Towards the D-Optimal Online Experiment Design for Recommender Selection

Da Xu
Walmart Labs
Sunnyvale, California, USA
Da.Xu@walmartlabs.com

Chuanwei Ruan*
Walmart Labs
Sunnyvale, California, USA
RuanChuanwei@gmail.com

Evren Korpeoglu
Walmart Labs
Sunnyvale, California, USA
EKorpeoglu@walmart.com

Sushant Kumar
Walmart Labs
Sunnyvale, California, USA
SKumar4@walmartlabs.com

Kannan Achan
Walmart Labs
Sunnyvale, California, USA
KAchan@walmartlabs.com

ABSTRACT

Selecting the optimal recommender via online exploration-exploitation is catching increasing attention where the traditional A/B testing can be slow and costly, and offline evaluations are prone to the bias of history data. Finding the optimal online experiment is nontrivial since both the users and displayed recommendations carry contextual features that are informative to the reward. While the problem can be formalized via the lens of multi-armed bandits, the existing solutions are found less satisfactorily because the general methodologies do not account for the case-specific structures, particularly for the e-commerce recommendation we study. To fill in the gap, we leverage the *D-optimal design* from the classical statistics literature to achieve the maximum information gain during exploration, and reveal how it fits seamlessly with the modern infrastructure of online inference. To demonstrate the effectiveness of the optimal designs, we provide semi-synthetic simulation studies with published code and data for reproducibility purposes. We then use our deployment example on Walmart.com to fully illustrate the practical insights and effectiveness of the proposed methods.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; • **Computer systems organization** → *Real-time systems*; • **Mathematics of computing** → *Statistical paradigms*.

KEYWORDS

Recommender system; Multi-armed bandit; Exploration-exploitation; Optimal design; Deployment infrastructure

---

*The author is now with Instacart.

1 INTRODUCTION

Developing and testing recommenders to optimize business goals are among the primary focuses of e-commerce machine learning. A crucial discrepancy between the business and machine learning world is that the target metrics, such as gross merchandise value (GMV), are difficult to interpret as tangible learning objectives. While a handful of surrogate losses and evaluation metrics have been found with particular empirical success [20, 44], online experimentation is perhaps the only rule-of-thumb for testing a candidate recommender's real-world performance. In particular, there is a broad consensus on the various types of bias in the collected history data [11], which can cause the "feedback-loop effect" if the empirical metrics are used without correction [18]. Recently, there has been a surge of innovations in refining online A/B testings and correcting offline evaluation methods [18, 29, 47, 52]. However, they still fall short in specific applications where either the complexity of the problem outweighs their potential benefits, or their assumptions are not satisfied. Since our work discusses the e-commerce recommendations primarily, we assume the online shopping setting throughout the paper. On the A/B testing side, there is an increasing demand for interleaving the tested recommenders and targeted customers due to the growing interests in personalization [43]. The process of collecting enough observations and drawing inference with decent power is often slow and costly (in addition to the complication of defining the buckets in advance), since the number of combinations can grow exponentially. As for the recent advancements for offline A/B testing [18], even though certain unbiasedness and optimality results have been shown in theory, the real-world performance still depends on the fundamental causal assumptions [22], e.g. unconfounding, overlapping and identifiability, which are rarely fulfilled in practice [49]. We point out that the role of both online and offline testings are irreplaceable regardless of their drawbacks; however, the current issues motivate us to discover more efficient solutions which can better leverage the randomized design of traffic.

The production scenario that motivates our work is to choose from several candidate recommenders who have shown comparable performances in offline evaluation. By the segment analysis, we find each recommender more favorable to specific customer groups, but again the conclusion cannot be drawn entirely due to the exposure and selection bias in the history data. In other words, while it is safe to launch each candidate online, we still need randomized experiments to explore each candidates' real-world performance for different customer groups. We want to design the experiment by accounting for the customer features (e.g. their segmentation information) to minimize the cost of trying suboptimal recommenders on a customer group. Notice that our goal deviates from the traditional controlled experiments because we care more about minimizing the cost than drawing rigorous inference. In the sequel, we characterize our mission as a recommender-wise exploration-exploitation problem, a novel application to the best of our knowledge. Before we proceed, we illustrate the fundamental differences between our problem and learning the ensembles of recommenders [24]. The business metrics, such as GMV, are random quantities that depend on the recommended contents as well as the distributions that govern customers' decision-making. Even if we have access to those distributions, we never know in advance the conditional distribution given the recommended contents. Therefore, the problem can not be appropriately described by any fixed objective for learning the recommender ensembles.

In our case, the exploration-exploitation strategy can be viewed as a sequential game between the developer and the customers. In each round $t = 1, \ldots, n$, where the role of n will be made clear later, the developer chooses a recommender $a_t \in \{1, \ldots, k\}$ that produces the content c_t, e.g. top-k recommendations, according to the front-end request r_t, e.g. customer id, user features, page type, etc. Then the customer reveal the *reward* y_t such as click or not-click. The problem setting resembles that of the *multi-armed bandits (MAB)* by viewing each recommender as the *action (arm)*. The front-end request r_t, together with the recommended content $c_t = a_t(x_t)$, can be think of as the *context*. Obviously, the context is informative of the reward because the clicking will depend on how well the content matches the request. On the other hand, an (randomized) experiment design can be characterized by a distribution π over the candidate recommenders, i.e. $0 < \pi_t(a) < 1$, $\sum_{i=1}^{k} \pi_t(i) = 1$ for $i = 1, \ldots, n$. We point out that a formal difference between our setting and classical *contextual bandit* is that the context here depends on the candidate actions. Nevertheless, its impact becomes negligible if choosing the best set of contents is still equivalent to choosing the optimal action. Consequently, the goal of finding the optimal experimentation can be readily converted to optimizing π_t, which is aligned with the bandit problems. The intuition is that by optimizing π_t, we refine the estimation of the structures between context and reward, e.g. via supervised learning, at a low exploration cost.

The critical concern of doing exploration in e-commerce, perhaps more worrying than the other domains, is that irrelevant recommendations can severely harm user experience and stickiness, which directly relates to GMV. Therefore, it is essential to leverage the problem-specific information, both the contextual structures and prior knowledge, to further design the randomized strategy for

higher efficiency. We use the following toy example to illustrate our argument.

