Ad Recommender System Analysis by the Multi-Armed Bandit Problem

Yohei Nishimura
ynishimura@wisc.edu

1. Introduction

Ad recommendation systems implement intriguing algorithms as practical tools and research subjects. In particular, the development and spread of the Internet have made it simple for companies and individuals to publish their content and, sometimes, sell their products. As a result, people are exposed to information overload, a volume of information they could not process in a lifetime. Furthermore, users encounter an immense of advertisements on the Internet. Because Internet advertisements cost less than other channels' ads, companies display thousands of promotions. In terms of a media, a supply side of the advertisement, it is essential to exhibit the ads to draw users' attention to maximize revenues.

For online promotion to be effective, companies have relied on targeting techniques on the users' information such as the historical data of browsing, shopping, or posting; however, users have become afraid of invading their privacy [Langheinrich et al., 1999]. Therefore, the research has been developed based on the users' feedback which does not require privacy information. Researchers have researched ad recommendation systems based on reinforcement learning algorithms in multi-armed bandit (MAB) problems [Scott, 2010] since there is a high affinity between the MAB setup and the online ad recommendation system. An advertiser seeks to maximize the number of clicks on a website by choosing the most effective ads, and each ad has a theoretical Click-Through Rate (CTR) that is unknown and assumed to not change over time. The advertiser's goal is to maximize the number of clicks over time, which is similar to a typical MAB problem.

This research project aims to discover and implement an agent's algorithm that goes beyond baselines: a random algorithm and an ϵ-greedy algorithm. We implement ad recommender agents using MAB problem setup with a simulated advertisement server environment. This goal can contribute to the explicit comparison of ad recommendation algorithms in the simplified simulated environment.

For the implementation, our code is open in the GitHub. Follow the instruction on README.md, and build the required environment. Note that some packages have a dependency on the Operating System: Ubuntu. We confirm the availability of Ubuntu 20.04 and 18.04. We recommend the same OS environment as ours.

2. Literature Review

2.1. Multi-Armed Bandit problem for ad recommender systems

[Slivkins, 2019] refers to ad selection as one representative of the multi-armed bandit (MAB) problems. In the problem, the agent observes whether the displayed ad is clicked; if it is clicked, the agent obtains the reward whose amount is one, while the agent does not gain the reward without clicking. The probability of the click on each ad does not change over time. We inherit the setting to our experiment in the gym-adserver [Falossi, 2021].

2.2. Algorithms for multi-arms bandit problem

2.2.1 Random/ϵ-greedy/Gradient Bandit Algorithm

We use a random algorithm and an ϵ-greedy algorithm as a baseline for more advanced algorithms.

In academic experiments such as [Theocharous et al., 2015], ϵ-greedy algorithm is widely used because of their balance between efficiency and effectiveness. Moreover, the ϵ-greedy algorithm is often employed in practice since it is efficient and delivers a robust result.

The [Sutton and Barto, 2018] provides the basic algorithms for MAB problems, including Gradient Bandit Algorithm using the softmax function.

2.2.2 Upper Confident Bound

[Auer et al., 2002] represents the application of the Upper Confident Bound (UCB) algorithm to the multi-armed bandit problem. This paper proposes UCB1, which achieves logarithmic regret without preliminary knowledge about the reward distributions. They denote the second term of the UCB score as $\sqrt{\frac{2\log t}{N_t(a)}}$.

In the textbook [Sutton and Barto, 2018], the authors use the coefficient c instead of $\sqrt{2}$ in the term. Therefore, we use the textbook's definition as the UCB score and select c by the sensitivity analysis.

Attributes	Values
Action Space	{1, ..., n}
Observation Space	$\{(imp_i, click_i)\}_{i=1}^n$
Actions	k ∈ {1, ..., n}
Rewards	1: Clicked, 0: Otherwise

Table 1. Gym-Adserver attributes

2.2.3 Thompson Sampling

The paper [Agrawal and Goyal, 2012] applies the concept of Thompson Sampling, which is proposed by [Thompson, 1933], to a practical algorithm for the MAB problem.

2.3. Metrics to evaluate algorithms in multi-bandit problem

[Li et al., 2010] adopts CTR as one of the primary metrics to evaluate the recommendation quality of personalized news articles in the MAB problem. We use the same metrics since our simulated environment is simplified and suitable for calculating CTR.

3. Methodology

3.1. Environment

[Falossi, 2021] provides a simulated environment in an OpenAI gym. This environment implements a typical multi-armed bandit scenario where an agent selects an advertisement to be displayed within k ads and counts it as one impression. If the displayed ad is clicked, the agent obtains the reward ($= 1$), while it gets zero as its reward without clicking. In this research, we use this environment for the evaluation of algorithms.

The table 1 shows the attributes of Gym-Adserver[1].

