Counterfactual Reward Modification for Streaming Recommendation with Delayed Feedback

Xiao Zhang[1,2], Haonan Jia[2,3], Hanjing Su[4], Wenhan Wang[4], Jun Xu[1,2,*], Ji-Rong Wen[1,2]
[1] Gaoling School of Artificial Intelligence, Renmin University of China
[2] Beijing Key Laboratory of Big Data Management and Analysis Methods
[3] School of Information, Renmin University of China [4] Tencent Inc.
{zhangx89, junxu, jrwen}@ruc.edu.cn, hnjia00@gmail.com, {justinsu, ezewang}@tencent.com

ABSTRACT

The user feedbacks could be delayed in many streaming recommendation scenarios. As an example, the user feedbacks to a recommended coupon consist of the immediate feedback on the click event and *the delayed feedback on the resultant conversion*. The delayed feedbacks pose a challenge of training recommendation models using instances with incomplete labels. When being applied to real products, the challenge becomes more severe as the streaming recommendation models need to be retrained very frequently and the training instances need to be collected over very short time scales. Existing approaches either simply ignore the unobserved feedbacks or heuristically adjust the feedbacks on a static instance set, resulting in biases in the training data and hurting the accuracy of the learned recommenders. In this paper, we propose a novel and theoretic sound counterfactual approach to adjusting the user feedbacks and learning the recommendation models, called CBDF (Counterfactual Bandit with Delayed Feedback). CBDF formulates the streaming recommendation with delayed feedback as a problem of sequential decision making and models it with a batched bandit. To deal with the issue of delayed feedback, at each iteration (episode), a counterfactual importance sampling model is employed to re-weight the original feedbacks and generate the modified rewards. Based on the modified rewards, a batched bandit is learned for conducting online recommendation at the next iteration. Theoretical analysis showed that the modified rewards are statistically unbiased, and the learned bandit policy enjoys a sub-linear regret bound. Experimental results demonstrated that CBDF can outperform the state-of-the-art baselines on a synthetic dataset, the Criteo dataset, and a dataset from Tencent's WeChat app.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies → Reinforcement learning**.

KEYWORDS

delayed feedback; batched bandit; counterfactual learning

ACM Reference Format:
Xiao Zhang[1,2], Haonan Jia[2,3], Hanjing Su[4], Wenhan Wang[4], Jun Xu[1,2,*], Ji-Rong Wen[1,2] . 2021. Counterfactual Reward Modification for Streaming Recommendation with Delayed Feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462892

[*] Corresponding author: Jun Xu.

1 INTRODUCTION

In streaming recommender systems, the models need to be updated frequently on fresh user feedbacks, in order to prevent them from becoming stale [8, 9, 18, 21, 37, 38]. In some applications, the user feedbacks may have not yet occurred when the training data are collected in a short period, resulting in the missing of a large number of user feedback events. We refer to this category of limited user feedback due to the time delay as the *delayed feedback*.

Streaming recommendation with delayed feedback is ubiquitous in real applications. Let's use coupon recommendation as an example. After a user clicks a recommended coupon (immediate feedback), she may use the coupon after some time (delayed conversion), or just leave it there (no conversion). Figure 1 shows the delay statistics of coupon recommendation in the Tencent's social media app, WeChat, where the blue histogram represents the percentage of conversion events w.r.t. different time intervals. We observed that a large proportion (almost 70%) of the conversion feedbacks are delayed (i.e., after the 0-th day).

Usually, the streaming recommendation models are trained on the log data collected in a short period of time (e.g., one day or several days). It is inevitable that there exist a large number of incompletely labeled instances in the training data because their conversion events could be delayed to the time after the data collection. Studies have shown that the incompletely labeled instances may seriously hurt the model training. Figure 2 shows an empirical study where a series of LinUCB [22] models were trained on a dataset with controlled proportions of delayed feedbacks. The LinUCB simply treats the incompletely labeled instances as no conversion. We observed that the recommendation performance steadily decreased with the increase of the proportion of delayed feedbacks (PRDF). The results clearly indicate that the delayed feedbacks could be damaging in terms of hurting the model training.

