# Improving Ad Click Prediction by Considering Non-displayed Events

Bowen Yuan
National Taiwan University
f03944049@csie.ntu.edu.tw

Jui-Yang Hsia[*]
National Taiwan University
hsiajuiyang5174@gmail.com

Meng-Yuan Yang[†]
ETH Zurich
meyang@ethz.ch

Hong Zhu
Huawei Noah's ark lab
zhuhong8@huawei.com

Chih-Yao Chang
Huawei Noah's ark lab
chang.chih.yao@huawei.com

Zhenhua Dong
Huawei Noah's ark lab
dongzhenhua@huawei.com

Chih-Jen Lin
National Taiwan University
cjlin@csie.ntu.edu.tw

## ABSTRACT

Click-through rate (CTR) prediction is the core problem of building advertising systems. Most existing state-of-the-art approaches model CTR prediction as binary classification problems, where displayed events with and without click feedbacks are respectively considered as positive and negative instances for training and offline validation. However, due to the selection mechanism applied in most advertising systems, a selection bias exists between distributions of displayed and non-displayed events. Conventional CTR models ignoring the bias may have inaccurate predictions and cause a loss of the revenue. To alleviate the bias, we need to conduct counterfactual learning by considering not only displayed events but also non-displayed events. In this paper, through a review of existing approaches of counterfactual learning, we point out some difficulties for applying these approaches for CTR prediction in a real-world advertising system. To overcome these difficulties, we propose a novel framework for counterfactual CTR prediction. In experiments, we compare our proposed framework against state-of-the-art conventional CTR models and existing counterfactual learning approaches. Experimental results show significant improvements.

## CCS CONCEPTS

• **Information systems** → **Computational advertising**.

## KEYWORDS

CTR prediction, Recommender system, Missing not at random, Selection bias, Counterfactual learning

[*]This author contributes equally with Bowen Yuan.
[†]Work done at Huawei.

*CIKM '19, November 3–7, 2019, Beijing, China*
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358058

**ACM Reference Format:**
Bowen Yuan, Jui-Yang Hsia, Meng-Yuan Yang, Hong Zhu, Chih-Yao Chang, Zhenhua Dong, and Chih-Jen Lin. 2019. Improving Ad Click Prediction by Considering Non-displayed Events. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3357384.3358058

## 1 INTRODUCTION

Online advertising is a major profit model for most Internet companies where advertisers pay publishers for promoting ads on their services. Advertising systems perform in the role of a dispatcher that decides which and where ads are displayed. Though there are many compensation methods in advertising systems, in this work, we focus on cost-per-click (CPC) systems, where advertisers are only charged if their ads are clicked by users. We give a simplified example of CPC systems in Figure 1.

Because the number of positions from publishers is limited, the system must select the most valuable ads to display. Therefore, when the system receives a request, all ads must be ranked according to the following expected revenue

$$\text{ECPM} = P \times R, \tag{1}$$

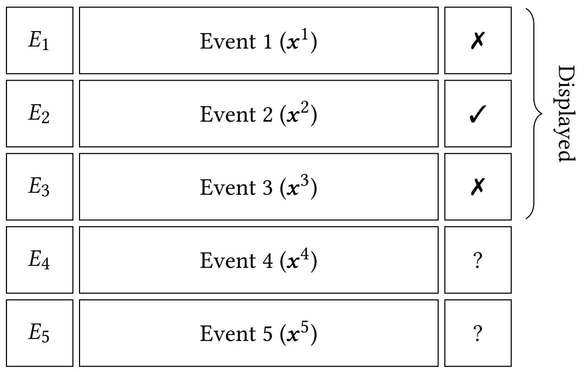
where

$$P = \Pr(\text{click} \mid \text{event}) \tag{2}$$

is the probability of an event associated with a (request, ad) pair being clicked, and $R$ is the price paid by advertisers if the event is clicked. Then the system selects and displays the top events on the publishers. In Figure 1, the example shows that five events bid for three advertising positions, so the system ranks them by (1). Event '1','2' and '3' are the top three, and get the chance to be displayed.

Once the event is displayed, the system will receive signals if a customer clicks it, which can be considered as the feedback of this event. Then the system gains revenue from the advertiser. As the example shown in Figure 1, among the three displayed events, event '2' is clicked, while the other two are not clicked.

To maximize the revenue, the estimation of (2), also referred as click-through rate (CTR) prediction, is the core problem of building advertising systems. To estimate CTR, the most widely used



$E_1$
Event 1 ($\boldsymbol{x}^1$)
✗
$E_2$
Event 2 ($\boldsymbol{x}^2$)
✓
$E_3$
Event 3 ($\boldsymbol{x}^3$)
✗
$E_4$
Event 4 ($\boldsymbol{x}^4$)
?
$E_5$
Event 5 ($\boldsymbol{x}^5$)
?
Displayed

**Figure 1: A simplified example to illustrate the procedure of selecting events and collecting feedbacks in CPC systems. The symbols '✓', '✗', and '?' indicate the event is clicked, not clicked, and not displayed, respectively.**

approach is to learn from historically displayed events via machine learning techniques. Specifically, assume that we have the following $L$ events

$$(y_1, \boldsymbol{x}^1), \cdots, (y_L, \boldsymbol{x}^L), \tag{3}$$

where $y_l$ and $\boldsymbol{x}^l$ are the label and the feature vector of the $l$th events, respectively. To model CTR prediction as binary classification problems, we assume that the label $y$ satisfies

$$y = \begin{cases} 1 & \text{the event was displayed and clicked,} \\ -1 & \text{the event was displayed but not clicked.} \end{cases}$$

Then we need to find a decision function $\hat{y}(\boldsymbol{x}) \in [-\infty, \infty]$ so that it can be used to predict the label. For example, in a typical classification setting the predicted label is

$$\begin{cases} 1 & \text{if } \hat{y}(\boldsymbol{x}) \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

For CTR prediction, a probability output may be needed, so we can transform $\hat{y}(\boldsymbol{x})$ to a value in $[0, 1]$ by a sigmoid function. To obtain a good decision function, many state-of-the-art models for CTR prediction problems have been proposed (e.g., logistic regression [4, 24, 28], factorization machines [27], field-aware factorization machines [15, 16], and deep learning approaches [8, 26, 38]). A review of some of these models will be given in Section 2.1.

Recall the example in Figure 1. Two events are not displayed so their feedbacks are not available. Most works mentioned above ignore non-displayed events. We think the reason supporting this decision is that some pioneering works (e.g., [28, Section 3]) thought causality exists between a display and a click, where the click is understood as a consequence of the display. Then, they give the following basic but far-reaching assumption:

Assumption 1.1. *The probability of an event being clicked but not displayed is zero.*

With the assumption, no click information is included for non-displayed events. Therefore, the probability of an event being clicked can be measured by CTR defined as

$$\text{CTR} = \frac{\text{number of clicks}}{\text{number of displayed events}} \times 100\%.$$

**Table 1: Main notation.**

| | |
|---|---|
| $L$ | numbers of displayed events |
| $\hat{L}$ | numbers of displayed and non-displayed events |
| $\boldsymbol{x}, y$ | feature vector and label |
| $D$ | numbers of features |
| $S_t$ | set of events logged by the uniform policy |
| $S_c$ | set of events logged by non-uniform policies |
| $\hat{y}(\cdot), \sigma(\cdot)$ | CTR model and imputation model |
| $m, n$ | numbers of requests and ads |
| $\Omega$ | set of displayed (request, ad) pairs |
| $Y, \hat{Y}, A$ | label matrix, output matrix, and the matrix of imputed labels |

However, in Section 2.2, by following some past works (e.g., [18, 31]), we argue the importance of handling non-displayed events. That is, through considering non-displayed events as unlabeled instances, CTR prediction should be modeled as a learning problem with labeled and unlabeled instances. Further, through an A/B test conducted in our online system, we show that due to the inconsistency between distributions of labeled and unlabeled samples, the ignorance of non-displayed events may cause a strong bias and inaccurate predictions. To eliminate the bias in CTR prediction, not only displayed events but also non-displayed events should be considered. However, because of lacking labels for non-displayed events, we need to solve a learning problem with partial labels, which is also referred to as "batch learning from logged bandit feedbacks" or "counterfactual learning" [31].

In Section 3, through a review on existing approaches of counterfactual learning, we point out some difficulties of applying them in a real-world CPC system. For example, many existing approaches required that events are sampled to be displayed through the importance sampling technique. However, as illustrated in Figure 1, most real-world CPC systems deterministically select the top events with the highest expected revenues. Therefore, these approaches are inapplicable to practical CTR prediction. Additionally, some existing approaches need to solve a difficult optimization problem, a situation not impractical for industrial-scale CTR prediction. Our main contribution in Section 4 is to propose a novel framework for counterfactual CTR prediction to overcome these difficulties. To address the issue of the training efficiency, we link the proposed framework to recommender systems with implicit feedbacks and develop efficient algorithms. In Section 5, experiments are conducted on a semi-synthetic small data set and a real-world data set collected from our online system. Results show our framework is superior to both conventional and counterfactual learning approaches. Table 1 gives main notation in this paper. Supplementary materials and code for experiments are at https://www.csie.ntu.edu.tw/~cjlin/papers/occtr.

## 2 HANDLING UNOBSERVED DATA IN CTR PREDICTION

In this section, we first give a brief review on the widely-used settings for CTR prediction. Next we show that under this setting, a bias may exist and cause inconsistency between offline and online evaluations.

## 2.1 A Review on CTR Prediction by Binary Classification

As mentioned in Section 1, the task of CTR prediction is to find a model, which inputs $\boldsymbol{x}$, outputs $\hat{y}(\boldsymbol{x})$ and generates a probability value as the prediction of (2). If we have $L$ displayed events as training samples shown in (3), most CTR works solve the following optimization problem.

$$\min_{\mathcal{W}} \quad \lambda R(\mathcal{W}) + \sum_{l=1}^{L} \ell(y_l, \hat{y}_l), \tag{4}$$

where $\hat{y}_l \equiv \hat{y}(\boldsymbol{x}^l)$, $\mathcal{W}$ is a set of model parameters, $\ell(a,b)$ is a loss function convex in $b$, and $R(\mathcal{W})$ is the regularization term with the parameter $\lambda$ decided by users. Logistic regression (LR) is a commonly used model for CTR prediction. Assume $D$ is the number of features in each vector $\boldsymbol{x}^l$, and a vector $\boldsymbol{w} \in \mathcal{R}^D$ is considered as the set $\mathcal{W}$ of model parameters. LR then sets

$$\hat{y}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$

and considers the following logistic loss

$$\ell(a,b) = \log(1 + e^{-ab}). \tag{5}$$

Past studies (e.g., [4, 16]) have shown that learning the effect of feature conjunctions is crucial for CTR prediction. However, due to the high dimensionality and the high sparsity of $\boldsymbol{x}$, LR may have difficulty to learn from all feature conjunctions of $\boldsymbol{x}$. To overcome the difficulty, factorization machine (FM) [27] and its variants such as field-aware factorization machine (FFM) [16] are proposed. They consider a low-rank approximation to the coefficient matrix of all feature conjunctions. The output of FM is

$$\hat{y}_{\text{FM}}(\boldsymbol{x}) = \sum_{j_1=1}^{D} \sum_{j_2 \geq j_1+1}^{D} (\boldsymbol{w}_{j_1}^T \boldsymbol{w}_{j_2}) x_{j_1} x_{j_2}, \tag{6}$$

where $\boldsymbol{w}_j \in \mathcal{R}^k, \forall j$ are the latent vectors and $k$ is a pre-specified value. FFM extends FM to assume that features are grouped to several "fields."

$$\hat{y}_{\text{FFM}}(\boldsymbol{x}) = \sum_{j_1=1}^{D} \sum_{j_2 \geq j_1+1}^{D} (\boldsymbol{w}_{j_1,f_{j_2}}^T \boldsymbol{w}_{j_2,f_{j_1}}) x_{j_1} x_{j_2}, \tag{7}$$

where $f_{j_1}$ and $f_{j_2}$ are the corresponding fields of features $j_1$ and $j_2$, respectively, and $\boldsymbol{w}_{j_1,f_{j_2}} \in \mathcal{R}^k$ is a latent vector which learns the latent effect between feature $j_1$ and the interacted features belonging to the field $f_{j_2}$. It has been shown in some past competitions (e.g., [14, 16]) and industry scenarios [15] that FFM gives state-of-the-art results on CTR prediction.

