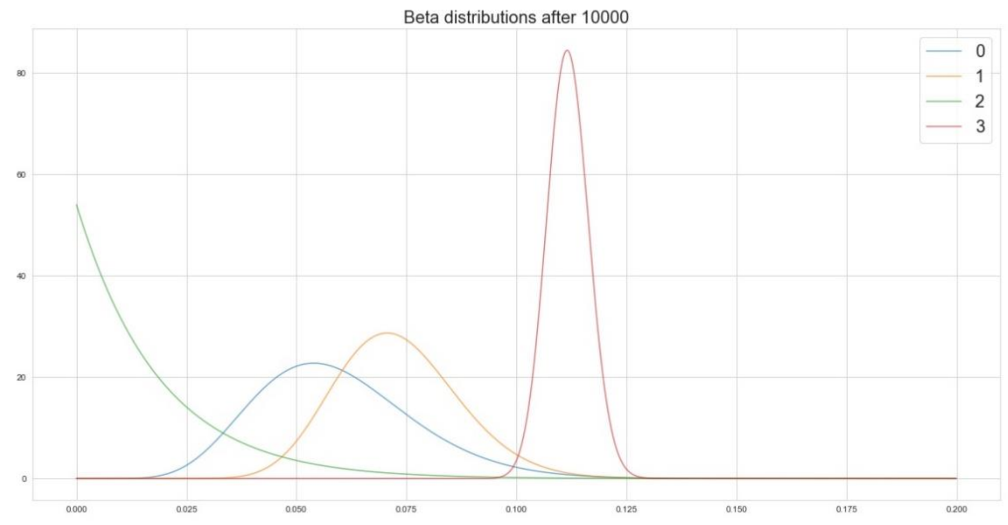


Figure 4.12 Beta distribution of 4 machines after 10000 runs

The posterior distribution of the arm 0 and arm 1 are now very close to each other but we can see the clear winner that is arm 3 whose posterior distribution is higher than the rest of the other arms.

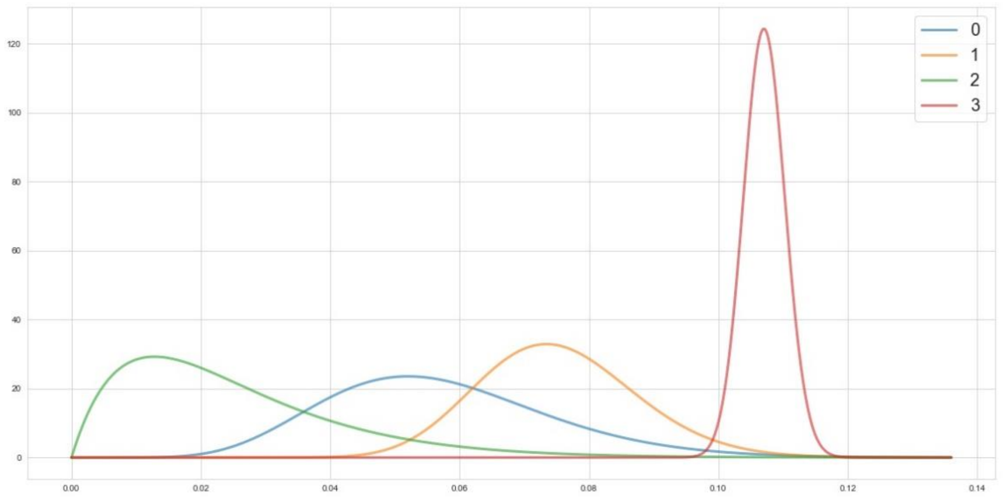
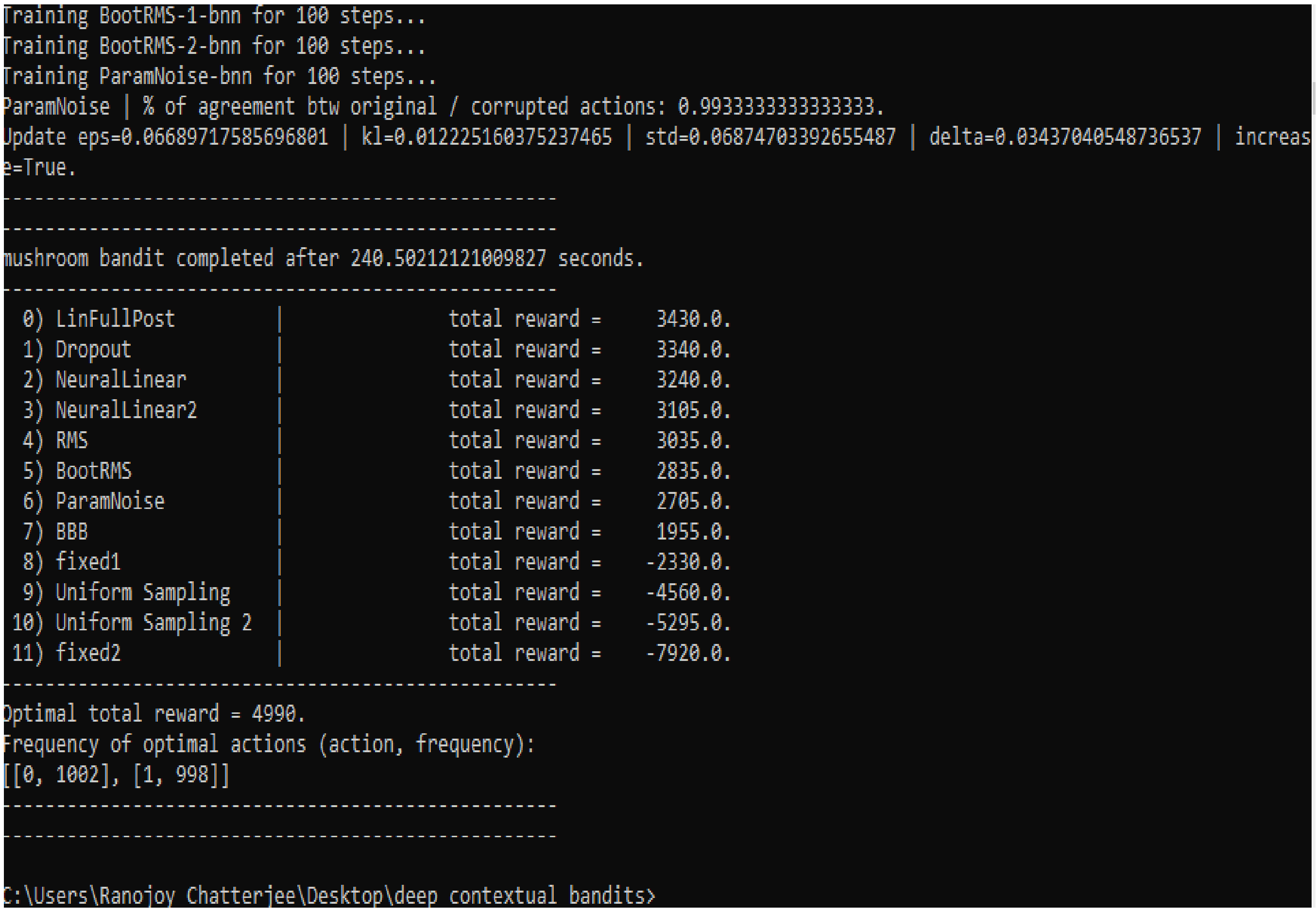