A common approach to training streaming recommendation models with delayed feedback is ignoring the unobserved feedbacks and directly employing the typical online learning algorithms

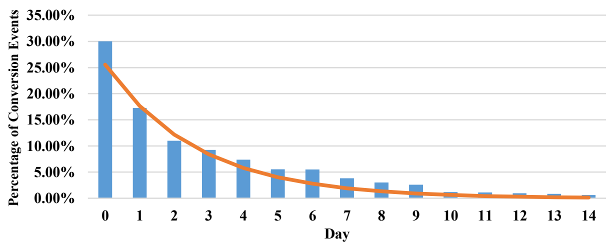


Figure 1: Percentage of coupon conversion events w.r.t. the number of days after users clicking the coupons, based on the data collected from Tencent's WeChat. The orange curve indicates that the delay time approximately follows an exponential distribution.

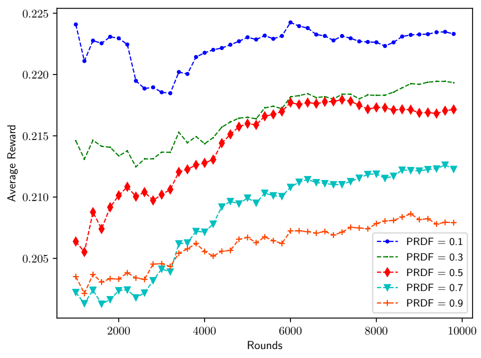


Figure 2: Recommendation performance (average reward) of LinUCB [22] w.r.t. the round of interaction between LinUCB and synthetic environment in Section 6.2, where the rewards are convex combinations of the click reward and the conversion reward (Eq. (1)). "PRDF" means the proportion of delayed feedbacks, i.e., the proportion of the number of hidden conversion events to the total number of conversion events.

[15, 16, 34]. These algorithms usually need to wait for a period of time until the delayed feedback could be correctly observed. In real-world applications, waiting a long time before training is usually not realistic, and only a small proportion of true feedbacks can be observed in the training set. There are also studies that heuristically adjusts the feedbacks on a static set of passively-received log data [10, 21, 27, 40, 41]. These approaches are designed under the batch learning framework, without considering the streaming nature of data in streaming recommendation. More importantly, in all of these models, the feedback distributions of the training instances are different from the real user feedback, resulting in high biases in reward estimation.

In this paper, we propose a novel and theoretical sound counterfactual approach to streaming recommendation with delayed feedback. To achieve unbiased estimates of the true rewards, we propose to estimate the delayed feedbacks through re-weighting the observed feedbacks with importance sampling. By dividing the collected user feedbacks with a counterfactual deadline, a survival model is estimated and subsequently used to determine the sampling weights. The approach, referred to as CBDF, formalizes the recommendation under delayed feedback as a problem of sequential decision making and is modeled as a batch bandit. At each iteration, it repeatedly performs the reward modification, updates the batched bandit model with the modified rewards, makes online recommendations, and collects user feedbacks for the next iteration. Theoretical analysis shows that the modified rewards are statistically unbiased, and the training of the batched bandit enjoys a sublinear regret bound. Experimental results based on a synthetic dataset, Criteo benchmark, and a real dataset collected from Tencent's WeChat showed that CBDF can outperform the state-of-the-art baselines in terms of the recommendation accuracy and convergence rate. Empirical analysis also verified the correctness of the theoretical conclusions.

2 RELATED WORK

Learning with Delayed Feedback has received considerable attention in the studies of predicting click-through rate (CTR) or conversion rate (CVR). Chapelle [10] assumed that the delay distribution is exponential, and proposed two generalized linear models for predicting the CVR and the delay time, respectively. This study has been extended to a nonparametric delayed model using the kernel density estimation [41]. Ktena et al. [21] focused on the CTR prediction problem in the presence of delayed feedbacks and compared different combinations of losses and models. Saito et al. [27] presented a dual learning algorithm for CVR prediction under delayed feedback, in which a CVR predictor and a biased estimator can be learned alternately using the observed conversions. Yasui et al. [40] reduced the delayed feedback problem to a feedback shift problem, which formulates a consistent empirical risk and corrected the delayed feedbacks using importance weighting.

