Personalized Recommendation via Parameter-Free Contextual Bandits

Liang Tang Yexi Jiang Lei Li Chunqiu Zeng Tao Li
School of Computing and Information Sciences, Florida International University
11200 S.W. 8th Street, Miami, FL 33199, USA
{ltang002,yjian004,lli003,czeng001,taoli}@cs.fiu.edu

ABSTRACT

Personalized recommendation services have gained increasing popularity and attention in recent years as most useful information can be accessed online in real-time. Most online recommender systems try to address the information needs of users by virtue of both user and content information. Despite extensive recent advances, the problem of personalized recommendation remains challenging for at least two reasons. First, the user and item repositories undergo frequent changes, which makes traditional recommendation algorithms ineffective. Second, the so-called *cold-start* problem is difficult to address, as the information for learning a recommendation model is limited for new items or new users. Both challenges are formed by the dilemma of exploration and exploitation.

In this paper, we formulate personalized recommendation as a contextual bandit problem to solve the exploration/exploitation dilemma. Specifically in our work, we propose a parameter-free bandit strategy, which employs a principled resampling approach called online bootstrap, to derive the distribution of estimated models in an online manner. Under the paradigm of probability matching, the proposed algorithm randomly samples a model from the derived distribution for every recommendation. Extensive empirical experiments on two real-world collections of web data (including online advertising and news recommendation) demonstrate the effectiveness of the proposed algorithm in terms of the click-through rate. The experimental results also show that this proposed algorithm is robust in the *cold-start* situation, in which there is no sufficient data or knowledge to tune the parameters.

Categories and Subject Descriptors:
H.3.5[Information Systems]: On-line Information Services;
I.2.6[Computing Methodologies]: Learning;
H.2.8[Database Applications]: Data Mining

Keywords: Recommender Systems;Personalization;Contextual Bandit;Probability Matching;Bootstrapping

SIGIR'15, August 09 - 13, 2015, Santiago, Chile.

DOI: http://dx.doi.org/10.1145/2766462.2767707.

1. INTRODUCTION

Personalized recommender systems promptly identify popular items and tailor the content according to users' interest. A user's interest often evolves over time. The uncertainties of information need can only be captured via collecting users' feedbacks in real time and adapting recommendation models to the interest changes. Further, a significant number of users/items might be completely new to the system, that is, they may have no consumption history at all, which is known as the *cold-start* problem [28]. Such a setting renders traditional recommendation approaches ineffective in providing reasonable recommendation results, as it is difficult to learn the match between user preferences and items in a *cold-start* situation.

The aforementioned issues are often recognized as an exploration/exploitation problem, in which we have to find a tradeoff between two competing goals: maximizing users' satisfaction in a long run, while exploring uncertainties of user interests [3]. For instance, a news recommender should prompt breaking news to users while maintaining user preferences based on aging news stories. In practice, such a dilemma is often formulated as a contextual bandit problem [35]. The problem setting consists of a series of trials. Each trial provides a context. An algorithm selects an arm to pull, and after pulling, it receives a reward. The reward is drawn from some unknown distribution determined by the selected arm with the context. The goal is to maximize the total received reward. In personalized recommendation, each trial is seen as a user visit. Every arm is an item (e.g., a news article or advertisement). Pulling an arm is recommending that item. A context is a set of user features. The reward is the user response (e.g., a click). Therefore, personalized recommendation can be seen as an instance of the contextual bandit problem.

Typical solutions of the contextual bandit problem involve unguided exploration (e.g., ϵ-greedy [34], epoch-greedy [18]) and guided exploration (e.g., LinUCB [19], EXP4 [4]). Most of the existing algorithms require an input parameter to control the importance of exploration, such as ϵ in greedy-based algorithms and α in LinUCB. In practice, however, it is often difficult to determine the optimal value for the input parameter. The EXP4 algorithm adopts the exponential weighting technique, but it is computationally expensive especially when the context is high-dimensional.

Another family of algorithms is *probability matching* [32], which randomly allocates the pulling opportunities according to the probability that an arm gives the largest expected

reward[1]. Compared with other methods, the benefit of probability matching is that the tradeoff between exploration and exploitation evolves with the learning process, rather than being arbitrarily set [8]. The pulling allocation is usually implemented by random sampling from the posterior distribution of Bayesian learning models [29, 21]. This strategy is also referred to as Thompson sampling or Bayesian bandits. It provides promising performance in many empirical studies [13, 14, 29, 36]. However, an improper prior for Bayesian learning models can lead to imbalanced exploration and exploitation and jeopardize the overall performance. Moreover, in the *cold-start* situation, there is no enough data for tuning the prior or parameters.

This paper proposes a non-Bayesian implementation of the *probability matching* strategy. The key idea is to apply the online bootstrap to maintain a collection of bootstrap replications for learning model coefficients. To make each recommendation decision, the model coefficient vector is randomly drawn from these bootstrap replications, rather than the posterior distribution. One advantage of this method is that it does not require a prior or predefined parameters that can affect the tradeoff of exploration and exploitation. In summary, the contribution of our work is three-fold:

- We propose a non-Bayesian method based on the *probability matching* strategy to solve the personalized recommendation problem. This method has no input parameter affecting the tradeoff between exploration and exploitation, and is suitable for the *cold-start* situation.
- We give both theoretical and empirical analyses to show that the performance of Thompson sampling depends on the choice of the prior.
- We conduct extensive experiments on real data sets to demonstrate the efficacy of the proposed method compared with other baseline algorithms. The results show that our method is relatively stable. Other algorithms can have a poor performance if the initial parameter or prior is not appropriate.

The rest of this paper is organized as follows. In Section 2, the preliminaries of our work are introduced, and the detailed algorithmic description is presented in Section 3. Extensive empirical evaluation results are reported in Section 4. Section 5 presents a brief summary of prior work relevant to bandit problems, probability matching and bootstrapping. Finally, Section 6 concludes the paper.

2. PRELIMINARIES

In this section, we briefly describe the online learning paradigm for the contextual bandit problem in the setting of personalized recommendation, and then discuss the framework of probability matching in solving the contextual bandit problem. Table 1 lists the important notations throughout the paper.

2.1 Personalized Recommendation and Contextual Bandits

[1]The traditional probability matching is based on the probability of an arm having the largest reward, not the largest expected reward.

Personalized recommender systems recommend items (e.g., movies, news articles) to users based on the predicted users' interests on these items. The user's response helps the system improve their future interest prediction [1]. However, the response to particular items can only be available after these items are recommended. If the items are never shown to the users, the recommender systems cannot collect the response on these items. This problem can be naturally modeled as a contextual bandit problem [35].

Table 1: Important Notations

Notation	Description
$a^{(i)}$	the i-th arm.
$\mathcal{A}$	the set of arms, $\mathcal{A} = \{a^{(1)}, ..., a^{(k)}\}$.
$\mathbf{x}_t$	the context of the t-th trial, and represented by a vector.
r_{t,a_t}	the reward of pulling the arm a_t in the t-th trial, $a_t \in \mathcal{A}$.
$\hat{r}_{t,a_t}$	the expected reward of pulling the arm a_t in the t-th trial, $a_t \in \mathcal{A}$.
$\mathbf{y}_t$	the observation received in the t-th trial, $\mathbf{y}_t = (\mathbf{x}_t, a_t, r_{t,a_t})$.
$\mathcal{D}_t$	the set of received observations from the beginning to the t-th trial, i.e., $\{\mathbf{y}_1, ..., \mathbf{y}_t\}$.
$\mathcal{D}_t^{(i)}$	the set of observations in $\mathcal{D}_t$ that are received only by pulling the arm $a^{(i)}$.
$\tilde{\mathcal{D}}_t^{(i)}$	a bootstrap sample of $\mathcal{D}_t^{(i)}$.
$n_t^{(i)}$	the number of observations in $\mathcal{D}_t^{(i)}$.
$f(\mathbf{x}, \boldsymbol{\theta})$	the reward prediction function using the context $\mathbf{x}$ and the model coefficient vector $\boldsymbol{\theta}$.
$\boldsymbol{\theta}_{a^{(i)}}$	the coefficient vector of the reward prediction model for the arm $a^{(i)}$.
$\hat{\boldsymbol{\theta}}_{a^{(i)}}$	the estimation of $\boldsymbol{\theta}_{a^{(i)}}$.
$\tilde{\boldsymbol{\theta}}_{a^{(i)}}$	a bootstrap replication of $\hat{\boldsymbol{\theta}}_{a^{(i)}}$, which is the estimation of $\boldsymbol{\theta}_{a^{(i)}}$ using a bootstrap sample.
$\mathcal{L}(\boldsymbol{\theta}; \mathbf{y})$	the likelihood of $\mathbf{y}$ given $\boldsymbol{\theta}$.

