Trustworthy Recommender Systems via Bayesian Bandits

Vivek Saravanan, Eric Song, Hien Bui, Xiqiang Liu
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92037
`{visarava,ejsong,h1bui,xil073}@ucsd.edu`

Abstract

Recommender systems have emerged as a simple yet powerful framework for the suggestion of relevant items to users. However, a potential issue arises when recommender systems overly recommend or spam undesired products to users in which the model loses the trust of the user. We propose a constrained bandit-based recommender system. We show this model outperforms Upper Confidence Bound (UCB) and Thompson sampling in terms of expected regret and does not lose the trust of the users.

1 Introduction

In recent years, online platforms and marketplaces have developed the capacity to host and sell a large variety of products and services across a wide range of industries. With this vast amount of products that are projected to exponentially grow, users have relied on and continue to rely on the platforms' recommendation system to aid in the decision making process. Some popular examples include Netflix for movie and TV show recommendations, and Yelp for restaurants, among others. The success of the recommender system (and subsequently the platform's success) is dependent on the quality of recommendations given out to users. When a product has a good track record and its quality is known beforehand, it is relatively easy to make a recommendation. Ideally, a system would maximize "good" products' recommendations, and minimize "bad" products' recommendations. However, a common problem arises when the system has little to no information about a new product. This challenge is known as the "cold start" problem and many new/ niche products fail to overcome it. It is up to the recommender system to balance how much time is spent on exploration (to gather information on how the product fares) and exploitation (to maximize the reward). Initially exploring too much, or "spamming" the user can have the negative consequence of users losing trust in the recommender system as a result of receiving a lot of ill-suited recommendations. This problem can be modeled using a multi-armed bandit framework. In this setting, the available actions are products that can be chosen by the user. There are many bandit algorithms that have their own unique method of balancing exploration and exploitation. In this project, our goal is to develop a recommender system that outperforms popular asymptotic bandit algorithms while maintaining the recommender's credibility.

In many previous works, many multi-armed bandit algorithms (MABs) have been utilized for recommender systems. These algorithms have been rigorously studied, and we referenced the algorithms and the intuition behind them in the work by Lattimore and Szepesvári [2]. In this, they highlight the pros and cons of a collection of bandit algorithms including Upper Confidence Bound, Thompson Sampling, and Bayesian Optimal Policy, etc. For our project, we compared the performances of these algorithms, and decided on the standard Bayesian Optimal Policy for the implementation. Additionally, we looked to existing literature on the development of fair and trustworthy recommender systems. Our project draws heavily on research from Che and Hörner [1].

In their study, they discuss a similar problem of needing to discover an optimal recommendation policy that does not lose user trust. Some key insights include the idea that spamming can be a way to promote early exploration. However, excessive spamming can backfire and harm the credibility of the recommender. A solution they propose is to start small and spam over a longer duration. Further, they also discuss how alternate avenues of gathering information such as product research can aid in developing credibility and can act as a substitute for costly exploration. In the context of this project, we adopt a similar approach that spams to a fraction of agents over time with the intent of maintaining credibility.

This report is divided into a few main sections. First, we formulate the bandit environment in our recommendation setting and how we assess trustworthiness. Next, we show how our bandit policy of choice: Bayesian Optimal Policy outperforms popular bandit algorithms: UCB and Thompson sampling. Lastly, we show how our constrained bandit model utilizing Bayesian Optimal Policy does not lose the trust of the users in our recommender system setting.

