Collaborative Thompson Sampling

Zhenyu Zhu, Liusheng Huang(✉), and Hongli Xu

University of Science and Technology of China, Hefei, Anhui, China
zzy7758@mail.ustc.edu.cn, {lshuang,honglixu}@ustc.edu.cn

Abstract. Thompson sampling is one of the most effective strategies to balance exploration-exploitation trade-off. It has been applied in a variety of domains and achieved remarkable success. Thompson sampling makes decisions in a noisy but stationary environment by accumulating uncertain information over time to improve prediction accuracy. In highly dynamic domains, however, the environment undergoes frequent and unpredictable changes. Making decisions in such an environment should rely on current information. Therefore, standard Thompson sampling may perform poorly in these domains. Here we present a collaborative Thompson sampling algorithm to apply the exploration-exploitation strategy to highly dynamic settings. The algorithm takes collaborative effects into account by dynamically clustering users into groups, and the feedback of all users in the same group will help to estimate the expected reward in the current context to find the optimal choice. Incorporating collaborative effects into Thompson sampling allows to capture real-time changes of the environment and adjust decision making strategy accordingly. We compare our algorithm with standard Thompson sampling algorithms on two real-world datasets. Our algorithm shows accelerated convergence and improved prediction performance in collaborative environments. We also provide a regret analysis of our algorithm on a non-contextual model.

Keywords: Thompson sampling · Bandits · Collaborative effect · Dynamic clustering

1 Introduction

Thompson sampling has received considerable attention in recent years due to its capability of balancing exploration-exploitation trade-off. The exploration-exploitation trade-off often occurs in online learning problems, where the algorithm tries to balance between exploiting current information to maximize immediate reward and exploring new information that may improve future performance. Theoretical analysis [3,17] and empirical results [6,7,14] revealed that Thompson sampling represents one of the most promising approaches to tackle this problem. It has been successfully applied in a variety of applications, including revenue management [11], Markov decision process [13,23] and recommendation systems [18,20].

H. Gao et al. (Eds.): CollaborateCom 2018, LNICST 268, pp. 17–32, 2019.
https://doi.org/10.1007/978-3-030-12981-1_2

While Thompson sampling is good at balancing exploration-exploitation tradeoff, it may perform poorly in highly dynamic and large-scale applications. In a stationary system, Thompson sampling is an effective and efficient method [24]. Each round, the algorithm applies probability matching heuristic and selects item corresponding to its probability of being optimal. This randomized manner [6] allows Thompson Sampling to continuously explore all possible candidates to accumulate information and explicit the optimal choice in the current context at the same time. However, in nonstationary settings, the environment undergoes frequent and unpredictable changes [12]. In this case, accumulated information of previous rounds becomes irrelevant to future performance. These issues also arise in large-scale settings where it is almost impossible to explore all the items to find the optimal choice. Thus, inferencing the expected reward of items based on a few observations will result in high variance and low accuracy. These drawbacks hinder the practical deployment of Thompson sampling in highly dynamic and large-scale domains [24].

One promising method to address these problems is to leverage on the collaborative effects. The collaborative effects often represent the potential connections between different users and different items in a real-world application. Although integrating collaborative effects into bandit settings has been reported in a series of works [21,22,29], little attention has been paid to incorporating collaboration into Thompson sampling. Previous works focus on applying the collaborative effects to LinUCB [9], a well-studied deterministic contextual bandit algorithm. They use feedback of all users in the same cluster to build a group vector as an estimator of the user vector. The selected item is then determined with upper confidence bound by the group vector. However, we cannot directly apply this strategy to Thompson sampling. Thompson sampling uses a Bayesian heuristic [6], it maintains and updates a posterior distribution of user vector. The distribution of the group vector is not an ideal estimator of the user vector, because it is much more concentrated than the user vector due to the abundance of feedback. Thus, directly applying this strategy will limit the exploration of Thompson sampling and lead to suboptimal performance.

This paper presents a collaborative Thompson sampling algorithm. The algorithm works under the assumption that the feedback of the user can be used to build an unbiased estimator of reward with high variance at the beginning, while the feedback of other users in the same cluster can be a biased estimator of reward with low variance. Combining these estimators to approximate reward may lead to improved prediction accuracy and accelerated convergence. This assumption has been illustrated in previous research [19] on standard statistical problems where the combined estimator outperforms the initial estimators in most cases. In our algorithm, the reward of items is estimated by a compound of both user preference and collaborative information. Specifically, the population of users is dynamically partitioned into clusters based on their similarity. The expected reward is a linear combination of two estimators, the collaborative estimator and the personal estimator. The collaborative estimator is constructed and shared by all users in the same cluster, while the personal estimator is constructed independently by each user. The combination of these two estimators generates a more sophisticated estimator of rewards, which lead to accelerated convergence and improved prediction performance.

We demonstrate that sharing collaborative information among similar users can not only improve the prediction performance of the algorithm but also help to find the optimal item efficiently. We compare our algorithm to standard Thompson sampling algorithm on two real-world datasets. The experimental results show that our algorithm outperforms standard Thompson sampling methods in terms of both prediction accuracy and convergence rate in collaborative environments. We also provide a regret analysis in a standard non-contextual setting.

2 Related Work

Thompson sampling was first proposed in 1933 [27] and has drawn much attention since 2010 [7,26]. It is proved by a series of theoretical analysis [3,17] and empirical evaluations [6,7,14] to be among the most effective bandit policies to tackle complex online problems. Adaptations of Thompson sampling have now been successfully applied in a wide variety of domains, including marketing [25], online advertising [1,15] and recommendation system [18].

