Utilizing Multi-Armed Bandit Algorithms for Advertising: An In-Depth Case Study on an Online Retail Platform's Advertising Campaign

Litian Shao *

Department of Computer Science and Engineering, University of Minnesota Twin Cities, Minnesota, USA

* Corresponding Author Email: shaox196@umn.edu

Abstract. In recent years, the advertising sector has increasingly embraced Multi-armed Bandit Algorithms (MAB) for their versatile applications. This article delves into four stochastic MAB algorithms: the Explore-Then-Commit (ETC), the Upper Confidence Bound (UCB), the Thompson-Sampling (TS), and their respective variants, applied in an online shopping platform's advertising campaign to optimize click-through rates. Our findings indicate that each of these algorithms successfully identifies the most effective advertisement. Both the UCB and TS algorithms demonstrate logarithmic regret relative to the time horizon. The ETC algorithm exhibits a delayed onset of logarithmic regret but surprisingly holds up well against the UCB algorithms. Notably, the TS algorithm class outshines the UCB class due to its inherent randomness, offering a more robust performance. Furthermore, two variants, the UCB algorithm with the Mini-max Optimal Strategy in the Stochastic case (MOSS) and the paralleled TS algorithm, show promising results. They not only maintain logarithmic regret but also enhance efficiency, indicating further potential in the UCB and TS algorithm classes for advanced applications.

Keywords: Multi-Armed Bandits; Upper Confidence Bound; Thompson Sampling.

1. Introduction

The Multi-armed Bandit Problems are a set of problems involving the decision making process of a player and an environment, with the environment acting as if it was a multi-armed bandit. When each action, or the arm as from a bandit in real life, is picked by the player, a reward is awarded to the player. The fundamental goal of such set of problems is to maximize the total cumulative reward obtained from a given number of trials, also called the horizon, and a policy with a balance of the exploration, a phase trying out arms and trying to find out the optimal arm, and exploitation, a stage making the best use of the most rewarding arm found, is crucial for the maximization of the cumulative regret. The MAB problems pertain to a wide range of real life statistical problems related to choices. One pertaining example is the industry of advertising, in which the advertising agents come with several drafts to shown to the audience, and uses certain set of policies to pursuit a click rate or like rate and the most welcomed advertisements are made the best use of.

In this case study, a dataset consisting of advertisement clicking data from an online shopping platform campaign is selected, and the goal is to find the specific advertisement as well as to minimize the cumulative regret (i.e. to maximize the cumulative reward) using stochastic MAB algorithms. The ETC algorithm, the classical UCB algorithm, the UCB MOSS algorithm, the TS algorithm and the paralleled TS algorithms are implemented and their result and performance are to be compared against each other in terms of both behaviors between different classes, such as the UCB algorithms and the TS algorithms, and the subtle differences in performance between variants from the same class, such as the classical UCB algorithm and the UCB MOSS algorithm, as to find the best class of algorithms pertaining to this specific advertisement campaign, while also trying to introduce a further step of improvement to origin versions of the algorithms with some simple modifications.

2. Theoretical Background and Introduction of the Algorithms

The stochastic MAB algorithms differ from each other, but they have a common background. In this paper, all algorithms discussed in this paper are stochastic MAB algorithms with finite arms and no side information offered to the player from each pull of the arm. Let n be the horizon, or the total number of times the player can pull the arm, and Δ_i be the reward gap between arm i and the real optimal arm.

2.1. Explore-Then-Commit Algorithm

The explore-then-commit algorithm is a rather straightforward implementation of the algorithm. It is clearly divided clearly into two phases: the exploration phase and the exploitation phase. In the exploration phase, the algorithm tries out each arm an equal number of times, after which it picks up the optimal arm in the exploration phase, or the arm with the largest average reward, until the end. The average reward is calculated by the division of the total reward, obtained by picking that arm, by the times that arm has been picked up in the exploration phase.

Now, let k denote the number of arms, m be the number of exploration trials for each arm (i.e. exploration in $m * k$ rounds), and $\text{argmax}_i \hat{\mu}_i(mk)$ be the optimal arm in the first $m * k$ rounds with the largest average reward, i being the notation for each arm, then the action taken by policy of the ETC algorithm at round t can be expressed by:

$$A(t) = \begin{cases} (t \bmod k) + 1, & \text{if } t \leq mk \\ \text{argmax}_i \hat{\mu}_i(mk), & \text{if } t > mk \end{cases} \quad (1)$$

Where A(t) represents the specific arm chosen at round t.