EXAMPLE 1. *Suppose that there are six items in total, and the front-end request consists of a uni-variate user feature $r_t \in \mathbb{R}$. The reward mechanism is given by the linear model:*

$$Y_t = \theta_1 \cdot I_1 \times X_t + \ldots + \theta_6 \cdot I_6 \times X_t;$$

where I_j is the indicator variable on whether item j is recommended. Consider the top-3 recommendation from four candidate recommenders as follow (in the format of one-hot encoding):

$$
\begin{aligned}
a_1(r_t) &= [1, 1, 1, 0, 0, 0]; \\
a_2(r_t) &= [0, 0, 0, 1, 1, 1]; \\
a_3(r_t) &= [0, 0, 1, 0, 1, 1]; \\
a_4(r_t) &= [0, 0, 0, 1, 1, 1].
\end{aligned}
\tag{1}
$$

If each recommender is explored with the probability, the role of a_1 is underrated since it is the only recommender that provides information about θ_1 and θ_2. Also, a_2 and a_4 give the same outputs, so their exploration probability should be discounted by half. Similarly, the information provided by S_3 can be recovered by S_1 and S_4 (or S_2) combined, so there is a linear dependency structure we may leverage.

The example is representative of the real-world scenario, where the one-hot encodings and user features may simply be replaced by the pre-trained embeddings. By far, we provide an intuitive understanding of the benefits from good online experiment designs. In Section 2, we introduce the notations and the formal background of bandit problems. We then summarize the relevant literature in Section 3. In Section 4, we present our optimal design methods and describe the corresponding online infrastructure. Both the simulation studies and real-world deployment analysis are provided in Section 5. We summarize the major contributions as follow.

- We provide a novel setting for online recommender selection via the lens of exploration-exploitation.
- We present an optimal experiment approach and describe the infrastructure and implementation.
- We provide both open-source simulation studies and real-world deployment results to illustrate the efficiency of the approaches studied.

2 BACKGROUND

We start by concluding our notations in Table 1. By convention, we use lower and upper-case letters to denote scalars and random variables, and bold-font lower and upper-case letters to denote vectors and matrices. We use $[k]$ as a shorthand for the set of: $\{1, 2, \ldots, k\}$. The randomized experiment strategy (policy) is a mapping from the collected data to the recommenders, and it should maximize the overall reward $\sum_{t=1}^{n} Y_t$. The interactive process of the online recommender selection can be described as follow.

1. The developer receives a front-end request $r_t \sim P_{\text{request}}$.

2. The developer computes the feature representations that combines the request and outputs from all candidate recommender: $\mathbf{X}_t := \left\{ \boldsymbol{\phi}(r_t, a_i(r_t)) \right\}_{i=1}^{k}$.

3. The developer chooses a recommender a_t according to the randomized experiment strategy $\pi(a|\mathbf{X}_t, \vec{h}_t)$.

4. The customer reveals the reward y_t.

In particular, the selected recommender a_t depends on the request, candidate outputs, as well as the history data:

$$a_t \sim \pi\left(a \,\middle|\, r_t, \{\boldsymbol{\phi}(r_t, a_i(r_t))\}_{i=1}^k, \vec{h}_t\right),$$

and the observation we collect at each round is given by:

$$\left(r_t, \{\boldsymbol{\phi}(r_t, a_i(r_t))\}_{i=1}^k, a_t, y_t\right).$$

We point out that compared with other bandit applications, the restriction on computation complexity per round is critical for real-world production. This is because the online selection experiment is essentially an additional layer on top of the candidate recommender systems, so the service will be called by tens of thousands of front-end requests per second. Consequently, the context-free exploration-exploitation methods, whose strategies focus on the cumulative rewards: $Q(a) = \sum_{j=1}^t y_j 1[a = j]$ and number of appearances: $N(a) = \sum_{j=1}^t 1[a = j]$ (assume up to round t) for $a = 1, \dots, k$, are quite computationally feasible, e.g.

- ϵ-**greedy**: explores with probability ϵ under the uniform exploration policy $\pi(a) = 1/k$, and selects $\arg\max_{a\in[k]} Q(a)$ otherwise (for exploitation);
- **UCB**: selects $\arg\max_{a\in[k]} Q(a) + CI(a)$, where $CI(a)$ characterizes the confidence interval of the action-specific reward $Q(a)$, and is given by: $\sqrt{(\log 1/\delta)/N(a)}$ for some pre-determined δ.

The more sophisticated **Thompson sampling** equips the sequential game with a Bayesian environment such that the developer:

- selects $\arg\max_{a\in[k]} \tilde{Q}(a)$, where $\tilde{Q}(a)$ is sampled from the posterior distribution $\text{Beta}(\alpha_t, \beta_t)$, and α_t and β_t combines the prior knowledge of average reward and the actual observed rewards.

For Thompson sampling, it is clear that the front-end computations can be simplified to calculating the uni-variate indices (Q, N, α, β). For MAB, taking account of the context often requires employing a parametric reward model: $y_a = f_{\boldsymbol{\theta}}(\boldsymbol{\phi}(r, a(r)))$, so during exploration, we may also update the model parameters $\boldsymbol{\theta}$ using

k, m, n	The number of candidate actions (recommenders); the number of top recommendations; the number of exploration-exploitation rounds. (may not known in advance).
R_t, A_t, Y_t	The frond-end request, action selected by the developer, and the reward at round t.
$\vec{h}_t, \pi(\cdot \mid \cdot)$	The history data collected until round t, and the randomized strategy (policy) that maps the contexts and history data to a probability measure on the action space.
$\mathcal{I}, a_i(\cdot)$	The whole set of items and the ith candidate recommender, with $a_i(\cdot) \in \mathcal{I}^{\otimes m}$.
$\boldsymbol{\phi}(r_t, a_i(r_t))$	The (engineered or pre-trained) feature representation in $\mathbb{R}^d$, specifically for the tth-round front-end request and the output contents for the ith recommender.

Table 1: A summary of the notations. By tradition, we use uppercase letters to denote random variables, and the corresponding lowercase letters as observations.

the collected data. Suppose we have an *optimization oracle* that returns $\hat{\boldsymbol{\theta}}$ by fitting the empirical observations, then all the above algorithms can be converted to the context-aware setting, e.g.