Formally, given a set of independent arms $\mathcal{A}$, a contextual bandit algorithm makes a decision for each trial $t = 1, 2, ..., n$. For the t-th trial, the context is a feature vector $\mathbf{x}_t$. The algorithm selects an arm $a_t \in \mathcal{A}$ to pull. By pulling the arm a_t, the algorithm receives a reward r_{t,a_t}, which is drawn from some unknown distribution determined by the arm a_t with the context $\mathbf{x}_t$. The goal of the algorithm is to maximize the total received reward $R = \sum_{t=1}^{n} r_{t,a_t}$. Contextual bandit algorithms can make use of the past t trial data $\mathcal{D}_t = \{(\mathbf{x}_1, a_1, r_{1,a_1}), ..., (\mathbf{x}_t, a_t, r_{t,a_t})\}$ to improve the decision for future trials [18], but the past data may not be sufficient for learning. Typically, in the t-th trial the algorithm first predicts the expected reward $\hat{r}_{t,a}$ for each arm a before making the decision. The expected reward is

$$\hat{r}_{t,a} = f(\mathbf{x}_t, \boldsymbol{\theta}_a),$$

where $\boldsymbol{\theta}_a$ is a vector of the unknown coefficients with respect to the arm a and f is a predefined prediction function. For instance, $f(\mathbf{x}_t, \boldsymbol{\theta}_a) = 1/(1 + \exp(-\mathbf{x}_t^T \boldsymbol{\theta}_a))$ is the logistic regression model. By learning from the past observations $\mathcal{D}_t$, the unknown coefficients $\boldsymbol{\theta}_a$ can be estimated.

Several recommendation algorithms consider both user and item information simultaneously, and represent the data as a feature-based user-item matrix. The recommendation can be achieved by utilizing feature-based matrix factorization techniques [9], and the goal is to fill the missing values in the matrix. Our problem setting is orthogonal to theirs as we expect that the total reward is maximized by running a series of trials.

2.2 Probability Matching

Probability matching is a widely used decision strategy in k-armed bandit algorithms [8, 29]. In this strategy, the probability of pulling the arm $a, a \in \mathcal{A}$ equals to the probability that a has the largest expected reward. In the t-th trial, the probability of arm $a^{(i)}$ having the largest expected reward is

$$\begin{aligned} Q(i) &= \Pr(\hat{r}_{t,a^{(i)}} = \max\{\hat{r}_{t,a^{(1)}}, ..., \hat{r}_{t,a^{(k)}}\}), \\ &= \Pr(f(\mathbf{x}_t, \boldsymbol{\theta}_{a^{(i)}}) = \max\{f(\mathbf{x}_t, \boldsymbol{\theta}_{a^{(1)}}), ..., f(\mathbf{x}_t, \boldsymbol{\theta}_{a^{(k)}})\}), \end{aligned}$$

where $i = 1, ..., k$. The selected arm is a random sample drawn from $Q(i)$. But $Q(i)$ does not need to compute explicitly. The algorithms usually draw a vector $\hat{\boldsymbol{\theta}}_{a^{(i)}}$ from the probability distribution of $\boldsymbol{\theta}_{a^{(i)}}$, $i = 1, ..., k$, and select the arm $a^{(i^*)}$, where

$$i^* = \arg\max_{i=1,...,k} f(\mathbf{x}_t, \hat{\boldsymbol{\theta}}_{a^{(i)}}).$$

In Thompson sampling [32], the probability distribution of $\boldsymbol{\theta}_{a^{(i)}}$ in the t-th trial is the posterior distribution, denoted as $\Pr(\boldsymbol{\theta}_{a^{(i)}}|\mathcal{D}_{t-1}^{(i)})$, where $\mathcal{D}_{t-1}^{(i)}$ denotes the set of observations in $\mathcal{D}_{t-1}$ that are only obtained by pulling the arm $a^{(i)}$ [8]. It assumes that the unknown coefficient $\boldsymbol{\theta}_{a^{(i)}}$ follows a predefined probability model, e.g., a Gaussian model,

$$\boldsymbol{\theta}_{a^{(i)}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are unknown parameters. Estimating $\Pr(\boldsymbol{\theta}_{a^{(i)}}|\mathcal{D}_t^{(i)})$ is to find $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$ such that $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ and $\Pr(\boldsymbol{\theta}_{a^{(i)}}|\mathcal{D}_t^{(i)})$ are close to each other. Sampling from the posterior distribution is similar to sampling from $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$.

Based on Thompson sampling, in the $(t+1)$-th trial, the sampling area of $\boldsymbol{\theta}_{a^{(i)}}$ is determined by the variance of $\Pr(\boldsymbol{\theta}_{a^{(i)}}|\mathcal{D}_t^{(i)})$. When t is not sufficiently large, $\Pr(\boldsymbol{\theta}_{a^{(i)}}|\mathcal{D}_t^{(i)})$ mainly depends on the prior $\Pr(\boldsymbol{\theta}_0)$, denoted by $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. In other words, if $\boldsymbol{\Sigma}_0$ is large, the algorithm performs more exploration; otherwise, it does less. Therefore $\boldsymbol{\Sigma}_0$ mainly controls the tradeoff between exploration and exploitation in early trials. Moreover, $\boldsymbol{\Sigma}_0^{-1}$ is the regularization weight for estimating $\boldsymbol{\mu}_t$. If $\boldsymbol{\Sigma}_0$ is too small, the sampling will only focus on a small area around $\boldsymbol{\mu}_0$, but $\boldsymbol{\mu}_0$ may not be accurate. For instance, let $f(\mathbf{x}_t, \boldsymbol{\theta}_a) = 1/(1 + \exp(-\mathbf{x}_t^T \boldsymbol{\theta}_a))$, then the prediction model is a logistic regression model. Let $\mathcal{D}_t^{(i)} = \{(\mathbf{x}_1^{(i)}, a^{(i)}, r_1^{(i)}), ..., (\mathbf{x}_{t^{(i)}}^{(i)}, a^{(i)}, r_{t^{(i)}}^{(i)})\}$. By Laplace approximation [33],

$$\boldsymbol{\Sigma}_t^{-1} = \boldsymbol{\Sigma}_0^{-1} + \nabla\boldsymbol{\Sigma}^{-1},$$

where

$$\nabla\boldsymbol{\Sigma}^{-1} = \sum_{j=1}^{t^{(i)}} f(\mathbf{x}_j^{(i)}, \boldsymbol{\mu}_t)(1 - f(\mathbf{x}_j^{(i)}, \boldsymbol{\mu}_t))\mathbf{x}_j^{(i)}\mathbf{x}_j^{(i)^T}.$$

$\boldsymbol{\Sigma}_0^{-1}$ is the start point of $\boldsymbol{\Sigma}_t^{-1}$. In the *cold-start* situation, since we do not know the values from the data $\mathcal{D}_t^{(i)}$ in advance, it is not easy to come up with $\boldsymbol{\Sigma}_0^{-1}$ that is balanced with $\nabla\boldsymbol{\Sigma}^{-1}$.

Therefore, the given $\Pr(\boldsymbol{\theta}_0)$ dominates the balance of exploration and exploitation in early trials. An improper estimation in earlier trials also affects later trials. "Good" arms might be underrated in earlier trials, and then would have few chances to be pulled later and be corrected. As a result, the algorithm would take a long time to converge to the optimal estimation.