The cumulative regret of the algorithm in can be expressed with the following inequalities (suppose arm 1 is optimal):

$$R(n) \leq m \sum_{i=2}^{k} \Delta_i + (n - mk) \sum_{i=2}^{k} \Delta_i \, e^{\frac{-m\Delta_i^2}{4}} \quad (2)$$

However, it can be noticed that the cumulative regret of the ETC algorithm is dependent upon the value of m. Though well-tuned ETC can achieve logarithmic cumulative regrets, the best choice of m generally requires knowledge of both the gap Δ_i and the horizon n in advance, the former one often being unavailable or unfeasible beforehand in real-world occasions [1]. Recent researches have indentified and proved that a clear division between the exploration phase and the exploitation phase must be sub-optimal [2].

2.2. Upper Confidence Bound Algorithm

The simplest and original upper confidence bound algorithm, also called UCB1 in some other literatures, takes a different strategy from the ETC algorithm, in that it does not have a rapid shift from the exploration stage to the exploitation stage. In each round before a choice is picked, the UCB algorithm evaluates the average reward value of each arm from previous rounds, as well as an extra confidence value, related to the times that arm has been picked in previous rounds, called the confidence level. Then the sum of both of the two components, serving as the UCB value, is updated for each arm, and the arm with the largest UCB value is picked in that round. Intuitively, the value of the confidence part of a specific arm declines as it is being picked up for a fixed horizon, so more privileges can be given to arms that are less frequently picked, thus acting as the exploring agent to some extent.

The policy of the UCB algorithm at round t can be expressed in the following expression:

$$\text{A}(t) = \text{argmax}_i(\hat{\mu}_i(\text{t} - 1) + \sqrt{\frac{4 \log n}{T_i(t-1)}}) \quad (3)$$

In which i still represents the arm number, n denotes the horizon, and $T_i(t - 1)$ is the times arm i has been picked up in previous (t – 1) rounds.

To avoid the denominator from the square root part being 0, in practice each arm is usually picked up once before the actual UCB algorithm steps in.

From the observation of the policy, the UCB algorithm takes a policy that only requires knowledge of the horizon n, without having to know the optimality gap Δ_i for each arm. It also tends to achieve a balance of exploration and exploitation by using an extra factor.

It has been proved by Auer and other co-authors in 2002 that the cumulative regrets of the original UCB algorithm is bounded by$O(\log n)$, and the cumulative regret can be expressed in the following inequality [3]:

$$R(n) \leq 3\sum_{i=1}^{k}\Delta_i + \sum_{i:\Delta_i>0}\frac{16\log n}{\Delta_i} \tag{4}$$

2.3. Thompson-sampling (TS) Algorithm

The Thompson-sampling algorithm differs from the ETC and the UCB algorithm, as it introduces randomness in the decision-making progress. The core of the algorithm is to formulate a distribution for each arm in each round, based on previous rounds, after which a random reward is generated for each arm based on their distribution, simulating pulls from the real environment. The arm with the largest simulated value is then picked, and the distribution is then updated in the next round.

It has been pointed out by Chapelle and Li in their literature review that though there lacks solid theoretical support for the TS algorithm, it has gained popularity because of its easiness to be implemented and recent promising results [4]. From trial to trial, the TS algorithm might perform significantly different, but overall it has better performance than the ETC and the UCB algorithm, while also appearing to suit a wide range of applications, thanks to the randomness.

The policy of Thompson-sampling at round t can be expressed by:

$$A(t) = argmax_i(\nu_t) \tag{5}$$

Where ν_t is the aforementioned set of random values generated for each arm using the updated distribution at each round?

To be more specific, if the reward distribution of the arm is believed to have Gaussian distribution, then the cumulative distribution function (CDF) of the distribution updated at the beginning of each round follows $\mathrm{N}(\hat{\mu}_i, \frac{1}{T_i(t)})$, a Gaussian distribution with its empirical average reward as the mean and the reciprocal of the time it has been chosen as the variance.

The TS algorithm has an asymptotically optimal regret, and with Gaussian distributions the regret R_n satisfies the following upper bound limit as horizon n approaches infinity:

$$\lim_{n\to\infty}\frac{R_n}{\log n} = \sum_{i:\Delta_i>0}\frac{2}{\Delta_i} \tag{6}$$

With symbols and letters having the same meaning mentioned earlier.