For evaluation, two metrics are commonly used [4, 9, 16]. Suppose $L$ is the number of events being evaluated. The first criterion is the negative logarithmic loss (NLL):

$$\text{NLL} \equiv -\frac{1}{L} \sum_{l=1}^{L} \log(1 + e^{-y_l \hat{y}_l}). \tag{8}$$

Another one is the area under the ROC curve (AUC):

$$\text{AUC} \equiv \frac{\sum_{l=1}^{L_p} \text{Rank}_l - \binom{L_p}{2}}{(L_p(L - L_p))}, \tag{9}$$

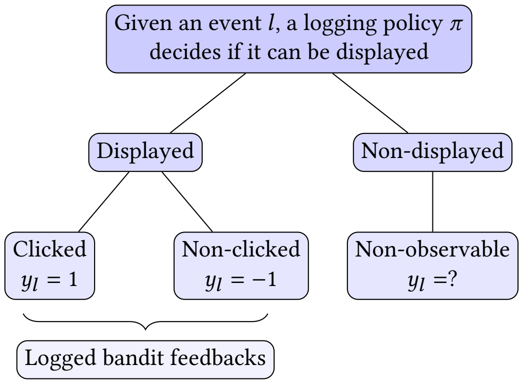


**Figure 2: A diagram of CTR prediction being formulized as a selective labels problem**

where $L_p$ is the number of positive instances, and $\text{Rank}_l$ is the rank of a positive event $l$ in all $L$ events, which are ranked in a descending order according to their predicted values.

## 2.2 The Importance of Handling Non-displayed Events[1]

Let $\Pr(\mathcal{C}, \boldsymbol{\mathcal{X}})$ be the distribution to generate events $\boldsymbol{x}$ and the corresponding clicks or not-clicks $y$. The probability value from CTR model $\hat{y}(\boldsymbol{x})$ obtained by solving the following expected risk minimization should be an optimal estimation of (2).

$$\min_{\hat{y}} R(\hat{y}), \text{ where } R(\hat{y}) = \int \ell(y, \hat{y}(\boldsymbol{x})) d \Pr(y, \boldsymbol{x}). \tag{10}$$

In practice, because $\Pr(\mathcal{C}, \boldsymbol{\mathcal{X}})$ is unknown, we conduct empirical risk minimization by minimizing the average loss of sampled results. Most CTR models conducted training and validation on the displayed events. Let $\Pr(\mathcal{C}, \boldsymbol{\mathcal{X}}, \mathcal{O} = 1)$ be the joint distribution of clicks, events being displayed. Models obtained by solving (4) implicitly assume that

$$\Pr(\mathcal{C}, \boldsymbol{\mathcal{X}}) \propto \Pr(\mathcal{C}, \boldsymbol{\mathcal{X}}, \mathcal{O} = 1),$$

such that

$$\min_{\hat{y}} R(\hat{y}), \text{ where } R(\hat{y}) = \int \ell(y, \hat{y}(\boldsymbol{x})) d \Pr(y, \boldsymbol{x}, \mathcal{O} = 1). \tag{11}$$

is equivalent to (10). However, in the subsequent discussion, we show that this assumption is not precise. We start by following [18] to formalize CTR prediction as a selective labels problem. As illustrated in Figure 2, we can view our system as a labeling platform, where customers help to label $L + L_{\text{U}}$ instances. However, due to the limitation of capacity, an algorithm $\pi$ (also called logging policy) selects $L$ instances and asks customers to label them. From this view, the displayed events are labeled instances (also called logged bandit feedbacks), while non-displayed events are unlabeled instances. Because of lacking labels, the click information of non-displayed events, is unknown rather than zero in Assumption 1.1.

Because (11) can be reformulated as

$$\bar{R}(\hat{y}) = \int \ell(y, \hat{y}(\boldsymbol{x})) \frac{d \Pr(y, \boldsymbol{x}, \mathcal{O} = 1)}{d \Pr(y, \boldsymbol{x})} d \Pr(y, \boldsymbol{x}), \tag{12}$$

[1]Some descriptions in this section are improved after the publication of the paper by following the explanation in the paper [35]

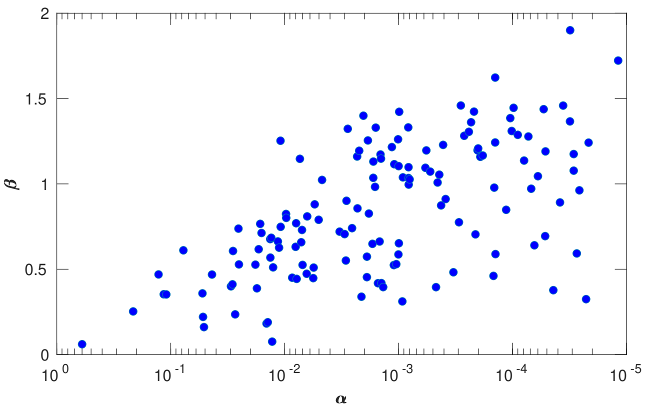


**Figure 3: The coordinates $(\alpha, \beta)$ of each dot indicate an ad's displayed ratio defined in (18) and the log-scaled CTR ratio between events from $S_c$ and $S_t$; see (19).**

from (11), whether $\bar{R}(\hat{y})$ is an unbiased expected risk depends on if events have the same chance to be displayed. That is, if there exists a constant $\Delta > 0$ such that

$$\Pr(C, \boldsymbol{X}) = \Delta \Pr(C, \boldsymbol{X}, O = 1), \tag{13}$$

then

$$\bar{R}(\hat{y}) = \int \ell(y, \hat{y}(\boldsymbol{x}))\Delta d \Pr(y, \boldsymbol{x}) = \Delta R(\hat{y}), \tag{14}$$

so (10) and (11) return the same optimal model. To investigate (13), we first express two sides respectively as

$$\begin{aligned} \Pr(C, \boldsymbol{X}, O = 1) &= \Pr(C \mid \boldsymbol{X}, O = 1) \Pr(\boldsymbol{X}, O = 1) \\ \Pr(C, \boldsymbol{X}) &= \Pr(C \mid \boldsymbol{X}) \Pr(\boldsymbol{X}). \end{aligned} \tag{15}$$

Because the tendency of a customer click can not be affected by if the event is displayed, given an event, we should have

$$\Pr(C \mid \boldsymbol{X}, O = 1) = \Pr(C \mid \boldsymbol{X}). \tag{16}$$

By using (16) to cancel terms in the division of the two expressions in (15) and through $\Pr(\boldsymbol{X}, O = 1) = \Pr(\boldsymbol{X}) \Pr(O = 1 \mid \boldsymbol{x})$, the task of investigating (13) is reduced to checking if the distribution of an event being displayed

$$\Pr(O = 1 \mid \boldsymbol{x}) \tag{17}$$

is uniform. Unfortunately, this condition can not be held for CTR prediction due to the non-uniform selection of displayed events as introduced in Section 1. Therefore, (17) is non-uniform and (13) does not hold, nor does (14). Then $R(\cdot)$ is not proportional to $\bar{R}(\cdot)$, so $\bar{R}(\cdot)$ is a biased expected risk of $\hat{y}(\boldsymbol{x})$ and the model obtained by solving (4) leads to a biased estimator of (2).

To better explain this, we deployed a uniform policy on a small part of live traffics in our online system, while the rest was still served by non-uniform policies. According to logging policies, the labeled events are grouped into the following two sets:

$$S_t = \{(y_l, \boldsymbol{x}^l) \mid \text{event } l \text{ was logged by the uniform policy}\},$$
$$S_c = \{(y_l, \boldsymbol{x}^l) \mid \text{event } l \text{ was logged by non-uniform policies}\}.$$

In Figure 3, we check how CTR estimated from logs of non-uniform policies differs from the true CTR. The x-axis is the displayed ratio of an ad, where each point is calculated by

$$\alpha_j = \frac{\text{\# displayed events associated with ad } j}{\text{\# (displayed+non-displayed) events associated with ad } j}. \tag{18}$$

The y-axis is the log-scaled ratio of an ad's observed CTR in $S_c$ to that in $S_t$, where each point is calculated by

$$\beta_j = \log\left(\frac{\text{the CTR of ad } j \text{ observed in } S_c}{\text{the CTR of ad } j \text{ observed in } S_t}\right). \tag{19}$$

As we mentioned earlier, for events logged by a uniform policy, the bias can be avoided. Thus an ad's CTR value from $S_t$ can be considered as the true $P$'s average over all events associated with this ad. From Figure 3, with $\beta_j > 0$, an ad's CTR from events in $S_c$ is higher than that from events in $S_t$. This observation confirms a gap between the biased CTR and the unbiased CTR when events are logged by non-uniform policies. More importantly, the gap varies on different ads. For example, for those ads with lower $\alpha$ scores, the gap enlarges, while for those ads with higher $\alpha$ scores, the gap narrows. It can be realized that for an ad favoured by the proceeding policies, more associated events can be displayed and fewer events are not labeled. Then the observed CTR is closer to the unbiased one obtained from the fully labeled set $S_t$. We can conclude that if we fit CTR models on events logged by non-uniform policies, a biased estimator is obtained. It may overestimate the expected revenue of ads with lower $\alpha$ scores and wrongly display too many of them.

The bias caused by using a non-uniform logging policy is often referred to as "selection bias" in the literature of econometrics [10], and "missing not at random" in the field of statistics [21]. Ideally, a simple way to avoid the selection bias is to directly train CTR models on $S_t$. However, in most scenarios of CTR prediction, the feature set is high dimensional and sparse, so a large training set is required for learning a model with good generalization ability. The higher CTR shown in Figure 3 by displaying events in $S_c$ indicates that if we deploy the uniform policy on more traffics, the overall revenue may decrease. Therefore, forming a large $S_t$ through deploying the uniform policy on a significant amount of traffic is not practically viable for many internet companies. In this paper, we will develop more practical ways to eliminate the bias carried in the data logged by non-uniform logging policies.

Now we aim to consider both displayed and non-displayed events. Assume we have the following $\hat{L}$ events

$$(y_1, \boldsymbol{x}^1), \cdots, (y_{\hat{L}}, \boldsymbol{x}^{\hat{L}}), \tag{20}$$

where $\hat{L}$ with $\hat{L} \gg L$ is the number of all possible events. To estimate $P$ without bias, we expect to solve the following optimization problem.

$$\min_{\mathcal{W}} \quad \lambda R(\mathcal{W}) + \sum_{l=1}^{\hat{L}} \ell(y_l, \hat{y}_l). \tag{21}$$

However, because feedbacks of non-displayed events can not be observed for most CPC systems, the corresponding $y_l$ for $l = L + 1, \cdots, \hat{L}$ are not available. Therefore, (21) can not be solved by conventional methods based on supervised learning. Solving (21) with selective labels is also referred to as "batch learning from logged bandit feedbacks" and "counterfactual learning" [31]. We give a brief review of existing approaches on this topic in the next section.

## 3 EXISTING APPROACHES FOR LEARNING FROM SELECTIVE LABELS

Existing approaches for solving (21) with selective labels broadly fall into the following three categories: direct method, the inverse-propensity-scoring method, and the doubly robust method.

### 3.1 Direct Method

As we mentioned above, the key challenge in solving (21) is that we lack labels of those non-displayed events. Direct methods are based on an intuitive idea that we can find a relationship $\sigma(\cdot)$ between labels and features of displayed events such that $\sigma(\boldsymbol{x}^l) \approx y_l$, for $l = 1, \cdots, L$. We call $\sigma(\cdot)$ as an imputation model, which can impute labels for all events and have the following $\hat{L}$ labeled samples

$$(\sigma_1, \boldsymbol{x}^1), \cdots, (\sigma_{\hat{L}}, \boldsymbol{x}^{\hat{L}}),$$

where $\sigma_l \equiv \sigma(\boldsymbol{x}^l)$. Then we solve the following standard supervised learning problem.

$$\min_{\mathcal{W}} \quad \lambda R(\mathcal{W}) + \sum_{l=1}^{\hat{L}} \ell(\sigma_l, \hat{y}_l). \tag{22}$$

Clearly, the performance of a direct method is determined by the quality of $\sigma(\cdot)$. Many past works found $\sigma(\cdot)$ via conducting machine learning techniques on the labeled data set. For example, ridge linear regression [6], and random forests [19] are considered.

### 3.2 Inverse-Propensity-Scoring (IPS) Method

By assuming the logging policy is stochastic, the IPS method proposed in [11] weights each labeled event with the inverse of its propensity score, which refers to the tendency or the likelihood of an event being logged. Specifically, given the event $l$, we can observe an indicator $O_l \in \{0, 1\}$, which indicates if the event was displayed. Following [30, 32], we assume that $O_l$ is sampled from a Bernoulli distribution as $O_l \sim \text{Bern}(z_l)$, where $z_l$ is the probability of the event being displayed. Here we consider $z_l$ as the propensity scores of event $l$. The IPS method solves the following problem.

$$\min_{\mathcal{W}} \quad \lambda R(\mathcal{W}) + \sum_{l=1}^{\hat{L}} \frac{\ell(y_l, \hat{y}_l)}{z_l} O_l,$$

which, according to the observations of $O_l$, $\forall l$, is equivalent to

$$\min_{\mathcal{W}} \quad \lambda R(\mathcal{W}) + \sum_{l=1}^{L} \frac{\ell(y_l, \hat{y}_l)}{z_l}. \tag{23}$$

The following proof has been given in past works, which shows that solving (23) can get an unbiased estimator of $P$.

$$\mathbb{E}_O[\sum_{l=1}^{\hat{L}} \frac{\ell(y_l, \hat{y}_l)}{z_l} O_l] = \sum_{l=1}^{\hat{L}} \frac{\ell(y_l, \hat{y}_l)}{z_l} \cdot \mathbb{E}_O[O_l] = (21).$$

Before solving (23), we need to know $z_l$ for $l = 1, \cdots, L$. The work [30] differentiates two settings for the assignment mechanism of $z_l$. The first is an experimental setting, where any event $z_l$ is decided by us and $O_l$ is generated according to $z_l$. Therefore we can directly solve (23) with known $z_l$. Many past works and data sets (e.g., [20, 31]) are based on this setting. The second is an observational setting, where $O_l$ is not controlled by us, so $z_l$ is implicit. In this case, we must estimate $z_l$ first by some techniques. For example, naive Bayes [30], survival models [37], and more generally, by assigning

$$O_l = \begin{cases} 1 & \text{the event was displayed,} \\ 0 & \text{the event was not displayed,} \end{cases}$$

we can find a relationship $o(\cdot)$ between $O_l$ and the given features such that $o(\boldsymbol{x}^l) \approx O_l$, for $l = 1, \cdots, \hat{L}$ by any machine learning techniques (e.g., logistic regression [30], gradient boosting [25]). Then $z_l \equiv o(\boldsymbol{x}^l)$ for $l = 1, \cdots, L$ can be used as the propensity scores in (23).