Multi-Armed Bandits with Delayed Feedback has attracted much research efforts in recent years. Vernade et al. [34] proposed stochastic bandits for the uncensored and censored feedbacks under the assumption that the distribution of delay is known. Héliou et al. [16] introduced a gradient-free online learning algorithm with delayed feedback, where the action space is compact and convex. Grover et al. [13] demonstrated that the high sample complexity of bandits under delayed feedback can be offset when the partial feedbacks are received. Bistritz et al. [5] showed the average strategy results in a Nash equilibrium under delayed feedback. Delayed feedback has also been studied in the context of Markov Decision Process (MDP) [20, 35]. Though contextual bandit has been extensively studied and the representative models include LinUCB [22, 28], EXP4 [4, 7], LinEXP3 [3, 24], etc. Applying them to the recommendation with delayed feedback is still hard due to the complex delay mechanisms and user behaviors.

3 BATCHED BANDIT FOR RECOMMENDATION

3.1 Problem Formulation

Streaming recommendation with delayed feedback can be formulated as a problem of sequential decision making. As shown in Figure 3, at each iteration, a batch of log data which consists of the decision history from recommendation models and the feedbacks (may be delayed) from users is collected. The recommendation model is updated incrementally using the batch data. Usually, the recommendation model is updated periodically (e.g., every X hours or days). New recommendations are made based on the updated

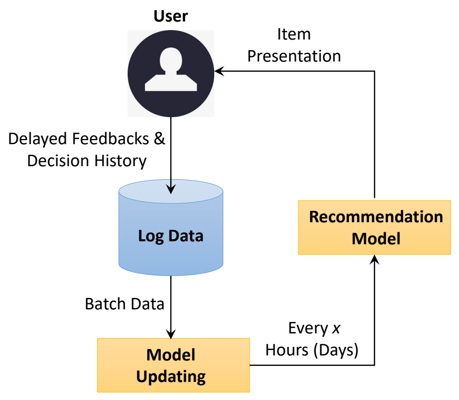


Figure 3: Sequential decision making under delayed feedback.

model, and the results and user feedbacks are collected for further updating the model.

In this paper, we propose to formulate the process as a batched bandit which can be represented by a 7-tuple $\langle \mathcal{S}, \mathcal{A}, \pi, R, \gamma, N, B \rangle$:

Context space $\mathcal{S} \subseteq \mathbb{R}^d$ denotes a context space that summarizes the information of both the user and items.

Action space $\mathcal{A}$ denotes a given action space and each action (arm) corresponds to a candidate item. Choosing an action $A \in \mathcal{A}$ means that the corresponding item is recommended.

Policy $\boldsymbol{\pi}$ describes the behaviors of an agent, which is a function of the input context $S \in \mathcal{S}$ and output a distribution over actions $\mathcal{A}$. $\boldsymbol{\pi}$ determines which action to take.

Reward R: in this paper, the reward is defined as a linear combination of two feedbacks: (a) The *immediate feedback* which can be immediately observed once the action is executed, coming from the user click event. The immediate feedback is determined by an observed variable $C \in \{0, 1\}$. The *click timestamp* of C occurs, denoted as c, is also stored; (b) The *delayed feedback* which is received at some time between the click timestamp and the data collection timestamp, coming from the resultant conversion event. The delayed feedback is determined by an observed variable $Y \in \{0, 1\}$. It is obvious that $Y = 0$ if $C = 0$. Different from the immediate feedback, $Y = 0$ does not mean no conversion because the conversion may occur after the data collection time. Let's use the true (may unobservable) variable $V \in \{0, 1\}$ to denote whether a conversion will eventually occur, and a delayed variable $D \in \{0, 1\}$ to denote whether a conversion can be correctly observed. It is also easy to know that $Y = D \times V$ if $V = 1$ and $Y = 0$ otherwise.