- **epoch-greedy**: explores under $\pi(a) = 1/k$ for a epoch, and selects $\arg\max_{a\in[k]} \hat{y}_a := f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\phi}(r, a(r)))$ otherwise;
- **LinUCB**: by assuming the reward model is linear, it selects $\arg\max_{a\in[k]} \hat{y}_a + CI(a)$ where $CI(a)$ characterizes the confidence of the linear model's estimation;
- **Thompson sampling**: samples $\hat{\boldsymbol{\theta}}$ from the reward-model-specific posterior distribution of $\boldsymbol{\theta}$, and selects $\arg\max_{a\in[k]} \hat{y}_a$.

We point out that the per-round model parameter update via the optimization oracle, which often involves expensive real-time computations, is impractical for most online services. Therefore, we adopt the stage-wise setting that divides exploration and exploitation (similar to epoch-greedy). The design of π thus becomes very challenging since we may not have access to the most updated $\hat{\boldsymbol{\theta}}$. Therefore, it is important to take advantage of the structure of $f_{\boldsymbol{\theta}}(\cdot)$, which motivates us to connect our problem with the optimal design methods in the classical statistics literature.

3 RELATED WORK

We briefly discuss the existing bandit algorithms and explain their implications to our problem. Depending on how we perceive the environment, the solutions can be categorized into the frequentist and Bayesian setting. On the frequentist side, the reward model plays an important part in designing algorithms that connect to the more general *expert advice* framework [12]. The *EXP4* and its variants are known as the theoretically optimal algorithms for the expert advice framework if the environment is adversarial [4, 6, 35]. However, customers often have a neutral attitude for recommendations, so it is unnecessary to assume adversarialness. In a neutral environment, the *LinUCB* algorithm and its variants have been shown highly effective [4, 12]. In particular, when the contexts are viewed as i.i.d samples, several regret-optimal variants of LinUCB have been proposed [2, 15]. Nevertheless, those solutions all require real-time model updates (via the optimization oracle), and are thus impractical as we discussed earlier.

On the other hand, several suboptimal algorithms that follow the *explore-then-commit* framework can be made computationally feasible for large-scale applications [38]. The key idea is to divide exploration and exploitation into different stages, like the *epoch-greedy* and *phased exploration* algorithms [1, 28, 39]. The model training and parameter updates only consume the back-end resources dedicated for exploitation, and the majority of front-end resources still take care of the model inference and exploration. Therefore, the stage-wise approach appeals to our online recommender selection problem, and it resolves certain infrastructural considerations that we explain later in Section 4.

On the Bayesian side, the most widely-acknowledge algorithms belong to the *Thompson sampling*, which has a long history and fruitful theoretical results [10, 40, 41, 45]. When applied to contextual bandit problems, the original Thompson sampling also requires per-round parameter update for the reward model [10]. Nevertheless, the flexibility of the Bayesian setting allows converting Thompson sampling to the stage-wise setting as well.

In terms of real-world applications, online advertisement and news recommendation [3, 30, 31] are perhaps the two major domains where contextual bandits are investigated. Bandits have also been applied to our related problems such as item-level recommendation [26, 32, 53] and recommender ensemble [9]. To the best of our knowledge, none of the previous work studies contextual bandit for the recommender selection.

4 METHODOLOGIES

As we discussed in the literature review, the stage-wise (phased) exploration and exploitation appeals to our problem because of their computation advantage and deployment flexibility. To apply the stage-wise exploration-exploitation to online recommender selection, we describe a general framework in Algorithm 1.

Algorithm 1: Stage-wise exploration and exploitation

Input: Reward model $f_\theta(\cdot)$; the restart criteria; the initialized history data $\vec{h}_t$.

1 **while** *total rounds* $\leq n$ **do**
2   **if** *restart criteria is satisfied* **then**
3 | Reset $\vec{h}_t$;
4   **end**
5   Play n_1 rounds of random exploration, for instance: $\pi(a \mid \cdot) = \frac{1}{k}$, and collect observation to $\vec{h}_t$;
6   Find the optimal $\hat{\theta}$ based on $\vec{h}_t$ (e.. via empirical-risk minimization);
7   Play n_2 rounds of exploitation with: $a_t = \arg\max_a f_{\hat{\theta}}(r_t, a(r_t))$;
8 **end**

The algorithm is deployment-friendly because Step 5 only involves front-end and cache operation, Step 6 is essentially a batch-wise training on the back-end, and Step 7 applies directly to the standard front-end inference. Hence, the algorithm requires little modification from the existing infrastructure that supports real-time mdoel inference. Several additional advantages of the stage-wise algorithms include:

- the number of of exploration and exploitation rounds, which decides the proportion of traffic for each task, can be adaptively adjusted by the resource availability and response time service level agreements;
- the non-stationary environment, which are often detected via the hypothesis testing methods as described in [5, 8, 33], can be handled by setting the restart criteria accordingly.

4.1 Optimal designs for exploration

This section is dedicated to improving the efficiency of exploration in Step 5. Throughout this paper, we emphasize the importance of leveraging the case-specific structures to minimize the number of exploration steps it may take to collect equal information for estimating θ. Recall from Example 1 that one particular structure is the relation among the recommended contents, whose role can be thought of as the design matrix in linear regression. Towards that end, our goal is aligned with the optimal design in the classical

statistics literature [37], since both tasks aim at optimizing how the design matrix is constructed. Following the previous buildup, the reward model has one of the following forms:

$$y_t = \begin{cases} \theta^\top \phi(r_t, a(r_t)), & \text{linear model} \\ f_\theta(\phi(r_t, a(r_t))), & \text{for some nonlinear } f_\theta(\cdot), \end{cases} \quad (2)$$

We start with the frequentist setting, i.e. θ do not admit a prior distribution. In each round t, we try to find a optimal design $\pi(\cdot \mid \cdot)$ such that the action sampled from π leads to a maximum information for estimating θ. For statistical estimators, the Fisher information is a key quantity for evaluating the amount of information in the observations. For the general f_θ, the Fisher information under (2) is given by:

$$M(\pi) = \sum_{a_i=1}^{k} \pi(a_i) \nabla_\theta f_\theta(\phi(r_t, a_i(r_t))) \cdot \nabla_\theta f_\theta(\phi(r_t, a_i(r_t)))^\top, \quad (3)$$

where $\pi(a_i)$ is a shorthand for the designed policy. For the linear reward model, the Fisher information is simplified to:

$$M(\pi) = \sum_{a_i=1}^{k} \pi(a_i) \phi(r_t, a_i(r_t)) \cdot \phi(r_t, a_i(r_t))^\top. \quad (4)$$

To understand the role Fisher information in evaluating the underlying *uncertainty* of a model, according to the textbook derivations for linear regression, we have:

- $\text{var}(\hat{\theta}) \propto M(\pi)^{-1}$;
- the prediction variance for $\phi_i := \phi(r_t, a_i(r_t))$ is given by $\text{var}(\hat{y}_i) \propto \phi_i M(\pi)^{-1} \phi_i^\top$.