### 3.3 Doubly Robust Method

Doubly robust method for counterfactual learning was first proposed in [6], which takes advantage of the two aforementioned approaches by combining them together. Therefore, the following optimization problem is solved.

$$\min_{\mathcal{W}} \quad \lambda R(\mathcal{W}) + \underbrace{\sum_{l=1}^{L} \Big( \frac{\ell(y_l, \hat{y}_l)}{z_l} - \ell(\sigma_l, \hat{y}_l) \Big)}_{\text{IPS method part}} + \underbrace{\sum_{l=1}^{\hat{L}} \ell(\sigma_l, \hat{y}_l)}_{\text{direct method part}} . \tag{24}$$

The analysis in [6] shows that the bias of the doubly robust method is the product of the two parts from the direct method and the IPS method. Therefore, the bias of the doubly robust method can be eliminated if either the direct method part or the IPS method part is unbiased. Empirical studies confirm that the doubly robust method performs better than solely a direct method or an IPS method.

### 3.4 Discussion on Feasibility of Existing Approaches for CPC Systems

Though the estimators obtained by above-mentioned methods are expected to give unbiased predictions of $P$, we show that there exist three main obstacles for applying these approaches for CTR prediction in a real-world CPC system.

First, from the theoretic guarantee established by past literatures, IPS works when the logging policy is stochastic such that with the increase of $L$, sufficient events with various propensity scores can be observed. That is, though the policy tends to select the events with high propensity scores, those with low propensity scores still can have a chance to be displayed and included in the labeled set. For events with low propensity scores, the formulation in (23) leads to larger corresponding instance weights, and then IPS can correct the selection bias. However, commercial products such as CPC systems usually avoid a stochastic policy because of the risk of revenue loss. Instead, as we introduced in Section 1, the policy makes decisions by considering if the expected revenue is large enough. Therefore, the logging policy of our CPC system can be considered as under the following deterministic setting.

$$O_l = \mathbb{I}(\text{ECPM}_l > t), \tag{25}$$

where $\text{ECPM}_l$ is the expected revenue of the event $l$, and $t$ is a dynamic threshold. This policy deterministically selects the events with high propensity scores to be displayed. In this case, the events with low propensity scores are absent in the labeled set. The deterministic logging policy makes theory and practice of the IPS method inapplicable to practical CTR prediction.

Secondly, for the direct method and the corresponding part in the doubly robust method, we need an imputation model $\sigma(\cdot)$ to generate labels for non-displayed events. However, as we have shown that displayed events may carry the selection bias in Section 2.2, many existing works [6, 19] find that imputation models learnt from displayed events will propagate the selection bias to the non-displayed events. Then both the direct and the doubly robust methods cannot give unbiased predictions.

The third issue is that for estimators obtained by solving optimization problems like (22) and (24), they involve $O(\hat{L})$ costs of time and space. However, $\hat{L}$ is extremely large considering all displayed and non-displayed events. Therefore, to have a practical framework for large-scale counterfactual CTR prediction, any $O(\hat{L})$ cost of storage and computation should be avoided.

## 4 A PRACTICAL FRAMEWORK FOR COUNTERFACTUAL CTR PREDICTION

To address issues mentioned in Section 3.4, we propose a practical framework for counterfactual CTR prediction. Our framework extends the doubly robust method by dropping the propensity scores to address the issue of using a deterministic logging policy. We call it as the propensity-free doubly robust method. To further improve the robustness of our framework, we propose a new way to obtain the imputation model by taking the advantage of the small but unbiased set $S_t$ defined in Section 2.2. Finally, by connecting the CTR prediction with recommender systems, we propose an efficient training algorithm so that the prohibitive $O(\hat{L})$ cost can be trickily avoided if some conditions are satisfied.

### 4.1 Propensity-free Doubly Robust Method

A recent work [19] describes an analogous scenario that many commercial statistical machine translation systems only deterministically provide the best translation result to users and then collect corresponding feedbacks. Then they find that in their scenario, the doubly robust method still can perform well even under a deterministic logging policy. Their analysis provides a reason behind the result that the direct method part of doubly robust imputes labels for non-displayed events, which remedies the absence of unlabeled events and improves the generalization ability of the model. This result motivates us to investigate if the similar idea can be extended on CTR prediction. Specifically, we follow [19] to set $z_l = 1$ for all displayed events logged by a deterministic policy. Then we modify (24) to have the following problem of the propensity-free doubly robust method.

$$\min_{\mathcal{W}} \quad \lambda R(\mathcal{W}) + \sum_{l=1}^{L} \Big( \ell(y_l, \hat{y}_l) - \omega \ell(\sigma_l, \hat{y}_l) \Big) + \omega \sum_{l=1}^{\hat{L}} \ell(\sigma_l, \hat{y}_l), \quad (26)$$

where $\omega$ is used as a hyper-parameter to balance the IPS part and the direct method part.

Recall that the concept of the doubly robust method is that the bias can be eliminated if either one of the two sub-estimators is unbiased. Compared with (24), we can see that the IPS part in (26) under the setting of $z_l = 1$ is essentially reduced to the naive binary classification problem (4), which definitely cannot give unbiased predictions. Therefore, we rely on the direct method part to get unbiased predictions. However, if we follow existing works of the doubly robust method (e.g., [6, 19]) to learn $\sigma(\cdot)$ from displayed events, the carried selection bias will be propagated to non-displayed events through the imputation model. In this case, the direct method part can not give unbiased predictions either. In order to avoid this situation, we design a new approach to obtain the imputation model in the next section.

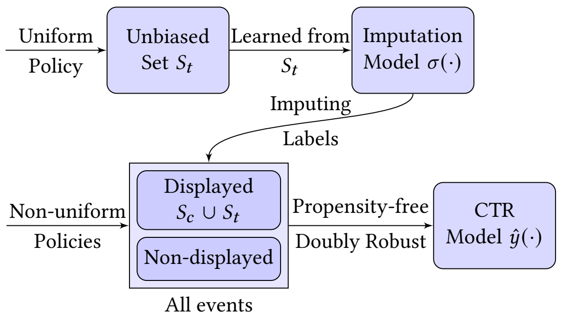


**Figure 4: A diagram showing components of our framework**

### 4.2 Unbiased Imputation Models

Recall that in Section 2.2, to help realize the severity of the selection bias, we collect a labeled set $S_t$ through deploying a uniform policy on a small percentage of our online traffic. Though $S_t$ is very small, it still contains some gold-standard unbiased information. We aim to leverage $S_t$ in our framework.

Past studies that have used $S_t$ include, for example, [3, 29]. By considering both $S_c$ and $S_t$ as training sets, they jointly learn two models respectively from the two sets in a multi-task fashion and conduct predictions by combining the two models. Our approach differs from theirs on the usage of $S_t$. As illustrated in Figure 4, we learn the imputation model $\sigma(\cdot)$ from $S_t$ and apply the model to impute labels of all displayed and non-displayed events; see the last term in (26).

Note that in our framework, learning the imputation model and learning the CTR model are two separate tasks. Different from the task of learning the CTR model which is generally complex and requires a large training set, the task of learning the imputation model is much more flexible. We can decide the imputation model and the corresponding feature set depending on the size of $S_t$. That is, when $S_t$ is large enough, the task can be as complex as that of learning the CTR model. But when the size of $S_t$ is limited, the task can be reduced to impute labels with a single value inferred from $S_t$ (e.g., the average CTR in $S_t$). Through the experimental results shown in Section 5.5, we will see that unbiased imputation models help our framework to obtain good performances even if $S_t$ is very small.

### 4.3 Efficient Training

By imputing labels for all possible events in the direct method part, (26) becomes a standard classification problem with $\hat{L}$ instances. Because $\hat{L}$ can be extremely large for a real-world CPC system, any cost of $O(\hat{L})$ will make the training process infeasible. Therefore, many existing optimization techniques, if directly applied, are unable to efficiently solve (26). For example, the stochastic gradient (SG) method popularly used for CTR prediction [8, 16] may face difficulties in the process of sampling $O(\hat{L})$ instances.[2] To overcome the difficulties, we connect CPC systems with a related but different field, called recommender systems, where an analogous issue has been well-studied.

In a typical recommender system, we observe part of a rating matrix $Y \in \mathcal{R}^{m \times n}$ of $m$ users and $n$ items. For each observed $(i, j)$ pair, $Y_{ij}$ indicates the rating of item $j$ given by user $i$. The goal is to predict the rating of an item not rated by a user yet. Following the idea, if we consider an event as a (request, ad) pair, an advertising system is naturally a recommender system. As illustrated in Figure 5, the $L$ displayed events can be considered as $L$ observed (request, ad) entries in a rating matrix $Y \in \mathcal{R}^{m \times n}$, where $m$ is the total number of requests and $n$ is the total number of ads. According to whether the ad was clicked by the user or not, these observed entries can be grouped into the following two sets

$$\Omega^+ = \{(i, j) \mid \text{ad } j \text{ was displayed and clicked in request } i\},$$
$$\Omega^- = \{(i, j) \mid \text{ad } j \text{ was displayed but not clicked in request } i\}.$$

By assigning ratings to observed entries as

$$Y_{ij} = \begin{cases} 1 & (i, j) \in \Omega^+, \\ -1 & (i, j) \in \Omega^-, \end{cases}$$

and replacing the index $l$ with the index $(i, j)$, we can re-write (26) as

$$\begin{aligned} \min_{\mathcal{W}} \quad & \lambda R(\mathcal{W}) + \sum_{(i,j)\in\Omega} \Big(\ell(Y_{ij}, \hat{Y}_{ij}) - \omega\ell(A_{ij}, \hat{Y}_{ij})\Big) \\ & + \omega \sum_{i=1}^{m}\sum_{j=1}^{n} \ell(A_{ij}, \hat{Y}_{ij}), \end{aligned} \tag{27}$$

where

$$\Omega = \Omega^+ \cup \Omega^-$$

is the set of displayed pairs with $|\Omega| = L$, $\hat{L} = mn$, $\hat{Y} \in \mathcal{R}^{m \times n}$ is the output matrix with $\hat{Y}_{ij} = \hat{y}(\boldsymbol{x}^{i,j})$, and $A \in \mathcal{R}^{m \times n}$ is the matrix of imputed labels with

$$A_{ij} = \sigma(\boldsymbol{x}^{i,j}). \tag{28}$$

Under the above setting, $\hat{L} = mn$ indicates the total number of possible events. The second term in (27) involves $O(mn) = O(\hat{L})$ operations, which are prohibitive when $m$ and $n$ are large. The same $O(mn)$ difficulty occurs in a topic called "recommender systems with implicit feedbacks" [12, 13, 33, 36]. It describes a scenario where a rating matrix $Y$ includes both positive and negative entries. However, only a subset $\Omega^+$ including some positive entries have been observed. An approach to handle this scenario is to treat all unobserved pairs as a negative rating $r$, such as 0 or -1 in [34, 36]. Let $A \in \mathcal{R}^{m \times n}$ be an artificial rating matrix with

$$A_{ij} = \begin{cases} 1 & (i, j) \in \Omega^+, \\ r & (i, j) \notin \Omega^+. \end{cases} \tag{29}$$

[2]The failure of SG on handling similar scenarios has been reported in several past works [33, 36].