Delay time γ is the time between the click timestamp and its resultant conversion timestamp, which is unobservable when $Y = 0$ and $V = 1$. In addition, we denote the *following-up time* by an observed variable e, which is the time between the click timestamp, and the smaller value of the data collection timestamp and the conversion timestamp. It is obvious that $\gamma = e$ when users convert before data collection.

Number of episodes N. The decision process of a batched bandit is partitioned into N episodes. Within an episode, the agent updates the policy using the collected data, and then interacts with users for multiple steps using the updated policy.

Batch size B is the number of steps in each episode. That is, in each episode, the agent interacts with the users B times using a fixed policy, collects mini-batch data, and stores them into a data buffer $\mathcal{D} = \{(S_i, A_i, C_i, Y_i, c_i, e_i)\}_{i=1}^{B}$ at the end of each episode. Finally, a new policy is trained on $\mathcal{D}$ at the beginning of the next episode.

Table 1 summarizes the notations used throughout the paper.

Table 1: A summary of notations.

Symbol	Explanation
$C \in \{0, 1\}$	Observed variable indicating whether a click event occurs with its click timestamp c
$Y \in \{0, 1\}$	Observed variable indicating whether a conversion occurs before the data is collected
$V \in \{0, 1\}$	True variable indicating whether a conversion will eventually occur (may be unobserved)
$D \in \{0, 1\}$	Delayed variable indicating whether a conversion can be correctly observed (may be unobserved)
$\gamma \in \mathbb{R}_+ \cup \{0\}$	Delay time between the click and its resultant conversion (may be unobserved)
$e \in \mathbb{R}_+ \cup \{0\}$	Following-up time since the click (equals the delay time when user converts)
N, B	Number of episodes, batch size, respectively

3.2 Problem Analysis

At each step of an episode, the agent observes a new context, chooses an action, and receives a reward that includes two parts: an immediate feedback and a delayed feedback. The agent in the batched bandit can only wait for the delayed feedbacks for at most B steps. As a result, the delayed feedbacks will be missing if the conversion events occur after the end of this episode. That is, the system observed $Y = 0$ while the true feedback could be $V = 1$.

Ideally, the recommendation model would be trained using the rewards based on the true delayed feedback V's. In practice, however, V is unobservable. Directly using the observed feedback Y's inevitably introduces bias in training and hurts the model accuracy. In the next section, we propose a novel counterfactual reward modification approach that adjusts the user feedbacks before training so that the bias can be relieved.

4 COUNTERFACTUAL BANDIT WITH DELAYED FEEDBACK

In this section, we propose a novel counterfactual approach tailored for streaming recommendation with delayed feedback, called CBDF (Counterfactual Bandit with Delayed Feedback).

4.1 Algorithm Overview

Figure 4 illustrates the overall algorithm architecture of CBDF. In each episode, given the collected data buffer $\mathcal{D}$, the agent re-weights the delayed feedback for each record in $\mathcal{D}$, obtaining the modified reward R^{mod} and a counterfactual buffer $\mathcal{D}^{\text{mod}}$. A new policy $\boldsymbol{\pi}$ is then updated on $\mathcal{D}^{\text{mod}}$ for the recommendation in the next episode.

Algorithm 1 shows the detailed procedure. It takes batch size, number of episodes, counterfactual deadline parameter, and action space as inputs. After the initialization, the main loop of CBDF

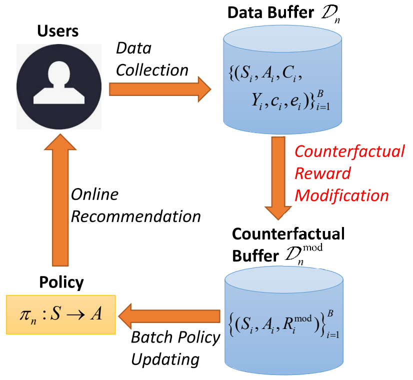

Users
Data Collection
Data Buffer $\mathcal{D}_n$
$\{(S_i, A_i, C_i, Y_i, c_i, e_i)\}_{i=1}^B$
Counterfactual Reward Modification
Counterfactual Buffer $\mathcal{D}_n^{\text{mod}}$
$\{(S_i, A_i, R_i^{\text{mod}})\}_{i=1}^B$
Batch Policy Updating
Policy
$\pi_n : S \to A$
Online Recommendation

Figure 4: The n-th episode of the proposed counterfactual approach to learning batched bandits with delayed feedback.