3.2. Algorithms

We compare the performances among five representative algorithms for the multi-armed bandit problem: random algorithm, ϵ-greedy algorithm, Gradient Bandit algorithm, Upper-Confidence-Bound Action Selection (UCB) algorithm, and Thompson Sampling algorithm.

3.2.1 Random and ϵ-greedy Algorithms

The random algorithm continues to select the arm at random through the episodes. The ϵ-greedy algorithm adopts the greedy algorithm with the probability $1-\epsilon$, while it randomly takes the arm with the probability ϵ. The algorithm has the hyperparameter ϵ, so it should be tested by the sensitivity analysis.

[1]https://github.com/falox/gym-adserver/blob/master/README.md

```
Algorithm: UCB algorithm
Take all arms once
for t = K+1, ..., T do
    Calculate the UCB score of each arm $\bar{\mu_i}(t)$
    Take $i$th arm based on $\arg\max_{i\in\{1,\ldots,K\}} \bar{\mu_i}(t)$
End for
```

Table 2. UCB algorithm.

3.2.2 Gradient Bandit Algorithms

We can use 'preference' for actions $H(a)$ for deciding the agent's action instead of the reward. We define the preference as the action probability by soft-max distribution such that

$$P(A_t = a) = \frac{\exp H_t(a)}{\sum_{b=1}^{k} \exp H_t(b)} = \pi_t(a)$$

where $H_t(a)$ denotes the preference for action a at time t, and $\pi_t(a)$ denotes that the probability of taking action a at time t. Note that we initialize $H_1(a) = 0$ for all a.

Based on the soft-max action preference, the stochastic gradient ascent algorithm updates the preference by the equations below:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R_t})(1 - \pi_t(A_t))$$
$$H_{t+1}(A_t) = H_t(A_t) - \alpha(R_t - \bar{R_t})\pi_t(A_t) \;\; (a \neq A_t)$$

where step size $\alpha > 0$ and R_t denotes the reward at time t and $\bar{R_t}$ does the average reward from 1 to time $t-1$.

3.2.3 UCB Algorithm

According to [Auer et al., 2002], we can define the new algorithm with the adjustment of the probability of the exploration compared to ϵ-greedy; we utilize the likelihood estimation of the mean in the arms' hidden distribution.

We define the estimated mean, named UCB score, $\bar{\mu_i}(t)$ as follows:

$$\bar{\mu_i}(t) = \hat{\mu_i}(t) + c\sqrt{\frac{\log t}{N_i(t)}} \qquad (1)$$

where $\hat{\mu_i}(t)$ is the sample mean and $N_i(t)$ is the number of the selection of the ith arm at the time t.

The UCB algorithm is shown in the table 2. In the table, K denotes the number of arms, and T denotes the number of episodes.

3.2.4 Thompson Sampling Algorithm

Thompson sampling was proposed in [Thompson, 1933] in 1933. This algorithm formulates probability-matching method using Bayesian statistics framework.

```
Algorithm: Thompson Sampling
For each arm i = 1, ..., N set $n_i$ = 0, $m_i$ = 0
for t = 1, ..., T do
    Sample $\tilde{\mu}_i$ from $Beta(\alpha + m_i, \beta + n_i - m_i)$
    Take $i$th arm based on $\arg\max_{i \in \{1,...,K\}} \bar{\mu_i}(t)$
    See Reward $R_i(t) \in \{0, 1\}$
    $n_i = n_i + 1, m_i = m_i + R_i(t)$
End for
```

Table 3. Thompson Sampling.

We formulate Thompson Sampling based on [Junya Honda, 2016]. In Thompson Sampling, we assume that the prior distribution generates the parameter μ_i in the hidden distribution. We assume that the prior distribution is the beta distribution because of its conjugation. Here, the beta distribution is formulated by the equation below:

$$f(x|\alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$$

where $B(\alpha, \beta)$ denotes a beta function.

Based on the prior distribution, we consider the posterior of μ_i. If the agent takes the ith arm n_i times and obtains one reward m_i times and zero $n_i - m_i$ times. Then, we obtain the posterior below by Bayesian theorem:

$$posterior(\mu|observation) = B(\alpha + m_i, \beta + n_i - m_i)$$

If we derive the expectation for each arm, we are supposed to marginalize by μ. Nevertheless, for the implementation, the paper [Agrawal and Goyal, 2012] presents a practical algorithm for the multi-arm bandit problem that applies the Thompson Sampling; we show the algorithm in table 3 (we modify the original algorithm of [Agrawal and Goyal, 2012] along with [Junya Honda, 2016]).

4. Empirical Study

ϵ-greedy, Gradient Bandit, and UCB algorithms have room to select their hyperparameters; ϵ, step size α, and c, respectively. Therefore, we first show the sensitivity analysis among the three algorithms to decide their hyperparameters. Then, we conclude which algorithm is the best performer in the simulated environment through the comparative analysis among five algorithms.