3. ALGORITHM

In this section, we present a non-Bayesian algorithm to implement the probability matching strategy. The basic idea is using the sampling distribution obtained by the bootstrap instead of the posterior distribution to sample the prediction model coefficients for each item. We first discuss an offline bootstrap method for solving the contextual bandit problem. Then, we present an online implementation of the bootstrap method along with an online optimization algorithm.

3.1 Bootstrap for Contextual Bandits

The bootstrap is a method to derive the distribution of an estimator by data resampling [10]. Instead of specifying a generative model for data generating process, it only uses the information from the observed data.

In the $(t+1)$-th trial, we have the previous t pulling observations, $\mathcal{D}_t$. Let $\mathcal{D}_t^{(i)}$ denote the observations only from pulling the arm $a^{(i)}$, $i \in \{1, ..., k\}$. k is the number of arms. $\mathcal{D}_t = \mathcal{D}_t^{(i)} \cup ... \cup \mathcal{D}_t^{(k)}$. Let $n_t^{(i)}$ be the number of observations in $\mathcal{D}_t^{(i)}$. When any $\mathcal{D}_t^{(i)}$ is not sufficient large (e.g., less than 30 observations [15]), $i = 1, ..., k$, we randomly select an arm. When all $\mathcal{D}_t^{(i)}$ are sufficient large, for the $(t+1)$-th trial, given the context $\mathbf{x}_{t+1}$, the offline bootstrap based contextual bandit algorithm has the following steps:

- For each $i = 1, ...k$,
  1. Randomly sample $n_t^{(i)}$ observations from $\mathcal{D}_t^{(i)}$ with replacement, denoted by $\tilde{\mathcal{D}}_t^{(i)}$;
  2. Estimate $\boldsymbol{\theta}_{a^{(i)}}$ using $\tilde{\mathcal{D}}_t^{(i)}$ based on maximum likelihood estimation, denoted by $\tilde{\boldsymbol{\theta}}_{a^{(i)}}$;
- Pull the arm $a^{(i^*)}$, where $i^* = \arg\max_{i=1,...,k} f(\mathbf{x}_{t+1}, \tilde{\boldsymbol{\theta}}_{a^{(i)}})$;
- Receive the reward r_{t+1} of pulling the arm $a^{(i^*)}$, $\mathcal{D}_{t+1}^{(i^*)} \leftarrow \mathcal{D}_t^{(i^*)} \cup \{(\mathbf{x}_{t+1}, a^{(i^*)}, r_{t+1})\}$.

where $\tilde{\boldsymbol{\theta}}_{a^{(i)}}$ is a bootstrap replication of $\hat{\boldsymbol{\theta}}_{a^{(i)}}$. If the step 1 and step 2 are repeated for many times, we can have a collection of bootstrap replications of $\hat{\boldsymbol{\theta}}_{a^{(i)}}$, which approximately represents the sampling distribution of $\hat{\boldsymbol{\theta}}_{a^{(i)}}$. Therefore, $\tilde{\boldsymbol{\theta}}_{a^{(i)}}$ can be seen as a random sample drawn from the distribution of $\hat{\boldsymbol{\theta}}_{a^{(i)}}$.

Let $\mathcal{L}(\boldsymbol{\theta}; \mathbf{y})$ denote the likelihood of an observation $\mathbf{y}$ by given $\boldsymbol{\theta}$, where $\boldsymbol{\theta}$ is the unknown coefficient vector. Assuming the observations are i.i.d and the arms are independent, we have the following lemma.

LEMMA 1. *The offline bootstrap based contextual bandit algorithm implements the probability matching strategy.*

PROOF. In the $(t+1)$-th trial, given a context $\mathbf{x}_{t+1}$, let $\mathbf{a} = \arg\max_{a^{(i)} \in \mathcal{A}} f(\mathbf{x}_{t+1}, \hat{\boldsymbol{\theta}}_{a^{(i)}})$, where $\hat{\boldsymbol{\theta}}_{a^{(1)}}$, ... ,$\hat{\boldsymbol{\theta}}_{a^{(k)}}$ are the model coefficient vectors estimations of the k arms. Since these estimations vary with different observations, $\hat{\boldsymbol{\theta}}_{a^{(1)}}, ..., \hat{\boldsymbol{\theta}}_{a^{(k)}}$ are random variables. Then, $\mathbf{a}$ is a random variable and randomized by the joint random variable $(\hat{\boldsymbol{\theta}}_{a^{(1)}}, ..., \hat{\boldsymbol{\theta}}_{a^{(k)}})$. Based on *probability matching*, the selected arm should be a sample randomly drawn from the distribution $\Pr(\mathbf{a})$.

Let $\pi_{a^{(i)}}$ denote some unknown distribution for generating the observations by pulling arm $a^{(i)}$, $i = 1, ..., k$. $\mathbf{Y}_1^{(i)}, \mathbf{Y}_2^{(i)}, ...$ are independent random variables following $\pi_{a^{(i)}}$. Let $\mathbf{D}_t^{(i)} = \{\mathbf{Y}_1^{(i)}, ..., \mathbf{Y}_{n_t^{(i)}}^{(i)}\}$. Obviously, $\mathcal{D}_t^{(i)}$ is the observed value of $\mathbf{D}_t^{(i)}$. $\hat{\boldsymbol{\theta}}_{a^{(i)}}$ is the maximum likelihood estimation of $\boldsymbol{\theta}_{a^{(i)}}$, so in the t-th trial,

$$\hat{\boldsymbol{\theta}}_{a^{(i)}} = \arg\max_{\boldsymbol{\theta} \in R^d} \log \mathcal{L}(\boldsymbol{\theta}; \mathbf{D}_t^{(i)}),$$

where d is the dimensionality of the context. $\hat{\boldsymbol{\theta}}_{a^{(i)}}$ is randomized by $\mathbf{D}_t^{(i)}$. Since every observation in $\tilde{\mathcal{D}}_t^{(i)}$ is drawn from $\pi_{a^{(i)}}$ and every random variable in $\mathbf{D}_t^{(i)}$ follows $\pi_{a^{(i)}}$, $\tilde{\mathcal{D}}_t^{(i)}$ is also a random sample of $\mathbf{D}_t^{(i)}$. Then, based on the bootstrap method, $\tilde{\boldsymbol{\theta}}_{a^{(i)}} = \arg\max_{\boldsymbol{\theta} \in R^d} \log \mathcal{L}(\boldsymbol{\theta}; \tilde{\mathcal{D}}_t^{(i)})$ is also a random sample of $\hat{\boldsymbol{\theta}}_{a^{(i)}}$.

To sum up all k arms, $\tilde{\boldsymbol{\theta}}_{a^{(1)}}$, ... ,$\tilde{\boldsymbol{\theta}}_{a^{(k)}}$ are random samples of $\hat{\boldsymbol{\theta}}_{a^{(1)}}, ..., \hat{\boldsymbol{\theta}}_{a^{(k)}}$, respectively. Since the arms are independent, $(\tilde{\boldsymbol{\theta}}_{a^{(1)}}, ..., \tilde{\boldsymbol{\theta}}_{a^{(k)}})$ is a sample randomly drawn from $\Pr(\hat{\boldsymbol{\theta}}_{a^{(1)}}, ..., \hat{\boldsymbol{\theta}}_{a^{(k)}})$. Then $a^{(i^*)} = \arg\max_{a^{(i)} \in \mathcal{A}} f(\mathbf{x}_{t+1}, \tilde{\boldsymbol{\theta}}_{a^{(i)}})$ can be seen as a sample randomly drawn from $\Pr(\mathbf{a})$. □

3.2 An Online Implementation

In real recommender systems, each recommendation decision must be made in real time. The algorithm cannot go through all previous observations to generate a bootstrap sample $\tilde{\mathcal{D}}_t^{(i)}$. On the other hand, the learning algorithm cannot estimate $\tilde{\boldsymbol{\theta}}_{a^{(i)}}$ utilizing the entire $\tilde{\mathcal{D}}_t^{(i)}$. Therefore, in our problem setting, we apply the online bootstrap method to solve the contextual bandit problem [23].