Algorithm 1: Counterfactual Bandit with Delayed Feedback (CBDF)

```
Require: Batch size $B$, number of episodes $N$, counterfactual deadline parameter $C_\xi$, action space $\mathcal{A} = \{A_j\}_{j\in[M]}$, where $[M] := \{1, 2, \dots, M\}$
 1: Initialize policy $\boldsymbol{\pi}_0 \leftarrow \mathbf{1}/M$
 2: Sample data buffer $\mathcal{D}_1 = \{(S_{0,b}, A_{I_{0,b}}, C_{0,b}, Y_{0,b}, c_{0,b}, e_{0,b})\}_{b=1}^B$ using $\boldsymbol{\pi}_0$ during timestamps $t_1^{\text{str}}$ and $t_1^{\text{end}}$
 3: for $n = 1$ to $N$ do
 4:     // Counterfactual Reward Modification
 5:     $\xi \leftarrow C_\xi \times (t_n^{\text{end}} - t_n^{\text{str}}) + t_n^{\text{str}}$ {counterfactual deadline}
 6:     $\mathcal{D}_n^{\text{mod}} \leftarrow \text{CRM}(\mathcal{D}_n, \xi, \mathcal{A})$ {Algorithm 2}
 7:     // Batch Policy Updating
 8:     Obtain $\boldsymbol{\pi}_n$ using batch policy updating Eq. (11) on $\mathcal{D}_n^{\text{mod}}$
 9:     // Online Recommendation
10:     for $b = 1$ to $B$ do
11:         Observe context $S_{n,b}$
12:         Choose $A_{I_{n,b}} \in \mathcal{A}$ following $\boldsymbol{\pi}_n(S_{n,b})$ using Eq. (12)
13:     end for
14:     $\mathcal{D}_{n+1} \leftarrow \{(S_{n,b}, A_{I_{n,b}}, C_{n,b}, Y_{n,b}, c_{n,b}, e_{n,b})\}_{b=1}^B$
15:     $t_{n+1}^{\text{str}}, t_{n+1}^{\text{end}} \leftarrow$ start and end timestamps of collecting $\mathcal{D}_{n+1}$
16: end for
```

repeatedly conducts three operations: counterfactual reward modification, batch policy updating, and online recommendation: (1) the *counterfactual reward modification* tries to eliminate the bias in the delayed feedbacks using counterfactual importance sampling; (2) the *batch policy updating* approximates the optimal policy for streaming recommendation based on the modified rewards; (3) the *online recommendation* offers recommendations on items to users following the updated policy (for B steps), and collects the context-action pairs, observed feedbacks (Y and C), click timestamp c, and following-up time e.

Next, we specify these operations with details. For convenience, we will simplify the subscripts of elements in the data buffer as $(S_i, A_i, C_i, Y_i, c_i, e_i)$ if not specified.

4.2 Counterfactual Reward Modification

Intuitively, the reward can be calculated as a linear combination of the immediate feedback and the delayed feedback:

$$R = \lambda \widehat{R} + (1 - \lambda)\widetilde{R}, \tag{1}$$

where $\widehat{R} = C \in \{0, 1\}$ indicates the immediate feedback on whether the user skips or clicks the presented item, and $\widetilde{R} = Y \in \{0, 1\}$ indicates the delayed feedback whether the user converts using this item after the click events and before the data collection time, and $\lambda \in [0, 1)$ is a weighting coefficient.