During this experiment, we set 10,000 as the number of terminal episodes.

4.1. Hypothesis

We hypothesize that the UCB and Thompson sampling are competitive algorithms among the five, and the two algorithms perform equally well; here, the better performance

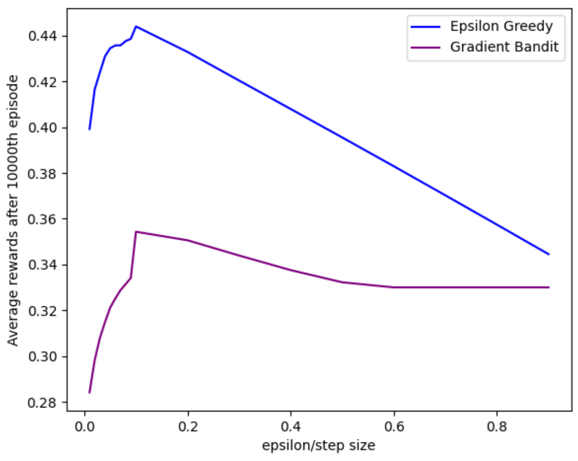

Figure 1. The figure shows the average rewards of ϵ-greedy and gradient bandit algorithm. The averages are calculated by running 20 times.

denotes that the average reward of the algorithm on the convergence is larger than other algorithms.

4.2. Sensitivity Analysis

4.2.1 ϵ-greedy Algorithm

ϵ-greedy algorithm has the hyperparameter of ϵ, which is the probability of taking a random choice for the action. We implement the algorithm with respect to ϵs from 0.01 to 0.09 in 0.01 increments and from 0.1 to 0.9 in 0.1 increments. Figure 1 shows the results; we use $\epsilon = 0.1$ to show the comparative analysis among the five algorithms.

4.2.2 Gradient Bandit Algorithms

The gradient bandit algorithm has the hyperparameter of step size α. We implement the algorithm in the gym-adserver environment with respect to α from 0.01 to 0.09 in 0.01 increments and from 0.1 to 0.9 in 0.1 increments. Figure 1 shows the results; along with the result, we take $\alpha = 0.06$ when we show the comparative analysis among the five algorithms.

4.2.3 UCB Algorithm

UCB algorithm has the hyperparameter of c to weigh the second term of the UCB score $\sqrt{\frac{\log t}{N_t(a)}}$. We implement the sensitivity analysis with respect to c from 0.01 to 0.09 in 0.01 increments, from 0.1 to 0.9 in 0.1 increments, and from 1 to 9 in 1 increment. Fig 2 shows the result, which leads us to take $c = 0.1$ as a hyperparameter of the UCB algorithm in the comparative analysis.

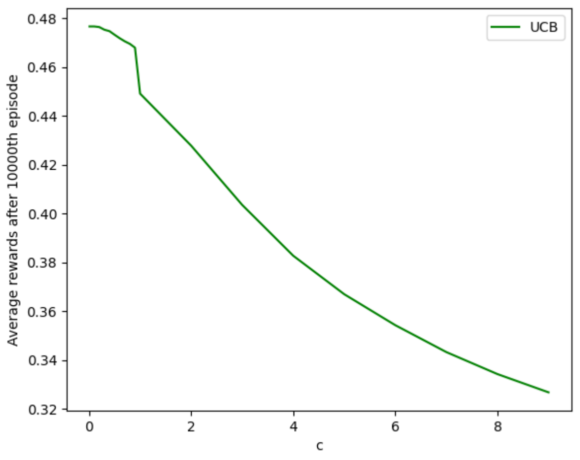

Figure 2. The figure shows the average rewards of UCB algorithm. The averages are calculated by running 20 times.

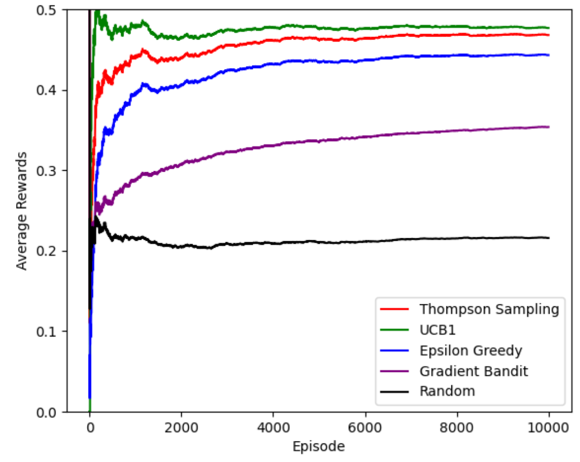

Figure 3. The figure shows the average rewards of each algorithm. This results is calculated by running 100 times.