Beyond the general settings of Thompson sampling, our work is also closely related to collaborative approaches with dynamic clustering. Clustering at user side or item side to provide collaborative information in bandit settings has been studied in a series of previous works. The work [22] incorporates online clustering into contextual bandit settings to divide users into groups and customizes the bandits to each group. The paper [5] also relies on clustering at the user side, with a constraint condition that once a user consumes an item, the item cannot be recommended to the same user again. The goal of their work is to maximize the number of consumed items recommended to users over time. In [29], the authors developed a collaborative contextual bandit algorithm, in which the adjacency graph is leveraged to share contexts and payoffs among neighboring users. In [21], the authors proposed COFIBA as an extension of [29], in which clustering is performed at both user and item side. They also used sparse graph representation to avoid expensive computation. The most similar work to ours is [8]. In this paper, the authors proposed an online collaborative algorithm to learn the underlying parameters of users and items. They defined the prior distribution of both the user vectors and the item vectors. At each round, the algorithm samples from the posterior distribution of both the user vector and the item vector. The expected reward of item is computed by logistic function with the product of user vector and item vector. Item with the highest expected reward will be selected. After observing the feedback, the posterior distributions of both user vector and item vector are updated with online gradient descent.

Our work shares the same assumption that collaborative effects can be leveraged to share information across users. In our work, we use a compound of both collaborative effects and personal preference to help the users to estimate the expected reward with a few observations.

3 Learning Model

We assume the user preference is parameterized by an unknown vector $\mu_i \in \mathbb{R}^d$ for each user $i \in \mathcal{U}$, where $i = 1, 2, 3, ..., n$ is the set of n users. We follow the settings of standard bandit problem. The parameter u_i of user i determines the payoff of item $j \in \mathcal{C}$ with contextual vector x_j. Formally, the payoff value $r_{i,j}$ is given by a function f and a random variable ϵ:

$$r_{i,j} = f(\mu_i, x_j) + \epsilon_{i,j} \tag{1}$$

Where i is the index of the user, j is the index of the item and ϵ is a zero-mean and bounded random noise. We assume that for any fixed user vector μ_i and contextual vector x_j, $f(\mu_i, x_j)$ is the expected payoff observed by user i for item j.

We model the learning process as a sequential decision problem with T rounds: at each round $t = 1, 2, 3..., T$, for each user i, the algorithm is provided with the set of candidate contents C_t. The contextual information of each content is represent by a vector x_j for $j = 1, 2, ..., |C_t|$. Our task is to select an item $\hat{j}_i^t$ from the candidate pool C_t and recommend to the user i. After that, we will observe the user feedback $r_{\hat{j}_i}^t$. When the user feedback is the behavior whether the selected item is clicked, the payoff is binary. The user feedback is used to evaluate the prediction performance of our algorithm in empirical evaluation, which is represented as the click-through rate $1/(nT)\sum_{t=1}^{T}\sum_{i=1}^{n} r_{\hat{j}_i}^t$ of recommended items over T rounds. The goal of our algorithm is to maximize the click-through rate of the selected items.

For theoretical analysis, the most popular performance measure is the total expected regret. Regret is defined as the gap of reward between the selected item and the optimal item. Total expected regret is formally defined as:

$$\mathbb{E}(\mathcal{R}(T)) = \sum_{i=1}^{n}\sum_{t=1}^{T} E[(r_i^* - r_{\hat{j}_i}^t)] \tag{2}$$

where r_i^* is the reward of the optimal item for user i, and $r_{\hat{j}_i}^t$ is the reward of the chosen item $\hat{j}_i^t$ at round t.

4 Collaborative Thompson Sampling Algorithm

In the learning model, since we do not know the true value of user vector μ^*, we maintain a posterior distribution over the user vector $P(\mu|\cdot)$. The posterior distribution is approximated by a multivariate Gaussian distribution [24] with the diagonal covariance matrix in linear and logistic Thompson sampling algorithms. If we want to exploit the immediate reward, we would choose for user i the item j that maximizes $\mathbb{E}[r_{i,j}|i,j] = \int \mathbb{E}[r_{i,j}|i,j,\mu_{i,j}]P(\mu_{i,j}|\cdot)d\mu_{i,j}$. However, to balance the exploration-exploitation trade-off, Thompson Sampling (TS) uses the probability matching heuristic and chooses for user i an item j according to its probability of being optimal. i.e., with probability [8]:

$$\int \mathbb{I}\big[\mathbb{E}[r_{i,j}|i,j,\mu_{i,j}] = \max_{j' \in C} \mathbb{E}[r_{i,j'}|i,j',\mu_{i,j'}]\big] P(\mu_{i,j}|\cdot) d\mu_{i,j} \tag{3}$$

The $\mathbb{I}[\cdot]$ is the indicator function. Its value is 1 if the condition holds. The $\mathbb{E}$ donates expectation of rewards in the current context, and the integral denotes expectation over the posterior distribution of user vector. In Thompson sampling, this integral can be estimated by drawing a sample $\tilde{\mu}_{i,j}$ from its posterior distribution $P(\mu_{i,j}|\cdot)$, and calculating its expected reward with $\mathbb{E}[r_{i,j}|\mu_{i,j}] = f(\tilde{\mu}_{i,j}, x_j)$ as described in the previous section. The algorithm will choose the item with the largest expected reward. The posterior distribution is then updated according to the feedback of the user (clicked or not). Intuitively, In stationary and infinite time-horizon settings, the algorithm accumulates more information about user preference over time and the posterior distribution of user vector will concentrate around its true μ^*. The optimal choice will be selected with high probability.