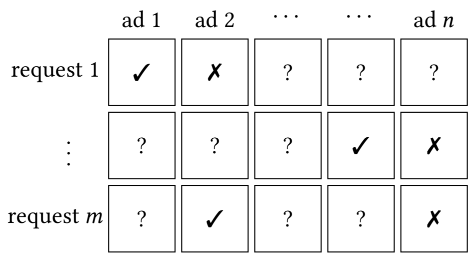


**Figure 5: An illustration of a rating matrix of $m$ requests and $n$ ads, where $L$ events are displayed. The symbols '✓', '✗', and '?' indicate the event is clicked, not clicked, and not displayed, respectively.**

To find a model fitting all $A_{ij}$ values, the cost of $O(mn)$ arises in solving the following problem with $mn$ instances.

$$\min_{\mathcal{W}} \quad \lambda R(\mathcal{W}) + \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}\ell(A_{ij}, \hat{Y}_{ij}). \tag{30}$$

where $C_{ij}$ is a cost parameter. To avoid the $O(mn)$ cost, existing studies [1, 33, 34, 36] impose some resctrictions on terms over values not in $\Omega^+$ and change (30) to

$$\begin{aligned} \min_{\mathcal{W}} \quad & \lambda R(\mathcal{W}) + \sum_{(i,j)\in\Omega^+} C_{ij}\ell(1, \hat{Y}_{ij}) + \sum_{(i,j)\notin\Omega^+} \bar{C}_{ij}\bar{\ell}(r, \hat{Y}_{ij}) \\ = & \lambda R(\mathcal{W}) + \sum_{(i,j)\in\Omega^+} \Big(C_{ij}\ell(1, \hat{Y}_{ij}) - \bar{C}_{ij}\bar{\ell}(r, \hat{Y}_{ij})\Big) \\ & + \sum_{i=1}^{m}\sum_{j=1}^{n} \bar{C}_{ij}\bar{\ell}(r, \hat{Y}_{ij}), \end{aligned} \tag{31}$$

where the following conditions should apply.

- The cost parameters $\bar{C}_{ij}$ in (31) cannot be arbitrary values. In practice, a constant value is often considered.
- A simple loss function $\bar{\ell}(\cdot, \cdot)$ is needed. In most existing works the squared loss is considered.
- The model function $\hat{y}(\cdot)$ can not be an arbitrary one. Specifically, given a feature vector $\boldsymbol{x}^{i,j}$, $\hat{y}(\boldsymbol{x}^{i,j})$ can be factorized into multiple parts, each of which is only related to $i$ or $j$ and convex in the corresponding parameters. For example, a modified FFM model proposed in [36] satisfies this condition.

Optimization methods have been proposed so that the $O(mn)$ term in the overall cost can be replaced with a smaller $O(m)+O(n)$ term. The main reason why $O(mn)$ can be eliminated is because results on all $i$ and all $j$ can be obtained by using values solely related to $i$ and $j$, respectively. A conceptual illustration is in the following equation.

$$\sum_{i=1}^{m}\sum_{j=1}^{n}(\cdots) = \sum_{i=1}^{m}(\cdots) \quad \times \text{ or } + \quad \sum_{j=1}^{n}(\cdots). \tag{32}$$

By considering the similarity between (27) and (31), with the corresponding restrictions in mind, we propose the following final realization of the framework (26):

$$\min_{\mathcal{W}} \quad \lambda R(\mathcal{W}) + \sum_{(i,j)\in\Omega} \left(\ell(Y_{ij}, \hat{Y}_{ij}) - \omega(A_{ij} - \hat{Y}_{ij})^2\right) + \omega \sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij} - \hat{Y}_{ij})^2. \tag{33}$$

For $(i, j)$ in $\Omega$, we allow general convex loss functions in the original IPS part to maintain the connection to conventional CTR training, while in the direct method part, we follow the above restriction for $\bar{\ell}$ in (31) to consider the squared loss. We then compare the two $O(mn)$ terms in (31) and (33). By respectively excluding $\bar{C}_{ij}$ and $\omega$, ours considers a more complicated setting of $A$ in (28), where all $mn$ entries can be various rather than only a constant value in (31). There remains a cost of $O(mn)$ caused by $A$ in (33). Inspired by the restriction of $\hat{y}(\cdot)$ that is used to have $\hat{Y}_{ij} = \hat{y}(\boldsymbol{x}^{i,j})$, to extend existing optimization methods for (31) on our proposed framework, an additional condition should be considered:

- The imputation model function $\sigma(\cdot)$ can not be an arbitrary one. Specifically, given a feature vector $\boldsymbol{x}^{i,j}$, $\sigma(\boldsymbol{x}^{i,j})$ can be factorized into multiple parts, each of which is only related to $i$ or $j$ but not required to be convex in parameters. Thus the choice of $\sigma(\cdot)$ is broader than that of $\hat{y}(\cdot)$. For example, the multi-view deep learning approach proposed in [7] satisfies this condition.

In our realization, we implement various imputation models from simple to complex, all of which satisfy the above condition; see details in Section 5.2.2.

To use (33), we consider the FFM model for $\hat{y}(\boldsymbol{x}^{i,j})$ and develop an optimization method. Details are in Section A of supplementary materials.

## 5 EXPERIMENTS

We begin by presenting our experimental settings including data sets and parameter selections. Then, we design and conduct a series of experiments to demonstrate the effectiveness and robustness of our proposed framework in Section 4.

### 5.1 Data Sets

We consider the following data sets for experiments, where statistics are described in Table 2.

- **CPC**: This is a large-scale data set for CTR prediction, which includes events of nine days from a real-world CPC system. We consider a two-slot scenario so that for each request, the two ads with the highest expected revenues from 140 candidates are displayed. To have training, validation and test sets under uniform and non-uniform policies, as illustrated in Figure 6, we segment the data to six subsets. The five subsets used in our experiments are described below.
  - $S_t$, $S_c$ defined in Section 2.2, are used separately or together as the training set depending the approach.
  - $S_{\text{va}}$ is the set obtained in the validation period under the uniform logging policy. As an unbiased set, it is used by all methods for the parameter selection.
  - $S_{\text{te}}$ is the set obtained in the test period under the uniform logging policy. As an unbiased set, it is used to check the test performance of all methods.

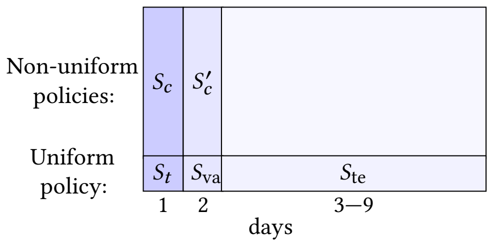


**Figure 6: An illustration of data segmentation according to training, validation, and test periods.**

**Table 2: Data statistics**

| Data set | $m$ | $n$ | $D$ | $\lvert S_c\rvert$ | $\lvert S_t\rvert$ | $\lvert S_{\text{va}}\rvert$ | $\lvert S_{\text{te}}\rvert$ |
|---|---|---|---|---|---|---|---|
| **Yahoo! R3** | 14,877 | 1,000 | 15,877 | 176,816 | 2,631 | 2,775 | 48,594 |
| **CPC** | 21.7M | 140 | 1.4M | 42.8M | 0.35M | 0.31M | 2.0M |

  - $S'_c$ is the set of events logged by non-uniform policies in the validation period. For methods that have involved $S_c$ for training in the validation process, after parameters are selected, $S'_c$ is included for training the final model to predict $S_{\text{te}}$.

  For the feature set, we collect user profiles, contextual information (e.g., time, location) of each request and side information of each advertisement. All are categorical features.
- **Yahoo! R3** [23]: The data set, from a music recommendation service, includes a training and a test set of rated (user, song) pairs in a rating matrix $R$. For the training set, users deliberately selected and rated the songs by preference, which can be considered as a stochastic logging policy by following [30, 32]. For the test set, users were asked to rate 10 songs randomly selected by the system. Because the logging policy of the test set is completely random, the ratings contained in the test set can be considered as the ground truth without the selection bias. In Section B.1 of supplementary material, we give details of modifying the problem to simulate a deterministic logging policy, which is what our framework intends to handle.

### 5.2 Considered Approaches

The considered approaches in our comparison fall into two main categories.

5.2.1 *Conventional Approaches for CTR Prediction.* This category includes approaches that solve the binary classification problem in (4) with the following CTR model.

- FFM: This is the CTR model reviewed in Section 2.1. We consider the slightly modified formulation proposed in [36].

For the training set, we consider the following three combinations of $S_c$ and $S_t$.

- $S_c$: This is a large set including events logged by non-uniform policies; see Section 2.2. This simulates the setting of most systems for CTR prediction.
- $S_t$: This is a small set including events logged by a uniform policy; see Section 2.2. Because the set is free from the selection bias, some works (e.g., [22]) deploy a model trained on $S_t$, though an issue is that $|S_t|$ is small.
- $S_c \cup S_t$: This is the set of all logged events.

*5.2.2 Counterfactual Learning Approaches.* This category includes the following approaches, where all of them consider FFM as the CTR model and $S_c \cup S_t$ as the training set.

- IPS: The approach was reviewed in Section 3.2. For the estimation of propensity scores, we follow [30] to consider the following Naive Bayes propensity estimator.

$$z_l \approx P(O_l = 1 \mid y_l = y) \approx \frac{P(y, O = 1)}{P(y)}, \tag{34}$$

  where $y = \{-1, 1\}$ is the label, $P(y, O = 1)$ is the ratio of events labeled as $y$ in the displayed set, and $P(y)$ is the ratio of the events labeled as $y$ in a selection-bias free set. We estimated them by respectively using $S_c \cup S_t$ and $S_t$.
- CausE: The "causal embedding" approach [3] jointly learns two CTR models $\hat{y}_c(\cdot)$ and $\hat{y}_t(\cdot)$ respectively from $S_c$ and $S_t$. They solve the following optimization problem.

$$\min_{\mathcal{W}_c, \mathcal{W}_t} \frac{1}{|S_c|} \sum_{(y,\boldsymbol{x}) \in S_c} \ell(y, \hat{y}_c(\boldsymbol{x})) + \frac{1}{|S_t|} \sum_{(y,\boldsymbol{x}) \in S_t} \ell(y, \hat{y}_t(\boldsymbol{x}))$$
$$+ \lambda_c R(\mathcal{W}_c) + \lambda_t R(\mathcal{W}_t) + \lambda_{\|t-c\|} \|\mathcal{W}_c - \mathcal{W}_t\|_2^2,$$

  where $\mathcal{W}_c$ and $\mathcal{W}_t$ are are parameters of $\hat{y}_c(\cdot)$ and $\hat{y}_t(\cdot)$, respectively, and the last term controls the discrepancy between $\mathcal{W}_c$ and $\mathcal{W}_t$. The original study in [3] only supports matrix factorization models. We make an extension to support more complex models (e.g., FFM). For the prediction, we follow [3] to use only $\hat{y}_c(\cdot)$.
- New: This approach is a realization of our proposed framework in (33), where we use the logistic loss defined in (5) as $\ell(\cdot, \cdot)$. We use a modified FFM model in [36] for the CTR model $\hat{y}(\cdot)$. It satisfies the conditions needed in Section 4.3. For the imputation model $\sigma(\cdot)$, we consider three settings.
  - avg: This is a simple setting is to impute labels with the average CTR in $S_t$. We use the following logit function to transform a probability to a model output in $[-\infty, \infty]$.

$$\sigma(\boldsymbol{x}^{i,j}) = \log(\frac{\overline{\text{CTR}}}{1 - \overline{\text{CTR}}}), \forall i, j$$

    where $\overline{\text{CTR}}$ is the average CTR from events in $S_t$.
  - item-avg: This is an item-wise extension of avg as

$$\sigma(\boldsymbol{x}^{i,j}) = \log(\frac{\overline{\text{CTR}}_j}{1 - \overline{\text{CTR}}_j}), \forall i$$

    where $\overline{\text{CTR}}_j$ is the average CTR of ad $j$ from events in $S_t$.
  - complex: This is a complex setting to learn $\sigma(\cdot)$ from $S_t$ by solving (4) with the modified FFM model in [36].

We further include two constant predictors as baselines.

- average ($S_c$): This is a constant predictor defined as

$$\hat{y}(\boldsymbol{x}^{i,j}) = \log(\frac{\overline{\text{CTR}}}{1 - \overline{\text{CTR}}}), \forall i, j$$

  where $\overline{\text{CTR}}$ is the average CTR from events in $S_c$.
- average ($S_t$): It is the same as average ($S_c$), except that the average CTR from events in $S_t$ is considered.