4.2.1 Importance Sampling for Delayed Rewards. Given a context $S \in \mathcal{S} \subseteq \mathbb{R}^d$, the delayed feedback can be expressed as a delayed reward function $\widetilde{R}(S, Y)$. As mentioned in the previous sections, the observed variable Y will not be consistent with the true but unobservable variable V if the user conversion occurs after the data collection time, resulting in the issue of *biased delayed rewards*. Ideally, the goal of recommendation would be maximizing the *expected true delayed reward*,

$$\mathbb{E}_V \left[\widetilde{R}(S, V) \right] = \Pr\{V = 1 \mid S\}, \tag{2}$$

rather than the *expected biased delayed reward*

$$\mathbb{E}_Y \left[\widetilde{R}(S, Y) \right] = \Pr\{Y = 1 \mid S\}, \tag{3}$$

where $\widetilde{R}(S, V)$ denotes the true delayed reward based on the true variable V, and $\widetilde{R}(S, Y)$ denotes the biased delayed reward based on the observed variable Y. Since those users whose $Y = 0$ may eventually convert (i.e., $V = 1$) after the data collection timestamp, resulting $\Pr\{Y = 1 \mid S\} \leq \Pr\{V = 1 \mid S\}$ that introduces the bias.

Instead of directly using the biased delayed rewards, we propose to modify the biased delayed rewards using importance sampling [6, 30, 40], obtaining the *modified delayed rewards*:

$$\widetilde{R}^{\text{mod}}(S, Y) = w\widetilde{R}(S, Y),$$

where w is the *importance weights* which is defined as

$$w = \frac{\Pr\{V = 1 \mid S\}}{\Pr\{Y = 1 \mid S\}}. \tag{4}$$

Therefore, the orignal reward in Eq. (1) becomes the *modified reward*:

$$R^{\text{mod}}(S, Y) = \lambda \widehat{R} + (1 - \lambda)\widetilde{R}^{\text{mod}}(S, Y). \tag{5}$$

Please note that the modified rewards $R^{\text{mod}}(S, Y)$ may be larger than 1 since it is possible that the importance weight $w \geq 1$. The intuition is that an observed conversion in delayed feedback environments deserves a higher reward than that of in full-information.

We show that the modified delayed rewards using the importance weights in Eq. (4) are unbiased when estimating the expected true delayed reward, shown in Theorem 4.1.

Theorem 4.1 (Unbiasedness of Reward Estimation). *For any context $S \in \mathcal{S}$, the modified delayed reward is an unbiased estimate of the expected true delayed reward, i.e.,*

$$\mathbb{E}_Y \left[\widetilde{R}^{\text{mod}}(S, Y) \right] = \mathbb{E}_V \left[\widetilde{R}(S, V) \right]. \tag{6}$$

Let $[T] := \{1, 2, \dots, T\}$. If the important weights $w_i \leq w_{\max}, i \in [T]$

and for a sequence of i.i.d contexts $\{S_i\}_{i=1}^T$ *the rewards are independent*[1], *then with probability at least* $1-\delta$

$$\left|\frac{1}{T}\sum_{i=1}^{T}\widetilde{R}^{\text{mod}}(S_i,Y_i)-\mathbb{E}_{S,V}\left[\widetilde{R}(S,V)\right]\right|\le\frac{w_{\max}}{\sqrt{T}}\sqrt{\frac{1}{2}\ln\frac{2}{\delta}}. \quad (7)$$

Proof of Theorem 4.1 can be found in the Appendix A.1. The probabilistic error bound of Eq. (7) indicates that the mean of modified delayed rewards converges to the expected true delayed reward with a convergence rate of order $T^{-1/2}$, and a smaller $w_{\max}$ incurs a tighter error bound.

4.2.2 *Counterfactual Estimation of Importance Weights.* Directly computing the importance weight in Eq. (4) is intractable because V is an unobservable variable. Inspired by [40], we proposed a counterfactual learning approach to predicting the weight w. As shown in Figure 5, given a set of records collected from the previous episode $\mathcal{D}=\{(S_i,A_i,C_i,Y_i,c_i,e_i)\}_{i=1}^B$ (collected during timestamps t^{str} and t^{end}), we introduce a *counterfactual deadline* $\xi\in(t^{\text{str}},t^{\text{end}})$ which simulates a virtual data collection timestamp, and splits $\mathcal{D}$ into the *observed set* and *hold-out set.* Then, the instances from the observed set whose conversions can be found in $\mathcal{D}$ (i.e., $Y=1$) are selected as the training instances. The conversions occurring in the hold-out set are considered as the delayed while 'observed' true variable V. Prediction models parameterized by $\{\boldsymbol{\beta}_A\}_{A\in\mathcal{A}}$ are learned based on the selected instances, which are further employed to predict the importance weights for all the instances from $\mathcal{D}$.

