Ensemble Contextual Bandits for Personalized Recommendation

Liang Tang Yexi Jiang Lei Li Tao Li
Florida International University
11200 S.W. 8th Street, Miami, FL 33199
{ltang002,yjian004,lli003,taoli}@cs.fiu.edu

ABSTRACT

The cold-start problem has attracted extensive attention among various online services that provide personalized recommendation. Many online vendors employ contextual bandit strategies to tackle the so-called *exploration/exploitation* dilemma rooted from the cold-start problem. However, due to high-dimensional user/item features and the underlying characteristics of bandit policies, it is often difficult for service providers to obtain and deploy an appropriate algorithm to achieve acceptable and robust economic profit.

In this paper, we explore ensemble strategies of multiple contextual bandit algorithms to obtain robust predicted *click-through rate* (CTR) of web objects. Specifically, the ensemble is acquired by aggregating different pulling policies of bandit algorithms, rather than forcing the agreement of prediction results or learning a unified predictive model. To this end, we employ a meta-bandit paradigm that places a hyper bandit over the base bandits, to explicitly explore/exploit the relative importance of base bandits based on user feedbacks. Extensive empirical experiments on two real-world data sets (news recommendation and online advertising) demonstrate the effectiveness of our proposed approaches in terms of CTR.

1. INTRODUCTION

Personalized recommendation services aim to identify popular items and tailor the content according to users' preferences. In practice, a large number of users or items might be completely new to the system, which refers to the *cold-start* problem [22]. This issue is often recognized as an exploration/exploitation problem, in which we have to find a trade-off between two competing goals: maximizing users' satisfaction in a long run, while exploring uncertainties of user interests [1]. For example, a news recommender should prompt breaking news to users while maintaining user preferences based on aging news stories.

The aforementioned issue is often modeled as a contextual bandit problem [32], which consists of a series of trials with corresponding contexts. By pulling an arm, one can obtain a reward, drawn from some unknown distribution determined by the selected arm and the context. The goal is to maximize the total received reward. In personalized recommendation, each trial can be treated as a user visit. Every arm is an item (e.g., a news article or advertisement). Pulling an arm is to recommend 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 regarded as an instance of the contextual bandit problem.

Recently, a series of algorithms have been reported to tackle the contextual bandit problem, including unguided exploration (e.g., ϵ-greedy [30], epoch-greedy [13], etc.) and guided exploration (e.g., LinUCB [14], EXP4 [2], Thompson sampling [6] etc.). These existing algorithms can achieve promising performance under specific settings. The performances of different policies[1] vary significantly in many recommendation applications. A common practice of picking the appropriate policy is to first evaluate these policies and then select the best one to deploy. However, in the cold-start situation, it is often difficult to conduct an unbiased offline evaluation due to the deficiency of the historical data. For online evaluation, e.g., A/B test [25], the online web traffic has to be split to multiple buckets for different policies, and therefore the number of testing policies running in parallel is restricted in order to obtain acceptable daily income.

In our work, we explore the possibility of utilizing ensemble strategies to obtain a robust policy that can achieve acceptable CTR in various recommender systems. As the predictive result of each contextual bandit algorithm is the pulled arm (item), it is not appropriate to adopt majority voting or consensus prediction as the ensemble. We hence resort to meta learning to build a hyper policy that adaptively allocates the pulling chances to different base policies based on the estimation of their performance. The proposed ensemble bandit algorithms may not produce the optimal CTR of base policies, but it can always approach to the best one, which gives a robust mechanism for online recommendation in the cold-start situations.

In summary, the contribution of our work is three-fold:

- We explore the possibility of stabilizing the CTR estimation of web objects by integrating the advantages of different bandit policies.
- We propose two ensemble strategies to address the cold-start problem in personalized recommendation applications. Both strategies employ a meta-bandit learning paradigm to achieve the robustness of the CTR.

[1]A policy refers to a decision maker, which is a combination of a contextual bandit algorithm, a predictive model and a specific parameter setting.

- We conduct extensive experiments on real-world data sets to demonstrate the efficacy of the proposed method compared with other baseline algorithms. The results show that our method is robust in terms of CTR.

The rest of this paper is organized as follows. In Section 2, we describe a brief summary of prior work relevant to contextual bandit problems, ensemble recommendation and meta learning. We then formulate the problem in Section 3, and present the detailed algorithmic description in Section 4. Extensive empirical evaluation results are reported in Section 5. Finally, Section 6 concludes the paper.

2. RELATED WORK

The robustness of contextual bandit algorithms for predicting CTR is one of the most prominent factors that dominate the economic profit of service providers. However, to the best of our knowledge, no research effort has been paid on exploring this important property. In our work, we employ ensemble strategies combined with a meta learning paradigm to stabilize the output of contextual bandit algorithms. In the following, we highlight the previous research that are most relevant to our work.

Contextual Bandits: When predicting the CTR of web data, the *cold-start* problem is often modeled as a contextual bandit problem with exploration/exploitation trade-off, where user features are regarded as contextual information. Typical solutions of this problem involve unguided exploration (e.g., ϵ-greedy [30], epoch-greedy [13]), guided exploration (e.g., LinUCB [14], EXP4 [2]) and probability matching (e.g., Thompson sampling [6, 23, 28]). Most existing methods require either a parameter to control the importance of exploration or prior information of Bayesian learning models; however in practice, it is difficult to determine the optimal value for the input due to the insufficiency of user feedbacks. Hence, the prediction performance of these algorithms is not stable along both exploration and exploitation phases, unless the selection policy/model converges.

Ensemble Recommendation: Ensemble based algorithms have been well explored to improve the performance of prediction [21, 29], and are often preferred in recommendation competitions, such as the Netflix Prize contest [12, 26, 31] and KDD Cups [18, 34, 33]. Typically, an ensemble method combines the prediction of different algorithms to obtain a final prediction [19], which is often referred to as "blending" [11]. The most basic blending strategy is to acquire the final prediction based on the mean over all the prediction results or the majority vote. Learning based approaches have also been proposed [35] to unify different recommendation algorithms. In our work, to obtain a robust policy, we resort to ensemble strategies that assimilate the advantages of different contextual bandit algorithms.