4.3. Comparative Analysis

We plot the performance of each algorithm on fig 3; it shows that the UCB algorithm and Thompson sampling perform better than others in ad recommender simulation environment, as we hypothesize. If we increase the number of epochs, both algorithms' performance would be the same.

Note that the performance is averaged over 100-time experiments, and the standard deviation of the calculation is shown in fig 4.

5. Conclusion

We experiment with the MAB algorithms as agents in the simulated ad recommender system; we conclude that the UCB and Thompson Sampling perform well in the simulated environment. Interestingly, ϵ-greedy algorithm performs third best; it provides some evidence for the practical use of this algorithm.

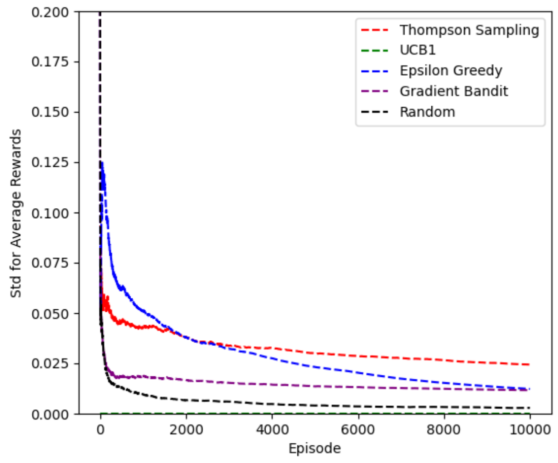

Figure 4. The figure shows the standard deviation of the average rewards of each algorithm. This results is calculated by running 100 times.

The simplified environment constrains this experiment. Therefore, we should analyze the ad recommendation in realistic conditions, such as non-stationary, or evaluate the algorithms by more long-term-basis metrics. Moreover, the actual experiment depends on the industry; the strategy of the ad recommendation by gaming companies is different from that of beauty companies, being evaluated in actual situations.

In the future study, we can introduce other algorithms for the multi-armed bandit problem, such as UCT [Kocsis and Szepesvári, 2006], to find more effective arms. Additionally, we can apply Non-Stationarity to MAB and formulate the ad recommendation problem in practice because it is common to the distributions changing over time. [Koulouriotis and Xanthopoulos, 2008] lists the algorithms for Non-stationary MAB and can be applied to ad recommender systems.

Regarding metrics, because reinforcement learning can deal with the long-term objective, we can consider the Life Time Value [Theocharous and Hallak, 2013] as metrics and introduce the Markov Decision Process into the ad recommender system to apply the recent reinforcement algorithms such as DQN [Zhao et al., 2019] or A2C [Mnih et al., 2016].

References

[Agrawal and Goyal, 2012] Agrawal, S. and Goyal, N. (2012). Analysis of thompson sampling for the multi-armed bandit problem. In Mannor, S., Srebro, N., and Williamson, R. C., editors, *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pages 39.1–39.26, Edinburgh, Scotland. PMLR. 2, 3

[Auer et al., 2002] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2–3):235–256. 1, 2

[Falossi, 2021] Falossi, A. (2021). gym-adserver. 1, 2

[Junya Honda, 2016] Junya Honda, A. N. (2016). *Theory and algorithms for bandit problems*. Kodansya. 3

[Kocsis and Szepesvári, 2006] Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, page 282–293, Berlin, Heidelberg. Springer-Verlag. 4

[Koulouriotis and Xanthopoulos, 2008] Koulouriotis, D. and Xanthopoulos, A. (2008). Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. *Applied Mathematics and Computation*, 196(2):913–922. 4

[Langheinrich et al., 1999] Langheinrich, M., Nakamura, A., Abe, N., Kamba, T., and Koseki, Y. (1999). Unintrusive customization techniques for web advertising. *Computer Networks*, 31(11):1259–1272. 1

[Li et al., 2010] Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. 2

[Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR. 4

[Scott, 2010] Scott, S. L. (2010). A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658. 1

[Slivkins, 2019] Slivkins, A. (2019). Introduction to multi-armed bandits. 1

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction (2nd ed.)*. MIT press. 1

[Theocharous and Hallak, 2013] Theocharous, G. and Hallak, A. (2013). Lifetime value marketing using reinforcement learning. *RLDM 2013*, page 19. 4

[Theocharous et al., 2015] Theocharous, G., Thomas, P. S., and Ghavamzadeh, M. (2015). Personalized ad recommendation systems for life-time value optimization with guarantees. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 1806–1812. AAAI Press. 1

[Thompson, 1933] Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294. 2

[Zhao et al., 2019] Zhao, X., Gu, C., Zhang, H., Liu, X., Yang, X., and Tang, J. (2019). Deep reinforcement learning for online advertising in recommender systems. *CoRR*, abs/1909.03602. 4