The major problem of Thompson sampling is that it has to estimate the expected reward $\mathbb{E}(r_{i,j}|\mu_{i,j})$ on the fly. As described in previous section, the feedback $r_{i,j}$ is determined by user vector $\mu_{i,j}$, contextual vector x_j and a random variable ϵ. Although we assume that the expectation of random variable ϵ is zero, the variance might be large when we try to estimate $\mathbb{E}(r_{i,j}|\mu_{i,j})$ with a few observations. In order to reduce the variance, standard Thompson sampling algorithm has to repeatedly select the same item for users to observe their feedback. Since it runs independent instance for each user, the algorithm converges slowly and is inaccurate in the first few rounds. Accurately estimating the expected reward $\mathbb{E}(r_{i,j}|\mu_{i,j})$ in the current context will help to boost the prediction performance and speed up convergence. To address this problem, we leverage on the collaborative effects. We assume that users may have similar preference and similar behavior, this indicates that the feedback of one user can be used to estimate the feedback of similar users. In other words, the feedback of users in the same cluster can be used to estimate the expected reward $\mathbb{E}(r_{i,j}|\mu_{i,j})$ in the current context. The expected reward is the compound of two different estimators, the personal estimator $\mathbb{E}[r_{i,j}|\mu_{i,j}]$ and a collaborative estimator $\mathbb{E}[r_{i,j}|G_s]$. where G_s is the cluster that user i belongs to.

The personal estimator $\mathbb{E}[r_{i,j}|\mu_{i,j}]$ estimates the reward by user vector $\mu_{i,j}$ which is trained by the user's previous feedback; it is an unbiased estimator of $r^*_{i,j}$. However, it suffers from sparsity of training data. The collaborative estimator $\mathbb{E}[r_{i,j}|G_i]$, on the other hand, is based on the feedback of similar users in the same cluster and is a direct approximation of reward with low variance. But it is a biased estimator of reward $r^*_{i,j}$ since user behavior may not be compatible with cluster behavior.

In our algorithm, both the personal estimator and the collaborative estimator are leveraged to estimate the reward of items in the current context. As a result, collaboration among users can not only capture additional information embedded in user similarity but also help to deal with the data sparsity issues. Both of these effects lead to an improvement of prediction performance and accelerated convergence.

4.1 Algorithm Description

In this section, we use logistic regression as an example to describe our algorithm. Our algorithm stores and updates a posterior distribution $P(\mu_{i,j}|D_t)$ of user vector $\mu_{i,j}$ for user $i \in \mathcal{U}$ and item $j \in \mathcal{C}$ at round t. We assume the prior of μ is a multivariate Gaussian distribution. Due to conjugacy property of normal distribution [24], its posterior distribution $P(\mu_{i,j}|D_t)$ remains normal after any number of rounds. More specifically, if the prior of $\mu_{i,j}$ at round t is given by $\mathcal{N}(\hat{\mu}_{i,j}(t), S_{i,j}(t))$, the posterior distribution at round $t+1$ is $\mathcal{N}(\hat{\mu}_{i,j}(t+1), S_{i,j}(t+1))$.

Our assumption is that the expected reward $\pi_{i,j}$, given contextual vector x_j and user vector $\mu_{i,j}$ is determined by the logistic function $\pi_{i,j} = 1/(1+\exp(-\mu_{i,j}^{\mathrm{T}} x_j))$. And user feedback $r_{i,j}$ is drawn from a Bernoulli distribution parameterized by $\pi_{i,j}$: $r_{i,j} \sim \text{Bernoulli}(\pi_{i,j})$.

As the closed form analysis of logistic regression in Bayesian inference is intractable, we then utilize the Laplace approximation [28] to approximate the posterior distribution of $\mu_{i,j}$ with a Gaussian distribution $\mathcal{N}(u_{i,j}|\hat{u}_{i,j}, S_{i,j})$, where $\hat{u}_{i,j}$ is the mode and $S_{i,j}$ is the Hessian matrix [4,8]. We update $\hat{\mu}_{i,\hat{j}}^{t+1}$ with online gradient descent $\hat{\mu}_{i,\hat{j}}^{t+1} \leftarrow \hat{\mu}_{i,\hat{j}}^{t} - \eta_t \nabla_{u_{i,\hat{j}}}^{t}$, where

$$\nabla_{u_{i,\hat{j}}}^{t} = {S_{u_{i,\hat{j}}}^{t-1}}^{-1}(\hat{\mu}_{i,\hat{j}}^{t} - \hat{\mu}_{i,\hat{j}}^{t-1}) + (\pi_{i,\hat{j}}^{t} - r_{i,\hat{j}}^{t})x_{\hat{j}}^{t} \tag{4}$$

$x_{\hat{j}}^{t}$ is the contextual vector of the selected item and $r_{i,\hat{j}}^{t}$ is the observed reward and η is the learning rate. We also need to update ${S_{u_{i,\hat{j}}}^{t}}^{-1}$:

$${S_{u_{i,\hat{j}}}^{t+1}}^{-1} = {S_{u_{i,\hat{j}}}^{t}}^{-1} + \pi_{i,\hat{j}}^{t}(1-\pi_{i,\hat{j}}^{t})x_{\hat{j}}^{t}{x_{\hat{j}}^{t}}^{\mathrm{T}} \tag{5}$$

Our algorithm also maintains and updates clusters over users $\mathcal{U}$ to approximate the expected rewards of items. This algorithm takes the simple viewpoint that clustering over users is determined by the similarity of their feedback. Each user i maintains a vector l_i and records his click-through rate of all items. If user i and user j are in the same cluster, then $||l_i - l_j||_2^2 < \lambda$, while if i and j are in different clusters then $||l_i - l_j||_2^2 > \lambda$. The clusters are initialized as follows: We first randomly select k items and recommend those items to all users $i \in \mathcal{U}$, the feedback of users on these items can be used to partition users into 2^k clusters. The users with the same feedback are grouped into the same cluster. Each cluster maintains its click-through rate of all items as a vector l_g for $g \in |\mathcal{G}_t|$. We denote by $|\mathcal{G}_t|$ the number of distinct partitions of users $\mathcal{U}$, and work under the assumption that $|\mathcal{G}|$ is significantly smaller than $|\mathcal{U}|$. The clusters $\mathcal{G}_t$ record the current partition of users $\mathcal{U}$ by the similarity of feedback. The clusters are updated each round to estimate the true partition of users.