3.2.1 Online Bootstrap

The basic idea of online bootstrap is to generate a random variable P_j, which is the proportion of times the j-th observation is picked to a bootstrap sample. Then in the online setting, when we receive the j-th observation, we know how many times this observation should appear in a bootstrap sample. After processing the j-th observation, we do not pick it again. Let $\mathcal{D}_t^{(i)}$ be the set of received observations by pulling the arm $a^{(i)}$. $n_t^{(i)}$ is the size of $\mathcal{D}_t^{(i)}$. $\tilde{\mathcal{D}}_t^{(i)}$ is a bootstrap sample of $\mathcal{D}_t^{(i)}$. $\mathcal{D}_t^{(i)}$ and $\tilde{\mathcal{D}}_t^{(i)}$ have the same number of elements. The elements in $\tilde{\mathcal{D}}_t^{(i)}$ are randomly resampled from $\mathcal{D}_t^{(i)}$ with replacement. For each resampling, each element in $\mathcal{D}_t^{(i)}$ has an identical probability of $1/n_t^{(i)}$ to be picked. There are $n_t^{(i)}$ independent chances of resampling. Therefore, $P_j \sim \text{Binom}(n_t^{(i)}, 1/n_t^{(i)})$. When $n_t^{(i)}$ is large, this binomial distribution is approximated to a Poisson distribution $\text{Pois}(n_t^{(i)} \cdot 1/n_t^{(i)}) = \text{Pois}(1)$ [23]. Then,

$$P_j \sim \text{Poisson}(1).$$

P_j does not depend on t.

3.2.2 Online Learning and Bootstrapping

The online learning algorithm processes every observation in a streaming manner. When a new observation is received, by online bootstrap, this observation should appear P_j times in the bootstrap sample, where $P_j \sim$ Poisson(1). We invoke the online learning algorithm to learn this observation P_j times. Then, the learned model is approximated to the model that learns observations in a bootstrap sample offline.

In our proposed algorithm, we apply this idea with the stochastic gradient ascent algorithm for updating the estimation of each bootstrap replication. Let $\tilde{\boldsymbol{\theta}}_{a^{(i)}}$ be a current bootstrap replication of $\boldsymbol{\theta}_{a^{(i)}}$. η_z is the current learning rate for this bootstrap replication in the stochastic gradient ascent algorithm, where z is the number of updated times. Usually, $\eta_z = 1/\sqrt{z+1}$ [16]. $\mathbf{y}_t = (\mathbf{x}_t, a^{(i)}, r_{t,a^{(i)}})$ is the received new observation. Based on online bootstrap, we draw a random integer p_t from Poisson(1). The new bootstrap replication of $\boldsymbol{\theta}_{a^{(i)}}$ is $\tilde{\boldsymbol{\theta}}_{a^{(i)}}^{(p_t)}$, where

$$\tilde{\boldsymbol{\theta}}_{a^{(i)}}^{(l)} = \tilde{\boldsymbol{\theta}}_{a^{(i)}}^{(l-1)} + \eta_{z+l-1} \nabla \log \mathcal{L}(\tilde{\boldsymbol{\theta}}_{a^{(i)}}^{(l-1)}; \mathbf{y}_t), \tag{1}$$

$l = 1, ..., p_t$, $\tilde{\boldsymbol{\theta}}_{a^{(i)}}^{(0)} = \tilde{\boldsymbol{\theta}}_{a^{(i)}}$, η_{z+p_t} is the new learning rate.

If we only maintain one bootstrap sample for each arm, the decisions made for different trials may not be fully independent. Our solution is to maintain a collection of independent bootstrap samples for each arm at the same time. In every trial, we randomly select one of them to make the reward prediction. Let B be the number of bootstrap samples for each arm. There are B independent bootstrap replications of $\hat{\boldsymbol{\theta}}_{a^{(i)}}$ maintained for each $a^{(i)} \in \mathcal{A}$, denoted by $\mathcal{B}_{a^{(i)}} = \{\tilde{\boldsymbol{\theta}}_{a^{(i)},1}, ... , \tilde{\boldsymbol{\theta}}_{a^{(i)},B}\}$. In each trial and arm $a^{(i)}$, we randomly select a bootstrap replication from $\mathcal{B}_{a^{(i)}}$, where every bootstrap replication has an equal probability to be selected. Let $\tilde{\boldsymbol{\theta}}_{a^{(i)}}$ be the selected replication. $\Pr(\tilde{\boldsymbol{\theta}}_{a^{(i)}})$ is represented by $\mathcal{B}_{a^{(i)}}$. When B and $n_t^{(i)}$ are sufficiently large, $\mathcal{B}_{a^{(i)}}$ provides an approximation of the sampling distribution of $\hat{\boldsymbol{\theta}}_{a^{(i)}}$ [23, 26]. The details of the online bootstrap algorithm for contextual bandits are stated in Algorithm 1.

Algorithm 1 *OnlineBootstrapBandit*

```
Receive the context x_t.
for i = 1, ..., k do
    Randomly select a vector from {θ̃_{a(i),1}, ..., θ̃_{a(i),B}}, denoted by θ̃_{a(i)}.
end for
Pull the arm a^{(i*)}, where i* = arg max_{i=1,...,k} f(x_t, θ̃_{a(i)}).
Receive the reward r_t.
y_t ← {x_t, a^{(i*)}, r_t}.
for j = 1, ..., B do
    Draw p from Pois(1)
    for z = 1, ..., p do
        η_{i,j} ← 1/√(n_{i,j} + 1)
        θ̃_{a(i*),j} ← θ̃_{a(i*),j} + η∇ log L(θ̃_{a(i*),j}; y_t)
        n_{i,j} ← n_{i,j} + 1
    end for
end for
```

In general, the time complexity of calculating a log-likelihood is $O(d)$, where d is the dimensionality of context feature vectors. Let $T(t)$ be the time cost of a trial in Algorithm 1. Generating a Poisson random variable is $O(p_t)$, where p_t is

the generated value [17]. Eq.(1) has p_t iterations, and hence updating one bootstrap estimation requires $O(p_t \cdot d + p_t) = O(p_t \cdot d)$. Based on the cumulative distribution function (CDF) of Poisson(1), $\Pr(p_t \leq p) = e^{-1} \sum_{z=0}^{p} \frac{1}{z!}$. For example, the probability of $p_t \leq 3$ is 0.981. Therefore, the time cost of updating one bootstrap estimation is $O(d)$ with a high probability. There are k arms and B bootstrap replications for each arm. As a result, $T(t) = O(Bkd)$ with a high probability.

In practice, the larger the B, the better the sampling distribution approximation. However, the memory cost and time cost will become significantly large. Thus, the choice of B depends on the actual computational power of the system. As the B bootstrap replications are independent, they can be easily implemented in a parallel system, where each computing node handles a few replications independently.

4. EVALUATION

We verify the proposed algorithm on two real-world data sets, including news recommendation data (i.e., Yahoo! Today News) and online advertising data (i.e., KDD Cup 2012, Track 2). We start with an introduction to these two data sets, and then describe the implementation of the baseline algorithms. Finally, we present experimental results of the proposed algorithm with comparison to the baselines.

4.1 Data Collections

4.1.1 Yahoo! Today News

Personalized news recommendation aims to display suitable news articles on the web page for different users based on the prediction of their individual interests. The prediction model is usually built upon user feedbacks on displayed news. However, the feedbacks are only available when the news articles are displayed to the users. Therefore, the problem of personalized news recommendation can be regarded as an instance of the contextual bandit problem.

The experimental data set is collected by Yahoo! Today module and published by Yahoo! research lab[2]. The news were randomly displayed on the Yahoo! Front Page from October 2nd, 2011 to October 16th, 2011. The data set contains 28,041,015 user visit events to the Today Module on Yahoo! Front Page. Each visit event is associated with the user's information, e.g., age, gender, behavior targeting features, etc., represented by a binary feature vector of dimension 136. This data set has been used for evaluating contextual bandit algorithms in other literatures [19, 8, 20]. 2 million user visit events are used in this evaluation.