2.4. Some Variants to the UCB and the TS algorithm

2.4.1. UCB algorithm with Mini-max Optimal Strategy in the Stochastic case

In some situations, there is only one run for the algorithms. Algorithms that are good in average cases are not necessarily good in terms of the worst case performance. Many companies in the industry try to take a safer strategy in MAB problems, by which they tend to choose policies, not with the best average case performance, but with the minimal cumulative regret in the class. Audibert and Bubeck has proved that for stochastic bandits, a variant of the UCB algorithm, hence called the UCB algorithm with Mini-max Optimal Strategy in the Stochastic case (UCB MOSS), has the cumulative regret that has proven to be minimal in the worst case scenario and no other algorithms may surpass [5].

The UCB MOSS algorithm modifies the confidence level of the original UCB algorithm, and the policy taken at round t can be represented by:

$$A(t) = \text{argmax}_i(\hat{\mu}_i(t-1) + \sqrt{\frac{4}{T_i(t-1)} \log \max(1, \frac{n}{kT_i(t-1)})} \quad (7)$$

Intuitively, if an arm is chosen too many times, with fixed horizon n, the value of $\frac{n}{kT_i(t-1)}$ decreases and falls below 1, and that is exactly when the max function kicks in to prevent assigning negative values under the square root, while also eliminating the confidence level to 0.

2.4.2. Paralleled Thompson-sampling Algorithm

The algorithms mentioned before all take one arm in one round and update the mean regret, the UCB value or the distribution at the beginning of the next round, whereas in reality it is rather difficult to receive real-time reward, as most feedbacks in surveys are received from customers in batches.

Karbasi, et al. has proposed a variant of the TS algorithm that makes a batch of pulls for several rounds instead of a single pull and update for each round [6]. They have been able to achieve the same asymptotic regret performance as the one of sequential TS algorithm, by reducing the time of batch queries to O(log n), with n being the time horizon.

However, to keep simplicity and see the basic impact of parallelism, it is also of great interest to see what happens if a simple fixed number of arms is chosen in each batch without complicated mathematical techniques, which is to be examined in the following sections.

3. Empirical Study Design

To look into how each of the algorithm above performs in practice, a dataset (https:/ / www.kaggle. Com /datasets / pavansanagapati /ad – displayclick - data - on -taobaocom/ data). Consisting of data clicking rate of advertisement and information related to the users and advertisements has been chosen.

The overall goal of the empirical study is to compare the cumulative regret of all aforementioned algorithms in a given horizon, as well as to observe if the optimal arm appears to be the same across different MAB algorithms.

To simplify the problem, a few baselines need to be set in the first place. First, it might be hard to compare hundreds or even dozens of arms at the same time, and there might be large uncertainties since many advertisements are only clicked once or twice in millions of clicks, and therefore it is rather the advertisement group than the individual advertisements that is studied and only top five advertisement groups with the highest click rates have entered the selection of object to study, as shown in Table 1.

Table 1. Selected advertisement groups.

	adgroup_id	cate_id	campaign_id
307068	684724	4281	125699
587051	747157	1665	359520
588194	759279	1665	136030
601421	715171	6261	166940
757617	608584	6261	291817

Secondly, there needs to be an appropriate choice of horizon. Having a horizon that is too small might have a relatively large regret compared to the size of the horizon, and algorithms might not be able to display their overall behaviors in the early stage. On the other hand, choosing a horizon that is too large may result in excessive time running the MAB algorithms, and when it comes to comparisons, such as the comparison between the ETC algorithm and the UCB/TS algorithm, might have a gap that is too large between one and the other, making it the comparison meaningless. The horizon has been chosen to be 100,000, after some early-stage experiments, as the choice keeps a good balance between both showing the characteristics of the algorithms and the time cost of running these algorithms.

For the ETC algorithm, the exploration phase, or the numeric value of m*k, is chosen to be 10% of the horizon, which in this case is 10,000.

For the paralleled TS algorithm, to keep the problem simple, it is decided that in each batch 10 arms will be picked, and their distributions will be updated after that batch. Like the UCB algorithm, each arm will be picked once before the main part of the algorithm starts, as to provide distributions of each arm in the first place.

The reward in each round is just represented by 0 or 1, meaning whether the user has made a click on that advertisement, and the cumulative regret in each round is calculated by the difference between the product of the empirical average reward of the arm with the highest average reward in previous rounds and the rounds played, and the actual total reward received by the player. Each algorithm will be played 100 times with given horizon and their regret and selection of the optimal arm will be recorded and compared.