Algorithm 1. Collaborative Logistic Thompson Sampling

```
Input:
    Set of users $\mathcal{U} = 1, 2, 3, ..., n$;
    Set of contents $\mathcal{C} = 1, 2, 3, ..., m$;
    Set of contextual vectors $x_1, x_2, x_3 ..., x_m$
    Hyper-parameter $\beta$
Init:
    $\mu^1_{i,j} = 0 \in \mathbb{R}^d$ and $S^1_{i,j} = I \in \mathcal{R}^{d \times d}$
    Prior distribution of user vector $\mu_{i,j} \sim \mathcal{N}(\mu^1_{i,j}, {S^1_{i,j}}^{-1})$
Initialize user clusters:
Random select $k$ items and recommend them to all users
for each $i \in \mathcal{U}$ do
  Suppose user feedback for all $k$ items are $r_{i,1}, r_{i,2}, ..., r_{i,k}$
  The cluster id of user $i$ is $\sum_0^{j=k-1} 2^j * r_{i,j+1}$
end for
Recommend contents and update parameters:
for $t = 1, 2, 3..., T$ do
  for each $i \in \mathcal{U}$ do
    Sampling $\tilde{\mu}^t_{i,j}$ from distribution $\mathcal{N}(\hat{\mu}^t_{i,j}, {S^t_{i,j}}^{-1})$
    Determine the cluster this user belongs to: $\hat{g}^t_i = \arg\max_{g \in \mathcal{G}} ||l^t_i - l^t_g||^2_2$
    for each $m \in \mathcal{C}$ do
      Compute personal expected reward $\pi^t_{i,j} = 1/(1 + \exp(\tilde{\mu}^{t\mathrm{T}}_{i,j} x_j))$
      Compute the compound of cluster expected reward and personal
      expected reward: $\mathbb{E}[r_{i,j,t}|i, j, \mu_{i,j}, \mathcal{G}] = \beta l^t_{\hat{g},j} + (1-\beta)\pi^t_{i,j}$
    end for
    Set $\hat{j}^t = \arg\max_{j \in C} \mathbb{E}[r_{i,j,t}|i, j, \mu_{i,j}, \mathcal{G}]$
    Recommend content $\hat{j}^t$ to user $i$ and observe payoff $r^t_{i,\hat{j}}$
    Update $\hat{\mu}^t_{i,\hat{j}}$, ${S^t_{i,\hat{j}}}^{-1}$ and the posterior distribution as follows:
      $\nabla^t_{u_{i,\hat{j}}} = {S^{t-1}_{i,\hat{j}}}^{-1}(\hat{\mu}^t_{i,\hat{j}} - \hat{\mu}^{t-1}_{i,\hat{j}}) + (\pi^t_{i\hat{j}} - r^t_{i,\hat{j}}) x^t_{\hat{j}}$
      $\hat{\mu}^{t+1}_{i,\hat{j}} \leftarrow \hat{\mu}^t_{i,\hat{j}} - \eta_t \nabla^t_{u_{i,\hat{j}}}$
      ${S^{t+1}_{i,\hat{j}}}^{-1} = {S^t_{i,\hat{j}}}^{-1} + \pi^t_{i\hat{j}}(1 - \pi^t_{i\hat{j}}) x^t_{\hat{j}} {x^t_{\hat{j}}}^{\mathrm{T}}$
      $P(\mu_{i,\hat{j}}|\mathbb{D}(t)) = \mathcal{N}(\hat{\mu}^{t+1}_{i,\hat{j}}, {S^{t+1}_{i,\hat{j}}}^{-1})$
    Update user click-through rate $l_i$
    Update cluster click-through rate $l_{b_{i,t}}$
  end for
end for
```

At each round t, collaborative Thompson sampling algorithm first draws a sample $\tilde{\mu}_{i,t}$ from posterior distribution $P(\mu_{i,j}|\mathcal{D}_t)$ and construct the personal estimator of reward $\pi_{i,j} = 1/(1 + \exp(\tilde{\mu}^{\mathrm{T}}_{i,j} x_j))$. Suppose that at current round t, user i belongs to cluster $\hat{g}$, the click-through rate of items $l^t_{\hat{g},j}$ represents the expected reward in the cluster. The expected reward used in Thompson sampling is a compound of personal estimator and collaborative estimator:

$$\mathbb{E}[r^t_{i,j}|i, j, \mu_{i,j}, \mathcal{G}] = \beta l^t_{\hat{g},j} + (1-\beta)\pi^t_{i,j} \tag{6}$$

The algorithm will select item $\hat{j}_i = \arg\max_j \mathbb{E}[r^t_{i,j}|i,j,\mu_{i,j},\mathcal{G}]$ to recommend to user i and observe the feedback $r^t_{i,\hat{j}}$. After observing the user feedback $r_{i,\hat{j}}$, the algorithm will update the posterior distribution. The user will be assigned to the closest clusters.

5 Regret Analysis

We consider the special case of two items in a non-contextual setting. There are n users and each user has 2 items to select from. The true reward of item j for user i is $u_{i,j}$. The collaborative Thompson Sampling algorithm assumes that the prior distribution of $u_{i,j}$ is Beta$(1,1)$. At round t, if the algorithm have observed $S_{i,j}$ successes (reward = 1) and $F_{i,j}$ failures (reward = 0), the posterior distribution of $\mu_{i,j}$ will be Beta$(S_{i,j}+1, F_{i,j}+1)$. At round t, the collaborative Thompson sampling algorithm samples $\tilde{\mu}_{i,j}$ from the posterior distributions of each item, and computes $\hat{\mu}_j = \frac{1}{n}\sum_{i=1}^n \tilde{\mu}_{i,j}$, the selected item for user i is determined by $\hat{j}^t = \arg\max_j \beta\hat{\mu}_j + (1-\beta)\tilde{\mu}_{i,j}$.