**Table 3: A performance comparison of different approaches. We show the relative improvements over the** average ($S_c$) **on test NLL and AUC scores. The best approach is bold-faced.**

| Metric / Approach \ Data set | NLL (Yahoo! R3) | AUC (Yahoo! R3) | NLL (CPC) | AUC (CPC) |
|---|---|---|---|---|
| average ($S_c$) | +0.0% | +0.0% | +0.0% | +0.0% |
| average ($S_t$) | **+79.1%** | +0.0% | +29.6% | +0.0% |
| FFM ($S_c$) | -7.7% | +36.4% | -16.9% | +32.2% |
| FFM ($S_t$) | -20.7% | +4.6% | +32.4% | +42.8% |
| FFM ($S_c \cup S_t$) | +0.2% | +36.4% | +10.6% | +34.6% |
| IPS | +62.2% | +23.2% | +28.5% | +31.2% |
| CausE | +6.8% | +37.8% | +25.3% | +39.4% |
| New (avg) | **+79.1%** | +51.8% | +32.9% | +48.6% |
| New (item-avg) | +76.8% | **+54.2%** | +32.4% | +48.2% |
| New (complex) | -0.2% | +37.4% | **+33.4%** | **+50.6%** |

**Table 4: The performance when $S_t$ becomes smaller sets. $|S_t|$ indicates the size of $S_t$ as the percentage of total data. We present the relative improvements over the** average ($S_c$) **on test NLL and AUC scores. The CPC set is considered and the best approach is bold-faced.**

| Metric / Approach \ $\lvert S_t \rvert$ | NLL (0.1%) | AUC (0.1%) | NLL (0.01%) | AUC (0.01%) |
|---|---|---|---|---|
| average ($S_t$) | +29.2% | +0.0% | +28.7% | +0.0% |
| FFM ($S_t$) | +28.6% | +18.6% | +26.5% | +1.2% |
| FFM ($S_c \cup S_t$) | -12.5% | +29.6% | -16.9% | +32.0% |
| IPS | +26.1% | +26.8% | +18.7% | +23.6% |
| CausE | +25.6% | +37.4% | +22.4% | +31.6% |
| New (avg) | **+33.5%** | **+48.4%** | +30.4% | **+47.2%** |
| New (item-avg) | +31.5% | +46.6% | +26.7% | +46.0% |
| New (complex) | +32.2% | +44.6% | **+32.5%** | +43.0% |

For evaluation criteria, we consider NLL and AUC respectively defined in (8) and (9) and report relative improvements over the method average ($S_c$) as

$$\frac{criterion_{approach} - criterion_{\text{average }(S_c)}}{|criterion_{\text{average }(S_c)}|},$$

where *criterion* is either NLL or AUC, and *approach* is the approach being measured. Because the performance of average ($S_c$) is unchanged in all experiments, results presented in different tables can be comparable. Details of our parameter selection procedure are in supplementary materials.

## 5.3 Comparison on Various Approaches

By comparing models in Table 3, we have the following observations.

- For two constant predictors, NLL scores of average ($S_t$) are much better than those of average ($S_c$) estimated from the biased $S_c$ set. The reason is that the distribution of $S_t$ is consistent with that of the test set. However, because average ($S_t$) is still a constant predictor, its AUC scores are the same as those of average ($S_c$).

- For FFM ($S_t$), it performs much better than FFM ($S_c$) and FFM ($S_c \cup S_t$) on **CPC** set. This result confirms our conjecture in Section 2.2 that conventional approaches considering only displayed events selected by non-uniform policies may cause a strong selection bias. However, the result of FFM ($S_t$) on **Yahoo! R3** set is not ideal. The reason is that the feature set of **Yahoo! R3** is too sparse by having only two identifier features[3] and $S_t$ is too small to learn an FFM model.
- IPS has good NLL scores, but its AUC scores are not competitive. In (34) to estimate propensity scores, the set $S_t$ is used. This may explain why the obtained NLL is similar to that by average ($S_t$). On the other hand, the unsatisfactory AUC results may be because the Naive Bayes estimator used in (34) is too simple.
- For CausE, it performs better than FFM ($S_c$) and FFM ($S_c \cup S_t$) but worse than FFM ($S_t$) on the **CPC** set. The reason might be that models learnt from $S_t$ are better than those learnt from $S_c$, but we follow [3] to use only $\hat{y}_c(\cdot)$ for prediction.
- Our proposed framework performs significantly better than others, though an exception is New (complex) on the **Yahoo! R3** set. The reason is analogous to the explanation of the bad result of FFM ($S_t$) on **Yahoo! R3** set where $S_t$ is too small. In contrast, for the **CPC** set, because $S_t$ is large enough to learn a complex imputation model, New (complex) performs the best.

## 5.4 Impact of Unbiased Imputation Models Learnt from $S_t$

In Section 4.2, we proposed unbiased imputation models because past works [6, 19] find that learning from biased data sets will propagate the carried selection bias to non-displayed events and worsens the performance. We conduct an experiment to verify this conjecture and the importance of using an unbiased imputation model. Detailed results are in Section B.3 of supplementary materials.

## 5.5 Impact of the Size of $S_t$

To examine the robustness of the approaches relying on $S_t$ to learn an unbiased imputation model, we shrink $S_t$ to see how they perform on the **CPC** set. For experiments in Section 5.3, $S_t$ includes 1% of the data. We now select two subsets respectively having 0.1% and 0.01% of total data. These subsets are then used as $S_t$ for experiments, though each model's parameters must be re-selected. The set $S_c$ remains unchanged.

The result is in Table 4. By comparing the third column of Table 3, which shows the result of using 1% data as $S_t$, we observe that with the shrinkage of $S_t$, the performance of all approaches gradually becomes worse. However, our proposed framework still performs significantly better than others. In particular, for other approaches AUC values are strongly affected by the size of $S_t$, but ours remain similar. Among the three settings of our framework, the simple setting New (avg) of using average CTR in $S_t$ as the imputation model is competitive. This might be because for small $S_t$, training a complicated imputation model is not easy.

[3]See details of the **Yahoo! R3** set in supplementary materials

## 6 CONCLUSION

In this work, through an A/B test on our system, we show that conventional approaches for CTR prediction considering only displayed events may cause a strong selection bias and inaccurate predictions. To alleviate the bias, we need to conduct counterfactual learning by considering non-displayed events. We point out some difficulties for applying counterfactual learning techniques in real-world advertising systems.

(1) For many advertising systems, the thresholding operation (25) is applied as a deterministic logging policy, but existing approaches may require stochastic policies.
(2) Learning imputation models from displayed events causes the selection bias being propagated to non-displayed events.
(3) The high cost of handling all displayed and non-displayed events causes some approaches impractical for large systems.

To overcome these difficulties, we propose a novel framework for counterfactual CTR prediction. We introduce a propensity-free doubly robust method. Then we propose a setting to obtain unbiased imputation models in our framework. For efficient training we link the proposed framework to recommender systems with implicit feedbacks and adapt algorithms developed there. Experiments conducted on real-world data sets show our proposed framework outperforms not only conventional approaches for CTR prediction but also some existing counterfactual learning methods.

## REFERENCES

[1] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *WWW*.
[2] Mathieu Blondel, Masakazu Ishihata, Akinori Fujino, and Naonori Ueda. 2016. Polynomial Networks and Factorization Machines: new Insights and Efficient Training Algorithms. In *ICML*.
[3] Stephen Bonner and Flavian Vasile. 2018. Causal Embeddings for Recommendation. In *RecSys*.
[4] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2015. Simple and scalable response prediction for display advertising. *ACM TIST* 5, 4 (2015), 61:1–61:34.
[5] Wei-Sheng Chin, Bo-Wen Yuan, Meng-Yuan Yang, and Chih-Jen Lin. 2018. An Efficient Alternating Newton Method for Learning Factorization Machines. *ACM TIST* 9 (2018), 72:1–72:31.
[6] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. In *ICML*.
[7] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*.
[8] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI*.
[9] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *ADKDD*.
[10] James J. Heckman. 1979. Sample selection bias as a specification error. *Econometrica* 47, 1 (1979), 153–161.
[11] Daniel G. Horvitz and Donovan J. Thompson. 1952. A Generalization of Sampling Without Replacement From a Finite Universe. *J. Amer. Statist. Assoc.* 47 (1952), 663–685.
[12] Cho-Jui Hsieh, Nagarajan Natarajan, and Inderjit Dhillon. 2015. PU Learning for Matrix Completion. In *ICML*.
[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*.
[14] Machael Jahrer, Andreas Töscher, Jeong-Yoon Lee, Jingjing Deng, Hang Zhang, and Jacob Spoelstra. 2012. Ensemble of Collaborative Filtering and Feature Engineered Model for Click Through Rate Prediction. In *KDD Cup 2012 Workshop*.
[15] Yuchin Juan, Damien Lefortier, and Olivier Chapelle. 2017. Field-aware factorization machines in a real-world online advertising system. In *WWW*.
[16] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *RecSys*.

[17] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42 (2009).

[18] Himabindu Lakkaraju, Jon Kleinberg, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. 2017. The selective labels problem: Evaluating algorithmic predictions in the presence of unobservables. In *KDD*.

[19] Carolin Lawrence, Artem Sokolov, and Stefan Riezler. 2017. Counterfactual Learning from Bandit Feedback under Deterministic Logging: A Case Study in Statistical Machine Translation. In *EMNLP*.

[20] Damien Lefortier, Adith Swaminathan, Xiaotao Gu, Thorsten Joachims, and Maarten de Rijke. 2016. Large-scale validation of counterfactual learning methods: A test-bed. In *NIPS Workshop on Inference and Learning of Hypothetical and Counterfactual Interventions in Complex Systems*.

[21] Roderick J. A. Little and Donald B. Rubin. 2002. *Statistical Analysis with Missing Data* (second ed.). Wiley-Interscience.

[22] David C. Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C. Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related Pins at Pinterest: The Evolution of a Real-World Recommender System. In *WWW*.

[23] Benjamin M. Marlin and Richard S. Zemel. 2009. Collaborative Prediction and Ranking with Non-random Missing Data. In *RecSys*.

[24] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad Click Prediction: a View from the Trenches. In *KDD*.

[25] Yusuke Narita, Shota Yasui, and Kohei Yata. 2019. Efficient Counterfactual Learning from Bandit Feedback. In *AAAI*.

[26] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2018. Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data. *ACM TOIS* 37 (2018), 5:1–5:35.

[27] Steffen Rendle. 2010. Factorization machines. In *ICDM*.

[28] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ADs. In *WWW*.

[29] Nir Rosenfeld, Yishay Mansour, and Elad Yom-Tov. 2017. Predicting counterfactuals from large historical data and small randomized trials. In *WWW*.

[30] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations As Treatments: Debiasing Learning and Evaluation. In *ICML*.

[31] Adith Swaminathan and Thorsten Joachims. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. *JMLR* 16 (2015), 1731–1755.

[32] Longqi Yang, Yin Cui, Yuan Xuan, Chenyang Wang, Serge Belongie, and Deborah Estrin. 2018. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In *RecSys*.

[33] Hsiang-Fu Yu, Mikhail Bilenko, and Chih-Jen Lin. 2017. Selection of Negative Samples for One-class Matrix Factorization. In *SDM*.

[34] Hsiang-Fu Yu, Hsin-Yuan Huang, Inderjit S. Dihillon, and Chih-Jen Lin. 2017. A Unified Algorithm for One-class Structured Matrix Factorization with Side Information. In *AAAI*.

[35] Bowen Yuan, Yaxu Liu, Jui-Yang Hsia, Zhenhua Dong, and Chih-Jen Lin. 2020. Unbiased Ad click prediction for position-aware advertising systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*. http://www.csie.ntu.edu.tw/~cjlin/papers/debiases/debiases.pdf

[36] Bowen Yuan, Meng-Yuan Yang, Jui-Yang Hsia, Hong Zhu, Zhirong Liu, Zhenhua Dong, and Chih-Jen Lin. 2019. *One-class Field-aware Factorization Machines for Recommender Systems with Implicit Feedbacks*. Technical Report. National Taiwan University. http://www.csie.ntu.edu.tw/~cjlin/papers/ocffm/imp_ffm.pdf

[37] Weinan Zhang, Tianxiong Zhou, Jun Wang, and Jian Xu. 2016. Bid-aware gradient descent for unbiased learning with censored data in display advertising. In *KDD*.

[38] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *KDD*.

# Supplementary Materials for "Improving Ad Click Prediction by Considering Non-displayed Events"

## A DETAILS OF EFFICIENT TRAINING

As we mentioned in Section 4.3, to solve (33) without $O(mn)$ costs, the CTR model and the imputation model should satisfy some conditions. In this section, we take a modified FFM model proposed in [36] as an example to present our efficient training algorithm. The modified FFM model, which is a multi-block convex function [2], is considered as both CTR and imputation models.

We note that [36] considers a slightly modified FFM model to make each sub-problem convex, where the idea is extended from related works in [2, 5]. If we let

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_F \end{bmatrix}$$

be considered as a concatenation of $F$ sub-vectors, where $\boldsymbol{x}_f \in \mathcal{R}^{D_f}$ includes $D_f$ features which belong to the field $f$, then the modified function is

$$\begin{aligned} \hat{y}(\boldsymbol{x}) &= \sum_{f_1=1}^{F} \sum_{f_2=f_1}^{F} (W_{f_1}^{f_2 T} \boldsymbol{x}_{f_1})^T (H_{f_2}^{f_1 T} \boldsymbol{x}_{f_2}) \\ &= \sum_{f_1=1}^{F} \sum_{f_2=f_1}^{F} \boldsymbol{p}_{f_1,f_2}{}^T \boldsymbol{q}_{f_2,f_1}, \end{aligned} \tag{A.1}$$

where

$$W_{f_1}^{f_2} = [\boldsymbol{w}_{1,f_2}, \cdots, \boldsymbol{w}_{D_{f_1},f_2}]^T \in \mathcal{R}^{D_{f_1} \times k},$$

$$H_{f_2}^{f_1} = \begin{cases} W_{f_2}^{f_1} & f_1 \neq f_2, \\ [\boldsymbol{h}_{1,f_1}, \cdots, \boldsymbol{h}_{D_{f_2},f_1}]^T & f_1 = f_2, \end{cases}$$

$$\boldsymbol{p}_{f_1,f_2} = W_{f_1}^{f_2 T} \boldsymbol{x}_{f_1} \text{ and } \boldsymbol{q}_{f_2,f_1} = H_{f_2}^{f_1 T} \boldsymbol{x}_{f_2}. \tag{A.2}$$

We do not show that (A.1) is a minor modification from (7); for details, please see [36].