4.1.2 KDD Cup 2012 Online Advertising

Online advertising systems deliver relevant advertisements (ads) to individual users to maximize the click-though rate (CTR) of the displayed ads. Sponsored search is one typical instance of online advertising. Given a user profile and a set of search keywords, the search engine selects an ad (advertisement) to display in the search result page. In practice, a huge amount of new ads will be continually imported into the ad pool. The system has to display these new ads to users and then collects the feedbacks to improve the CTR prediction. Hence, the ad selection problem is an instance of the contextual bandit problem, where an arm is an ad, a trial is an ad impression for a search activity, the context is the user profile with the search keywords, and the reward is the click count of the user.

The experimental data set is collected by a search engine and published by KDD Cup 2012[3]. In this data set, each instance is an ad impression, which consists of the user profile, search keywords, displayed ad information and the click count. The user profile contains the user's gender and age. In our work, the context is represented as a binary feature vector, each entry of which denotes whether a query token is contained in the search query or not. The user's profile information is also appended to the context vector using the binary format. The dimension of the context feature for this data set is 1,070,866. 1 million user visit events are used in the experiments.

4.2 Experimental Setup

For evaluation purpose, we use the averaged reward as the metric, which is the total reward divided by the total number of trials, i.e., $\frac{1}{n}\sum_{t=1}^{n} r_t$, where n is the number of trials. In the aforementioned data sets, the averaged reward is the overall CTR (click-through rate) of the corresponding items (news articles or ads). The higher the CTR, the better the performance. In the experiments, to avoid the leakage of business-sensitive information, we report the relative CTR, which is the overall CTR of an algorithm divided by the overall CTR of random selection.

To demonstrate the efficacy of our proposed approach, we implement the following algorithms as baselines:

- `Random`: it randomly selects an arm to pull.
- `Exploit`: it selects the arm of the largest predicted reward and has no exploration. `Exploit` is equivalent to ϵ-`greedy`(0), `LinUCB`(0) and `TSNR`$(+\infty)$.
- ϵ-`greedy`(ϵ): it randomly selects an arm with probability ϵ and selects the arm of the largest predicted reward with probability $1-\epsilon$.
- `LinUCB`(α): it is an extension of the UCB algorithm for contextual bandit problems [19]. In each trial, it pulls the arm of the largest score, which is a linear combination of the mean and standard deviation of the predicted reward. Given a context $\mathbf{x}$, the score is $\hat{\boldsymbol{\mu}}^T\mathbf{x} + \alpha\sqrt{\mathbf{x}^T\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{x}}$, where $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ are the estimated mean and covariance of the posterior distribution $\Pr(\boldsymbol{\theta}|\mathcal{D}_t)$, and α is a predefined parameter. When $\alpha = 0$, it becomes the `Exploit` policy that has no exploration.
- `TS`(q_0): thompson sampling with logistic regression [8], described in Section 2.2, it randomly draws the coefficients from the posterior distribution, and selects the arm of the largest predicted reward. The priori distribution is $\mathcal{N}(\mathbf{0}, q_0^{-1}\mathbf{I})$.
- `TSNR`(q_0): it is similar to `TS`(q_0), but in the stochastic gradient ascent, there is no regularization by the prior. The priori distribution $\mathcal{N}(\mathbf{0}, q_0^{-1}\mathbf{I})$ is only used in the calculation of the posterior distribution for the coefficients sampling, but not in the stochastic gradient ascent. When q_0 is arbitrarily large, the variance approaches 0 and `TSNR` becomes `Exploit`.

[2]http://webscope.sandbox.yahoo.com/catalog.php.

[3]http://www.kddcup2012.org/c/kddcup2012-track2.

- `Bootstrap`: it is proposed in this paper and described in Algorithm 1.

In the following experiments, the reward in a single recommendation activity is the user click, which is a binary value. Therefore, logistic regression is applied as the learning model. Since the contextual bandit algorithms are online algorithms, stochastic gradient ascent is used as the learning algorithm [7]. Notice that the algorithms digest the data in an online manner, and hence all the user visits in the data sets are used for the testing purpose.

Thompson sampling with logistic regression is described in [8]. The prediction function $f(\mathbf{x}, \boldsymbol{\theta}) = (1+\exp(-\boldsymbol{\theta}^T\mathbf{x}))^{-1}$. The posterior distribution is obtained by Laplace approximation and the unknown coefficient vector $\boldsymbol{\theta}$ is assumed to be normally distributed [7]. Let $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ denote the posterior distribution after receiving t observations. The estimated $\boldsymbol{\mu}_t$ is the *maximum a posteriori* (MAP) estimation , which is learned by the stochastic gradient ascent method. The inverse of the estimated covariance $\boldsymbol{\Sigma}_t^{-1} = \boldsymbol{\Sigma}_0^{-1} + \sum_{j=1}^{t} y_j(1-y_j)\mathbf{x}_j\mathbf{x}_j^T$, where $\boldsymbol{\Sigma}_0$ is the given priori covariance, $y_t = f(\mathbf{x}_t, \boldsymbol{\theta}_t)$, and $\mathbf{x}_1, \ldots, \mathbf{x}_t$ are the context feature vectors [8].

As we mentioned in Section 2, to decrease the exploration in Thompson sampling, we can give a small variance to the prior. But when the variance of the prior is small, the weight for the regularization in the stochastic gradient ascent will be high. As a result, the learned $\boldsymbol{\mu}_t$ would focus on a small area around the mean of the prior. To solve this conflict, we propose a variation of Thompson sampling, `TSNR`. In stochastic gradient ascent, it ignores the regularization by the prior in the learning step. The prior is only used to compute the variance of the posterior.

4.3 Evaluation Method

The experiments on the Yahoo! Today news is evaluated by the *replayer* method [20], which provides an unbiased offline evaluation by utilizing the historical logs. It shows that, for a testing algorithm, the CTR estimated by this *replayer* approaches the real CTR of the deployed online system if the items in historical user visits are random uniformly recommended. Therefore, we apply the *replayer* to evaluate the performance of various algorithms on the Yahoo! Today News data. The basic idea of *replayer* is to replay each user visit to the testing algorithm. If the recommended item is equal to displayed news in the log, this visit is regarded as a matched visit. The estimated CTR is the sum of the user clicks in the matched visits over the total number of matched visits.

However, the *replayer* only works for a small item pool. When the number of items is large, the number of matched visits for each item would be very small. As a result, the CTR estimation based on a small number of matched visits is not reliable and would have a large variance [8]. For the Yahoo! Today news, the number of recommending articles is less than 50. But for online advertising data, the number of ads is usually over 16,000. We evaluate the KDD Cup 2012 online ads data using a simulation method, which is used in [8] for the same purpose. To this end, we select 100 ads from the entire ads pool. The context data of these ads are real and given in the data, but the rewards are simulated using a weight vector $\mathbf{w}$ for each ad. Given a context $\mathbf{x}$, the click of an ad is generated with a probability $(1+\exp(-\mathbf{w}^T\mathbf{x}))^{-1}$. For each user visit and each arm, the weight vector $\mathbf{w}$ is drawn from a fixed normal distribution that is randomly generated before the testing.

4.4 Experimental Results

We consider the performances of algorithms in two situations: cold start and warm start. In cold start, there is no training data at the beginning for every algorithm. The algorithm can only learn the model by exploration. In warm start, we have 10,000 records of user activities for training. We train the logistic regression model using these data first. Tables 3 and 2 report the results of Yahoo! News data and KDD Cup 2012 online ads data, respectively. For each algorithm, we enumerate different parameter values. Except for `LinUCB` and `Exploit`, all other algorithms are randomized algorithms. For each trial, we also randomly shuffle the pool of recommending items. Thus, the performance of `LinUCB` and `Exploit` may vary in different runs. We run each algorithm with each parameter value 10 time, and keep track of the mean, standard deviation, minimum and maximum of the overall CTR. The best mean is highlighted in **bold**. The worst mean is marked with an asterisk $(*)$.