Theorem 1. *If $0 \le |\bar{\mu}_j - \mu_{i,j}| \le \gamma$ for any $i \in \mathcal{U}$ and $j \in \mathcal{C}$, where $\bar{\mu}_j = \frac{1}{n}\sum_{i=1}^n \mu_{i,j}$, collaborative Thompson Sampling algorithm has expected regret:*

$$\mathbb{E}(\mathcal{R}(T)) = \mathcal{O}(\frac{\ln T}{\Delta} + \frac{\Delta(1-\beta)^2 \ln T}{(\Delta + \Delta\beta - 4\gamma\beta)^2} + \frac{\Delta(1-\beta)^4}{(\Delta + \Delta\beta - 4\gamma\beta)^4} + 18\Delta) \tag{7}$$

Proof. Our proof follows the outline of [2] and only considers one user. The optimal item is the first item. Firstly, we assume that the second item has been selected $L = 24(\ln T)/\Delta^2$ times. The total expected regret before the algorithm selects the second item again is ΔL, where $\Delta = u_1 - u_2$. After the second item has been selected for L times, the following events happen with high probability: the posterior distribution of μ_2 is tightly concentrated around its true value and $\frac{1}{n}\sum_{i=1}^n \tilde{\mu}_{i,j}$ is very close to $\bar{\mu}_j$. Therefore, when we draw a sample $\tilde{\mu}_2$ from its posterior distribution, the value of $\tilde{\mu}_2$ is roughly μ_2. And the value of $\hat{\mu}_j = \frac{1}{n}\sum_{i=1}^n \tilde{\mu}_{i,j}$ is roughly $\bar{\mu}_j$. We then try to estimate how many times the second item will be selected between two consecutive selections of the first item. The selected item is determined by $\hat{j}_t = \arg\max_j \beta\hat{\mu}_j + (1-\beta)\tilde{\mu}_j$. With the above approximations of $\hat{\mu}_j$ and $\tilde{\mu}_j$, we instead estimate how many times we need to sample from the posterior distribution of μ_1 until we get a sample $\tilde{\mu}_1$ that satisfies $\tilde{\mu}_1 > \mu_2 - \beta(\Delta - 2\gamma)/(1-\beta)$.

Using Lemma 6 of [2] for $y = \mu_2 + \Delta/2 - \beta(\Delta - 2\gamma)/(1-\beta)$ and $\Delta' = \frac{\Delta + \Delta\beta - 4\gamma\beta}{1-\beta}$, we get a regret bound of:

$$\mathbb{E}(\mathcal{R}(T)) < \frac{24\ln T}{\Delta} + \frac{4\Delta(1-\beta)^2 \ln T}{(\Delta + \Delta\beta - 4\gamma\beta)^2} + \frac{\Delta(1-\beta)^4}{(\Delta + \Delta\beta - 4\gamma\beta)^4} + 18\Delta \tag{8}$$

This regret bound gives us some insight into collaborative Thompson sampling. Firstly, the performance of collaborative Thompson sampling depends on the strength of collaborative effects. If γ is large, it implies that the users have very different preferences and the collaborative effects are very weak. Collaboration among users will result in poor performance in both convergence rate and prediction accuracy. But if the users are carefully clustered and users in the same cluster have similar preference, the algorithm will outperform standard Thompson sampling in terms of prediction and convergence. And also, the optimal value of parameter β is closely related to the strength of collaborative effects.

6 Experiment

To evaluate the performance of our collaborative Thompson Sampling algorithm, we compared our algorithm to standard Thompson sampling algorithm on two real-world large-scale datasets, the Yahoo! front page today module dataset and the Avazu click-through dataset.

6.1 Dataset Description

Yahoo! The first dataset we used to evaluate our algorithm was the Yahoo! front page today module dataset. This dataset was provided by Yahoo! in the "ICML 2012 Exploration & Exploitation Challenge". The dataset contains more than 45 million user visits to the Today Module. The variables recorded in one line are as follows: a timestamp, six user features, one recommended article (id) and its payoff (0 or 1), candidate articles and their features. The recommended article was selected uniformly at random from the candidate article pool. It makes the dataset ideal for unbiased, offline evaluation of exploration-exploitation approaches. Both the users and the articles are associated with a six-dimension feature vector (including a constant feature), constructed using a conjoint analysis with a bilinear model. More detailed description of how this dataset was generated and how the features were extracted can be found in [10].

We preprocessed the Yahoo! dataset to evaluate our collaborative Thompson Sampling algorithm. As the dataset only provides us with user features, we are unable to identify the users by their id. Instead, we performed K-means clustering over visits based on the similarity of user features and summarized their click-through rate of all articles. Each visit was assigned to a cluster, and these clusters were treated as users in our experiment. We then computed the click-through-rate of all articles in a cluster. If an article were never recommended to any user in this cluster, the click-through rate of this article is set to mean of CTR of all visits. In our experiment, if article j is recommended to a cluster i, the payoff $r_{i,j}$ is drawn from a Bernoulli distribution: $r_{i,j} \sim \text{Bernoulli}(\pi_{i,j})$, where $\pi_{i,j}$ is the click-through rate.

Avazu. The Avazu click-through rate dataset was presented by Avazu, which is an international corporation specializing in cross-device advertising. The dataset was provided for an online challenge to build and test models, which aim at predicting the click-through rate of mobile ads. What it contains was divided into two parts, a training set, and a test set. There are over 40 million records in the training set and approximately 4 million records in the test set. We used the training set to build and evaluate our algorithm because we did not know the payoff of the test data. Each line in the training set represents the event of an ad impression on a website or a mobile application. The variables contained in each line can be categorized as follows: id, timestamp, payoff(clicked or not), banner position, three site variables, three app variables, five device variable, and 9 anonymized categorical variables(C1, C14–C21). The payoff was one if it was clicked, and 0 otherwise.