In recommendation, each $\boldsymbol{x}$ is usually a combination of user feature $\boldsymbol{u}$ and item feature $\boldsymbol{v}$. For $\boldsymbol{x}^l$ corresponding to user $i$ and item $j$, $\boldsymbol{x}^l$ can be written as

$$\boldsymbol{x}^{i,j} = \begin{bmatrix} \boldsymbol{u}^i \\ \boldsymbol{v}^j \end{bmatrix}.$$

Under any given field $f$, the vector $\boldsymbol{x}_f^{i,j}$ consists of two sub-vectors respectively corresponding to request $i$ and ad $j$.

$$\boldsymbol{x}_f^{i,j} = \begin{cases} \boldsymbol{u}_f^i & f \leq F_u, \\ \boldsymbol{v}_f^j & f > F_u. \end{cases} \tag{A.3}$$

where $F_{\boldsymbol{u}}$ is the number of user fields and $F_{\boldsymbol{v}}$ is number of item fields. The corresponding output function is

$$\begin{aligned} \hat{y}(\boldsymbol{x}^{i,j}) &= \sum_{f_1=1}^{F_u+F_v} \sum_{f_2=f_1}^{F_u+F_v} (W_{f_1}^{f_2 T} \boldsymbol{x}_{f_1}^{i,j})^T (H_{f_2}^{f_1 T} \boldsymbol{x}_{f_2}^{i,j}) \\ &= \sum_{f_1=1}^{F} \sum_{f_2=f_1}^{F} \boldsymbol{p}_{f_1,f_2}^{i,j}{}^T \boldsymbol{q}_{f_2,f_1}^{i,j}, \end{aligned} \tag{A.4}$$

where

$$\boldsymbol{p}_{f_1,f_2}^{i,j} = W_{f_1}^{f_2 T} \boldsymbol{x}_{f_1}, \ \boldsymbol{q}_{f_2,f_1}^{i,j} = H_{f_2}^{f_1 T} \boldsymbol{x}_{f_2}. \tag{A.5}$$

With the reformulation, our proposed (33) has a multi-block objective function. At each cycle of the block CD method we sequentially solve one of $F(F+1)$ convex sub-problems by the following loop.

```
1: for $f_1 \leftarrow \{1 \cdots F\}$ do
2:     for $f_2 \leftarrow \{f_1 \cdots F\}$ do
3:         Solve a sub-problem of $W_{f_1}^{f_2}$ by fixing others
4:         Solve a sub-problem of $H_{f_2}^{f_1}$ by fixing others
5:     end for
6: end for
```

If we consider to update the block $W_{f_1}^{f_2}$ corresponding to fields $f_1$ and $f_2$, then the convex sub-problem is to minimize

$$\begin{aligned} &\frac{\lambda}{2} R(W_{f_1}^{f_2}) + \sum_{(i,j) \in \Omega} (\ell(A_{ij}, \hat{Y}_{ij}) - \omega(A_{ij} - \hat{Y}_{ij})^2) \\ &\quad + \sum_{i=1}^{m} \sum_{j=1}^{n} \omega(A_{ij} - \hat{Y}_{ij})^2, \end{aligned} \tag{A.6}$$

where l2 regularization is considered. That is, $R(W) = \|W\|_F^2$. In [36], the sub-problem is a special case of (A.6), where the objective function is quadratic because of the squared loss function. Thus solving the sub-problem in [36] is the same as solving a linear system by taking the first derivative to be zero. In contrast, ours in (A.6) is a general convex function, so some differentiable optimization techniques such as gradient descent, Quasi-Newton or Newton are needed. All these methods must calculate the gradient (i.e., the first derivative) or the Hessian-vector product (where Hessian is the second-order derivative), but the $O(mn)$ cost is the main concern.

In next content, by following [34, 36], we develop an efficient Newton method for solving (A.6), where any $O(mn)$ cost can be avoided.

### A.1 Newton Methods for Solving Sub-problems

We discuss details in solving the sub-problem of $W_{f_1}^{f_2}$ in (A.6). Following [36], for easy analysis, we write (A.6) to a vector form

$$f(\tilde{\boldsymbol{w}}),$$

where $\tilde{\boldsymbol{w}} = \text{vec}(W_{f_1}^{f_2})$ and vec$(\cdot)$ stacks the columns of a matrix. We consider Newton methods to solve the sub-problem.

An iterative procedure is conducted in a Newton method. Each Newton iteration minimizes a second-order approximation of the function to obtain a updating direction $\boldsymbol{s}$

$$\min_{\boldsymbol{s}} \nabla f(\tilde{\boldsymbol{w}})^T \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T \nabla^2 f(\tilde{\boldsymbol{w}})\boldsymbol{s}. \tag{A.7}$$

Because $f(\tilde{\boldsymbol{w}})$ is convex, the direction $\boldsymbol{s}$ can be obtained by solving the following linear system

$$\nabla^2 f(\tilde{\boldsymbol{w}})\boldsymbol{s} = -\nabla f(\tilde{\boldsymbol{w}}). \tag{A.8}$$

We follow [36] to solve (A.7) by a iterative procedure called the conjugate gradient method (CG). The main reason of using CG is that $\nabla^2 f(\tilde{\boldsymbol{w}})$ is too large to be stored, and each CG step requires mainly the product between $\nabla^2 f(\tilde{\boldsymbol{w}})$ and a vector $\boldsymbol{v}$

$$\nabla^2 f(\tilde{\boldsymbol{w}})\boldsymbol{v}, \tag{A.9}$$

which, with the problem structure, may be conducted without explicitly forming $\nabla^2 f(\tilde{\boldsymbol{w}})$.

A difference from [36] is that the sub-problem in [36] is a quadratic function because of the squared loss. Thus, an optimization procedure like the Newton method is not needed. Instead, only one linear system is solved by CG.

To ensure the convergence of the Newton method, after obtaining an updating direction $\boldsymbol{s}$, a line search procedure is conducted to find a suitable step size $\theta$. Then we update the $\tilde{\boldsymbol{w}}$ by

$$\tilde{\boldsymbol{w}} \leftarrow \tilde{\boldsymbol{w}} + \theta \boldsymbol{s}.$$

We follow the standard backtracking line search to check a sequence $1, \beta, \beta^2, \cdots$ and choose the largest $\theta$ such that the sufficient decrease of the function value is obtained

$$f(\tilde{\boldsymbol{w}} + \theta\boldsymbol{s}) - f(\tilde{\boldsymbol{w}}) \le \theta\nu\nabla f^T(\tilde{\boldsymbol{w}})\boldsymbol{s}. \tag{A.10}$$

where $\beta, \nu \in (0, 1)$ are pre-specified constants.

## A.2 Algorithm Details

To discuss techniques for addressing the issue of $O(mn)$ complexity, let

$$\tilde{\boldsymbol{w}} = \text{vec}(W_{f_1}^{f_2})$$

and re-write (A.6) as

$$f(\tilde{\boldsymbol{w}}) = \frac{\lambda}{2}\|\tilde{\boldsymbol{w}}\|_2^2 + L^+(\tilde{\boldsymbol{w}}) + L^-(\tilde{\boldsymbol{w}}), \tag{A.11}$$

where

$$L^+(\tilde{\boldsymbol{w}}) = \sum_{(i,j)\in\Omega} \ell(Y_{ij}, \hat{Y}_{ij}) - \omega\frac{1}{2}(A_{ij} - \hat{Y}_{ij})^2 \tag{A.12}$$

$$L^-(\tilde{\boldsymbol{w}}) = \omega\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{1}{2}(A_{ij} - \hat{Y}_{ij})^2.$$

Then the gradient and Hessian-vector product can be respectively re-written as

$$\nabla \tilde{f}(\tilde{\boldsymbol{w}}) = \lambda\tilde{\boldsymbol{w}} + \nabla L^+(\tilde{\boldsymbol{w}}) + \nabla L^-(\tilde{\boldsymbol{w}}),$$

$$\nabla^2 \tilde{f}(\tilde{\boldsymbol{w}})\boldsymbol{v} = \lambda\boldsymbol{v} + \nabla^2 L^+(\tilde{\boldsymbol{w}})\boldsymbol{v} + \nabla^2 L^-(\tilde{\boldsymbol{w}})\boldsymbol{v}.$$

*A.2.1 The Computation of the Right-hand Side of the Linear System.* To efficiently calculate the gradient, [36] takes a crucial property of recommender systems into account.

For each request $i$ and ad $j$ pairs, there are $F(F + 1)$ numbers of feature embedding vectors according to different field combinations of $f_1$ and $f_2$. From (A.2) and (A.3),

$$\boldsymbol{p}_{f_1,f_2}^{i,j} \rightarrow \begin{cases} \boldsymbol{p}_{f_1,f_2}^{i} = W_{f_1}^{f_2\,T}\boldsymbol{u}_{f_1}^{i} & f_1 \le F_u, \\ \boldsymbol{p}_{f_1,f_2}^{j} = W_{f_1}^{f_2\,T}\boldsymbol{v}_{f_1}^{j} & f_1 > F_u, \end{cases}$$

and

$$\boldsymbol{q}_{f_1,f_2}^{i,j} \rightarrow \begin{cases} \boldsymbol{q}_{f_2,f_1}^{i} = H_{f_2}^{f_1\,T}\boldsymbol{u}_{f_2}^{i} & f_1 \le F_u, \\ \boldsymbol{q}_{f_2,f_1}^{j} = H_{f_2}^{f_1\,T}\boldsymbol{v}_{f_2}^{j} & f_1 > F_u. \end{cases}$$

When updating $\text{vec}(W_{f_1}^{f_2})$, the value $\hat{Y}_{ij}$ can be written as

$$\begin{aligned} \boldsymbol{x}_{f_1}^{i,j\,T} W_{f_1}^{f_2}\boldsymbol{q}_{f_2,f_1}^{i,j} + \text{const.} &= \text{vec}(W_{f_1}^{f_2})^T(\boldsymbol{q}_{f_2,f_1}^{i,j} \otimes \boldsymbol{x}_{f_1}^{i,j}) + \text{const.} \\ &= \tilde{\boldsymbol{w}}^T(\boldsymbol{q}_{f_2,f_1}^{i,j} \otimes \boldsymbol{x}_{f_1}^{i,j}) + \text{const.} \\ &= \tilde{\boldsymbol{w}}^T\,\text{vec}(\boldsymbol{x}_{f_1}^{i,j}\boldsymbol{q}_{f_2,f_1}^{i,j\;T}) + \text{const.} \end{aligned} \tag{A.13}$$

and its first derivative $(\hat{Y}_{ij})'$ with respect to $\tilde{\boldsymbol{w}}$ is

$$\tilde{\boldsymbol{x}}_{f_1}^{i,j} = \text{vec}(\boldsymbol{x}_{f_1}^{i,j}\boldsymbol{q}_{f_2,f_1}^{i,j\;T}).$$

Then $\nabla L^+(\tilde{\boldsymbol{w}})$ and $\nabla L^-(\tilde{\boldsymbol{w}})$ can be computed as

$$\begin{aligned} \nabla L^+(\tilde{\boldsymbol{w}}) &= \sum_{(i,j)\in\Omega}(\ell'(Y_{ij}, \hat{Y}_{ij}) - \omega(\hat{Y}_{ij} - A_{ij}))(\hat{Y}_{ij})' \\ &= \sum_{(i,j)\in\Omega}\text{vec}((\ell'(Y_{ij}, \hat{Y}_{ij}) - \omega(\hat{Y}_{ij} - A_{ij}))\boldsymbol{x}_{f_1}^{i,j}\boldsymbol{q}_{f_2,f_1}^{i,j\;T}) \end{aligned} \tag{A.14}$$

where $\ell'(Y, \hat{Y})$ indicates the derivative with respect to $\hat{Y}$, and

$$\begin{aligned} \nabla L^-(\tilde{\boldsymbol{w}}) &= \sum_{i=1}^{m}\sum_{j=1}^{n}\text{vec}(\omega(\hat{Y}_{ij} - A_{ij})(\hat{Y}_{ij})' \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n}\text{vec}(\omega(\hat{Y}_{ij} - A_{ij})\boldsymbol{x}_{f_1}^{i,j}\boldsymbol{q}_{f_2,f_1}^{i,j\;T}). \end{aligned} \tag{A.15}$$

From (A.4),

$$\hat{Y}_{ij} = \sum_{\alpha=1}^{F}\sum_{\beta=\alpha}^{F}\boldsymbol{p}_{\alpha,\beta}^{i,j\;T}\boldsymbol{q}_{\beta,\alpha}^{i,j}.$$

The $\nabla L^+(\tilde{\boldsymbol{w}})$ evaluation requires the summation of $O(|\Omega|)$ terms. With $|\Omega| \ll mn$, $\nabla L^+(\tilde{\boldsymbol{w}})$ can be easily calculated but the bottleneck is on $\nabla L^-(\tilde{\boldsymbol{w}})$, which sum up $O(mn)$ terms.