Meta Learning: In machine learning community, the goal of meta learning is to accumulate experience on the performance of multiple learning algorithms [8]. Meta learning has been widely used in algorithm selection [17, 20]. Due to the uncertainty of learning algorithms, i.e., we do not know in advance the predictive performance, a lot of work has modeled algorithm selection as a multi-armed bandit problem [7, 27] and tries to balance the trade-off between exploring algorithm capabilities and exploiting the predictive power of algorithms. In addition, some recent research efforts [9, 16, 24] focus on meta learning of exploration/exploitation strategies, where the base learners are bandit algorithms.

3. PROBLEM FORMULATION

In this section, we formulate the problem studied in this paper. Let $\mathcal{A}$ denote the set of items (or bandit arms), $\mathcal{A} = \{a_1, ..., a_k\}$, and $\Pi = \{\pi_1,...,\pi_m\}$ be a given set of bandit policies, where each policy is a contextual bandit algorithm with a specific parameter setting. $\pi_i(\mathbf{x}) = a$ indicates that the policy π_i pulls a with respect to $\mathbf{x}$, where $a \in \mathcal{A}$ and $\mathbf{x}$ is a context feature vector. Let $\mathcal{D}$ be the space of $\mathbf{x}$ and $p(\mathbf{x})$ denote the probability density of $\mathbf{x}$. After the pulling, π_i receives a reward r, where r is a value drawn from the conditional distribution $p(\cdot|\mathbf{x}, a)$. In this paper, we only consider $r \in \{0, 1\}$, i.e., a non-click/click on a specific item. Thus, $p(\cdot|\mathbf{x}, a)$ is a Bernoulli distribution.

Each policy $\pi_i \in \Pi$ aims to maximize the expected received reward denoted by $\mathrm{E}[r_{\pi_i}]$, where

$$\begin{aligned}\mathrm{E}[r_{\pi_i}] &= \int_{\mathcal{D}} \sum_{a\in\mathcal{A}} \sum_{r\in\{0,1\}} r \cdot p(r|\mathbf{x}, a) \cdot p(a = \pi_i(\mathbf{x})|\mathbf{x}) \cdot p(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= \int_{\mathcal{D}} \sum_{\pi_i(\mathbf{x})\in\mathcal{A}} \sum_{r\in\{0,1\}} r \cdot p(r, \pi_i(\mathbf{x}), \mathbf{x})\, \mathrm{d}\mathbf{x}.\end{aligned}$$

The expected reward $\mathrm{E}[r_{\pi_i}]$ is known as the CTR of the policy π_i, which is often used as the performance metric.

Existing studies propose different contextual bandit algorithms and show their empirical performances on various real-world data sets [5, 6, 14]. It is known that a bandit algorithm with different parameter settings can have different performance [23]. The choice of parameters depends on the distribution of the real data, which is unknown in the *cold-start* situation. Therefore, given a set of policies Π, individual polices in Π can have very different performance for a particular recommender system. Let π^* denote the best policy in terms of the performance, i.e.,

$$\pi^* = \arg\max_{\pi_i\in\Pi} \mathrm{E}[r_{\pi_i}]. \quad (1)$$

For different recommendation problems, π^* is different and not known in advance. The goal of this paper is to develop an ensemble contextual bandit policy such that its performance can be close to the performance of π^*.

4. ALGORITHM

This section presents two ensemble bandit algorithms, `HyperTS` and `HyperTSFB`, for solving the contextual recommendation problem in the cold-start situation. The idea of these two algorithms is to distribute the trials to the base bandit policies. Given a set of policies $\Pi = \{\pi_1, ..., \pi_m\}$ and a context $\mathbf{x}$, both algorithms make two decisions to deicide which arm to pull:

1. Select a policy from Π, denoted by π_i;
2. Select the arm $a = \pi_i(\mathbf{x})$ to pull.

For the first decision, if we know which base policy is the best one, i.e., π^*, we can always select π^*. However, the performance of each policy is unknown at the beginning. To estimate their expected rewards, we need to select them and observe the received rewards. Same as the second decision, the exploration-exploitation dilemma also exists in the first decision. In this paper, we put our focus on the first decision.

To address the policy selection problem in the first decision, both of the proposed algorithms leverage non-contextual Thompson sampling [6, 28]. Generally, in each trial, the algorithms randomly select a policy $\pi_i \in \Pi$, where the probability of selecting π_i is equal to the probability of π_i being π^*,

i.e., $p(\hat{\mathrm{E}}[r_{\pi_i}] = \max_{\pi_j \in \Pi} \hat{\mathrm{E}}[r_{\pi_j}])$, where $\hat{\mathrm{E}}[r_{\pi_i}]$ is the estimated expected reward of the policy π_i. It is difficult to directly compute this probability [23]. Thus, in each trial, we randomly draw a value, denoted by $\tilde{r}_{\pi_i}$, from the distribution of $\hat{\mathrm{E}}[r_{\pi_i}]$ for each $\pi_i \in \Pi$, and then select the policy that has the maximum value of $\tilde{r}_{\pi_i}$. In the following, we present two approaches to estimate $\mathrm{E}[r_{\pi_i}]$,

4.1 HyperTS Algorithm

HyperTS estimates the expected reward $\mathrm{E}[r_{\pi_i}]$ of each policy $\pi_i \in \Pi$ using Monte Carlo method. Concretely, let $\mathbf{x}_1$,...,$\mathbf{x}_n$ be the contexts of n trials in which π_i is selected. $\mathbf{x}_1$,...,$\mathbf{x}_n$ are samples drawn from $p(\mathbf{x})$. For an input context $\mathbf{x}_j$, π_i pulls the arm a_j and receives the reward r_j, where a_j is seen as a sample from $p(a = \pi_i(\mathbf{x}_j)|\mathbf{x}_j)$, r_j is seen as a sample drawn from $p(r|\mathbf{x}_j, a_j)$. Thus, $(\mathbf{x}_j, a_j, r_j)$ is a sample drawn from the joint distribution $p(\mathbf{x}, a, r)$, $j = 1, ..., n$. The Monte Carlo estimate is $\hat{\mathrm{E}}[r_{\pi_i}] = \frac{1}{n}\sum_{j=1}^{n} r_j$. The rewards $r_1, ..., r_n \in \{0, 1\}$ are the sample drawn from the Bernoulli distribution $p(r)$, which is a marginal distribution of $p(\mathbf{x}, a, r)$. Therefore, $\hat{\mathrm{E}}[r_{\pi_i}]$ follows the Beta distribution:

$$\hat{\mathrm{E}}[r_{\pi_i}] \sim \mathrm{Beta}(1 + \alpha_{\pi_i}, 1 + \beta_{\pi_i}), \tag{2}$$

where $\alpha_{\pi_i} = \sum_{j=1}^{n} r_j$ and $\beta_{\pi_i} = n - \alpha_{\pi_i}$. For the prior, $\mathrm{Beta}(1, 1)$ is used.