As ID represents a single ad impression, it cannot be used to identify an ad. Instead, we identified the ads with their site variables (site id, site domain, site category) if it is displayed on a website or app variables (app id, app domain, app category) otherwise. We identified the users with the combination of device variables (device id, device IP, device model and C19) rather than device id because most device ids are the same (null). In the Avazu dataset, we had the records of more than 1 million users and 20 thousand ads. However, not all records can be used in the evaluation of our algorithm, as we wanted to compare our algorithm to standard Thompson sampling algorithms in bandit settings in which the goal is to minimize the regret in the long term. If the users were provided with only a few ads, the performance would not be distinguishable among algorithms. Most users only appeared once in the dataset and were not suitable to evaluate our algorithm. In preprocessing, we eliminated the users and ads with less than 1000 records. The number of users and ads end up being 544 and 81 respectively. We used the remaining more than 1 million records in the experiments. However, we found that the dataset is quite sparse, most users were only recommended with few ads. To address this problem, we added one additional impression each ad to each user, and the reward was set to 0.13, which is the mean click-through rate of the remaining impressions. We emphasize that preprocessing did not provide any additional collaborative effects to the dataset, so it can still be used to evaluate the performance of collaborative Thompson sampling.

6.2 Algorithms

To the best of our knowledge, there were no previous works that focus on incorporating collaborative effects into Thompson sampling. To evaluate the prediction performance of our algorithm, we applied collaboration to a set of standard Thompson sampling algorithms.

Beta Thompson Sampling is a non-contextual bandit algorithm. The reward of content j follows a Bernoulli distribution with mean θ. The mean reward of each content is estimated using a Beta distribution because it is the conjugate distribution of Bernoulli distribution.

Linear Thompson Sampling follows the settings of standard contextual linear bandit. The vector parameter u_i of user i determined the payoff of a content $j \in \mathcal{C}$

with contextual vector x_j. Formally, the payoff value $r_{i,j}$ is computed by a linear function and a random variable ϵ: $r_{i,j} = \mu_i^{\mathrm{T}} x_j + \epsilon_{i,j}$

Logistic Thompson Sampling works under the assumption that given a contextual vector x_j for content j, the probability that user i click content j is given by $1/(1 + \exp - u_{i,j}^{\mathrm{T}} x_j)$, where $u_{i,j}$ is the user vector to be learned. We used Laplace approximation to approximate the posterior distribution of user vector $u_{i,j}$.

All algorithms require parameter tuning to achieve the optimal performance. There are shared parameters such as the exploration-exploitation tradeoff parameter, the learning rate of rewards. There are also private parameters of each algorithm such as the number of clusters in our algorithm. We tuned standard Thompson sampling algorithm to find the optimal combination of the shared parameter with grid search. The shared parameters were used in both collaborative Thompson sampling and standard Thompson sampling. We then tuned collaborative Thompson sampling to determine their private parameters based on the shared parameters.

6.3 Results

All experiments are aimed at comparing the prediction performance of collaborative Thompson sampling (CTS) and standard Thompson sampling (STS). The shared parameters were determined by standard Thompson sampling via grid search and were used by both CTS and STS. This gives rise to reliable estimation of the actual effect of collaboration under the same experimental conditions.

6.4 Yahoo! Dataset

In this experiment, click-through rate (CTR) was used to evaluate the performance of all Thompson sampling algorithms. CTR was computed in every 200 rounds for each algorithm. We also used the CTR to show the trend of learning procedure.

Performance Comparison. The results on the Yahoo! webscope dataset are presented in Fig. 1. We plotted click-through rate (CTR) of recent 200 rounds (left) and CTR of all rounds (right). The results of standard Thompson sampling are shown with the dotted line and the results of corresponding collaborative Thompson sampling are shown with the full line in the same color.

As we can see, all collaborative Thompson sampling algorithms outperformed corresponding standard Thompson sampling algorithms that do not take collaborative effects into account. All CTSs achieved higher prediction accuracy and accelerated convergence compared to STSs. The results revealed that the ability of CTS to share collaborative information among users allows it to provide an accurate estimation of expected rewards and balance exploration-exploitation trade-off at both the personal level and the system level.

In Fig. 1, We can conclude that collaborative Thompson sampling outperformed standard Thompson sampling in terms of CTR. In these simulations,

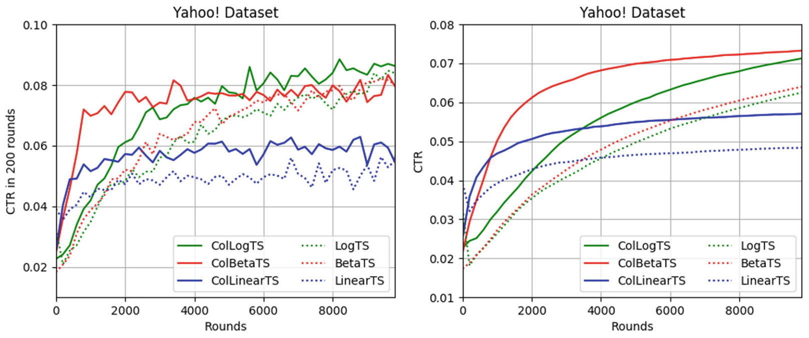

Fig. 1. Plots on the Yahoo! dataset reporting click-through rate (CTR) of all impressions over time, i.e., the fraction of the recommended articles get clicked. Left: CTR in 200 rounds; Right: CTR in all previous rounds (Color figure online)

the performance of Beta TS and Logistic TS was asymptotical to collaborative Thompson sampling. Their CTR of 200 rounds (left) was about the same value with collaborative approaches after 10000 rounds of training and the gap of CTR (right) between CTS and STS was closing over time. These results can be concluded that the collaborative effects accelerate the convergence of Thompson sampling without harming its prediction performance. This finding was consistent with our prediction in regret analysis. For Linear Thompson sampling, the collaborative Thompson sampling consistently achieved better click-through rate. It revealed that incorporating collaborative in a linear setting may help to improve the prediction performance as well.