2 Methods

2.1 Setting

The setting for our bandit problem is as follows. Suppose a product with unknown quality and reward ω is released at time $t = 0$, and an infinite number of agents or users arrives at each time $t > 0$. We are interested in understanding whether to recommend this unknown product or a known product with some constant reward c. We define the quality of the unknown product as good if $\phi = 1$ or bad if $\phi = 0$. The model is given some prior p^0, defined to be the probability that the unknown product is good ($\phi = 1$) where $p^0 \in [0, 1]$. At time $t = 0$, the model receives a signal $\sigma \in \{g, n\}$ (g refers to good news, n refers to no news) about the quality of the product with probabilities:

$$P(\sigma = g|\phi) = \begin{cases} \rho & \text{if } \phi = 1 \\ 0 & \text{if } \phi = 0 \end{cases} \tag{1}$$

for some constant $\rho \in [0, 1]$ and where $g(n)$ refers to the model receiving good (no) news/signal. Thus the designer can receive good or no news if the product is good, but will only receive no news if the product is bad. For time $t > 0$, the signal received by the model is determined as:

$$P(\sigma = g|\phi) = \begin{cases} \alpha_t & \text{if } \phi = 1 \\ 0 & \text{if } \phi = 0 \end{cases} \tag{2}$$

For some $\alpha_t \in [0, 1]$. If at some time t^* good news is received ($\sigma = g$), then the model will permanently recommend the product to all agents for $t \geq t^*$ ($\alpha_t = 1$). Otherwise, the model will recommend the product to a fraction α_t of the incoming agents at time t. Furthermore, p_t is defined to be the "no news" posterior where we receive no news. This is defined to be:

$$p_0 = \begin{cases} 1 & \text{if } \sigma_0 = g \\ \frac{(1-\rho)\cdot p^0}{(1-\rho)\cdot p^0 + (1-p^0)} & \text{if } \sigma_0 = n \end{cases} \tag{3}$$

and

$$p_{t+1} = \begin{cases} 1 & \text{if } \sigma_t = g \\ \frac{(1-\rho\alpha_t)p_t}{(1-\rho\alpha_t)p_t + (1-p_t)} & \text{if } \sigma_t = n \end{cases} \tag{4}$$

for all $t > 0$.

We can represent this problem as a one-armed bandit setting. In this case, the stochastic arm represents our unknown product with reward Bernoulli distributed and $p = \rho\alpha_t$ and the deterministic arm represents our known product with constant reward c.

2.2 Trustworthiness

In our model, agents are unaware of the information given only to the model. Thus, agents will form a rational belief about the model's prediction and act accordingly. Suppose g_t is the probability the model has received good news by time t. Then by the martingale property, we define g_t as:

$$g_t = \frac{p^0 - p_t}{1 - p_t} \tag{5}$$

Thus, our agents will become increasingly pessimistic about the quality of the unknown product as time progresses with the lack of good news. This can be observed there the inverse relationship of g_t and p_t. In addition, each agent will have some incentives for following the recommendation based off of their beliefs of the model's recommendation $q_t(p_t)$ defined as:

$$q_t(p_t) = \frac{g_t + (1 - g_t)\alpha_t p_t}{g_t + (1 - g_t)\alpha_t} \tag{6}$$

Therefore, an agent will only consume the unknown product if and only if their incentive to consume the unknown product is greater than the cost, or the reward of the known product:

$$q_t(p_t) \geq c \tag{7}$$

We describe the case in which there exists some time t such that $q_t(p_t) < c$ as the model losing the trust of the user. This leads us to define a trustworthy recommender system in this setting as a model which satisfies equation (7) for all t.

2.3 Models

Based on the bandit setting, we have two very similar models. Both models determine the best policy through the Bayesian Optimal Policy. In this case, ω is defined as:

$$\omega^t_{\alpha_t}(p_t) = r\alpha^* + \omega^{t+1}(p_{t+1}) \tag{8}$$

where r is defined as:

$$r = (1 - c)p_t + (-c)(1 - p_t) + \rho p_t \omega^{t+1}(1) - \rho p_t \omega^{t+1}(p_{t+1}) \tag{9}$$

We define $\omega^t_{\alpha_t}(1)$ as:

$$\omega^t_{\alpha_t}(1) = (1 - c)(n - t - 1) \tag{10}$$

and $\omega^{n+1}_{\alpha_{n+1}}(p_{n+1}) = 0$ for all p_{n+1}. We then use backwards induction to calculate all values of $\omega^t_{\alpha_t}(p_t)$. Due to the peculiar nature of our bandit setting, both models will use a variation of the retirement policy where we continuously pull the unknown/stochastic arm until we lose confidence and then permanently pull the deterministic arm. We add in an additional retirement case where if we receive good news at some time t^* ($\sigma = g$) then $\alpha_t = 1$ and we permanently pull the unknown/stochastic arm.