Figure 4.13 Beta distribution of 4 machines after 200000 runs

The final posterior distribution of the arms where we can clearly see it begins to converge to the best alternative of the arms, which in our case is the arm 3.

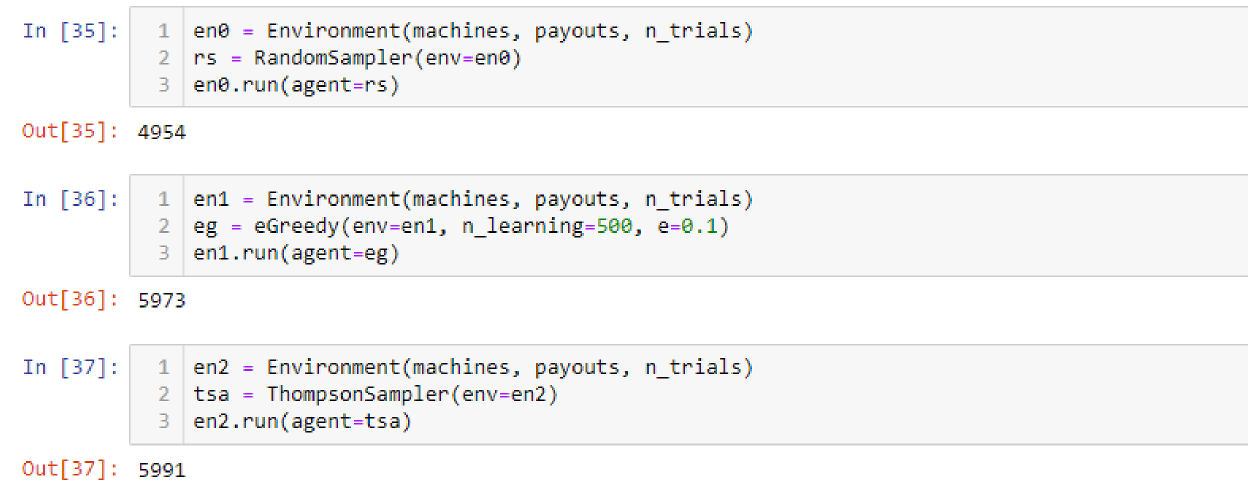


```
Training BootRMS-1-bnn for 100 steps...
Training BootRMS-2-bnn for 100 steps...
Training ParamNoise-bnn for 100 steps...
ParamNoise | % of agreement btw original / corrupted actions: 0.9933333333333333.
Update eps=0.06689717585696801 | kl=0.012225160375237465 | std=0.06874703392655487 | delta=0.03437040548736537 | increas
e=True.
---------------------------------------------------
---------------------------------------------------
mushroom bandit completed after 240.50212121009827 seconds.
---------------------------------------------------
  0) LinFullPost        |              total reward =     3430.0.
  1) Dropout            |              total reward =     3340.0.
  2) NeuralLinear       |              total reward =     3240.0.
  3) NeuralLinear2      |              total reward =     3105.0.
  4) RMS                |              total reward =     3035.0.
  5) BootRMS            |              total reward =     2835.0.
  6) ParamNoise         |              total reward =     2705.0.
  7) BBB                |              total reward =     1955.0.
  8) fixed1             |              total reward =    -2330.0.
  9) Uniform Sampling   |              total reward =    -4560.0.
 10) Uniform Sampling 2 |              total reward =    -5295.0.
 11) fixed2             |              total reward =    -7920.0.
---------------------------------------------------
Optimal total reward = 4990.
Frequency of optimal actions (action, frequency):
[[0, 1002], [1, 998]]
---------------------------------------------------
---------------------------------------------------


C:\Users\Ranojoy Chatterjee\Desktop\deep contextual bandits>
```