Algorithm 1 `HyperTS`(Π)

Input: Π : the set of base bandit policies.
for $i = 1, ..., m$ **do**
  $\alpha_{\pi_i} \leftarrow 0$, $\beta_{\pi_i} \leftarrow 0$
end for
for $t = 1, 2, ...$ **do**
  for $i = 1, ..., m$ **do**
    Draw r_i from $\mathrm{Beta}(1 + \alpha_{\pi_i}, 1 + \beta_{\pi_i})$.
  end for
  Pull the arm $a = \pi_j(\mathbf{x}_t)$, where $j = \arg\max_{i=1,...,m} r_i$.
  Receive the reward $r_{\mathbf{x}_t,a}$ and feed π_j with $r_{\mathbf{x}_t,a}$.
  if $r_{\mathbf{x}_t,a} = 1$ **then**
    $\alpha_{\pi_j} \leftarrow \alpha_{\pi_j} + 1$
  else
    $\beta_{\pi_j} \leftarrow \beta_{\pi_j} + 1$
  end if
end for

Algorithm 1 shows the pseudo-code of `HyperTS`. In each trial, r_i is a sample of $\hat{\mathrm{E}}[r_{\pi_i}]$, drawn from a Beta distribution. The selected policy is the one having the maximum r_i.

4.2 HyperTSFB Algorithm

In `HyperTS`, the expected reward of each base policy is estimated only from the feedback when that policy is selected. The feedback of the decision made by other policies is not utilized. If the number of policies in Π is large, the total number of trials needed for exploring the performance of base policies will be large and the total reward will be smaller. To improve the estimation efficiency, we propose `HyperTSFB` (`HyperTS` with shared feedback), an algorithm that fully utilizes every received feedback for expected reward estimation.

Given the context $\mathbf{x}$, `HyperTSFB` requires each base policy $\pi_i \in \Pi$ to provide the probability of π_i pulling the arm a, i.e., $p(a = \pi_i(\mathbf{x})|\mathbf{x})$. Then, even though the policy π_i is not selected in the trial, `HyperTSFB` can still utilizes the feedback for $\mathbf{x}$ to estimate the expected reward of π_i. For some policy, $p(a = \pi_i(\mathbf{x})|\mathbf{x})$ can be computed directly. For instance, if π_i denotes `random` policy, then $p(a = \pi_i(\mathbf{x})|\mathbf{x}) = 1/k$, $k = |\mathcal{A}|$; if π_i is ϵ-`greedy`, then

$$p(a = \pi_i(\mathbf{x})|\mathbf{x}) = \begin{cases} \epsilon/k + (1 - \epsilon) & \text{if } a = a^* \\ \epsilon/k & \text{if } a \neq a^*, \end{cases}$$

where a^* is the arm that has the maximum predicted reward by the input $\mathbf{x}$. For some policy, $p(a = \pi_i(\mathbf{x})|\mathbf{x})$ can be difficult to compute, e.g. contextual Thompson sampling. We can invoke $\pi_i(\mathbf{x})$ multiple times to estimate this probability according to the frequency of a being output.

Once we can compute $p(a = \pi_i(\mathbf{x})|\mathbf{x})$, $\mathrm{E}[r_{\pi_i}]$ can be rewritten as Eq. (3) and importance sampling [36] can be leveraged for estimation.

$$\begin{aligned}
\mathrm{E}[r_{\pi_i}] &= \sum_{a\in\mathcal{A}} \int_{\mathcal{D}} \sum_{r\in\{0,1\}} r \cdot p(r|\mathbf{x}, a) p(a = \pi_i(\mathbf{x})|\mathbf{x}) p(\mathbf{x})\, \mathrm{d}\mathbf{x} \\
&= \sum_{a\in\mathcal{A}} \int_{\mathcal{D}} \sum_{r\in\{0,1\}} r \frac{p(a = \pi_i(\mathbf{x})|\mathbf{x})}{p(a|\mathbf{x})} p(r|\mathbf{x}, a) p(a|\mathbf{x}) p(\mathbf{x})\, \mathrm{d}\mathbf{x} \\
&= \sum_{a\in\mathcal{A}} \int_{\mathcal{D}} \sum_{r\in\{0,1\}} (r \cdot w^{\pi_i}_{a,\mathbf{x}}) \cdot p(r|\mathbf{x}, a) p(a|\mathbf{x}) p(\mathbf{x})\, \mathrm{d}\mathbf{x} \\
&= \sum_{a\in\mathcal{A}} p(a) \int_{\mathcal{D}} \sum_{r\in\{0,1\}} (r \cdot w^{\pi_i}_{a,\mathbf{x}}) \cdot p(r, \mathbf{x}|a)\, \mathrm{d}\mathbf{x} \\
&= \sum_{a\in\mathcal{A}} p(a)\, \mathrm{E}_{r,\mathbf{x}}[r \cdot w^{\pi_i}_{a,\mathbf{x}}|a].
\end{aligned} \tag{3}$$

In Eq. (3), $w^{\pi_i}_{a,\mathbf{x}} = \frac{p(a=\pi_i(\mathbf{x})|\mathbf{x})}{p(a|\mathbf{x})}$ is the importance weight, and $\mathrm{E}_{r,\mathbf{x}}[r \cdot w^{\pi_i}_{a,\mathbf{x}}|a]$ is the expected reward of π_i by pulling a. Eq. 3 states that the expected reward estimation can be separated into multiple estimations for different arms. In the importance weight, $p(a|\mathbf{x})$ is the probability of the arm a being pulled given the context $\mathbf{x}$,

$$p(a|\mathbf{x}) = \sum_{\pi_i \in \Pi} p(a = \pi_i(\mathbf{x})|\mathbf{x}) p(\pi_i = \arg\max_{\pi_j \in \Pi} \hat{\mathrm{E}}[r_{\pi_j}]).$$

In the implementation, we use a sampling-based method to obtain the value of $p(a|\mathbf{x})$. For each given context $\mathbf{x}$, we invoke `HyperTSFB` multiple times and then estimate $p(a|x)$ according to the frequency of a being selected. $p(a)$ is the marginal probability of a being selected, which is simply approximated by the ratio of a being pulled in all previous trials done by `HyperTSFB`.