The key difference between the two models is the determination of α^*. In our first best policy, we recommend the unknown product to all agents regardless of the signal we receive. Thus, α^*_{FB} is defined as:

$$\alpha^*_{FB} = \begin{cases} 1 & \text{if } r > 0 \\ 0 & \text{if } r < 0 \end{cases} \tag{11}$$

In our second best policy, we only recommend the unknown product to some fraction α_t of all users at time t if we received no news by then. Thus α^*_{SB} is defined as:

$$\alpha^*_{SB} = \begin{cases} \alpha_t & \text{if } r > 0 \\ 0 & \text{if } r < 0 \end{cases} \tag{12}$$

where α_t is equal to:

$$\alpha_t = \min\left\{1, \frac{(1-c)(p^0 - p_t)}{(1-p^0)(c-p_t)}\right\} \tag{13}$$

3 Experiments

3.1 Policy Comparison

In this experiment, we compare the expected regret of the Bayesian Optimal Policy to popular asymptotically optimal bandit algorithms UCB and Thompson Sampling in the one-armed bandit setting. All algorithm descriptions and equations are shown in the supplementary section. In this case, the stochastic arm has a reward bernoulli distributed with some $p \in [0, 1]$ and the deterministic arm has a constant reward of 0.5. We set the priors for the density estimation variant of Bayesian Optimal Policy as well as Thompson Sampling to be beta distributed with $\alpha = 1$ and $\beta = 1$. We plotted the expected regret for all values of p from $[0, 1]$ for all algorithms below in figure 1. We observe both variants of the Bayesian Optimal Policy outperform Thompson Sampling and UCB.

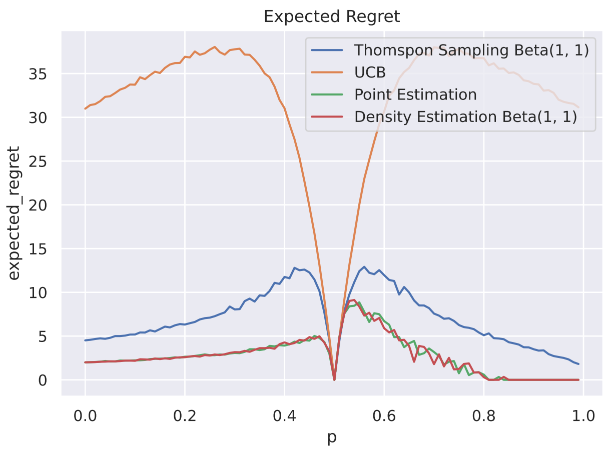

Figure 1: Regret Comparison of UCB, Thompson, and BOP

3.2 Trustworthy Recommender Systems

In the second experiment, we compare how α changes within our recommender system problem setting. Here, we initialized $\{p^0, \rho, c, n\} = \{\frac{1}{2}, \frac{1}{4}, \frac{2}{3}, 1000\}$. In figure 2, we plot how α_t versus t across both polices. We observe that our first best policy spams to all agents and then immediately stops recommending the product. This is because our first best policy violates equation (7) where the agents lose the trust of the policy due to over recommendation of the unknown product. However, we observe a different case in our second best policy where the model only recommends the product to a fraction of the agents. Gradually, this fraction increases until the model loses confidence in the unknown product and stop recommending it. In this case, equation 7 is satisfied for all t and as a result, the policy does not lose the trust of the consumers.