In order to deal with the $O(mn)$ cost, in [36], an efficient approach has been provided to compute

$$\nabla L^-(\tilde{\boldsymbol{w}}) = \sum_{i=1}^{m}\sum_{j=1}^{n}\text{vec}(\omega(\hat{Y}_{ij} - r)\boldsymbol{x}_{f_1}^{i,j}\boldsymbol{q}_{f_2,f_1}^{i,j\;T}). \tag{A.16}$$

where all $A_{ij}$ are treated as a constant $r$. We can leverage their technique by breaking (A.15) into two parts

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\text{vec}(\omega\hat{Y}_{ij}\boldsymbol{x}_{f_1}^{i,j}\boldsymbol{q}_{f_2,f_1}^{i,j\;T}) \tag{A.17}$$

and

$$-\sum_{i=1}^{m}\sum_{j=1}^{n}\text{vec}(\omega A_{ij}\boldsymbol{x}_{f_1}^{i,j}\boldsymbol{q}_{f_2,f_1}^{i,j\,T}). \tag{A.18}$$

It is clear that (A.17) is actually a specific case in [36] with $r = 0$. Furthermore, since a FFM imputation model is used, $\hat{Y}_{ij}$ and $A_{ij}$ have an identical structure.

$$A_{ij} = \sum_{\alpha=1}^{F}\sum_{\beta=\alpha}^{F}\boldsymbol{p'}_{\alpha,\beta}^{i,j\,T}\boldsymbol{q'}_{\beta,\alpha}^{i,j} \tag{A.19}$$

where

$$\boldsymbol{p'}_{f_1,f_2}^{i,j} \rightarrow \begin{cases}\boldsymbol{p'}_{f_1,f_2}^{i} = W'^{f_2\,T}_{f_1}\boldsymbol{u}_{f_1}^{i} & f_1 \le F_u,\\ \boldsymbol{p'}_{f_1,f_2}^{j} = W'^{f_2\,T}_{f_1}\boldsymbol{v}_{f_1}^{j} & f_1 > F_u,\end{cases}$$

and

$$\boldsymbol{q'}_{f_1,f_2}^{i,j} \rightarrow \begin{cases}\boldsymbol{q'}_{f_2,f_1}^{i} = H'^{f_1\,T}_{f_2}\boldsymbol{u}_{f_2}^{i} & f_1 \le F_u,\\ \boldsymbol{q'}_{f_2,f_1}^{j} = H'^{f_1\,T}_{f_2}\boldsymbol{v}_{f_2}^{j} & f_1 > F_u.\end{cases}$$

We have that $W'$ and $H'$ are the imputation model's embedding matrixes. Therefore, we can apply same computation method on (A.17) and (A.18).

The computation of (A.17) depends on the following three cases

$$\begin{cases} f_1, f_2 \le F_u,\\ f_1, f_2 > F_u,\\ f_1 \le F_u, f_2 > F_u.\end{cases}$$

Let

$$P_{f_1}^{f_2} = \left[\boldsymbol{p}_{f_1,f_2}^{1},\cdots,\boldsymbol{p}_{f_1,f_2}^{m}\right]^T, Q_{f_2}^{f_1} = \left[\boldsymbol{q}_{f_2,f_1}^{1},\cdots,\boldsymbol{q}_{f_2,f_1}^{n}\right]^T,$$

$$P'^{f_2}_{f_1} = \left[\boldsymbol{p'}_{f_1,f_2}^{1},\cdots,\boldsymbol{p'}_{f_1,f_2}^{m}\right]^T, Q'^{f_1}_{f_2} = \left[\boldsymbol{q'}_{f_2,f_1}^{1},\cdots,\boldsymbol{q'}_{f_2,f_1}^{n}\right]^T,$$

$$U_{f_1} = \left[\boldsymbol{u}_{f_1}^{1},\cdots,\boldsymbol{u}_{f_1}^{m}\right]^T.$$

- For the first case, (A.17) can be computed by

$$U_{f_1}^T\,\text{diag}(\boldsymbol{z})Q_{f_2}^{f_1},$$

where

$$z_i = \omega(na_i + \sum_{j=1}^{n} b_j + d_i),$$

$$\boldsymbol{d} = \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(P_{\alpha}^{\beta}(Q_{\beta}^{\alpha\,T}\mathbf{1}_{n\times 1})) \in \mathcal{R}^m,$$

$$a_i = \sum_{\alpha=1}^{F_u}\sum_{\beta=\alpha}^{F_u}\boldsymbol{p}_{\alpha,\beta}^{i\,T}\boldsymbol{q}_{\beta,\alpha}^{i}, \text{ and } b_j = \sum_{\alpha=F_u+1}^{F}\sum_{\beta=\alpha}^{F}\boldsymbol{p}_{\alpha,\beta}^{j\,T}\boldsymbol{q}_{\beta,\alpha}^{j}.$$

Similarly, (A.18) can be computed by

$$-U_{f_1}^T\,\text{diag}(\boldsymbol{z'})Q_{f_2}^{f_1},$$

where

$$z'_i = \omega(na'_i + \sum_{j=1}^{n} b'_j + d'_i),$$

$$\boldsymbol{d'} = \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(P'^{\beta}_{\alpha}(Q'^{\alpha\,T}_{\beta}\mathbf{1}_{n\times 1})) \in \mathcal{R}^m,$$

$$a'_i = \sum_{\alpha=1}^{F_u}\sum_{\beta=\alpha}^{F_u}\boldsymbol{p'}_{\alpha,\beta}^{i\,T}\boldsymbol{q'}_{\beta,\alpha}^{i}, \text{ and } b'_j = \sum_{\alpha=F_u+1}^{F}\sum_{\beta=\alpha}^{F}\boldsymbol{p'}_{\alpha,\beta}^{j\,T}\boldsymbol{q'}_{\beta,\alpha}^{j}.$$

The overall summation of (A.14) and (A.15) can be written as

$$U_{f_1}^T\,\text{diag}(\bar{\boldsymbol{z}})Q_{f_2}^{f_1},$$

where

$$\bar{z}_i = z_i - z'_i + \sum_{j\in\Omega_i}(\ell'(Y_{ij},\hat{Y}_{ij}) - \omega(\hat{Y}_{ij} - A_{ij}))$$

and $\Omega_i = \{j \mid (i,j) \in \Omega\}$.

- For the second case, let

$$V_{f_1} = \left[\boldsymbol{v}_{f_1}^{1},\cdots,\boldsymbol{v}_{f_1}^{n}\right]^T.$$

Then (A.17) can be computed as

$$V_{f_1}^T\,\text{diag}(\boldsymbol{z})Q_{f_2}^{f_1},$$

where

$$z_j = \omega(\sum_{i=1}^{m} a_i + mb_j + e_j),$$

and

$$\boldsymbol{e} = \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(Q_{\beta}^{\alpha}(P_{\alpha}^{\beta\,T}\mathbf{1}_{m\times 1})) \in \mathcal{R}^n,$$

Similarly, (A.18) can be computed as

$$-V_{f_1}^T\,\text{diag}(\boldsymbol{z'})Q'^{f_1}_{f_2},$$

where

$$z'_j = \omega(\sum_{i=1}^{m} a'_i + mb'_j + e'_j),$$

and

$$\boldsymbol{e'} = \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(Q'^{\alpha}_{\beta}(P'^{\beta\,T}_{\alpha}\mathbf{1}_{m\times 1})) \in \mathcal{R}^n,$$

The overall summation of (A.15) and (A.14) can be written as

$$V_{f_1}^T\,\text{diag}(\bar{\boldsymbol{z}}')Q_{f_2}^{f_1},$$

where

$$\bar{z}_j = z_j - z'_j + \sum_{i\in\bar{\Omega}_j}(\ell'(Y_{ij},\hat{Y}_{ij}) - \omega(\hat{Y}_{ij} - A_{ij}))$$

and $\bar{\Omega}_j = \{i \mid (i,j) \in \Omega\}$.

- Considering the third case, (A.17) can be computed as

$$\omega((U_{f_1}^T\boldsymbol{a})\boldsymbol{q}_{\text{o}}^T + (U_{f_1}^T\mathbf{1}_{m\times 1})\boldsymbol{q}_{\text{b}}^T + U_{f_1}^T T),$$

where

$$T = \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(P_{\alpha}^{\beta}(Q_{\beta}^{\alpha\,T}Q_{f_2}^{f_1})),$$

$$\boldsymbol{q}_{\text{o}} = Q_{f_2}^{f_1\,T}\mathbf{1}_{n\times 1}, \text{ and } \boldsymbol{q}_{\text{b}} = Q_{f_2}^{f_1\,T}\boldsymbol{b}.$$

Similarly, the (A.18) can be computed as

$$-\omega((U_{f_1}^T\boldsymbol{a'})\boldsymbol{q}_{\text{o}}{}^T + (U_{f_1}^T\mathbf{1}_{m\times 1})\boldsymbol{q}'_{\text{b}}{}^T + U_{f_1}^T T'),$$

where

$$T' = \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(P'^{\beta}_{\alpha}(Q'^{\alpha\,T}_{\beta}Q_{f_2}^{f_1}))$$

and

$$\boldsymbol{q}'_{\text{b}} = Q_{f_2}^{f_1 T}\boldsymbol{b}'.$$

The overall summation of (A.15) and (A.14) can be written as

$$U_{f_1}^T \begin{bmatrix} \boldsymbol{z}_1^T \\ \vdots \\ \boldsymbol{z}_m^T \end{bmatrix} + \omega((U_{f_1}^T \boldsymbol{a})\boldsymbol{q}_{\text{o}}^T + (U_{f_1}^T \mathbf{1}_{m\times 1})\boldsymbol{q}_{\text{b}}^T + U_{f_1}^T T)$$

$$- \omega((U_{f_1}^T \boldsymbol{a}')\boldsymbol{q}_{\text{o}}{}^T + (U_{f_1}^T \mathbf{1}_{m\times 1})\boldsymbol{q}_{\text{b}}'{}^T + U_{f_1}^T T'),$$

where

$$\boldsymbol{z}_i = \sum_{j\in\Omega_i} (\ell'(Y_{ij}, \hat{Y}_{ij}) - \omega(\hat{Y}_{ij} - A_{ij}))\boldsymbol{q}_{f_2,f_1}^j.$$

*A.2.2 The computation of Matrix-vector Products.* From the representation of $\hat{Y}_{ij}$ in (A.13), (A.14) and (A.15),

$$\nabla^2 L^+(\tilde{\boldsymbol{w}}) = \sum_{(i,j)\in\Omega} (\ell''(Y_{ij}, \hat{Y}_{ij}) - \omega)\tilde{\boldsymbol{x}}_{f_1}^{i,j}\tilde{\boldsymbol{x}}_{f_1}^{i,jT} \quad \text{(A.20)}$$

and

$$\nabla^2 L^-(\tilde{\boldsymbol{w}}) = \omega \sum_{i=1}^{m}\sum_{j=1}^{n} \tilde{\boldsymbol{x}}_{f_1}^{i,j}\tilde{\boldsymbol{x}}_{f_1}^{i,jT}, \quad \text{(A.21)}$$

where $\ell''(Y, \hat{Y})$ indicates the second derivative with respect to $\hat{Y}$.

Following the calculation in [36], the two products $\nabla^2 L^+(\tilde{\boldsymbol{w}})\operatorname{vec}(M)$ and $\nabla^2 L^-(\tilde{\boldsymbol{w}})\operatorname{vec}(M)$ can be computed as

$$\begin{aligned} &\nabla^2 L^+(\tilde{\boldsymbol{w}})\operatorname{vec}(M) \\ &= \sum_{(i,j)\in\Omega} (\ell''(Y_{ij}, \hat{Y}_{ij}) - \omega)\tilde{\boldsymbol{x}}_{f_1}^{i,j}\tilde{\boldsymbol{x}}_{f_1}^{i,jT}\operatorname{vec}(M) \\ &= \sum_{(i,j)\in\Omega} (\ell''(Y_{ij}, \hat{Y}_{ij}) - \omega)\operatorname{vec}(\boldsymbol{x}_{f_1}^{i,j}(\boldsymbol{x}_{f_1}^{i,jT} M\boldsymbol{q}_{f_2,f_1}^{i,j})\boldsymbol{q}_{f_2,f_1}^{i,j}{}^T), \end{aligned} \quad \text{(A.22)}$$

and

$$\begin{aligned} &\nabla^2 L^-(\tilde{\boldsymbol{w}})\operatorname{vec}(M) \\ &= \omega \sum_{i=1}^{m}\sum_{j=1}^{n} \tilde{\boldsymbol{x}}_{f_1}^{i,j}\tilde{\boldsymbol{x}}_{f_1}^{i,jT}\operatorname{vec}(M) \\ &= \omega \sum_{i=1}^{m}\sum_{j=1}^{n} \operatorname{vec}(\boldsymbol{x}_{f_1}^{i,j}(\boldsymbol{x}_{f_1}^{i,jT} M\boldsymbol{q}_{f_2,f_1}^{i,j})\boldsymbol{q}_{f_2,f_1}^{i,j}{}^T). \end{aligned} \quad \text{(A.23)}$$

The cost of evaluating $\nabla^2 L^+(\tilde{\boldsymbol{w}})\operatorname{vec}(M)$ is proportional to $O(|\Omega|)$ and can be computed directly. However, $\nabla^2 L^-(\tilde{\boldsymbol{w}})\operatorname{vec}(M)$ needs a delicate derivation. We adopt the results in [36] below.