In Eq. (3), the expected reward of π_i by pulling a is

$$\begin{aligned}
\mathrm{E}_{r,\mathbf{x}}[r \cdot w^{\pi_i}_{a,\mathbf{x}}|a] &= \int_{-\infty}^{+\infty} y \cdot p(r \cdot w^{\pi_i}_{a,\mathbf{x}} = y|a)\, \mathrm{d}y \\
&= \int_{-\infty}^{+\infty} y \cdot p(r = 1, w^{\pi_i}_{a,\mathbf{x}} = y|a)\, \mathrm{d}y \\
&= \int_{-\infty}^{+\infty} y \cdot p(w^{\pi_i}_{a,\mathbf{x}} = y|r = 1, a) p(r = 1|a)\, \mathrm{d}y \\
&= p(r = 1|a) \cdot \int_{-\infty}^{+\infty} y \cdot p(w^{\pi_i}_{a,\mathbf{x}} = y|r = 1, a)\, \mathrm{d}y \\
&= \mathrm{E}_{r,\mathbf{x}}[r|a] \cdot \mathrm{E}_{\mathbf{x}}[w^{\pi_i}_{a,\mathbf{x}}|r = 1, a].
\end{aligned} \tag{4}$$

In Eq. (4), $\mathrm{E}_{r,\mathbf{x}}[r|a]$ is the expected reward of pulling arm a, which is also the overall CTR of a and determined by the popularity of a. The importance weight $w^{\pi_i}_{a,\mathbf{x}}$ is proportional to the probability of π_i pulling a given $\mathbf{x}$. $\mathrm{E}_{\mathbf{x}}[w^{\pi_i}_{a,\mathbf{x}}|r = 1, a]$

reflects how likely will π_i pulls a if the reward of a is 1. Intuitively, Eq. 4 states that the expected reward of π_i by pulling a is determined by the popularity of a and the likelihood of π_i pulling a if the reward of a is 1. Since $r \in \{0, 1\}$, we use a Beta distribution to model $\hat{\mathrm{E}}_{r,\mathbf{x}}[r|a]$, i.e.

$$\hat{\mathrm{E}}_{r,\mathbf{x}}[r|a] \sim \mathrm{Beta}(1 + \alpha_a, 1 + \beta_a), \tag{5}$$

where α_a is the number of trials that the received reward is 1 by pulling a, β_a is the number of trials that the received reward is 0 by pulling a. For the prior, uniform distribution $\mathrm{Beta}(1, 1)$ is used. $\mathrm{E}_{\mathbf{x}}[w^{\pi_i}_{a,\mathbf{x}}|r = 1, a]$ is estimated by the mean of importance weights in previous trials in which a is pulled and the reward is 1. Assuming for one policy and one arm those importance weights will converge in one distribution, based on Central Limit Theory, the sample mean follows a normal distribution when the sample size is sufficient large, i.e.

$$\hat{\mathrm{E}}_{\mathbf{x}}[w^{\pi_i}_{a,\mathbf{x}}|r = 1, a] \sim \mathcal{N}(\mu_{a,i}, \sigma^2_{a,i}/n_{a,i}), \tag{6}$$

where $\mu_{a,i}$ and $\sigma^2_{a,i}$ are the mean and variance of the distribution of the importance weights, $n_{a,i}$ is the sample size to calculate the the mean and variance. If the sample size $n_{a,i}$ is not sufficient large (less than 30 [10]), we draw $\hat{\mathrm{E}}_{\mathbf{x}}[w^{\pi_i}_{a,\mathbf{x}}|r = 1, a]$ from uniform distribution $\mathcal{U}(0, 1)$.

Algorithm 2 shows the pseudo-code of `HyperTSFB`. For trial $t = 1, 2, ...$, given the context $\mathbf{x}_t$, the sampled expected reward of π_i is r_{π_i}, which is calculated based on two other sampled values from each arm (Line 8 to 21). As for the received reward, $r_{\mathbf{x}_t,a}$, the estimated parameters of every base policy and arm a are updated (Line 24 to 34).

Algorithm 2 `HyperTSFB`(Π)

Input: Π : the set of base bandit policies.

1: **for** $j = 1, ..., k$ **do**
2: $\alpha_{a_j} \leftarrow 0$, $\beta_{a_j} \leftarrow 0$, $n_{a_j} \leftarrow 0$
3: **for** $i = 1, ..., m$ **do**
4: $n_{a_j,i} \leftarrow 0$
5: **end for**
6: **end for**
7: **for** $t = 1, 2, ...$ **do**
8: **for** $j = 1, ..., k$ **do**
9: Draw r_{a_j} from $\mathrm{Beta}(1 + \alpha_{a_j}, 1 + \beta_{a_j})$
10: **end for**
11: **for** $i = 1, ..., m$ **do**
12: $r_{\pi_i} \leftarrow 0$
13: **for** $j = 1, ..., k$ **do**
14: **if** $n_{a_j,i} < 30$ **then**
15: Draw $w^{\pi_i}_{a_j}$ from $\mathcal{U}(0, 1)$
16: **else**
17: Draw $w^{\pi_i}_{a_j}$ from $\mathcal{N}(\mu_{a_j,i}, \sigma^2_{a_j,i}/n_{a_j,i})$
18: **end if**
19: $r_{\pi_i} \leftarrow r_{\pi_i} + n_{a_j}/t \cdot r_{a_j} \cdot w^{\pi_i}_{a_j}$
20: **end for**
21: **end for**
22: Pull the arm $a = \pi_{s^*}(\mathbf{x}_t)$, where $s^* = \arg\max_{i=1,...,m} r_{\pi_i}$.
23: Receive the reward $r_{\mathbf{x}_t,a}$, feed each $\pi_i \in \Pi$ with $r_{\mathbf{x}_t,a}$
24: $n_{a_j} \leftarrow n_{a_j} + 1$
25: **if** $r_{\mathbf{x}_t,a} = 1$ **then**
26: **for** $i = 1, ..., m$ **do**
27: $w_t \leftarrow \frac{p(a=\pi_i(\mathbf{x}_t)|\mathbf{x}_t)}{p(a|\mathbf{x}_t)}$
28: Update $\mu_{a,i}$ and $\sigma^2_{a,i}$ by w_t
29: $n_{a,i} \leftarrow n_{a,i} + 1$
30: **end for**
31: $\alpha_a \leftarrow \alpha_a + 1$
32: **else**
33: $\beta_a \leftarrow \beta_a + 1$
34: **end if**
35: **end for**

5. EVALUATION

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

5.1 Data Collections

5.1.1 Yahoo! Today News

Yahoo! Today News data set is collected by Yahoo! Today module and published by *Yahoo! Research Lab*[2]. The news articles 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 *Yahoo! 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 [6, 14, 15]. 10 million user visit events are used in this evaluation.