Specifically, given the i-th instance $(S_i,A_i,C_i,Y_i,c_i,e_i)$ in $\mathcal{D}$, its modified reward in Eq. (5) can be simplified as $R_i^{\text{mod}}=\lambda C_i+(1-\lambda)w_iY_i$, and the i-th importance weight w_i can be represented as

$$w_i=\frac{\Pr\{V_i=1\mid S_i\}}{\Pr\{Y_i=1\mid S_i\}}=\frac{1}{\Pr\{D_i=1\mid V_i=1,S_i\}} \quad (8)$$
$$=\frac{1}{1-\Pr\{D_i=0\mid V_i=1,S_i\}}=\frac{1}{1-\exp\left\{-\exp(\langle\boldsymbol{\beta}_{A_i},S_i\rangle)e_i\right\}},$$

where: (a) D_i is the delayed variable indicating whether the i-th conversion can be correctly observed, and $\Pr\{D_i=0\mid V_i=1,S_i\}$ means the V_i being incorrectly observed given the context S_i; (b) the last equality uses a survival distribution model to represent the conditional probability, by assuming the distribution of delayed conversions can be approximated using exponential distributions (cf. Figure 1), where $h_{A_i}(S_i):=\exp(\langle\boldsymbol{\beta}_{A_i},S_i\rangle)$ is the hazard function [19, 39], $\boldsymbol{\beta}_{A_i}\in\mathbb{R}^d$ is the model parameters corresponding to action A_i. $h_{A_i}(S_i)e_i$ can be explained as an estimate of the logarithm probability that $V_i=1$ is unobserved within time span e_i.

Then, we construct a subset of the observed set for training the prediction models $\{\boldsymbol{\beta}_A\}_{A\in\mathcal{A}}$. Using the counterfactual deadline ξ, $\mathcal{D}$ is split into an observed set (indexed by $\{i:c_i\le\xi,i\in[B]\}$) and a hold-out set, where c_i is the click timestamp of S_i in $\mathcal{D}$. From the observed set, we further select some instances as the training set

$$\mathcal{D}_A=\{(S_i,A_i,Y_i,c_i,Y_i^{\text{obs}},e_i^{\text{obs}}):c_i\le\xi,Y_i=1,A_i=A,i\in[B]\}$$

for all $A\in\mathcal{A}$, where $Y_i=1$ indicates the conversion of the i-th instance can be found in $\mathcal{D}$, either in the observed set or in the hold-out set; Y_i^{obs} indicates whether the conversion can be observed before the counterfactual deadline given the context S_i, i.e., $Y_i^{\text{obs}}=\begin{cases}1 & c_i+e_i\le\xi\\ 0 & \text{otherwise}\end{cases}$, and $e_i^{\text{obs}}=\min\{\xi-c_i,e_i\}$ is the *counterfactual following-up time* since the click event.

[1]The conditional independence assumption of the rewards is commonly used in the bandits literature, which can be ensured using a master algorithm as a theoretical construct [2, 11].

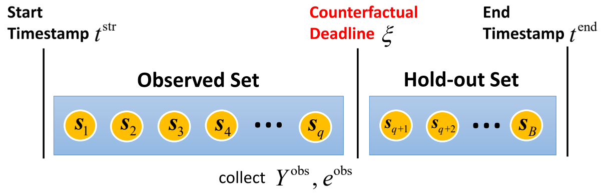

Start
Timestamp t^{str}
Counterfactual
Deadline ξ
End
Timestamp t^{end}
Observed Set
Hold-out Set
S_1 S_2 S_3 S_4 ... S_q
S_{q+1} S_{q+2} ... S_B
collect Y^{obs}, e^{obs}

Figure 5: Data partition in counterfactual deadline approach in each episode, where Y^{obs} indicates whether the conversion can be observed before the counterfactual deadline, e^{obs} is the counterfactual following-up time since the click event, and q is the number of instances in the observed set.