If first situation considered, the $\nabla^2 L^-(\tilde{\boldsymbol{w}})\operatorname{vec}(M)$ can be computed as

$$\operatorname{vec}(U_{f_1}^T \operatorname{diag}(\boldsymbol{z})Q_{f_2}^{f_1}),$$

where

$$\boldsymbol{z} = \omega n\left((U_{f_1}M)\odot Q_{f_2}^{f_1}\right)\mathbf{1}_{k\times 1}$$

The overall summation of (A.22) and (A.23) can be written as

$$U_{f_1}^T \operatorname{diag}(\bar{\boldsymbol{z}})Q_{f_2}^{f_1},$$

where

$$\bar{z}_i = z_i + \sum_{j\in\Omega_i} (\ell''(Y_{ij}, \hat{Y}_{ij}) - \omega)\boldsymbol{u}_{f_1}^i{}^T M\tilde{\boldsymbol{q}}_{f_2,f_1}^i$$

For the second case, details are omitted because an analogous derivation of the first case can be used. If the third case is considered, (A.22) can be computed as

$$\omega\operatorname{vec}(U_{f_1}^T (U_{f_1}M)(Q_{f_2}^{f_1 T}Q_{f_2}^{f_1})).$$

The overall summation of (A.22) and (A.23) can be written as

$$\operatorname{vec}(U_{f_1}^T \begin{bmatrix} \boldsymbol{\tau}_1^T \\ \vdots \\ \boldsymbol{\tau}_m^T \end{bmatrix})) + \omega\operatorname{vec}(U_{f_1}^T (U_{f_1}M)(Q_{f_2}^{f_1 T}Q_{f_2}^{f_1})),$$

where

$$\boldsymbol{\tau}_i = \sum_{j\in\Omega_i} (\ell''(Y_{ij}, \hat{Y}_{ij}) - \omega)(\boldsymbol{u}_{f_1}^i{}^T M\tilde{\boldsymbol{q}}_{f_2,f_1}^j)\tilde{\boldsymbol{q}}_{f_2,f_1}^j.$$

For complexity analysis please see [36].

*A.2.3 Line Search.* To check the sufficient decrease condition in (A.10), for the current iterate $\tilde{w}$ and a direction $S$, we must calculate

$$f(\tilde{\boldsymbol{w}} + \theta\operatorname{vec}(S)) - f(\tilde{\boldsymbol{w}}). \quad \text{(A.24)}$$

Because

$$\frac{1}{2}\lambda\|\tilde{\boldsymbol{w}}\| + L^-(\tilde{\boldsymbol{w}})$$

is quadratic, (A.24) is equivalent to

$$\begin{aligned} &L^+(\tilde{\boldsymbol{w}} + \theta\operatorname{vec}(S)) - L^+(\tilde{\boldsymbol{w}}) + \theta\nabla(\frac{1}{2}\lambda\|\tilde{\boldsymbol{w}}\|^2 + L^-(\tilde{\boldsymbol{w}}))^T\operatorname{vec}(S) \\ &\quad + \frac{1}{2}\theta^2\operatorname{vec}(S)^T\nabla^2(\frac{1}{2}\lambda\|\tilde{\boldsymbol{w}}\|^2 + L^-(\tilde{\boldsymbol{w}}))\operatorname{vec}(S) \\ &= L^+(\tilde{\boldsymbol{w}} + \theta\operatorname{vec}(S)) - L^+(\tilde{\boldsymbol{w}}) + \theta(\lambda\tilde{\boldsymbol{w}} + \nabla L^-(\tilde{\boldsymbol{w}}))^T\operatorname{vec}(S) \\ &\quad + \frac{1}{2}\theta^2\operatorname{vec}(S)^T(\lambda I + \nabla^2 L^-(\tilde{\boldsymbol{w}}))\operatorname{vec}(S). \end{aligned} \quad \text{(A.25)}$$

The value $L^+(\tilde{\boldsymbol{w}})$ is available from the previous iteration. We then calculate the following two values so that (A.25) can be easily obtained for any $\theta$.

$$(\lambda\tilde{\boldsymbol{w}} + \nabla L^-(\tilde{\boldsymbol{w}}))^T\operatorname{vec}(S) \quad \text{(A.26)}$$

and

$$\operatorname{vec}(S)^T(\lambda I + \nabla^2 L^-(\tilde{\boldsymbol{w}}))\operatorname{vec}(S). \quad \text{(A.27)}$$

Besides, we also have to cache the right hand side of (A.10), which is

$$\nabla f(\tilde{\boldsymbol{w}})^T\operatorname{vec}(S).$$

For the value in (A.27) we can calculate the following matrix-vector product first.

$$\begin{aligned} &(\lambda I + \nabla^2 L^-(\tilde{\boldsymbol{w}}))\operatorname{vec}(S) \\ &= \lambda\operatorname{vec}(S) + \nabla^2 L^-(\tilde{\boldsymbol{w}})\operatorname{vec}(S), \end{aligned}$$

where the second term can be conducted by the procedure in Section A.2.2.

With the above cached values, to evaluate the function-value difference at each new $\theta$, we only need to calculate

$$L^+(\tilde{\boldsymbol{w}} + \theta\operatorname{vec}(S)). \quad \text{(A.28)}$$

From (A.12), new $\hat{Y}_{ij}$ values are needed. That is, from (A.4) and (A.5),

$$\hat{Y}_{ij} + \theta\boldsymbol{x}_{f_1}^{i,jT}S\boldsymbol{q}_{f_2,f_1}^{i,j} \quad \forall i, j \in \Omega. \quad \text{(A.29)}$$

We can avoid calculating (A.29) for any new $\theta$ by applying a similar trick. Specifically, with the cached variables

$$\Delta Y_{ij} = \boldsymbol{x}_{f_1}^{i,jT} S \boldsymbol{q}_{f_2,f_1}^{i,j},$$

we can calculate (A.28) by

$$\begin{aligned} &L^{+}(\tilde{\boldsymbol{w}} + \theta\,\mathrm{vec}(S)) \\ = &\sum_{(i,j)\in\Omega} \ell(Y_{ij}, \hat{Y}_{ij} + \theta\Delta Y_{ij}) - \omega\frac{1}{2}(A_{ij} - (\hat{Y}_{ij} + \theta\Delta Y_{ij}))^2 \end{aligned}$$

with the complexity proportional to $O(|\Omega|)$.

## B ADDITIONAL DETAILS OF EXPERIMENTS

### B.1 Data Preparation of the Yahoo! R3 Set

To simulate events logged by the deterministic policy of (25), we modify the data set by the following procedures.

(1) We apply rating-based matrix factorization technique [17] on (user, song) pairs in the training set to find a matrix $\tilde{R}$ with

$$\tilde{R}_{ij} \approx R_{ij}, \quad \forall (i,j) \in \Omega,$$

where parameters including the size of latent factors and the $l2$ regularization's coefficient are selected through a five-fold cross validation on a grid of values.

(2) We apply a thresholding operation to generate a new set as

$$\Omega_{\text{det}} = \{(i,j) \mid \tilde{R}_{ij} > 2.5, \forall (i,j) \in \Omega\}.$$

To generate $S_c$, where each instance includes a feature vector $\boldsymbol{x}$ and a binary label $y$, we apply one-hot encoding and rating binarization as follows

$$S_c = \{(Y_{ij}, \boldsymbol{x}^{i,j}) \mid \forall (i,j) \in \Omega_{\text{det}}\},$$

where

$$\boldsymbol{x}^{i,j} = [\underbrace{\overbrace{\cdots, 1,}^{i} \cdots, 1,}_{14,877+j} \cdots]^T \in \mathcal{R}^{15,877},$$

and

$$Y_{ij} = \begin{cases} 1 & R_{ij} = 5, \\ -1 & R_{ij} < 5. \end{cases}$$

Besides, we randomly split out 5% pairs from the original test set and conduct one-hot encoding and binarization to generate $S_t$ defined in Section 2.2. Analogously, the rest 95% is further split into two subsets in ratios of 5% and 90% to generate $S_{\text{va}}$ and $S_{\text{te}}$ for parameter selections and evaluations, respectively.

### B.2 Parameter Selection

For parameter selections, we conduct grid searches by training multiple models on the corresponding training set of each approach. We consider to tune the following parameters.

- $\lambda$: the $l2$ regularization coefficient for FFM, IPS, CausE and New.
- $\lambda_c, \lambda_t$: $l2$ regularization's coefficients respectively for $\mathcal{W}_c$ and $\mathcal{W}_t$ in CausE.
- $\lambda_{\|t-c\|}$: the coefficient of the term in CausE, which controls the discrepancy between $\mathcal{W}_c$ and $\mathcal{W}_t$.
- $k_1$: the number of latent factors of each approach for the CPC set.
- $k_2$: the number of latent factors of each approach for the Yahoo! R3 set.
- $\omega$: the coefficient to balance the IPS part and the direct method part in New.
- $T$: the number of training iterations of each approach.

The search range of each parameter is listed in Table II. We then choose parameters achieving the highest NLL on $S_{\text{va}}$. For building the final model to predict the test set, we should include data in the validation period. To this end, we consider the following new training set.

- For approaches using $S_c$, we use $S_c \cup S_c'$ as the new training set, where for the **Yahoo! R3** set, because all events in the original training set have been included in $S_c$, $S_c' = \emptyset$. For the **CPC** set, as shown in Figure 6, $S_c'$ is the subset of events logged by non-uniform policies in the day of collecting $S_{\text{va}}$.
- For approaches using $S_t$, we use $S_t \cup S_{\text{va}}$ as the new training set.
- For approaches using $S_c \cup S_t$, we use $S_c \cup S_t \cup S_{\text{va}} \cup S_c'$ as the new training set.

We finally compute NLL and AUC scores of evaluating $S_{\text{te}}$.

### B.3 Impact of Unbiased Imputation Models Learnt from $S_t$

In section 4.2, we proposed unbiased imputation models because past works [6, 19] find that learning from biased data sets will propagate the carried selection bias to non-displayed events and worsen the performance. To verify this, we follow these works to learn biased imputation models from $S_c \cup S_t$. That is, in the workflow presented in Figure 4, $S_t$ is replaced by $S_t \cup S_c$. We then re-select the parameters for the imputation model by a validation procedure. From Table I, we observe that the performance of using

**Table I: A comparison between learning the imputation model via $S_t$ and $S_c \cup S_t$. All results are relative improvements over the** average ($S_c$). **The best approach is bold-faced.**

| Approach \ Data set | Metric | Yahoo! R3 NLL | Yahoo! R3 AUC | CPC NLL | CPC AUC |
|---|---|---|---|---|---|
| $S_t$ | New (avg) | **+79.1%** | +51.8% | +32.9% | +48.6% |
| | New (item-avg) | +76.8% | **+54.2%** | +32.4% | +48.2% |
| | New (complex) | -0.2% | +37.4% | **+33.4%** | **+50.6%** |
| $S_c \cup S_t$ | New (avg) | +0.5% | +31.2% | +4.34% | +32.8% |
| | New (item-avg) | -0.17% | +37.4% | +9.0% | +38.4% |
| | New (complex) | -1.4% | +36.8% | +19.2% | +43.0% |

**Table II: Search range of parameters considered in different approaches.**

| | FFM | IPS | CausE | New |
|---|---|---|---|---|
| $\lambda$ | $2^{\{0,\cdots,7\}}$ | $2^{\{0,\cdots,7\}}$ | - | $2^{\{0,\cdots,7\}}$ |
| $\lambda_c, \lambda_t$ | - | - | $2^{\{-4,\cdots,-10\}}$ | - |
| $\lambda_{\|t-c\|}$ | - | - | $2^{\{-4,\cdots,-10\}}$ | - |
| $k_1$ | $2^5$ | $2^5$ | $2^5$ | $2^5$ |
| $k_2$ | $2^{\{3,\cdots,6\}}$ | $2^{\{3,\cdots,6\}}$ | $2^{\{3,\cdots,6\}}$ | $2^{\{3,\cdots,6\}}$ |
| $\omega$ | - | - | - | $2^{\{-8,-12,-16\}}$ |
| $T$ | $[1, 100]$ | $[1, 100]$ | $[1, 100]$ | $[1, 100]$ |

an imputation model learnt from $S_t$ is significantly better than from $S_c \cup S_t$. This result confirms the importance of using an unbiased imputation model.