Therefore, the goal of optimal online experiment design can be explained as minimizing the uncertainty in the reward model, either for parameter estimation or prediction. In statistics, a *D-optimal design* minimizes $\det |M(\pi)^{-1}|$ from the perspective of estimation variance , and the *G-optimal design* minimize $\max_{i \in \{1,\ldots,k\}} \phi_i M(\pi)^{-1} \phi_i^\top$ from the perspective of prediction variance. A celebrated result states the equivalence between D-optimal and G-optimal designs.

THEOREM 1 (KIEFER-WOLFOWITZ [27]). *For a optimal design π^*, the following statements are equivalent:*

- $\pi^* = \max_\pi \log \det |M(\pi)|$;
- π^* *is D-optimal;*
- π^* *is G-optimal.*

Theorem 1 suggests that we use convex optimization to find the optimal design for both parameter estimation and prediction:

$$\max_\pi \log \det |M(\pi)| \text{ s.t. } \sum_{a_i=1}^{k} \pi(a_i) = 1.$$

However, a drawback of the above formulation is that it does not involve the observations collected in the previous exploration rounds. Also, the optimization problem does not apply to the Bayesian setting if we wish to use Thompson sampling. Luckily, we find that optimal design for the Bayesian setting has a nice connection to the above problem, and it also leads to a straightforward solution that utilizes the history data as a prior for the optimal design.

We still assume the linear reward setting, and the prior for θ is given by $\theta \sim N(0, \mathbf{R})$ where $\mathbf{R}$ is the covariance matrix. Unlike in the frequentist setting, the Bayesian design focus on the design

optimality in terms of certain utility function $U(\pi)$. A common choice is the expected gain in *Shannon information,* or equivalently, the Kullback-Leibler divergence between the prior and posterior distribution of θ. The intuition is that the larger the divergence, the more information there is in the observations. Let y be the hypothetical rewards for $\phi(r_t, a_1(r_t)), \ldots, \phi(r_t, a_k(r_t))$. Then the gain in Shannon information is given by:

$$
\begin{aligned}
U(\pi) &= \int \log p(\theta|y, \pi) p(y, \theta|\pi) \mathrm{d}\theta \mathrm{d}y \\
&= C + \frac{1}{2} \log \det \left| M(\pi) + \mathbf{R}^{-1} \right|,
\end{aligned}
\tag{5}
$$

where C is a constant. Therefore, maximizing $U(\pi)$ is equivalent to maximizing $\log \det \left| M(\pi) + \mathbf{R}^{-1} \right|$.

Compared with the objective for the frequentist setting, there is now an additive $\mathbf{R}$ term inside the determinant. Notice that $\mathbf{R}$ is the convariance of the prior, so given the previous history data, we can simply plug in the empirical estimation of $\mathbf{R}$. In particular, let $\vec{\phi}_{t-1}$ be the collection of feature vectors from the previous exploration rounds: $\left[\phi(x_1, a_1(x_1)), \ldots, \phi(x_{t-1}, a_{t-1}(x_{t-1})) \right]$. Then $\mathbf{R}$ is simply estimated by $\left(\vec{\phi}_{t-1} \vec{\phi}_{t-1}^{\mathsf{T}} \right)^{-1}$. Therefore, the objective for Bayesian optimal design, after integrating the prior from the history data, is given by:

$$
\text{maximize}_\pi \log \det \left| M(\pi) + \lambda \phi_{t-1} \phi_{t-1}^{\mathsf{T}} \right|, \text{ s.t. } \sum_{a_i=1}^{k} \pi(a_i) = 1, \quad (6)
$$

where we introduce the hyper parameter λ to control influence of the history data. We refer the interested readers to [13, 16, 25, 34] for the historic development of this topic.

Moving beyond the linear setting, the results stated in the Kiefer-Wolfowitz theorem also holds for nonlinear reward model [46]. Unfortunately, it is very challenging to find the exact optimal design when the reward model is nonlinear [51]. The difficulty lies in the fact that $\nabla_\theta f_\theta(\phi(r_t, a(r_t)))$ now depends on θ, so the Fisher information (4) is also a function of the unknown θ. One solution which we find computationally feasible is to consider a local linearization of $f_\theta(\cdot)$ using the Taylor expansion:

$$
\begin{aligned}
f_\theta(\phi(r_t, a(r_t))) \approx f_{\theta_0}(\phi(r_t, a(r_t))) + \\
\nabla_\theta f_\theta(\phi(r_t, a(r_t)))|_{\theta=\theta_0}(\theta - \theta_0),
\end{aligned}
\tag{7}
$$

where θ_0 is some local approximation. In this way, we have:

$$
\nabla_\theta f_\theta \approx \nabla_\theta f_\theta(\phi(r_t, a(r_t)))|_{\theta=\theta_0}, \tag{8}
$$

and the local Fisher information will be given by $M(\pi; \theta_0)$ according to (4). The effectiveness of local linearization completely depends on the choice of θ_0. Using the data gathered from previous exploration rounds is a reasonable way to estimate θ_0. We plug in θ_0 to obtain the optimal designs for the following exploration rounds:

$$
\text{maximize}_\pi \log \det \left| M(\pi; \theta_0) \right| \text{ s.t. } \sum_{a_i=1}^{k} \pi(a_i) = 1. \tag{9}
$$

We do not study the Bayesian optimal design under the nonlinear setting because even the linearization trick will be complicated. Moreover, by the way we construct θ_0 above, we already pass a certain amount of prior information to the design.

