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.
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 \( \pi \) over the candidate recommenders, i.e. \( 0 < \pi_t(a) < 1 \), \( \sum_{a=1}^{k} \pi_t(i) = 1 \) 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 \( \pi_t \), which is aligned with the bandit problems. The intuition is that by optimizing \( \pi_t \), 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 \( t \in \mathbb{R} \). The reward mechanism is given by the linear model:

\[
Y_t = \theta_1 \cdot I_{1} \times X_{1} + \ldots + \theta_6 \cdot I_{6} \times X_{6};
\]

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):

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

If each recommender is explored with the probability, the role of \( a_3 \) is underrated since it is the only recommender that provides information about \( \theta_1 \) and \( \theta_2 \). Also, \( a_2 \) and \( a_4 \) give the same outputs, so their exploration probability should be discounted by half. Similarly, the information provided by \( S_1 \) can be recovered by \( S_1 \) and \( S_4 \) (or \( S_3 \)) 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:

\[
X_t := \{ \phi(r_t, a_t(r_t)) \}_{t=1}^{k}.
\]
3. The developer chooses a recommender \( a_t \) according to the randomized experiment strategy \( \pi(a_t | X_t, \hat{H}_t) \).
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:

\[
a_t \sim \pi[a | r_t, \{ \phi(r_t, a_t(r_t)) \}_{i=1}^k, \tilde{h_t}],
\]

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

\[
\left(r_t, \{ \phi(r_t, a_t(r_t)) \}_{i=1}^k, a_t, y_t \right).
\]

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: \( Q(a) = \sum_{j=1}^T y_j | a = j \) and number of appearances: \( N(a) = \sum_{j=1}^T 1 | a = j \) (assume up to round \( t \)) for \( a = 1, \ldots, k \), are quite computationally feasible, e.g.

- **\( \epsilon \)-greedy**: explores with probability \( \epsilon \) under the uniform exploration policy \( \pi(a) = 1/k \) for a epoch, 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 \( \delta \).

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

- selects \( \arg \max_{a \in [k]} Q(a) \) for a epoch, where \( Q(a) \) is sampled from the posterior distribution Beta(\( \alpha_t, \beta_t \)), and \( \alpha_t \) and \( \beta_t \) 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 \( Q, N, \alpha, \beta \). For MAB, taking account of the context often requires employing a parametric reward model: \( y_a = f_\theta(\phi(a(r))) \), so during exploration, we may also update the model parameters \( \theta \) using the collected data. Suppose we have an optimization oracle that returns \( \theta \) 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 \( \arg \max_{a \in [k]} y_a := f_\theta(\phi(a(r))) \) 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 \( \theta \) from the reward-model-specific posterior distribution of \( \theta \), 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 \( \pi \) thus becomes very challenging since we may not have access to the most updated \( \theta \). Therefore, it is important to take advantage of the structure of \( f_\theta(\cdot) \), 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.

| \( k, m, n \) | 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 front-end request, action selected by the developer, and the reward at round \( t \). |
| \( \tilde{h_t}, \pi(\cdot | \cdot) \) | 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. |
| \( I_t, a_t(\cdot) \) | The whole set of items and the \( i^{th} \) candidate recommender, with \( a_t(\cdot) \in I_t^{\text{item}} \). |
| \( \phi(r_t, a_t(r_t)) \) | The (engineered or pre-trained) feature representation in \( \mathbb{R}^d \), specifically for the \( i^{th} \)-round front-end request and the output contents for the \( j^{th} \) 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.
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 recommendation [26, 32, 55] and recommender ensemble [9, 51]. 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 propose a general framework in Algorithm 1.