As depicted in the tables, the baseline algorithms of ϵ-`greedy` and `LinUCB`, which take into account both exploration and exploitation, exhibit a common trend: when the controlling parameter is small, i.e., with more exploitation, the algorithms can achieve better performance in terms of the CTR, whereas the deviation is high; Comparatively, when the parameter is set to be larger, i.e., with more exploration, the CTR shrinks, but the deviation decreases. Hence, the problem of personalized recommendation requires a tradeoff between exploration and exploitation. Further, the performance of ϵ-`greedy`, `LinUCB` highly depends on the parameter setting. The parameters of both algorithms explicitly or implicitly control the balance of the exploration and exploitation. If the parameter setting is perfect, the performance approaches the optimal. If the parameter setting is improper, the performance is poor. This conclusion is also mentioned in the empirical studies of [5, 31].

`TS` and `TSNR` are two types of Thompson sampling. `TS` is the straightforward implementation. It makes use of the given priori distribution and iteratively maximizes the posterior mean, where the inverse of the priori variance is the regularization weight. If the prior variance is large, e.g., `TS(0.001)`, the sampling area of `TS` will be too large, which may significantly sacrifice the exploitation part. If the prior variance is small, e.g., `TS(10)`, the regularization weight is large and the learned $\hat{\boldsymbol{\theta}}$ will be close to the given prior $\mathbf{0}$. Obviously, $\mathbf{0}$ is not a good guess of $\boldsymbol{\theta}$. Thus, for all parameter settings, the performance of `TS` is not satisfactory in terms of the CTR.

`TSNR` only maximizes the likelihood using stochastic gradient ascent and ignores the regularization from the prior. Therefore, the priori distribution only affects the sampling area of $\hat{\boldsymbol{\theta}}$ in the exploration part but not learning steps. As shown in Table 3 and 2, when the priori variance is appropriate, e.g., `TSNR(1000.0)`, it can achieve good performance. In `TSNR(1000.0)`, the variance is $q_0^{-1}\mathbf{I} = 1/1000.0\mathbf{I}$, which is quite small, meaning that we do not have to explore the areas that are far away from the estimated $\hat{\boldsymbol{\theta}}$. However, when q_0 is arbitrarily large, the variance approaches 0 and there will be no exploration at all. Consequently, `TSNR` becomes `Exploit`. As reported in the tables, the performance of `Ex-`

Table 2: Relative CTR on KDD Cup 2012 Online Ads Data.

Algorithm	Cold Start				Warm Start			
	mean	std	min	max	mean	std	min	max
`Bootstrap(1)`	**1.9933**	0.01291	1.9692	2.0098	**1.9990**	0.005678	1.9878	2.0083
`Bootstrap(5)`	1.9883	0.01106	1.9686	2.0012	1.9964	0.004983	1.9848	2.0022
`Bootstrap(10)`	1.9862	0.009128	1.9672	1.9977	1.9890	0.005434	1.9829	2.0003
`Bootstrap(30)`	1.9824*	0.01492	1.9566	2.0088	1.9886*	0.006086	1.9753	1.9954
ϵ`-greedy(0.01)`	**1.9941**	0.007293	1.9834	2.0060	**1.9971**	0.004908	1.9886	2.0038
ϵ`-greedy(0.1)`	1.9089	0.004887	1.8965	1.9145	1.8952	0.002741	1.8910	1.8986
ϵ`-greedy(0.3)`	1.7039	0.003797	1.6990	1.7101	1.6973	0.009368	1.6834	1.7193
ϵ`-greedy(0.5)`	1.5018*	0.004335	1.4965	1.5114	1.4983*	0.006319	1.4845	1.5067
`Exploit`	**1.8185***	0.05235	1.7228	1.8934	**1.9241***	0.007046	1.9152	1.9370
`LinUCB(0.01)`	1.8551	0.03543	1.7977	1.9059	**1.9279**	0.006951	1.9178	1.9371
`LinUCB(0.1)`	**1.9168**	0.005466	1.9070	1.9267	1.9202	0.004434	1.9112	1.9266
`LinUCB(0.3)`	1.8665	0.003644	1.8609	1.8726	1.8610	0.003271	1.8550	1.8661
`LinUCB(0.5)`	1.7808	0.007009	1.7669	1.7913	1.7903	0.0051	1.7823	1.7988
`LinUCB(1.0)`	1.6693*	0.004738	1.6634	1.6762	1.6742*	0.003179	1.6704	1.6792
`TS(0.001)`	1.3587	0.009703	1.3366	1.3736	1.3518	0.01002	1.3297	1.3673
`TS(0.01)`	1.4597	0.007215	1.4504	1.4749	1.4891	0.006421	1.4771	1.4994
`TS(0.1)`	**1.5714**	0.004855	1.5647	1.5791	**1.5905**	0.004176	1.5826	1.5967
`TS(1.0)`	1.5345	0.003435	1.5262	1.5384	1.5421	0.003741	1.5376	1.5480
`TS(10.0)`	0.9388*	0.4236	0.3064	1.5675	1.3174*	0.003157	1.3115	1.3212
`TSNR(0.01)`	1.4856*	0.01466	1.4657	1.5078	1.5700*	0.02163	1.5499	1.6298
`TSNR(0.1)`	1.7931	0.01284	1.7774	1.8167	1.8716	0.01035	1.8518	1.8870
`TSNR(1.0)`	1.9826	0.005853	1.9704	1.9921	1.9952	0.006996	1.9833	2.0047
`TSNR(10.0)`	**2.0118**	0.007808	1.9941	2.0208	2.0095	0.005107	2.0022	2.0198
`TSNR(100.0)`	2.0039	0.008942	1.9912	2.0215	**2.0097**	0.004586	2.0022	2.0187
`TSNR(1000.0)`	2.0047	0.01022	1.9894	2.0228	2.0088	0.004644	1.9966	2.0151

`ploit` is relatively poor compared with `Bootstrap`, ϵ`-greedy` and `LinUCB` in terms of the mean and deviation of the CTR.

The performance of `Bootstrap` is comparable with the ones of ϵ`-greedy` and `TSNR` in terms of the CTR, as it takes a non-Bayesian strategy based on bootstrapping without considering the prior and posterior distributions. The tradeoff between exploration and exploitation is handled in an evolving manner as the data size increases. In addition, when we use different numbers of bootstrap samples, the averaged reward varies very slightly. The reason here is straightforward: the number of bootstrap samples dominates the accuracy of the approximation towards the sampling distribution, rather than the one controlling the balance of the exploration and exploitation.

4.4.1 On Bootstrap Sample Size

The sample size of bootstrap determines the approximation accuracy of the bootstrap method. The sample size is larger, the approximation accuracy is higher. Therefore, it is not a parameter for the recommendation model. We argue that once the sample size is large enough, it would not affect the performance much, as it is not a dominant factor to control the exploration/exploitation or the recommendation model. To verify this claim, we set different bootstrap sample sizes from 1 to 500, and report the result in Figure 1. As depicted in this figure, the performance is still relative stable comparing to other algorithms. It is interesting to see that even if $B = 1$, in which the sampled bootstrap replications have some dependence for the same arm, the performance of

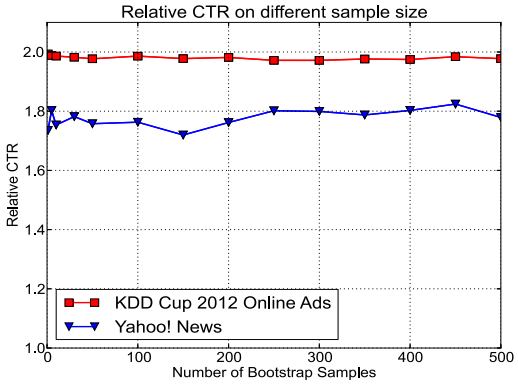

Figure 1: Relative CTR on different bootstrap sample sizes.

`Bootstrap` is not poor. Hence, `Bootstrap` provides a better and safe choice for personalized recommendation when we do not have any prior information about the data, and in this sense, we argue that our proposed method is parameter-free in terms of the exploration/exploitation tradeoff.