The CTR of the first relatively small fraction of rounds showed the performance of these algorithms in a cold-start regime. It is shown in Fig. 1 that collaborative Thompson sampling algorithms converge much faster than standard Thompson sampling algorithms. They were able to locate the optimal choice in fewer rounds. Thus, CTSs are more suitable to address the cold start problem. In highly dynamic environments, this property of CTS allows it to accurately estimate the expected rewards of new items and adapt to dynamic changes of environment. Therefore, it is reasonable to expect that our algorithm will perform well in highly dynamic systems with strong collaborative effects such as news recommendation, where new contents regularly become available for the recommendation and the value of news changes over time.

Note that Linear Thompson sampling (blue) performed poorly in terms of click-through rate compared to other algorithms. A possible explanation is that linear TS maintains one user vector for each user to estimate the expected reward of all contents to share knowledge among contents. While the other algorithms maintain independent parameter for each content. The knowledge learned from different contents may conflict with each other. Although Linear Thompson sampling did not perform well in this experiment, they have some advantage over other algorithms. For example, Linear TS converged much faster than Beta TS. Another advantage of Linear Thompson sampling is it is easy to implement because it does not require to approximate the complex distribution of parameters used in logistic regression or sample from the beta distribution.

6.5 Avazu Dataset

In this experiment, regret was used to evaluate the performance of all Thompson sampling algorithms. Regret is the most popular measure in the multi-armed bandit problem. It is defined as the gap of reward between the optimal item and the selected item. Regret was computed in every 100 rounds to show the current performance of algorithms. We also used the total regret to show the performance of our algorithm in the long run.

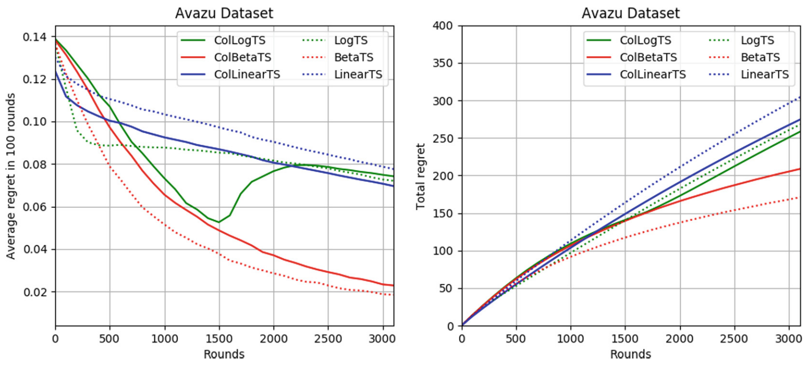

Fig. 2. Plots on the Avazu dataset reporting regret over time. Left: average regret of last 100 rounds; Right: total regret

Performance Comparison. The results on the Avazu dataset are summarized in Fig. 2. We plotted regret of recent 100 rounds (left) and cumulated regret of all rounds (right). The results of standard Thompson sampling were shown with the dotted line, and the results of corresponding collaborative Thompson sampling were shown with the full line in the same color.

As we can see, applying collaborative effects to three standard Thompson sampling algorithms in Avazu dataset revealed different effects. For Linear TS model (blue lines), collaborative Thompson sampling outperformed standard algorithm on both prediction accuracy and convergence rate. Collaborative TS has lower regret in each round and lower total regrets. For Logistic TS model (green lines), the collaborative Thompson sampling also provided slight performance improvement in terms of total regret. These results indicated that applying collaborative effects to contextual settings is profitable in this dataset. However, for the Beta TS model (red lines), collaborative TS did not outperform standard TS. The standard TS converged faster with lower total regret. It indicated that utilizing collaborative in a non-contextual model in this dataset may harm its performance. These effects might result from the sparsity of data. In

Avazu dataset, the ads were identified by their website and application domains rather than their id or contents. Thus many different ads were treated as one. Therefore, users may be recommended thousands of ads impressions without knowing what kind of ads they are. The sparsity of data weakened the collaborative effects among users and ads. Thus collaborative Thompson sampling cannot provide improved performance and accelerated convergence.

These different performances of collaborative TS resulted from the strength of the collaborative effects of these datasets. Although Avazu and Yahoo! dataset are both generated by real online web applications, they are different in the strength of collaborative effects. Firstly, Avazu dataset records the click on ad impressions via mobile apps or websites. The ads are identified by website domain or mobile domain rather than their id. It means that different ads may be identified as the same one in this dataset. In Yahoo! dataset, the articles are identified by their id. Therefore, the collaborative effects of the same article are much stronger. Secondly, for Yahoo! dataset, it is reasonable to predict that certain articles such as breaking news will draw much attention from all users and it is natural to expect that the true value of news can be estimated by feedback of other users. But users may not have the similar preference for ads in Avazu dataset. Thirdly, the Avazu dataset is quite sparse. The users are often recommended with the same ads for thousands of times and no other ads are ever recommended. The sparsity of data weakens the collaboratives effect embedded in the dataset. As our algorithm exploits the collaborative effects of the data, it is reasonable that our algorithm exhibited more significant effects in Yahoo! dataset than Avazu dataset.

To summarize, collaborative TS significantly outperforms standard TS in strong collaborative environments. It is especially effective in the cold-start period. Utilizing the collaborative effects in Thompson sampling results in accelerated convergence and improved prediction accuracy. It makes the algorithm more suitable for dynamic applications with strong collaborative effects. On the other hand, in weak collaborative environments, collaborative TS can still exploit the collaborative effects to improve the prediction accuracy and lower the regret. But in these domains, the users should be carefully clustered to exploit the collaborative information.