Algorithm 1: Stage-wise exploration and exploitation

\textbf{Input:} Reward model $f_0(\cdot)$; the restart criteria; the initialized history data $\hat{h}_t$.

\begin{algorithmic}
\While{total rounds $\leq n$}
\If{restart criteria is satisfied}
\State Reset $\hat{h}_t$.
\EndIf
\State Play $n_1$ rounds of random exploration, for instance: $\pi(a | r) = \frac{1}{2}$, and collect observation to $\hat{h}_t$;
\State Find the optimal $\hat{\theta}$ based on $\hat{h}_t$ (e.g., via empirical-risk minimization);
\State Play $n_2$ rounds of exploitation with:
\State $a_t = \arg \max_a \ell_0(r_t, a(r_t))$.
\EndWhile
\end{algorithmic}

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 model inference. Several additional advantages of the stage-wise algorithms include:

- the number 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 $\theta$. 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:

\begin{equation}
\begin{aligned}
    y_t &= \begin{cases}
    \theta^T \phi(r_t, a(r_t)), & \text{linear model} \\
    f_0(\phi(r_t, a(r_t))), & \text{for some nonlinear } f_0(\cdot),
\end{cases}
\end{aligned}
\end{equation}

We start with the frequentist setting, i.e. $\theta$ do not admit a prior distribution. In each round $t$, we try to find a optimal design $\pi(\cdot | r_t)$ such that the action sampled from $\pi$ leads to a maximum information for estimating $\theta$. For statistical estimators, the Fisher information is a key quantity for evaluating the amount of information in the observations. For the general $f_0$, the Fisher information under (2) is given by:

\begin{equation}
M(\pi) = \sum_{a_t=1}^{k} \pi(a_t) \left( \nabla_{\theta} f_0(\phi(r_t, a_t(r_t))) - \nabla_{\theta} f_0(\phi(r_t, a_t(r_t)))^{\top} \right),
\end{equation}

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

\begin{equation}
M(\pi) = \sum_{a_t=1}^{k} \pi(a_t) (\phi(r_t, a_t(r_t)) \cdot \phi(r_t, a_t(r_t)))^{\top}.
\end{equation}

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 $\phi_t := \phi(r_t, a_t(r_t))$ is given by $\text{var}(\hat{y}_t) \propto \phi_t M(\pi)^{-1} \phi_t^{\top}$.

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 $\max_{\pi \in \{1, \ldots, k\}} \phi_t M(\pi)^{-1} \phi_t^{\top}$ 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 $\pi^*$, the following statements are equivalent:

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

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

\begin{equation}
\max_{\pi} \log \det |M(\pi)| \text{ s.t. } \sum_{a_t=1}^{k} \pi(a_t) = 1.
\end{equation}

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 $\theta$ is given by $\theta \sim N(0, R)$ where $R$ 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 $\theta$. The intuition is that the larger the divergence, the more information there is in the observations. Let $y$ be the hypothetical rewards for $\phi(r_1, a_1(r_1)), \ldots, \phi(r_t, a_k(r_t))$. Then the gain in Shannon information is given by:

$$U(\pi) = \int \log p(\theta, \pi, y) p(y, \theta | \pi) d\theta dy$$

$$= C + \frac{1}{2} \log |M(\pi) + R^{-1}|,$$  \hspace{1cm} (5)

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

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 $M(\pi) + R^{-1}$ 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 $\hat{\theta}_0 = \hat{\theta}$ and $\eta_a := \nabla_\theta f_\theta(\phi(r_i, a(r_i)))|_{\theta=\theta_0}$; the convergence criteria.

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

Referring to the standard analysis of Frank-Wolfe algorithm [23], we show that it takes the solver at most $O(d \log d + d^2)$ updates to achieve a multiplicative $(1 + \epsilon)$ optimal solution. Each update has an $O(kd^2)$ computation complexity, but $d$ is usually small in practice (e.g. $d = 6$ in Example 1), which will we 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:
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 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.
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.

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 follows. 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., $\geq 3$ 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) against the total number of positive rating in the testing data. The reported results are averaged over ten runs.

Dataset. We use the MoiveLens 1M 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., $\geq 2.5$ 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 $\Phi(\cdot)$. 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.

Candidate recommenders. We employ the four classical recommendation models: user-based collaborative filtering (CF) [36], 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: $\epsilon$-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 $\epsilon$-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 $\epsilon$-greedy, which is expected. The usefulness of context in the simulation is suggested by the significantly 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 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.

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 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:

\[ \theta_0 + \theta_1 \langle z_u, z_i \rangle + \ldots + \theta_m \langle z_u, z_i \rangle + \theta^\top [\text{user feats}, \text{items feats}] \]

and \( z_u \) and \( z_i \) are the user and item embeddings.

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.
[54] 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.

[55] 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.

[56] 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.