We also investigate the scalability of `Bootstrap` (see Figure 2). We focus on the time cost for each model update with online estimation on different bootstrap sample sizes. As presented in Figure 2, the number of bootstrap samples increases from 50 to 500 (10 times), whereas the milliseconds per update increases about 3 times. Hence, a real-

Table 3: Relative CTR on Yahoo! News Data.

Algorithm	Cold Start				Warm Start			
	mean	std	min	max	mean	std	min	max
`Bootstrap(1)`	1.7350*	0.08327	1.6032	1.9123	1.7029*	0.1392	1.4299	1.8358
`Bootstrap(5)`	**1.8025**	0.07676	1.6526	1.9127	1.8366	0.07996	1.7118	1.9514
`Bootstrap(10)`	1.7536	0.07772	1.6338	1.8814	**1.8403**	0.08518	1.6673	1.9296
`Bootstrap(30)`	1.7818	0.08857	1.6092	1.9025	1.8311	0.08699	1.7230	1.9396
ϵ`-greedy(0.01)`	**1.7708**	0.09383	1.6374	1.9503	**1.8466**	0.05494	1.7846	1.9755
ϵ`-greedy(0.1)`	1.7375	0.04992	1.6452	1.8003	1.8132	0.03502	1.7621	1.8721
ϵ`-greedy(0.3)`	1.5486	0.03703	1.4812	1.5930	1.5976	0.02739	1.5591	1.6491
ϵ`-greedy(0.5)`	1.3819*	0.02341	1.3489	1.4169	1.3753*	0.02884	1.3173	1.4020
`Exploit`	**1.1782***	0.2449	0.9253	1.5724	**1.1576***	0.00198	1.1554	1.1607
`LinUCB(0.01)`	**1.6349**	0.08967	1.4849	1.7360	**1.8103**	0	1.8103	1.8103
`LinUCB(0.1)`	1.2037	0.02321	1.1682	1.2577	1.2394	0	1.2394	1.2394
`LinUCB(0.3)`	1.1661	0.01073	1.1552	1.1926	1.1650	1.863e-08	1.1650	1.1650
`LinUCB(0.5)`	1.1462	0.01215	1.1136	1.1571	1.1752	1.317e-08	1.1752	1.1752
`LinUCB(1.0)`	1.1361*	0.01896	1.0969	1.1594	1.1594*	1.317e-08	1.1594	1.1594
`TS(0.001)`	**1.2203**	0.026	1.1842	1.2670	**1.2725**	0.03175	1.2301	1.3422
`TS(0.01)`	1.1880	0.02895	1.1585	1.2466	1.2377	0.01886	1.2132	1.2713
`TS(0.1)`	1.1527	0.01988	1.1289	1.1811	1.1791	0.02225	1.1437	1.2169
`TS(1.0)`	1.1205	0.0142	1.1009	1.1472	1.1362	0.02203	1.0971	1.1599
`TS(10.0)`	0.7669*	0.1072	0.5445	0.9526	0.8808*	0.01557	0.8483	0.9031
`TSNR(0.01)`	1.2173*	0.03369	1.1430	1.2561	1.2972*	0.02792	1.2479	1.3394
`TSNR(0.1)`	1.2285	0.01948	1.1915	1.2610	1.3028	0.02121	1.2701	1.3461
`TSNR(1.0)`	1.2801	0.02365	1.2558	1.3303	1.3250	0.03148	1.2486	1.3634
`TSNR(10.0)`	1.6657	0.03285	1.6025	1.7125	1.6153	0.05608	1.5210	1.7128
`TSNR(100.0)`	**1.7816**	0.07609	1.7093	1.9278	1.8399	0.1134	1.5240	1.9200
`TSNR(1000.0)`	1.7652	0.09946	1.6123	1.9346	**1.8769**	0.03731	1.8409	1.9656

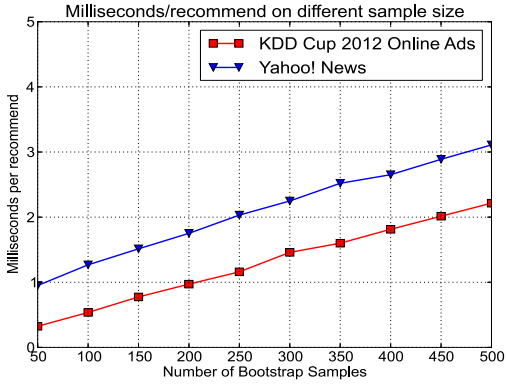

Figure 2: Time cost on different bootstrap sample sizes.

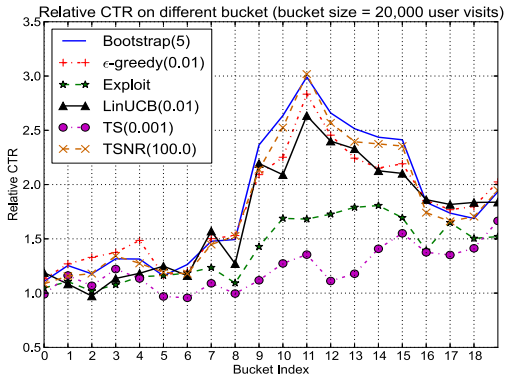

Figure 3: Relative CTR on different time buckets for Yahoo! Today News.

world production server cluster can easily handle more than 1000 bootstrap samples for personalized recommendation services, e.g., online advertising or news recommendation.

4.4.2 *On Time Bucket*

Besides the overall CTR of each algorithm, we also evaluate the CTR on individual time bucket. The CTR on each bucket is only calculated by the clicks collected in that bucket. The entire testing data is split into 20 time buckets. For Yahoo! Today news data, each bucket has 100,000 user visits. For KDD Cup 2012 online ad data, each bucket has 50,000 user visits. All the user visit events are order by the time. Figures 3 and 4 show the relative CTR on individual buckets. As shown in Figure 3, `Bootstrap`(5), `TSNR`(100) and ϵ`-greedy` are superior to other baselines starting from the 8-th bucket. Also, on different buckets the lift of each algorithm with respect to `random` is different. The main reason is that the user interests on these news articles may change over time. It is worthy to note that the large lifts of CTR are only in the middle buckets. In the last a few buckets, the lifts become smaller, although the prediction models have more feedback to learn. In other words, when

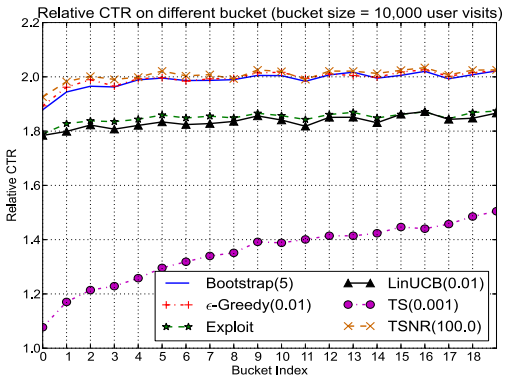

Figure 4: Relative CTR on different time buckets for KDD Cup 2012 Online Ads.

a news article becomes aging, there is no much space for a recommender system to improve its CTR. Therefore, the recommender system should be able to promptly learn the predictive model in an online manner. If it takes a long time to cumulate user feedbacks, user preferences may have changed even if the prediction model can be trained well.

The reward of the KDD Cup 2012 online ads is simulated by the logistic regression function and a fixed weight vector with some random noises. For each ad, the relation between the context and reward is much simpler than the Yahoo! Today news data. Thus, the data is easy to learn by a logistic regression model. The CTR curves in Figure 4 are very stable. `Bootstrap`(5), `TSNR`(100) and ϵ-`greedy` converge very quickly staring from the second bucket. The performance of `TS`(0.001) increases slowly. The prior variance of `TS`(0.001) is $(1/0.001)\mathbf{I} = 1000\mathbf{I}$, which is very large. The sampled coefficients are usually far away from the posterior mean in the early stage. But when it learns more data, the posterior variance becomes smaller and the performance increases gradually.

To summarize, our findings from the experiments are threefold: (1) For solving the contextual bandit problem, the algorithms of ϵ-`greedy` and `LinUCB` can achieve the optimal performance, but the input parameters that control the exploration need to be tuned carefully; (2) The probability matching strategies, e.g., Thompson sampling, can have the optimal result, but it highly depends on the selection of the prior; and (3) Our proposed approach, `Bootstrap`, is a safe choice of building predictive models for contextual bandit problems under the scenario of *cold-start*.