4 Conclusion

In this report, we have shown how the Bayesian Optimal Policy outperforms popular asymptotically optimal algorithms such as Upper Bound Confidence and Thompson Sampling. We have also shown

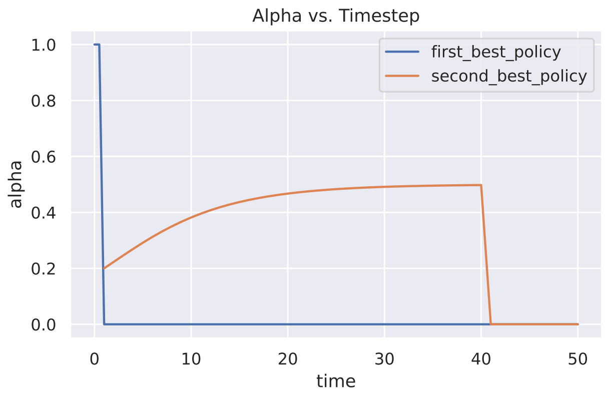

Figure 2: Recommendation Rate vs. Time

how we applied Bayesian Optimal Policy in recommender systems, and how a constrained variant of our model does not lose the trust of users in "cold-start" cases where the quality of the product is unknown at the time of recommendation. Potential future work involves generalizing our problem to k armed bandits where we compare our product to $k-1$ other known products.

References

[1] Yeon-Koo Che and Johannes Hörner. Recommender systems as mechanisms for social learning. *The Quarterly Journal of Economics*, 133(2):871–925, 2018.

[2] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

A Supplementary Information

A.1 Bandit Problems

A **bandit** problem is a sequential game between a **learner** and a **environment** [2]. The game is played over n rounds, where n is called the **horizon**. At round $t \in [1, n]$, the learner selects an action A_t from a set of possible actions $\mathcal{A}$. The actions here is often mentioned as the **arms**. Subsequently, the environment would provide a reward $X_t \in \mathbb{R}$. If the bandit environment is stochastic, the reward provided with each action is not deterministic. Instead, it would follow a certain probability distribution.

Under such a framework, the goal of the learner is to maximize the cumulative reward $\sum_{t=1}^{n} X_t$ it receives by choosing the action with the highest reward.

To achieve such goal, the learner would have a **policy** π, which is a algorithm mapping from its observed **history** $H_{t-1} = (A_1, X_1, A_2, X_2, \ldots, A_{t-1}, X_{t-1})$ to its actions.

In order to evaluate the performance of a learner, one could look at the measurement of the learner's policy regret R_n:

$$R_n = n \max_{a \in \mathcal{A}} \mu_a - \mathbb{E}\left[\sum_{t=1}^{n} X_t\right] \tag{14}$$

In plain words, regret is the difference between the cumulative rewards obtained through optimal action and the expected cumulative reward obtained through the learner's policy. A smaller regret would correspond to a better-performing policy.

A.2 Exploration vs. Exploitation

One core trade-off in the study of multi-armed bandit is exploration vs. exploitation. At any given round, the learner essentially has two options to take: either take some potentially sub-optimal actions or choose the action with the maximum empirical mean reward.

Both options have their pros and cons. If the learner explores too much, then the learner would be wasting much time even if the learner already has a good sense of the distributions of rewards for each arm. If the learner exploits too early, it may unaware of the possibility of better actions.

Thus, the study of multi-armed bandit is finding an algorithm with a good balance of exploration and exploitation. The remaining subsections in this section will highlight some of the well-known algorithms for this purpose.

In this project, we aim to examine the performance of several bandit algorithms by analyzing the regrets of these policies under specific settings.

A.3 Environment

To test our algorithm, we used a one-armed Bayesian bandit environment. This setting involved 2 arms, one with a known deterministic reward of $^1/_2$, and the other following an unknown Bernoulli distribution. Because we were testing the Bayesian Optimal Policy (see next section), the environment also included a Beta prior of $Q = \mathrm{Beta}(\alpha, \beta)$. Our horizon was set to $n = 1000$. We chose 20 different parameters for the value of the unknown arm, ranging from 0 to 1. Each parameter was run under these settings and simulated 1000 times. For each simulation, the regret was recorded and after all simulations, the average regret and variance were calculated.