5.1.2 KDD Cup 2012 Online Advertising

KDD Cup Online Advertising 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 (or user visit), 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 features for this data set is 1,070,866. One issue of this data set is that the click information is extremely sparse due to the large pool of the ads. To alleviate this problem, we only select the top 50 ads that have the most impressions in the evaluation. The generated data set contains 9 million user visit events.

5.2 Evaluation Methods

The experiments on *Yahoo! Today News* data set is evaluated by the *Replayer* method [15], which provides an unbiased offline evaluation by utilizing the historical log. It has been shown that the CTR estimated by this *Replayer* approaches the real CTR of the deployed online system if the items in historical user visits are randomly and uniformly recommended [15]. However, for *KDD Cup* data set, the search ads in historical logs are not uniformly recommended. We hence evaluate this data set using a simulation method [6]. In the simulation method, we first train a logistic regression model for each ad using the entire data offline. Then, for each impression with context $\mathbf{x}$, the click of an ad is generated with a probability $(1+\exp(-\mathbf{w}^T\mathbf{x}))^{-1}$.

[2]http://webscope.sandbox.yahoo.com/catalog.php.
[3]http://www.kddcup2012.org/c/kddcup2012-track2.

Although the evaluated CTR is not the real CTR, it provides a methodology for comparing different policies in such high dimensional data.

5.3 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. 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. The base policies used for the ensemble contain multiple types of algorithms, including:

- `Random`: it randomly selects an arm to pull.
- `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) [14]: 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, and α is a predefined parameter for controlling the balance of exploration and exploitation.
- `Softmax` (τ) [3]: it randomly selects an arm a_i with probability $\frac{\exp(r_{a_i}/\tau)}{\sum_{a_j\in\mathcal{A}}\exp(r_{a_j}/\tau)}$, where r_{a_i} is the predicted reward for the arm a_i.
- `Epoch-greedy` [13]: it defines an epoch with length L, in which a one-step exploration is first performed, and the rest trials in the epoch are used for exploitation.
- `TS` (q_0) [6]: thompson sampling with logistic regression, which 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 parameter sampling, but not in the learning algorithm.

In the 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 in all policies (except for `Random`). Since the contextual bandit algorithms are online algorithms, *stochastic gradient ascent* is used as the learning algorithm [4]. Notice that the algorithms digest the data in an online manner, hence all the user visits in the data sets are used for the testing purpose.

In our problem setting, we utilize ensemble strategies to obtain a unified policy. To evaluate the effectiveness of the ensemble, we empirically compare the following ensemble algorithms:

- `HyperRandom`: it randomly selects a policy from the policy pool, and then performs the recommendation based on the selected policy.
- `HyperTS`: The one proposed in Section 4.1.
- `HyperTSFB`: The one proposed in Section 4.2.

5.4 Result Analysis

For each policy, we test its performance on the entire data to obtain the overall CTR. To emphasize the robustness of our proposed ensemble strategy, we also split the data into multiple time buckets, and evaluate how the policies perform on each individual time bucket.

5.4.1 On Overall CTR

For base policies, most of them are randomized except for `LinUCB`. For each trial, we randomly shuffle the pool of items to be recommended. Thus, the performance of `LinUCB` may vary in different runs. We run each policy 10 times, and calculate the mean, standard deviation, minimum and maximum of the overall CTR. Table 1 reports the results of *Yahoo! Today News* data and *KDD Cup* data. The mean values of the top 5 base policies are highlighted in **bold**, and the comparable mean values of ensemble strategies are emphasized by **bold***.

As depicted in the table, the performance of the base policies varies significantly with different parameter values. Except for `Random` (pure exploration) and ϵ-greedy(0.0) (pure exploitation), all the base policies take into account both exploration and exploitation. A common trend among these policies is that, with more exploitation, the algorithms can achieve better performance in terms of the overall CTR after a long run over the data, whereas the deviation is high. Comparatively, with more exploration, the CTR shrinks, but the deviation decreases. Therefore, the performance of these base policies highly depends on the parameter setting. If the parameter that controls the relative importance of exploration and exploitation is perfectly set, then the performance approaches to the optimal; otherwise, the performance is relatively poor. However in practice, it is often difficult to determine the optimal value for the input parameter of each policy, primarily due to the online learning process of the algorithms as well as the unknown data distribution for learning models.

Our proposed ensemble strategies can achieve comparable performance with the top ranked policies in terms of the overall CTR, by virtue of the bandit property of the hyper model. It is worthy to note that the two ensemble strategies have no parameter to choose. Intuitively, with more trials, the base policies with the bandit property can produce better results as there are more data used for learning. By employing the ensemble strategies that select base policies based on their corresponding overall CTR, we can certainly obtain a unified policy with acceptable CTR. From Table 1, we observe that: (1) The `HyperRandom` policy performs pure exploration on the base policies. This may work well at the beginning of the learning process; however, after a long run, it cannot obtain acceptable performance due to the randomness of the policy selection. (2) The bandit-based ensemble strategies, i.e., `HyperTS` and `HyperTSFB`, are able to achieve comparable performance with the top ranked base policies.

The advantages of the meta-bandit policies involve two aspects: (1) by exploring/exploiting multiple base policies that have different parameter settings, *the meta-bandit policies are able to absorb the merits of good policies, and hence produce robust results in terms of the overall CTR*; and (2) *there is no parameter setting required for meta-bandit policies.* `HyperTS` does not have the mechanism of sharing feedbacks among different policies, then for each base policy, the digested click traffic may not be sufficient to produce a high-quality learning model and an accurate CTR estimate.

Table 1: Relative CTR on the experimental data.