REMARK 1. *Before we proceed, we briefly discuss what to expect from the optimal design in theory. Optimizing the* $\log \det |\cdot|$ *objective is essentially finding the minimum-volume ellipsoid, also known as the* John ellipsoid *[19]. According to the previous results from the geometric studies, using the proposed optimal design will do no worse than the uniform exploration if the reward model is misspecified. Also, we can expect an average* $\sqrt{d}$ *improvement in the frequentist's linear reward setting [7], which means it only takes* $o(1/\sqrt{d})$ *of the previous exploration steps to estimate* θ *to the same precision.*

4.2 Algorithms

In this section, we introduce an efficient algorithm to solve the optimal designs in (9) and (6). We then couple the optimal designs to the stage-wise exploration-exploitation algorithms. The infrastructure for our real-world production is also discussed. We have shown earlier that finding the optimal design requires solving a convex optimization programming. Since the problem is often of moderate size as we do not expect the number of recommenders k to be large, we find the *Frank-Wolfe* algorithm highly efficient [17, 23]. We outline the solution for the most general non-linear reward case in Algorithm 2. The solutions for the other scenarios are included as special cases, e.g. by replacing $M(\pi)$ with $M(\pi) + \mathbf{R}^{-1}$ for the Bayesian setting.

Algorithm 2: The optimal design solver

Input: A subroutine for computing the $M(\pi; \theta_0)$ in (4) or (8); the estimation $\theta_0 = \hat{\theta}$ and $\eta_a := \nabla_\theta f_\theta(\phi(r_t, a(r_t)))|_{\theta=\theta_0}$; the convergence criteria.

1 Initialize π^{old}: $\pi^{\text{old}}(a) = \frac{1}{k}$, $a = 1, \ldots, k$;
2 **while** *convergence criteria not met* **do**
3   find $\tilde{a} = \arg\max_a \eta_a^{\mathsf{T}} M(\pi^{\text{old}}; \theta_0)^{-1} \eta_a$;
4   compute $\lambda_a = \dfrac{\eta_{\tilde{a}}^{\mathsf{T}} M(\pi^{\text{old}}; \theta_0)^{-1} \eta_{\tilde{a}} / d - 1}{\eta_a^{\mathsf{T}} M(\pi^{\text{old}}; \theta_0)^{-1} \eta_a - 1}$;
5   **for** $a = 1, \ldots, k$ **do**
6        $\pi^{\text{new}}(a) = (1 - \lambda_a) \pi^{\text{old}}(a) + \lambda_a 1[a = \tilde{a}]$;
7   **end**
8      $\pi^{\text{old}} = \pi^{\text{new}}$
9 **end**

Referring to the standard analysis of Frank-Wolfe algorithm [23], we show that the it takes the solver at most $O(d \log \log k + d/\epsilon)$ updates to achieve a multiplicative $(1 + \epsilon)$ optimal solution. Each update has an $O(kd^3)$ computation complexity, but d is usually small in practice (e.g. $d = 6$ in Example 1), which we will illustrate with more detail in Section 5.

By treating the optimal design solver as a subroutine, we now present the complete picture of the stage-wise exploration-exploitation with optimal design. To avoid unnecessary repetitions, we describe the algorithms for nonlinear reward model under frequentist setting (Algorithm 3), and for linear reward model under the Thompson sampling. They include the other scenarios as special cases.

To adapt the optimal design to the Bayesian setting, we only need to make a few changes to the above algorithm:

Algorithm 3: Stage-wise exploration-exploitation with optimal design.

Input: Reward model $f_{\boldsymbol{\theta}}(\cdot)$; restart criteria; initialize
history data $\vec{h}_t$; optimal design solver; initialize $\hat{\boldsymbol{\theta}}$.

1 while *total rounds* $\leq n$ **do**

2   **if** *restart criteria is satisfied* **then**

3 | Reset $\vec{h}_t$;

4   **end**

5   Compute the optimal exploration policy π^* using the
optimal design solver under $\boldsymbol{\theta}_0 = \hat{\boldsymbol{\theta}}$; Play n_1 rounds of
exploration under π^* and collect observation to $\vec{h}_t$;

6   Optimize and update $\hat{\boldsymbol{\theta}}$ based on $\vec{h}_t$;

7   Play n_2 rounds of exploitation with:
     $a_t = \arg\max_{a\in[k]} f_{\hat{\boldsymbol{\theta}}}\big(r_t, a(r_t)\big)$;

8 end

- the optimal design solver is now specified for solving (6);
- instead of optimizing $\hat{\theta}$ via the empirical-risk minimization, we update the posterior of θ using the history data $\vec{h}_t$;
- in each exploitation round, we execute Algorithm 4 instead.

Algorithm 4: Optimal design for Thompson sampling at exploitation rounds.

---

1 Compute the posterior distribution $P_{\boldsymbol{\theta}}$ according to the
prior and collected data ;

2 Sample $\hat{\boldsymbol{\theta}}$ from $P_{\boldsymbol{\theta}}$;

3 Select $a_t = \arg\max_a \hat{\boldsymbol{\theta}}^\top \boldsymbol{\phi}\big(r_t, a(r_t)\big)$;

**4** Collect the observation to $\vec{h}_t$;

For Thompson sampling, the computation complexity of exploration is the same as Algorithm 3. On the other hand, even with a conjugate prior distribution, the Bayesian linear regression has an unfriendly complexity for the posterior computations. Nevertheless, under our stage-wise setup, the heavy lifting can be done at the back-end in a batch-wise fashion, so the delay will not be significant. In our simulation studies, we observe comparable performances from Algorithm 3 and 4. Nevertheless, each algorithm may experience specific tradeoff in the stage-wise setting, and we leave it to the future work to characterize their behaviors rigorously.

4.3 Deployment Infrastructure

In the ideal setting, the online recommender selection can be viewed as another service layer, which we refer to as the *system bandit service*, on top of the model service infrastructure.