4. Analysis and Comparison of the Results

The following figure, Figure 1, containing cumulative regret information for each algorithm, is obtained a program implemented using based on the empirical design. More in-depth analysis will be mentioned in the coming to sub-section.

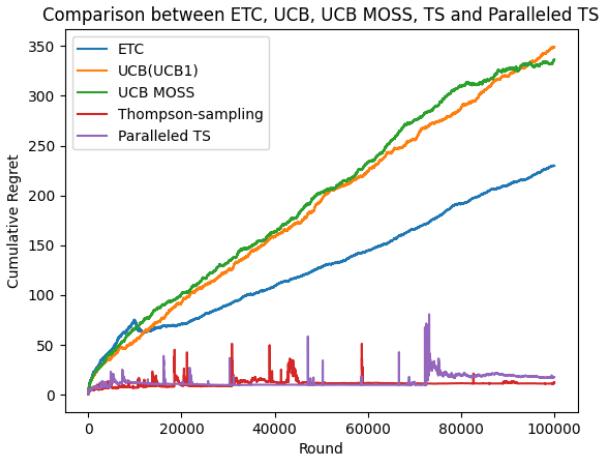

Fig 1. Comparison between ETC, UCB, UCB MOSS, TS and Paralleled TS (Photo/Picture credit: Original).

4.1. Comparison between the ETC, the UCB (UCB 1) and the TS algorithm

From all trials running the algorithm, all of the ETC, UCB and TS algorithm were able to identify the optimal arm (i.e. choosing the optimal arm the most times), the arm with group id 715171.

From the observation of figure 1, it can be noticed that the TS algorithm significantly outperformed both the ETC algorithm and the UCB algorithm, and its cumulative regret started to converge to display logarithmic behaviors at very early stage, though the curve was not smooth with some large fluctuations. However, even with these fluctuations, the TS algorithm still worked much better than the other two algorithms.

One interesting thing is that the ETC algorithm appeared to have a better cumulative regret than the UCB algorithm. Though the UCB algorithm started to display logarithmic cumulative regret very early, but it is not obvious. With horizon n = 100,000, the cumulative regret of the UCB algorithm grew much faster than the ETC algorithm, and more extended discussion will be covered in the next section.

4.2. Comparison between the Variants and Their Prototypes

Both the UCB MOSS algorithm and the paralleled TS algorithm have correctly indentified the optimal arm, just as previous three algorithms did.

The UCB MOSS algorithm did not appear to have less cumulative regret than the original UCB algorithm before the first 90,000 rounds of the trials, but it is reasonable since the core of the UCB MOSS algorithm is to cope with the worst case cumulative regret, but not the average case. However, though the UCB moss algorithm seems to have grown faster than the original UCB algorithm at first, as it approaches the end of the trials, the UCB MOSS algorithm converged at a higher speed and its cumulative regret was overtaken by the original UCB algorithm near round 950,000, so it is still meaning for to use the UCB MOSS variant if the number of trials is limited.

Speaking of the paralleled TS algorithm, it had a similar performance in terms of the cumulative regret to the sequential TS algorithm, but appeared to converge slower in the later phase of the play. The time consumptions for both the sequential and paralleled TS algorithms were measured. Over 100 trials with the same configuration of software and hardware, the sequential TS algorithm averaged a running time of 31.52 seconds, while the paralleled TS algorithm had an average running time of 29.83 seconds. Though the time difference was marginal in this particular case, with a larger scale of trials, the difference of time cost might be significant.

5. Discussion

It is an interesting observation that the ETC algorithm had outperformed both the UCB algorithm and the UCB MOSS algorithm in terms of cumulative regret in this particular setting of the study. Even though the UCB algorithm and the UCB MOSS algorithm had displayed logarithmic behavior in cumulative regret later and the ETC seemed to grow linearly, the algorithms of the UCB classes rose more rapidly in the beginning. This might have to do with the over-exploration with the UCB algorithm, which usually uses a series of inequalities in the process of building the upper bound, resulting in a looser confidence level, as pointed out by a recent study by Hao, et al [7]. This results in an underestimation in the optimal arm, and sometimes unnecessary emphasis on excessive exploration [8]. The ETC algorithm has also worked well probably because the reward in this study followed a Bernoulli distribution, but not a large discrete set of values, in which goal of finding the optimal arm was identical to finding the arm that had been clicked most frequently. With Bernoulli distribution, the ETC algorithm was efficient in finding the optimal arm in the exploration phase with the first 10% trials, and the rest of the pulls, the exploitation phase was solely committed for the optimal arm, which actually yield a better result for the cumulative regret in this case, while the UCB algorithms might have accumulated too much regret on extended explorations.