Algorithm	Yahoo! Dataset				KDD Dataset			
	mean	std	min	max	mean	std	min	max
Random	1.0000	0.0120	0.9706	1.0149	1.0000	0.0018	0.9972	1.0029
ϵ-greedy(0.0)	2.0404	0.0509	1.9527	2.1101	2.3761	0.1398	2.1687	2.6573
ϵ-greedy(0.01)	**2.0546**	0.0685	1.9267	2.1776	**2.6204**	0.0266	2.5572	2.6477
ϵ-greedy(0.1)	**2.0668**	0.0404	1.9811	2.1114	**2.5282**	0.0107	2.5136	2.5429
ϵ-greedy(0.3)	1.8001	0.0303	1.7454	1.8507	2.1974	0.0047	2.1892	2.2041
ϵ-greedy(0.5)	1.5265	0.0234	1.4866	1.5550	1.8511	0.0029	1.8470	1.8555
LinUCB(0.01)	1.8897	0.0561	1.7832	1.9645	2.3696	0.0885	2.1814	2.4318
LinUCB(0.1)	1.3450	0.0169	1.3215	1.3643	2.2158	0.0036	2.2109	2.2227
LinUCB(0.3)	1.1961	0.0076	1.1877	1.2118	1.9469	0.0046	1.9404	1.9560
Softmax(0.01)	1.9572	0.0572	1.8877	2.0578	**2.5435**	0.0102	2.5196	2.5586
Softmax(0.1)	1.1138	0.0133	1.0905	1.1308	1.1946	0.0025	1.1881	1.1971
Softmax(1.0)	1.0063	0.0119	0.9809	1.0199	1.0147	0.0012	1.0125	1.0166
Epoch-greedy(5)	1.9512	0.0378	1.8750	2.0237	2.3627	0.0064	2.3490	2.3721
Epoch-greedy(10)	**2.0517**	0.0841	1.8673	2.1752	**2.5278**	0.0075	2.5141	2.5378
Epoch-greedy(100)	**2.0473**	0.0742	1.9198	2.1616	**2.5794**	0.0645	2.4374	2.6379
Epoch-greedy(500)	**2.0473**	0.0325	1.9751	2.1022	2.5209	0.0853	2.3721	2.6176
TS(0.01)	1.2100	0.0121	1.1909	1.2238	2.0258	0.0021	2.0228	2.0296
TS(0.1)	1.1654	0.0074	1.1540	1.1801	1.9715	0.0039	1.9652	1.9794
TS(1.0)	1.1401	0.0112	1.1249	1.1603	1.6673	0.0024	1.6625	1.6707
TS(10.0)	0.8779	0.0990	0.6907	1.0322	1.2194	0.5090	0.4646	1.8784
TSNR(0.01)	1.2728	0.0184	1.2486	1.3060	2.0039	0.0025	2.0003	2.0072
TSNR(0.1)	1.2823	0.0108	1.2570	1.2956	2.0589	0.0025	2.0545	2.0621
TSNR(1.0)	1.3471	0.0152	1.3208	1.3741	2.2051	0.0019	2.2024	2.2089
TSNR(10.0)	1.8847	0.0389	1.8013	1.9270	2.4945	0.0031	2.4882	2.4993
HyperRandom	1.1856	0.0099	1.1702	1.2041	1.7379	0.0158	1.7115	1.7597
HyperTS	2.0095	0.0708	1.8719	2.1187	2.5175	0.1279	2.2364	2.6587
HyperTSFB	**2.1183***	0.0572	2.0115	2.1842	**2.6536***	0.0101	2.6390	2.6709

Without enough click traffic, the base policies that exhibit relatively poor performance at the beginning of the trails may have very limited opportunities to be explored/exploited in the subsequent trials, even though they are good policies if running solely. This is the primary reason that the performance of the HyperTS policy is inferior to the one of the HyperTSFB policy, as indicated in Table 1.

5.4.2 On CTR of Time Buckets

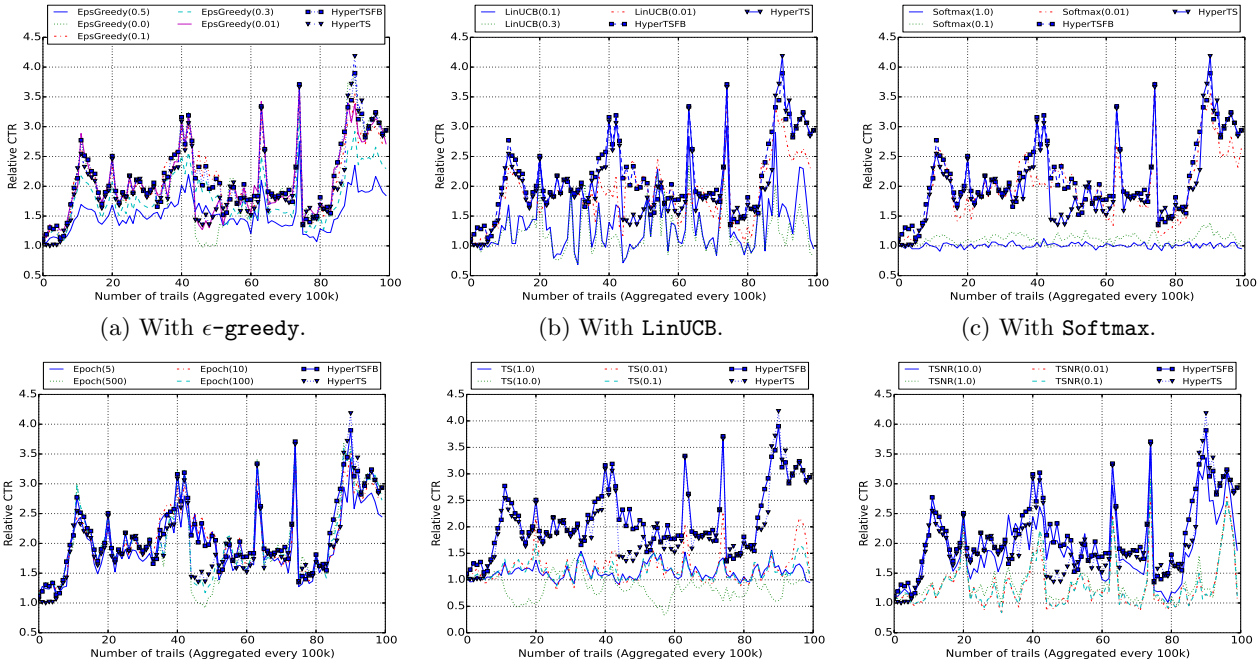
Besides the overall CTR of each policy, we also evaluate the CTR on individual time bucket. The CTR on each bucket is calculated by the clicks collected in that bucket. The entire *Yahoo! Today News* data is split into 100 time buckets, where each bucket has 100,000 impressions on news articles. The *KDD Cup* data set is split into 90 time buckets, with 100,000 impression for each bucket. All the user visit events are order by the time.