5. RELATED WORK

Our work is primarily relevant to three active areas of research, namely (1) modeling multi-armed bandit problems, (2) probability matching for contextual bandit problems, and (3) bootstrapping for statistics estimation.

Multi-armed bandit problem: The primary challenge in multi-armed bandit problems is to balance the tradeoff between exploration and exploitation. In practice, for solving a *cold-start* problem, neither a pure exploitation or a pure exploration works best, as there are always uncertainties of user preferences to explore. A wide range of application-oriented problems have been modeled as the context-free multi-armed bandit problem, such as online advertising [25, 30], web search and content optimization [2, 27], network optimization [11], game playing [12], routing [6], etc. A list of computational strategies have been proposed in the past decades, including ϵ-greedy, EXP3 [4], upper confidence bound (UCB) [5, 19], etc. The basic paradigm for solving the context-free bandit problem is to run multiple trials to estimate the reward distribution. However, these solutions are either sub-optimal or require careful settings of the balancing parameters. In our work, we present a parameter-free algorithm to avoid the process of parameter tuning, i.e., without the input parameter to allocate the importance of exploration.

Probability matching for contextual bandit: A more general version of the bandit problem is called contextual bandit, which has not been well studied. Here the contextual information is related to the specific environment of pulling the arms with a learning problem; for example, in online advertising systems, the context might involve a user's query, or the web page on which an ad is placed. Several interesting approaches have been reported by following the ϵ-greedy or UCB [4, 19] paradigms.

Another family of algorithms for solving contextual bandit problems is *probability matching* [32], which pulls different arms according to the probability that the corresponding arm has the largest expected reward. Instead of requiring a controlling parameter for the balance of exploration/exploitation, the strategy of probability matching enables the learning process to automatically adjust the tradeoff [8]. Representative work along this stream involves [8, 14, 22], which follow the Bayesian paradigm to estimate the uncertainties. However in many practical scenarios, an inappropriate prior for Bayesian learning models may lead to imbalanced exploration/exploitation in the early stage of learning, and consequently jeopardize the overall performance. Comparatively in our work, we propose a non-Bayesian algorithm based on the framework of probability matching, which utilizes bootstrapping for coefficient estimation, and does not require the input of priors.

Bootstrapping for statistics estimation: Bootstrapping, in statistics, is a useful technique for estimating some statistical properties of an estimator [10]. This technique resamples the observed data with replacement, and then utilizes the resampled data sets to infer the properties of the estimator. In machine learning, it is often used in the bootstrap aggregation (bagging) to ensemble different learning algorithms. However, the traditional bootstrap method operates in an offline way in which the observations are all already given. In some situations, the observations arrive from a data stream and the bootstrapping must be made in an online manner. To handle the online setting, [23] presents an online paradigm of data resampling using a Poisson distribution. [26] makes use of this method to improve the robustness of the learning model for large data sets. In our work, we employ the idea of online bootstrapping to the scenario of estimating cofficients of contextual bandit models.

6. CONCLUSION

In personalized recommender systems, the dilemma of exploration/exploitation in the cold-start situation remains a challenging issue due to the uncertainty of user preferences. In this paper, we formulate the problem of personalized recommendation as a contextual bandit problem to balance the tradeoff between these two competing goals. We propose a

parameter-free strategy for bandit problems, which employs a principled resampling approach called online bootstrap, to derive the sampling distributions of learning model estimators. The proposed algorithm is essentially an ensemble method to achieve optimal rewards without specifying exploration parameters. Extensive empirical experiments on two real-world data sets, i.e., online advertising and news recommendation, demonstrate the efficacy of our proposed approach in terms of averaged rewards.

As for the future work, the recommend items, e.g., advertisements or news articles, may have some underlying relations with each other. For example, two advertisements may belong to the same categories, or come from business competitors, or have other same features. In the future, we plan to consider the potential correlations among different items, or say, arms [24]. It is interesting to model these correlations as constraints, and incorporate them into the contextual bandit modeling process. A second direction to extend our proposed contextual model is to consider the temporal information of user preferences, as the interests of users may often evolve over time.

Acknowledgement

The work is partially supported by NSF grants CNS-1126619, IIS-1213026, and CNS-1461926, DHS grant 2010-ST-062-000039, Purdue VACCINE/DHS 4112-35822, and an FIU Dissertation Year Fellowship.

7. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 17(6):734–749, 2005.

[2] D. Agarwal, B.-C. Chen, and P. Elango. Explore/exploit schemes for web content optimization. In *ICDM*, pages 1–10. IEEE, 2009.

[3] D. Agarwal, B.-C. Chen, P. Elango, N. Motgi, S.-T. Park, R. Ramakrishnan, S. Roy, and J. Zachariah. Online models for content optimization. In *NIPS*, pages 17–24, 2008.

[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

[5] P. Auer and N. C.-B. P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[6] B. Awerbuch and R. Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97–114, 2008.

[7] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2006.

[8] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *NIPS*, pages 2249–2257, 2011.

[9] T. Chen, Z. Zheng, Q. Lu, W. Zhang, and Y. Yu. Feature-based matrix factorization. *arXiv preprint arXiv:1109.2271*, 2011.

[10] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*, volume 57. 1994.

[11] Y. Gai, B. Krishnamachari, and R. Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *TON*, 20(5):1466–1478, 2012.

[12] S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of uct with patterns in monte-carlo go. Technical report, 2006.

[13] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *ICML*, pages 13–20, 2010.

[14] O.-C. Granmo. Solving two-armed bernoulli bandit problems using a bayesian learning automaton. *International Journal of Intelligent Computing and Cybernetics*, 3(2):207–234, 2010.

[15] R. V. Hogg and E. A. Tanis. *Probability and Statistical Inference*. Prentice Hall, 1996.

[16] D. Hsu, N. Karampatziakis, J. Langford, and A. J. Smola. Parallel online learning. *CoRR*, abs/1103.4204, 2011.

[17] D. E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. 1997.

[18] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*, 2007.

[19] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670. ACM, 2010.

[20] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, pages 297–306, 2011.

[21] B. C. May, N. Korda, A. Lee, and D. S. Leslie. Optimistic bayesian sampling in contextual-bandit problems. *Journal of Machine Learning Research*, 13:2069–2106, 2012.

[22] B. C. May, N. Korda, A. Lee, and D. S. Leslie. Optimistic bayesian sampling in contextual-bandit problems. *The Journal of Machine Learning Research*, 13(1):2069–2106, 2012.

[23] N. C. Oza and S. Russell. Online bagging and boosting. In *IEEE international conference on Systems, man and cybernetics*, volume 3, pages 2340–2345, 2005.

[24] S. Pandey, D. Chakrabarti, and D. Agarwal. Multi-armed bandit problems with dependent arms. In *ICML*, pages 721–728, 2007.

[25] S. Pandey and C. Olston. Handling advertisements of unknown quality in search advertising. In *NIPS*, pages 1065–1072, 2006.

[26] Z. Qin, V. Petricek, N. Karampatziakis, L. Li, and J. Langford. Efficient online bootstrapping for large scale learning. *arXiv preprint arXiv:1312.5021*, 2013.

[27] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *ICML*, pages 784–791. ACM, 2008.

[28] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, pages 253–260. ACM, 2002.

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

[30] A. Slivkins. Multi-armed bandits on implicit metric spaces. In *NIPS*, pages 1602–1610, 2011.

[31] L. Tang, R. Rosales, A. Singh, and D. Agarwal. Automatic ad format selection via contextual bandits. In *CIKM*, pages 1587–1594, 2013.

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

[33] L. Tierney and J. B. Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86, 1986.

[34] M. Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *KI 2010: Advances in Artificial Intelligence*, pages 203–210. 2010.

[35] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *ECML*, pages 437–448. 2005.

[36] X. Zhao, W. Zhang, and J. Wang. Interactive collaborative filtering. In *ACM CIKM*, pages 1411–1420, 2013.