On the other hand, the TS algorithm works well with a wide range of problems, which includes problems with Bernoulli distributions [9]. As Scott pointed out in his 2010, progressed made in Bayesian computation has made it easier for the application of randomized probability model to be applied in almost all payoff distribution, which alleviates the workload of researchers in traditional model designs [10]. The TS algorithm indeed has demonstrated its potentiality in this particular problem, having the lowest cumulative regret and outperforming the other two classes of algorithms to a great extent. The fluctuations in the cumulative regret for the TS algorithm might have also been cause by sparse distributions of the click, since the decision were constructed on simulations of real arms, which might differ from the reality to a great extent in a few trials. Thus, the TS algorithm probably works better if it can be run several times to exclude extreme cases, though overall it has an outstanding performance.

The UCB MOSS algorithm had similar regret performance to the original UCB algorithm. Though it was indented to improve the worst case, the UCB MOSS algorithm also converged at faster speed than the UCB algorithm as more pulls of arms were being conducted. With large horizons and limited number of total experiments, the UCB MOSS algorithm seemed to be the better choice over the original UCB algorithm.

The paralleled TS algorithm did not save a huge amount of time in this particular scenario, since distributions still needed to be constructed for each batch, but it still had similar regret performance

compared to the sequential TS algorithm. For cases where the regret is more admissible and time efficiency is more emphasized, the paralleled TS algorithm may find its place.

6. Conclusion

This study implements and evaluates five algorithms – the Explore-Then-Commit algorithm, Upper Confidence Bound algorithm, Thompson-Sampling algorithm, UCB MOSS algorithm, and the parallel Thompson-Sampling algorithm – to address a Multi-Armed Bandit problem centered on the click rates of five advertisement groups. Each algorithm successfully identified the optimal arm. The analysis revealed that, due to the Bernoulli distribution characteristic of this problem, the ETC algorithm surpassed the UCB family in terms of cumulative regret, although the latter exhibited logarithmic trends towards the end. The Thompson-Sampling algorithm demonstrated superior performance over both the ETC and UCB algorithms, achieving logarithmic regret early on, albeit with some discrete fluctuations. Regarding the variations of the UCB and TS algorithms – the UCB MOSS and parallel TS algorithms – their performance closely mirrored that of their original counterparts. However, the UCB MOSS algorithm displayed a more rapid convergence, attributed to modifications in its confidence level calculation. Simultaneously, the parallel TS algorithm offered computational efficiency by selecting multiple arms in batches, streamlining the simulation process.

References

[1] Lattimore, T., & Szepesvári, C. (2020). Bandit algorithms. Cambridge University Press.

[2] Garivier, A., Lattimore, T., & Kaufmann, E. (2016). On explore-then-commit strategies. Advances in Neural Information Processing Systems, 29.

[3] Kuleshov, V., & Precup, D. (2014). Algorithms for multi-armed bandit problems. arXiv preprint arXiv:1402.6028.

[4] Chapelle, O., & Li, L. (2011). An empirical evaluation of thompson sampling. Advances in neural information processing systems, 24.

[5] Audibert, J. Y., & Bubeck, S. (2009, June). Minimax Policies for Adversarial and Stochastic Bandits. In COLT (Vol. 7, pp. 1-122).

[6] Karbasi, A., Mirrokni, V., & Shadravan, M. (2021). Parallelizing thompson sampling. Advances in Neural Information Processing Systems, 34, 10535-10548.

[7] Hao, B., Abbasi Yadkori, Y., Wen, Z., & Cheng, G. (2019). Bootstrapping upper confidence bound. Advances in neural information processing systems, 32.

[8] Auer, P., & Ortner, R. (2010). UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. Periodica Mathematica Hungarica, 61(1-2), 55-65.

[9] Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., & Wen, Z. (2018). A tutorial on thompson sampling. Foundations and Trends® in Machine Learning, 11(1), 1-96.

[10] Scott, S. L. (2010). A modern Bayesian look at the multi-armed bandit. Applied Stochastic Models in Business and Industry, 26(6), 639-658.