For the purpose of illustration, we compare the ensemble strategies, i.e., HyperTS and HyperTSFB, with each type of contextual bandit policies, including ϵ-greedy, LinUCB, Softmax, Epoch-greedy, TS and TSNR. At each time bucket, these policies are executed independently, and the relative CTR for each policy is calculated. The results for *Yahoo! Today News* data set and *KDD Cup* data set are reported in Figure 1 and 2, respectively. We can observe that the policy of HyperTSFB achieves consistent performance on both data sets. Although in some time buckets the relative CTR of HyperTSFB is slightly lower than the one of some specific base policies, its overall performance is quite robust compared with other baselines.

From Figure 1 we observe that the CTR curves of different policies have wide fluctuations. This is because the CTR estimated in *Yahoo! news* data is close to the real CTR in each bucket. The real CTR of an online recommender system usually varies over the time. For instance, popular news articles may become unpopular since the news is aging. User interests may change from daytime to nighttime. In contrast, the CTR curves for *KDD Cup* data, as described in Figure 2, are quite flat except the first few time buckets, and the reason is straightforward. The click of *KDD Cup* data is simulated by a group of logistic regression models. The time factor is not included in those models. Despite different characteristics of the two data sets, our proposed meta-bandit policy performs in a very robust way. This further demonstrates the generalization capability of our proposed method in dealing with different recommendation problems.

Another interesting phenomenon is that the CTR lift of HyperTSFB is significantly higher than the one of HyperTS at the first few time buckets on both *Yahoo! Today News* data and *KDD Cup* data. The reason here is straightforward: by sharing feedbacks among different policies, the data used for exploring/exploiting the policies become rich, and hence there are more data used for training the underlying learning model and estimating the performance of base policies. Therefore, at the initial time buckets, HyperTSFB outperforms HyperTS in terms of CTR.

To further demonstrate the robustness of our proposed methods, we consider to rank all the policies based on their CTR lift, and then examine if the result of HyperTSFB is in the top ranked list. Specifically in each time bucket, we rank



(a) With ϵ-greedy. (b) With LinUCB. (c) With Softmax.

(d) With Epoch-greedy. (e) With TS. (f) With TSNR.

Figure 1: Comparison on Yahoo! News Data.

the base and ensemble policies based on the CTR lift, and then count the number of times that a policy appears in the top@k ranked list. Next, we calculate the ratio of this count with the total number of buckets for each policy. Finally, we rank the policies based on their ranking ratios. For this evaluation, 4 best performed policies of *Yahoo! Today News* data and *KDD Cup* data are reported in Table 2 and 3, respectively, in which we consider the top@1, top@3 and top@5 results.

Table 2: CTR ranking on buckets for Yahoo! data.

top@1		top@3		top@5	
Policies	Ratio	Policies	Ratio	Policies	Ratio
HyperTSFB	88.89%	HyperTSFB	100.00%	HyperTSFB	100.00%
ϵ(0.01)	11.11%	ϵ(0.01)	97.78%	ϵ(0.01)	100.00%
-	-	Epoch(100)	78.89%	Epoch(100)	80.00%
-	-	Softmax(0.01)	16.67%	Softmax(0.01)	80.00%

Table 3: CTR ranking on buckets for KDD data.

top@1		top@3		top@5	
Policies	Ratio	Policies	Ratio	Policies	Ratio
ϵ(0.0)	18.00%	ϵ(0.0)	43.00%	HyperTSFB	72.00%
HyperTSFB	12.00%	Epoch(500)	43.00%	Epoch(500)	68.00%
Epoch(500)	12.00%	HyperTSFB	39.00%	ϵ(0.0)	61.00%
LinUCB(0.01)	11.00%	Epoch(100)	34.00%	Epoch(100)	57.00%

As observed in Table 2, our proposed policy HyperTSFB on *Yahoo! Today News* data always achieves the 1st place in the top@1, top@3 and top@5 results. Also in Table 3, HyperTSFB reaches the 2nd place of top@1, 3rd place of top@3, and 1st place of top@5. The results indicate that HyperTSFB is able to achieve promising performance in most time buckets. Such an observation further confirms the robust capability of our proposed policy in handling different online recommendation problems.

In addition, the performance of base policies with different parameter settings may vary significantly over different experimental data. From Table 2, we observe that ϵ-greedy(0.01) and Epoch(100) perform very well on *Yahoo! Today News* data, indicating that for this data set, the bandit policies can have achieve striking performance with a limited exploration. Comparatively in Table 3, ϵ-greedy(0.0) and Epoch(500) are able to produce promising results, meaning that higher CTR can be achieved on *KDD Cup* data based on the policies with much less exploration.

6. CONCLUDING REMARKS

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. A lot of contextual bandit policies have been proposed to tackle this dilemma; however, the prerequisite of the input parameters limits the predictive power of the policies. In real-world applications, these policies cannot be easily evaluated under different parameters as they may require too much web traffic and affect the profit of service providers.

In this work, we explore ensemble strategies of multiple contextual bandit policies to obtain robust predicted CTR. Specifically, we employ a meta-bandit paradigm that places a hyper bandit over the base bandits, to explicitly explore/exploit the relative importance of base bandits based on user feedbacks. The proposed approach does not have the restriction on the number of policies being involved, and can always obtain an acceptable CTR close the the optimal. Extensive empirical evaluation on two data sets demonstrates the efficacy of our proposed approach in terms of CTR.