An overview of the concept is provided in Figure 1. The system bandit module takes the request (which contains the relevant context), and the wrapped recommendation models. When the service is triggered, depending on the instruction from *request distributor* (to explore or exploit), the module either queries the pre-computed reward, finds the best model and outputs its content, or run the optimal design solver using the pre-computed quantities and choose a recommender to explore. The request distributor is essential for
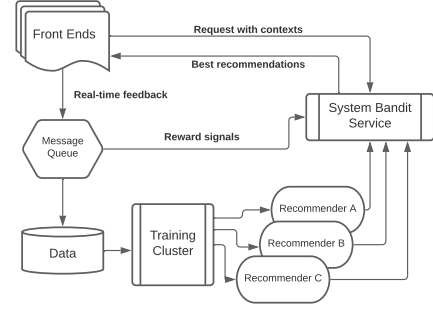

Figure 1: High-level overview of the system bandit service.

keeping the resource availability and response time agreements, since the optimal-design computations can cause stress during the peak time. Also, we initiate the scoring (model inference) for the candidate recommenders in parallel to reduce the latency whenever there are spare resources. The pre-computations occur in the back-end training clusters, and their results (updated parameters, posterior of parameters, prior distributions) are stored in such as the mega cache for the front end.

The logging system is another crucial component which maintains storage of the past reward signals, contexts, policy value, etc. The logging system works interactively with the training cluster to run the scheduled job for pre-computation. Another detail is that the rewards are often not immediately available, e.g. for the conversion rate, so we set up an *event stream* to collect the data. The system bandit service listens to the event streams and determines the rewards after each recommendation.

For our deployment, we treat the system bandit serves as a middleware between the online and offline service. The details are presented in Figure 2, where we put together the relevant components from the above discussion. It is not unusual these days to leverage the "near-line computation", and our approach takes the full advantage of the current infrastructure to support the optimal online experiment design for recommender selection.
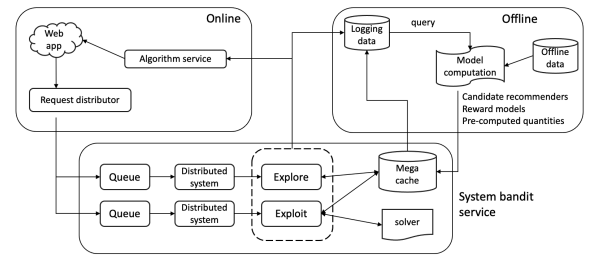
Figure 2: The deployment details of the optimal online experiment design for recommender selection.

5 EXPERIMENTS

We first provide simulation studies to examine the effectiveness of the proposed optimal design approaches. We then discuss the relevant testing performance on *Walmart.com*.

5.1 Simulation

For the illustration and reproducibility purposes, we implement the proposed online recommender selection under a semi-synthetic setting with a benchmark movie recommendation data. To fully reflect the exploration-exploitation dilemma in real-world production, we convert the benchmark dataset to an online setting such that it mimics the interactive process between the recommender and user behavior. A similar setting was also found in [9] that studies the non-contextual bandits as model ensemble methods, with which we also compare in our experiments. We consider the linear reward model setting for our simulation[1].

Data-generating mechanism. In the beginning stage, 10% of the full data is selected as the training data to fit the candidate recommendation models, and the rest of the data is treated as the testing set which generates the interaction data adaptively. The procedure can be described as follow. In each epoch, we recommend one item to each user. If the item has received a non-zero rating from that particular user in the testing data, we move it to the training data and endow it with a positive label if the rating is high, e.g. ≥ 3 under the five-point scale. Otherwise, we add the item to the rejection list and will not recommend it to this user again. After each epoch, we retrain the candidate models with both the past and the newly collected data. Similar to [9], we also use the **cumulative recall** as the performance metric, which is the ratio of the total number of successful recommendations (up to the current epoch) againt the total number of positive rating in the testing data. The reported results are averaged over ten runs.

Dataset. We use the *MoiveLens 1M* [2] dataset which consists of the ratings from 6,040 users for 3,706 movies. Each user rates the movies from zero to five. The movie ratings are binarized to $\{0, 1\}$, i.e. ≥ 2.5 or < 2.5, and we use the metadata of movies and users as the contextual information for the reward model. In particular, we perform the one-hot transformation for the categorical data to obtain the feature mappings $\boldsymbol{\phi}(\cdot)$. For text features such as movie title, we train a word embedding model [36] with 50 dimensions. The final representation is obtained by concatenating all the one-hot encoding and embedding.
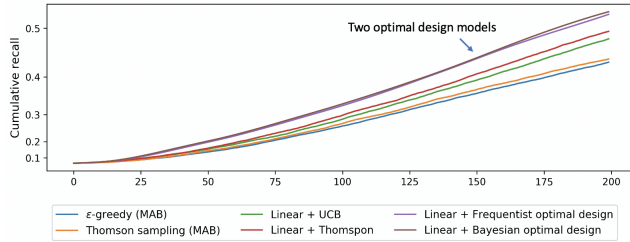
Figure 3: The comparisons of the cumulative recall (reward) per epoch for different bandit algorithms.

Candidate recommenders. We employ the four classical recommendation models: user-based collaborative filtering (CF) [54], item-based CF [42], popularity-based recommendation, and a matrix factorization model [21]. To train the candidate recommenders

during our simulations, we further split the 10% initial training data into equal-sized training and validation dataset, for grid-searching the best hyperparameters. The validation is conducted by running the same generation mechanism for 20 epochs, and examine the performance for the last epoch. For the user-based collaborative filtering, we set the number of nearest neighbors as 30. For item-based collaborative filtering, we compute the cosine similarity using the vector representations of movies. For relative item popularity model, the ranking is determined by the popularity of movies compared with the most-rated movies. For matrix factorization model, we adopt the same setting from [9].

Baselines. To elaborate the performance of the proposed methods, we employ the widely-acknowledged exploration-exploitation algorithms as the baselines:

- The multi-armed bandit (MAB) algorithm without context: ϵ**-greedy** and **Thompson sampling**.
- Contextual bandit with the exploration conducted in the LinUCB fashion (**Linear+UCB**) and Thompson sampling fashion (**Linear+Thompson**).

We denote our algorithms by the **Linear+Frequentis optimal design** and the **Linear+Bayesian optimal design**.

Ablation studies

We conduct ablation studies with respect to the contexts and the optimal design component to show the effectiveness of the proposed algorithms. Firstly, we experiment on removing the user context information. Secondly, we experiment with our algorithm without using the optimal designs.