7 Conclusion

We introduced collaborative Thompson sampling algorithm. The algorithm exploits collaborative effects to accelerate convergence and improve prediction performance. It works under the assumption that users can be partitioned into groups based on their similarity and users in the same cluster can collaborate to accurately estimate the expected reward of items with a few observations. We carried out empirical experiments on two real-world datasets comparing our algorithm with standard Thompson sampling algorithms. The experiments showed that our algorithm outperformed standard Thompson sampling algorithm in terms of efficiency and prediction accuracy in collaborative environments. Thus, the algorithm is more suitable for dynamic applications with strong collaborative effects. We also provided a theoretical analysis of its total expected regret.

Our algorithm can adapt to any Thompson sampling algorithms and clustering technique in a collaborative environment. One direction of the experimental research is to develop a robust dynamic clustering algorithm to exploit collaborative effects. Another line of theoretical research would be providing regret analysis of collaborative Thompson sampling on contextual settings.

Acknowledgments. This paper is supported by the National Science Foundation of China under Grant 61472385 and Grant U1709217.

References

1. Agarwal, D., Long, B., Traupman, J., Xin, D., Zhang, L.: Laser: a scalable response prediction platform for online advertising. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, pp. 173–182. ACM (2014)
2. Agrawal, S., Goyal, N.: Analysis of Thompson sampling for the multi-armed bandit problem. In: Conference on Learning Theory, pp. 39.1–39.26 (2012)
3. Agrawal, S., Goyal, N.: Thompson sampling for contextual bandits with linear payoffs. In: International Conference on Machine Learning, pp. 127–135 (2013)
4. Banerjee, A.: On Bayesian bounds. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 81–88. ACM (2006)
5. Bresler, G., Chen, G.H., Shah, D.: A latent source model for online collaborative filtering. In: Advances in Neural Information Processing Systems, pp. 3347–3355 (2014)
6. Brodén, B., Hammar, M., Nilsson, B.J., Paraschakis, D.: Ensemble recommendations via Thompson sampling: an experimental study within e-Commerce. In: 23rd International Conference on Intelligent User Interfaces, pp. 19–29. ACM (2018)
7. Chapelle, O., Li, L.: An empirical evaluation of Thompson sampling. In: Advances in Neural Information Processing Systems, pp. 2249–2257 (2011)
8. Christakopoulou, K., Banerjee, A.: Learning to interact with users: a collaborative-bandit approach. In: Proceedings of the 2018 SIAM International Conference on Data Mining, pp. 612–620. SIAM (2018)
9. Chu, W., Li, L., Reyzin, L., Schapire, R.: Contextual bandits with linear payoff functions. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 208–214 (2011)
10. Chu, W., et al.: A case study of behavior-driven conjoint analysis on Yahoo!: front page today module. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1097–1104. ACM (2009)
11. Ferreira, K., Simchi-Levi, D., Wang, H.: Online network revenue management using Thompson sampling (2017)
12. Glaze, C.M., Filipowicz, A.L., Kable, J.W., Balasubramanian, V., Gold, J.I.: A bias-variance trade-off governs individual differences in on-line learning in an unpredictable environment. Nat. Hum. Behav. **2**(3), 213 (2018)
13. Gopalan, A., Mannor, S.: Thompson sampling for learning parameterized Markov decision processes. In: Conference on Learning Theory, pp. 861–898 (2015)
14. Gopalan, A., Mannor, S., Mansour, Y.: Thompson sampling for complex online problems. In: International Conference on Machine Learning, pp. 100–108 (2014)

15. Graepel, T., Candela, J.Q., Borchert, T., Herbrich, R.: Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. Omnipress (2010)
16. Johnson, C.C.: Logistic matrix factorization for implicit feedback data. In: Advances in Neural Information Processing Systems, vol. 27 (2014)
17. Kaufmann, E., Korda, N., Munos, R.: Thompson sampling: an asymptotically optimal finite-time analysis. In: Bshouty, N.H., Stoltz, G., Vayatis, N., Zeugmann, T. (eds.) ALT 2012. LNCS (LNAI), vol. 7568, pp. 199–213. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34106-9_18
18. Kawale, J., Bui, H.H., Kveton, B., Tran-Thanh, L., Chawla, S.: Efficient Thompson sampling for online matrix-factorization recommendation. In: Advances in Neural Information Processing Systems, pp. 1297–1305 (2015)
19. Lavancier, F., Rochet, P.: A general procedure to combine estimators. Comput. Stat. Data Anal. **94**, 175–192 (2016)
20. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th International Conference on World Wide Web, pp. 661–670. ACM (2010)
21. Li, S., Karatzoglou, A., Gentile, C.: Collaborative filtering bandits. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 539–548. ACM (2016)
22. Nguyen, T.T., Lauw, H.W.: Dynamic clustering of contextual multi-armed bandits. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 1959–1962. ACM (2014)
23. Ouyang, Y., Gagrani, M., Nayyar, A., Jain, R.: Learning unknown Markov decision processes: a Thompson sampling approach. In: Advances in Neural Information Processing Systems, pp. 1333–1342 (2017)
24. Russo, D.J., Van Roy, B., Kazerouni, A., Osband, I., Wen, Z., et al.: A tutorial on Thompson sampling. Found. Trends® in Mach. Learn. **11**(1), 1–96 (2018)
25. Schwartz, E.M., Bradlow, E.T., Fader, P.S.: Customer acquisition via display advertising using multi-armed bandit experiments. Mark. Sci. **36**(4), 500–522 (2017)
26. Scott, S.L.: A modern Bayesian look at the multi-armed bandit. Appl. Stoch. Models Bus. Ind. **26**(6), 639–658 (2010)
27. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika **25**(3/4), 285–294 (1933)
28. Wolfinger, R.: Laplace's approximation for nonlinear mixed models. Biometrika **80**(4), 791–795 (1993)
29. Wu, Q., Wang, H., Gu, Q., Wang, H.: Contextual bandits in a collaborative environment. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 529–538. ACM (2016)