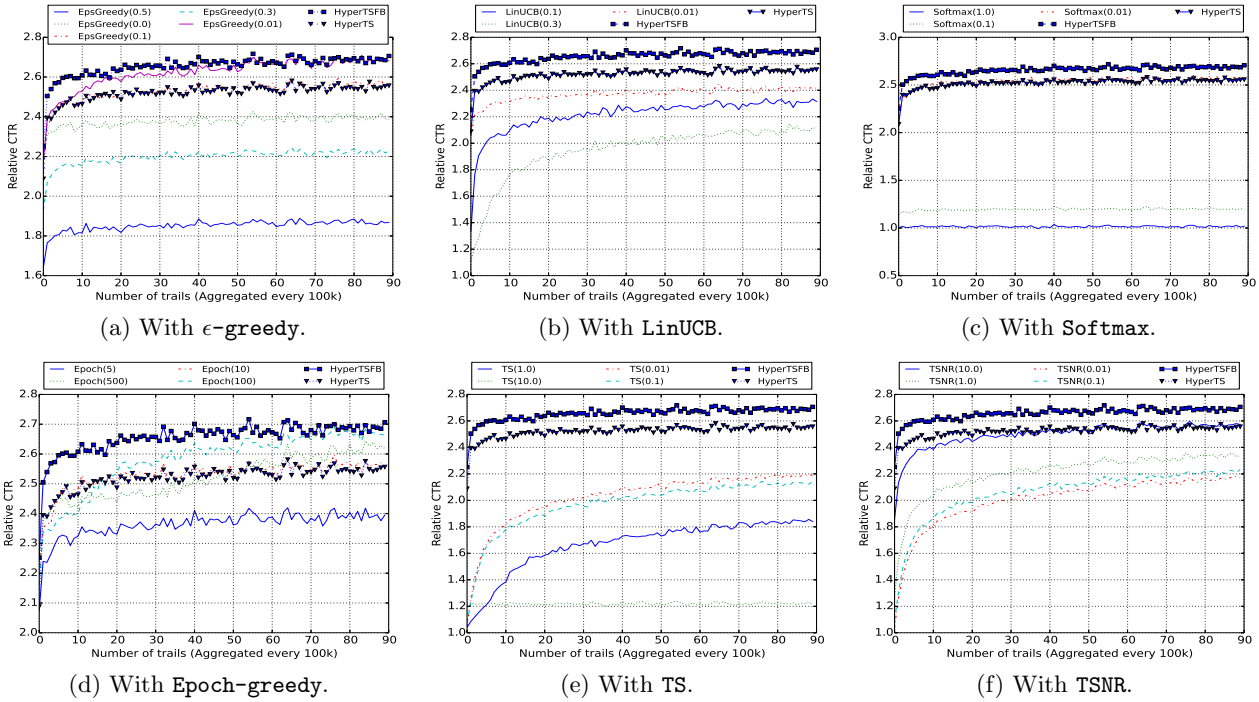
7. REFERENCES

[1] 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*, 2008.

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

[3] A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[4] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. 2006.



(a) With ϵ-`greedy`. (b) With `LinUCB`. (c) With `Softmax`.

(d) With `Epoch-greedy`. (e) With `TS`. (f) With `TSNR`.

Figure 2: Comparison on KDDCup Data.

[5] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski. A contextual-bandit algorithm for mobile context-aware recommender system. In *NIPS*, 2012.

[6] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *NIPS*, 2011.

[7] M. Gagliolo and J. Schmidhuber. Algorithm selection as a bandit problem with unbounded losses. In *Learning and Intelligent Optimization*, pages 82–96. Springer, 2010.

[8] C. Giraud-Carrier. Metalearning-a tutorial. In *ICMLA*, 2008.

[9] C. Hartland, N. Baskiotis, S. Gelly, M. Sebag, O. Teytaud, et al. Change point detection and meta-bandits for online learning in dynamic environments. *CAp*, 2007.

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

[11] M. Jahrer, A. Töscher, and R. Legenstein. Combining predictions for accurate recommender systems. In *SIGKDD*, 2010.

[12] Y. Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 2009.

[13] J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. *NIPS*, 2007.

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

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

[16] F. Maes, L. Wehenkel, and D. Ernst. Meta-learning of exploration/exploitation strategies: The multi-armed bandit case. In *Agents and Artificial Intelligence*, pages 100–115. 2013.

[17] A. Maurer and T. Jaakkola. Algorithmic stability and meta-learning. *JMLR*, 6(6), 2005.

[18] T. G. McKenzie, C.-S. Ferng, Y.-N. Chen, C.-L. Li, C.-H. Tsai, K.-W. Wu, Y.-H. Chang, C.-Y. Li, W.-S. Lin, S.-H. Yu, et al. Novel models and ensemble techniques to discriminate favorite items from unrated ones for personalized music recommendation. 2011.

[19] R. Polikar. Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE*, 6(3):21–45, 2006.

[20] A. Prodromidis, P. Chan, and S. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. *Advances in distributed and parallel knowledge discovery*, 3, 2000.

[21] J. B. Schafer, J. A. Konstan, and J. Riedl. Meta-recommendation systems: user-controlled integration of diverse recommendations. In *CIKM*, 2002.

[22] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, 2002.

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

[24] J. Seiler. *Meta learning in recommendation systems*. Master Thesis, 2013.

[25] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *RecSys handbook*. 2011.

[26] J. Sill, G. Takács, L. Mackey, and D. Lin. Feature-weighted linear stacking. *arXiv preprint arXiv:0911.0460*, 2009.

[27] C. Tekin and M. van der Schaar. Decentralized online big data classification-a bandit framework. *arXiv preprint arXiv:1308.4565*, 2013.

[28] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

[29] M. Tiemann and S. Pauws. Towards ensemble learning for hybrid music recommendation. In *RecSys*, 2007.

[30] M. Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *KI*. 2010.

[31] A. Töscher, M. Jahrer, and R. M. Bell. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, 2009.

[32] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *ECML*. 2005.

[33] K.-W. Wu, C.-S. Ferng, C.-H. Ho, A.-C. Liang, C.-H. Huang, W.-Y. Shen, J.-Y. Jiang, M.-H. Yang, T.-W. Lin, C.-P. Lee, et al. A two-stage ensemble of diverse models for advertisement ranking in kdd cup 2012. 2012.

[34] M. Wu. Collaborative filtering via ensembles of matrix factorizations. In *KDDCUP*, volume 2007, 2007.

[35] K. Yu, A. Schwaighofer, and V. Tresp. Collaborative ensemble learning: Combining collaborative and content-based information filtering via hierarchical bayes. In *UAI*, 2002.

[36] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *ICML*, 2004.