Results. The results on cumulative recall per epoch are provided in Figure 3. It is evident that as the proposed algorithm with optimal design outperforms the other bandit algorithms by significant margins. In general, even though ϵ-greedy gives the worst performance, the fact that it is improving over the epochs suggests the validity of our simulation setup. The Thompson sampling under MAB performs better than ϵ-greedy, which is expected. The usefulness of context in the simulation is suggested by the slightly better performances from Linear+UCB and Linear+Thompson. However, they are outperformed by our proposed methods by significant margins, which suggests the advantage of leveraging the optimal design in the exploration phase. Finally, we observe that among the optimal design methods, the Bayesian setting gives a slightly better performance, which may suggest the usefulness of the extra steps in Algorithm 4.
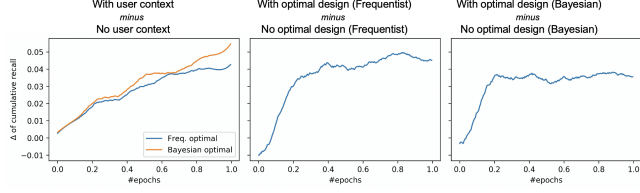
The results for the ablation studies are provided in Figure 4. The left-most plot shows the improvements from including contexts for bandit algorithms, and suggests that our approaches are indeed capturing and leveraging the signals of the user context. In the middle and right-most plots, we observe the clear advantage of conducting the optimal design, specially in the beginning phases of exploration, as the methods with optimal design outperforms their counterparts. We conjecture that this is because the optimal designs aim at maximizing the information for the limited options, which is more helpful when the majority of options have not been explored such as in the beginning stage of the simulation.

Finally, we present a case study to fully illustrate the effectiveness of the optimal design, which is shown in Figure 5. It appears that

---

[1]https://github.com/StatsDLMathsRecomSys/D-optimal-recommender-selection.
[2]https://grouplens.org/datasets/movielens/1m/

Figure 4: The difference in cumulative recall for system bandit under different settings. The left figure shows difference in performance between using and not using user context, under the frequentist and Bayesian setting, respectively. The other two figures compare using and not using the optimal design (uniform selection) in the exploration stages, both using the linear reward model.

in our simulation studies, the matrix factorization and popularity-based recommendation are found to be more effective. With the optimal design, the traffic concentrates more quickly to the two promising candidate recommenders than without the optimal design. The observations are in accordance with our previous conjecture that optimal design gives the algorithms more advantage in the beginning phases of explorations.
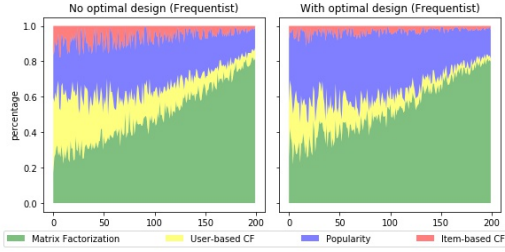
Figure 5: The percentage of traffic routed to each candidate recommender with and without using the optimal design under the frequentist setting.

5.2 Deployment analysis

We deployed our online recommender selection with optimal design to the similar-item recommendation of grocery items on Walmart.com. A webpage snapshot is provided in Figure 6, where the recommendation appears on the item pages. The baseline model for the similar-item recommendation is described in our previous work of [48], and we experiment with three enhanced models that adjust the original recommendations based on the *brand affinity*, *price affinity* and *flavor affinity*. We omit the details of each enhanced model since they are less relevant. The reward model leverages the item and user representations also
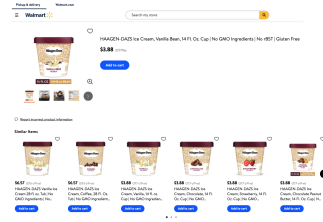

Figure 6: Item-page similar-item recommendation for groceries on Walmart.com.

described in our previous work. Specifically, the item embeddings are obtained from the Product Knowledge Graph embedding [50], and the user embeddings are constructed via the temporal user-item graph embedding [14]. We adopt the frequentist setting where the reward is linear function of ⟨item emb, user emb⟩, plus some user and item contextual features:

$$\theta_0 + \theta_1 \langle \mathbf{z}_u, \mathbf{z}_{I_1} \rangle + \ldots + \theta_m \langle \mathbf{z}_u, \mathbf{z}_{I_m} \rangle + \boldsymbol{\theta}^\top [\text{user feats, items feats}],$$

and $\mathbf{z}_u$ and $\mathbf{z}_I$ are the user and item embeddings.
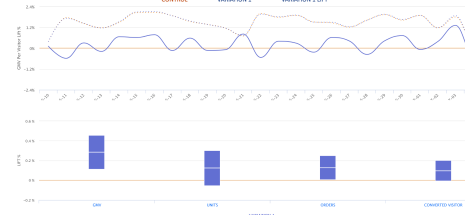
Figure 7: The testing results for the proposed stage-wise exploration-exploitation with optimal design.

We conduct a posthoc analysis by examining the proportion of traffic directed to the frequent and infrequent user groups by the online recommender selection system (Figure 8). Interestingly, we observe different patterns where the brand and flavor-adjusted models serve the frequent customers more often, and the unadjusted baseline and price-adjusted model get more appearances for the infrequent customers. The results indicate that our online selection approach is actively exploring and exploiting the user-item features that eventually benefits the online performance. The simulation studies, on the other hand, reveal the superiority over the standard exploration-exploitation methods.
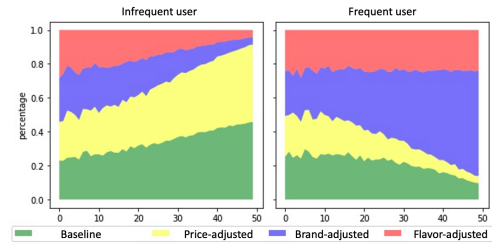
Figure 8: The analysis on the proportion of traffic directed to the frequent and infrequent customer.

6 DISCUSSION

We study optimal experiment design for the critical online recommender selection. We propose a practical solution that optimizes the standard exploration-exploitation design and shows its effectiveness using simulation and real-world deployment results.

REFERENCES

[1] Yasin Abbasi-Yadkori, András Antos, and Csaba Szepesvári. 2009. Forced-exploration based algorithms for playing in stochastic linear bandits. In *COLT Workshop on On-line Learning with Limited Feedback*, Vol. 92. 236.

[2] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*. 1638–1646.

[3] Shipra Agrawal and Navin Goyal. 2012. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*. 39–1.

[4] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multiarmed bandit problem. *SIAM journal on computing* 32, 1 (2002), 48–77.

[5] Peter Auer and Chao-Kai Chiang. 2016. An algorithm with nearly optimal pseudo-regret for both stochastic and adversarial bandits. In *Conference on Learning Theory*. 116–120.