Figure 4.14 CMAB score on mushroom data set

The screenshot shows that the score of the Contextual Bandit when used on Mushroom data set from UCI [11] with Thompson Sampling has the best score of 4990. The data set has two labels only one is poisonous and edible.



```
In [35]:
en0 = Environment(machines, payouts, n_trials)
rs = RandomSampler(env=en0)
en0.run(agent=rs)

Out[35]: 4954

In [36]:
en1 = Environment(machines, payouts, n_trials)
eg = eGreedy(env=en1, n_learning=500, e=0.1)
en1.run(agent=eg)

Out[36]: 5973

In [37]:
en2 = Environment(machines, payouts, n_trials)
tsa = ThompsonSampler(env=en2)
en2.run(agent=tsa)

Out[37]: 5991
```

Figure 4.15 MAB score on Mushroom data set

Using the same UCI Mushroom data set on the MAB using Thompson Sampling, we get a higher score than the CMAB. The same test parameters were maintained in both CMAB and MAB. While running a cross validation accuracy precision on MAB and Logistic regression, the mean score of MAB was found to be 0.967 whereas the mean score of Logistic Regression was 0.996.

Chapter 5 – Result & Analysis

Under the same condition and data set, MAB produced a better result than CMAB as was stated above with the MAB score being 5991 and CMAB score 4990. According to literature review it was found that CMAB will out-perform MAB, but in our case, it was found to be not true, and MAB outperformed random and ε-greedy algorithms.

The justification can be said that due to simplification of the data set, the problem transformed into a 2 armed bandit problem, and since the number of non-poisonous mushroom were greater, so the posterior probability of the non-poisonous was higher than that of the poisonous which made the algorithm choose the non-poisonous more than that of the poisonous, which in turn made the reward greater than the CMAB.

Chapter 6 – Future Work

The topic of recommender system is one of the most sorts after research topic in this day and age, from recommending a song or a drug for cancer treatment depends on a versatile and adaptable algorithm. Introduction of MAB in the art of movies has already increased Netflix customer base by 20%. This idea is also being used in retention system, companies like Bellwethr are using the same strategies to retent customer and reduce the churn rate. This method can be used to build a better search engine and also can be used to build better routing technique . This application has many utility and can be used in many number of places.

References

[1] A. Juliani, “Simple Reinforcement Learning with Tensorflow Part 1.5: Contextual Bandits.” [Online]. Available: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-1-5-contextual-bandits-bff01d1aad9c.

[2] V. Kuleshov and D. Precup, “Algorithms for multi-armed bandit problems,” Feb. 2014.

[3] M. Lu, T., Pál, D., & Pál, “Contextual multi-armed bandits,” *Proc. Thirteen. Int. Conf. Artif. Intell. Stat.*, pp. 485–492.

[4] “Random Sampling,” in *Encyclopedia of Survey Research Methods*, 2455 Teller Road, Thousand Oaks California 91320 United States of America: Sage Publications, Inc.

[5] “Research Methodology.” [Online]. Available: https://research-methodology.net/sampling-in-primary-data-collection/random-sampling/.

[6] A. Wong, “Solving the Multi-Armed Bandit Problem.” [Online]. Available: https://towardsdatascience.com/solving-the-multi-armed-bandit-problem-b72de40db97c.

[7] A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and R. E. Schapire, “Taming the monster: A fast and simple algorithm for contextual bandits,” in *31st International Conference on Machine Learning, ICML 2014*, 2014.

[8] C. Riquelme, G. Tucker, and J. Snoek, “Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling,” Feb. 2018.

[9] M. Waskom, “Seaborn: Statistical Data Visualization.” [Online]. Available: https://seaborn.pydata.org/.

[10] M. D. Hunter, J., Dale, D., Dorettboom, M., & Team, “Matplotlib,” 2012. [Online]. Available: https://matplotlib.org/.

[11] C. L. Blake and C. J. Merz, “UCI Repository of machine learning databases,” *Univ. Calif.*, 1998.