A.4 Upper Confidence Bound Algorithm

The UCB algorithm operates on the core principle of optimism in the face of uncertainty. For bandits, this means that exploration of unknown arms is favored. In each round, an upper confidence bound is calculated for each arm using the observed data. There are various methods to calculate the upper confidence bound, and this difference is what creates multiple variants of the UCB algorithm. For example, the upper confidence bound for a simple version: UCB(δ) takes in an error probability δ.

$$\mathrm{UCB}_i(t-1,\delta) = \begin{cases} \infty & \text{if } T_i(t-1) = 0 \\ \hat{\mu}_i(t-1) + \sqrt{\frac{2\log(1/\delta)}{T_i(t-1)}} & \text{otherwise} \end{cases} \tag{15}$$

In the beginning, the upper confidence bound for each round is set to infinity. Then, each arm is pulled once and the reward is observed to calculate the empirical mean. From here, the algorithm continues and chooses the arm with the higher confidence bound and discovers the optimal arm in this process.

The general UCB algorithm is defined as:

Algorithm 1 Upper Confidence Bound

```
Input $k$
  for $t \in 1, 2, \dots, n$ do
      $A_t \leftarrow \mathrm{argmax}_i\, \mathrm{UCB}_i(t-1)$
      $\mathrm{UCB(t)} \leftarrow X_t$
  end for
```

A.5 Thompson Sampling

Thompson sampling is a family of algorithms that uses a prior distribution before estimating the expectation for each arm. After defining the priors for each arm, a posterior distribution of the reward is calculated with the prior and the observed reward. A choice of conjugate priors aids in the computation of the posterior in each round.

A.5.1 Bernoulli Arms

In a Bernoulli setting, the family of Beta distributions is conjugate, and the computation of the posterior given the observed reward is also a beta distribution. Given a prior distribution $\mathrm{Beta}(\alpha, \beta)$, the posterior distribution would be $\mathrm{Beta}(\alpha + x, \beta + 1 - x)$, where $x \in \{0, 1\}$ is the reward the agent receives for the round.

A.6 Gaussian Arms

For the Gaussian case, the posterior update is calculated using the signal and prior variance in conjunction with the observed reward.

$$Q(\cdot|x) = \mathcal{N}\left(\frac{\mu_P/\sigma_P^2 + x/\sigma^2}{1/\sigma_P^2 + 1/\sigma^2}, \left(\frac{1}{\sigma^2} + \frac{1}{\sigma_P^2}\right)^{-1}\right) \tag{16}$$

The general algorithm for Thompson Sampling is defined as:

Algorithm 2 Thompson Sampling

```
Input $k$
  for $t \in 1, 2, \dots, n$ do
      for $i \in [1, k]$ do
          Sample $\hat{\theta}_i \sim Q(\cdot|A_1, X_1, A_2, X_2, \dots, A_{t-1}, X_{t-1})$
      end for
      Pick action $A_t \leftarrow \mathrm{argmax}_{i \in [k]}\, \hat{\theta}_i$
  end for
```

The performance of the UCB and Thompson Sampling algorithms will serve as a baseline for the focus of our project: **Bayesian Optimal Policy**.

A.7 Bayesian Optimal Policy

In this project, we attempted to modify and potentially improve the Bayesian Optimal Policy (BOP) for certain circumstances. The BOP takes into account a prior beta distribution as well as the reward and number of pulls for each arm to determine the optimal action at each round. In our environment, because the behavior of one of the arms is known, the policy comes down to exploring the unknown arm until the known arm yields a better reward. Since the posterior distribution is calculated based on the results of the action of the pulled arm, we know that the posterior of the action at the last round will be 0, since there are no more rounds after. Thus, we can derive our optimal expected cumulative reward backward, and then iterate forwards based on the results of our empirical reward and number of pulls. The posterior at the start of a given round is $\mathrm{Beta}(\alpha + S_t, \beta + t - 1 - S_t)$, with $S_t = \sum_{s=1}^{t-1} Z_s$. If we let $p_t(s) = (\alpha + s)/(\alpha + \beta + t - 1)$, then:

$$\mathbb{E}[Z_t|\mathfrak{F}_t] = \frac{\alpha + S_t}{\alpha + \beta + t - 1} = p_t(S_t) \tag{17}$$

$$\mathbb{P}(S_{t+1} = S_t + 1|S_t) = p_t(S_t) \tag{18}$$

$$\mathbb{P}(S_{t+1} = S_t|S_t) = 1 - p_t(S_t) \tag{19}$$

Thus, the optimal expected cumulative reward is as follows:

$$w_t(s) = \max{(n - t + 1)\mu_2, p_t(s) + p_t(s)w_{t+1}(s + 1) + (1 - p_t(s))w_{t+1}(s)} \tag{20}$$

This policy, while more computationally expensive, yields a higher reward and lower regret than the aforementioned baselines.

Algorithm 3 Bayesian Optimal Policy for Bernoulli Arms

```
Input $\alpha, \beta, n, p$
  $s$: Total reward of the Bernoulli arm
  $q$: Number of times the Bernoulli arm is pulled
  $n$: Horizon Length
  $p$: Reward of the deterministic arm
  for $s \in [1, n], q \in [1, n]$ do
      $w_{n+1}(s, q) \leftarrow 0$
  end for
  for $t = \{n, n-1, \ldots, 0\}$ do
      $w_t^1 \leftarrow \frac{\alpha+s}{\alpha+p+q} + \frac{\alpha+s}{\alpha+\beta+q} w_{t+1}(s+1, q+1) + (1 - \frac{\alpha+s}{\alpha+\beta+q}) w_{t+1}(s, q+1)$
      $w_t^2 \leftarrow p + w_{t+1}(s, q)$
      $w_t(s, q) \leftarrow \max(w_t^1, w_t^2)$
  end for
  for t ={0, 1, ..., n} do
      Take action $a_t = \mathrm{argmax}_i\, w_t^i$
      Update $s$ and $q$
  end for
```

A.8 Density Estimation

Originally, the BOP is a density estimation problem that involves updating the preset prior probability based on the results of the reward of the pulled arm at a given round. This BOP algorithm is as follows:

$$\max_{\pi=\{\pi^j\}} R(\pi) = \mathbb{E} \sum_{j=i}^{n} \int_{\mathcal{E}} \mu_{A^j}(\nu^j) \rho^j(\nu) d(\nu) \tag{21}$$

A.9 Point Estimation

The modification we are making is to use point estimation, which uses the empirical reward from all previous rounds to calculate the probabilities for the next round, instead of the prior and posterior distributions. Specifically, the policy uses s_k and q_k, which denote the cumulative reward and the number of times pulled for a given arm k, respectively. The utilization of s_k and q_k would essentially replace the sampling of the prior and updating of the posterior. Thus, for our version, a prior is not required. This policy maximizes the reward as defined by:

$$\max_{\pi=\{\pi^j\}} R(\pi) = \mathbb{E}\sum_{j=i}^{n} \mu_{A^j}(\hat{\nu}^j) \tag{22}$$

In this equation, $\mu_{A^j}(\hat{\nu}^j)$, in terms of s_k and q_k, can be rewritten as:

$$\mu_{A^j}(\hat{\nu}^j) = \frac{s_{A^j}}{q_{A^j}} \tag{23}$$

At a given round, the optimal expected cumulative reward from round i would be the max of the calculations of $w^i(s,q)$ for both arms. The equations for both arms is as follows:

$$w_2^i(s,q) = \frac{1}{2} + w^{i+1}(s,q), w_1^i(s,q) = \frac{s}{q} + \frac{s}{a}w^{i+1}(s+1,q+1) + (1-\frac{s}{q})w^{i+1}(s,q+1)$$