[6] Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert Schapire. 2011. Contextual bandit algorithms with supervised learning guarantees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 19–26.

[7] Sébastien Bubeck, Nicolo Cesa-Bianchi, and Sham M Kakade. 2012. Towards minimax policies for online linear optimization with bandit feedback. In *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, 41–1.

[8] Sébastien Bubeck and Aleksandrs Slivkins. 2012. The best of both worlds: Stochastic and adversarial bandits. In *Conference on Learning Theory*. 42–1.

[9] Rocío Cañamares, Marcos Redondo, and Pablo Castells. 2019. Multi-armed recommender system bandit ensembles. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 432–436.

[10] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*. 2249–2257.

[11] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and Debias in Recommender System: A Survey and Future Directions. *arXiv preprint arXiv:2010.03240* (2020).

[12] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. 2011. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 208–214.

[13] PAK Covey-Crump and SD Silvey. 1970. Optimal regression designs with previous observations. *Biometrika* 57, 3 (1970), 551–566.

[14] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. 2020. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations (ICLR)*.

[15] Miroslav Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. 2011. Efficient optimal learning for contextual bandits. *arXiv preprint arXiv:1106.2369* (2011).

[16] Otto Dykstra. 1971. The augmentation of experimental data to maximize [X' X]. *Technometrics* 13, 3 (1971), 682–688.

[17] Marguerite Frank, Philip Wolfe, et al. 1956. An algorithm for quadratic programming. *Naval research logistics quarterly* 3, 1-2 (1956), 95–110.

[18] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. 2018. Offline a/b testing for recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 198–206.

[19] Elad Hazan and Zohar Karnin. 2016. Volumetric spanners: an efficient exploration basis for learning. *The Journal of Machine Learning Research* 17, 1 (2016), 4062–4095.

[20] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.

[21] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining*. 263–272.

[22] Guido W Imbens and Donald B Rubin. 2015. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press.

[23] Martin Jaggi. 2013. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th international conference on machine learning*. 427–435.

[24] Michael Jahrer, Andreas Töscher, and Robert Legenstein. 2010. Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 693–702.

[25] Mark E Johnson and Christopher J Nachtsheim. 1983. Some guidelines for constructing exact D-optimal designs on convex design spaces. *Technometrics* 25, 3 (1983), 271–277.

[26] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. 2015. Efficient Thompson Sampling for Online Matrix-Factorization Recommendation. In *Advances in neural information processing systems*. 1297–1305.

[27] Jack Kiefer and Jacob Wolfowitz. 1960. The equivalence of two extremum problems. *Canadian Journal of Mathematics* 12 (1960), 363–366.

[28] John Langford and Tong Zhang. 2008. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*. 817–824.

[29] Minyong R Lee and Milan Shen. 2018. Winner's Curse: Bias Estimation for Total Effects of Features in Online Controlled Experiments. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 491–499.

[30] Lihong Li, Wei Chu, John Langford, Taesup Moon, and Xuanhui Wang. 2012. An unbiased offline evaluation of contextual bandit algorithms with generalized linear models. In *Proceedings of the Workshop on On-line Trading of Exploration and Exploitation 2*. 19–36.

[31] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. 661–670.

[32] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. 2016. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 539–548.

[33] Haipeng Luo, Chen-Yu Wei, Alekh Agarwal, and John Langford. 2018. Efficient contextual bandits in non-stationary worlds. In *Conference On Learning Theory*. 1739–1776.

[34] Lawrenca S Mayer and Arlo D Hendrickson. 1973. A method for constructing an optimal regression design after an initial set of input values has been selected. *Communications in Statistics-Theory and Methods* 2, 5 (1973), 465–477.

[35] H Brendan McMahan and Matthew Streeter. 2009. Tighter bounds for multi-armed bandits with expert advice. (2009).

[36] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. arXiv:1310.4546 [cs.CL]

[37] Timothy E O'Brien and Gerald M Funk. 2003. A gentle introduction to optimal design for regression models. *The American Statistician* 57, 4 (2003), 265–267.

[38] Herbert Robbins. 1952. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.* 58, 5 (1952), 527–535.

[39] Paat Rusmevichientong and John N Tsitsiklis. 2010. Linearly parameterized bandits. *Mathematics of Operations Research* 35, 2 (2010), 395–411.

[40] Daniel Russo and Benjamin Van Roy. 2016. An information-theoretic analysis of Thompson sampling. *The Journal of Machine Learning Research* 17, 1 (2016), 2442–2471.

[41] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. 2017. A tutorial on thompson sampling. *arXiv preprint arXiv:1707.02038* (2017).

[42] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (Hong Kong, Hong Kong) (*WWW '01*). Association for Computing Machinery, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

[43] Anne Schuth, Katja Hofmann, and Filip Radlinski. 2015. Predicting search satisfaction metrics with interleaved comparisons. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 463–472.

[44] Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender systems handbook*. Springer, 257–297.

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

[46] Lynda V White. 1973. An extension of the general equivalence theorem to nonlinear models. *Biometrika* 60, 2 (1973), 345–348.

[47] Yuxiang Xie, Nanyu Chen, and Xiaolin Shi. 2018. False discovery rate controlled heterogeneous treatment effect detection for online controlled experiments. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 876–885.

[48] Da Xu, RUAN Chuanwei, Kamiya Motwani, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Methods and apparatus for item substitution. US Patent App. 16/424,799.

[49] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Adversarial Counterfactual Learning and Evaluation for Recommender System. *Advances in Neural Information Processing Systems* 33 (2020).

[50] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Product knowledge graph embedding for e-commerce. In *Proceedings of the 13th international conference on web search and data mining*. 672–680.

[51] Min Yang, Stefanie Biedermann, and Elina Tang. 2013. On optimal designs for nonlinear models: a general and efficient algorithm. *J. Amer. Statist. Assoc.* 108, 504 (2013), 1411–1420.

[52] Xuan Yin and Liangjie Hong. 2019. The identification and estimation of direct and indirect effects in A/B tests through causal mediation analysis. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2989–2999.

[53] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. 2013. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 1411–1420.

[54] Z. Zhao and M. Shang. 2010. User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop. In *2010 Third International Conference on Knowledge Discovery and Data Mining*. 478–481.