Note that $Y_i=1$ for all of the instances in $\mathcal{D}_A$. Therefore, $Y_i^{\text{obs}}=0$ also means the variable Y_i can not be correctly observed in the observed set, i.e., the unobservable variable $D_i=0$ if the data collection timestamp is set as the counterfactual deadline. Motivated by this observation, $\mathcal{D}_A$ can be used to estimate the importance weight in Eq. (8) because it is also represented by the variable D_i. More specifically, for each action $A\in\mathcal{A}$, the maximum likelihood estimate of $\boldsymbol{\beta}_A$ can be found by minimizing a negative log-likelihood[2]

$$\boldsymbol{\beta}_A\leftarrow\arg\min_{\boldsymbol{\beta}\in\mathbb{R}^d}\mathcal{L}_{\boldsymbol{\beta}}(\mathcal{D}_A)$$
$$:=\sum_{i=1}^{|\mathcal{D}_A|}\left\{(1-Y_i^{\text{obs}})h_ie_i^{\text{obs}}-Y_i^{\text{obs}}\ln\left[1-\exp\left(-h_ie_i^{\text{obs}}\right)\right]\right\}, \quad (9)$$

where $|\mathcal{D}_A|$ denotes the size of the action-specific training set $\mathcal{D}_A$, and $h_i=\exp(\langle\boldsymbol{\beta},S_i\rangle)$ the hazard function specific to S_i and $\boldsymbol{\beta}$, and $h_ie_i^{\text{obs}}$ denotes the estimate of the logarithm probability that $Y_i^{\text{obs}}=0$ (i.e., $V_i=1$ is unobserved within time span e_i^{obs}).

Finally, given the prediction models $\{\boldsymbol{\beta}_A\}_{A\in\mathcal{A}}$ and Eq. (8), the modified reward for the i-th instance in $\mathcal{D}$ can be predicted as

$$R_i^{\text{mod}}=\lambda C_i+\frac{(1-\lambda)Y_i}{1-\exp\left\{-\exp(\langle\boldsymbol{\beta}_{A_i},S_i\rangle)e_i\right\}}. \quad (10)$$

We summarize the above steps in Algorithm 2, called CRM.

4.3 Model Training and Online Recommendation

Based on the modified rewards, the batch policy updating in [14] can be employed for updating the batch policy and conducting online recommendation.

4.3.1 *Batch Policy Updating.* In the n-th episode, for each action $A\in\mathcal{A}$, we store the context vectors and modified rewards into $S_A^{n-1}\in\mathbb{R}^{N_A^{n-1}\times d}$ and $\boldsymbol{R}_A^{n-1}\in\mathbb{R}^{N_A^{n-1}}$, respectively, where N_A^{n-1} denotes the number of times action A has been executed at the end of this episode. Then we can update the covariance matrix incrementally as $\Phi_A^n=\Phi_A^{n-1}+S_A^{n-1\intercal}S_A^{n-1}$ corresponding to the

[2]We update the parameter $\boldsymbol{\beta}_A$ using the gradient descent on $\mathcal{D}_A$, and set the updated $\boldsymbol{\beta}_A$ as the initial value of the next episode.

Algorithm 2: Counterfactual Reward Modification (CRM)

```
Require: Buffer $\mathcal{D} = \{(S_i, A_i, C_i, Y_i, c_i, e_i)\}_{i=1}^{B}$, counterfactual deadline $\xi$, action space $\mathcal{A}$
Ensure: Counterfactual buffer $\mathcal{D}^{\text{mod}} = \{(S_i, A_i, R_i^{\text{mod}})\}_{i=1}^{B}$
1: Initialize $\mathcal{D}_A \leftarrow \emptyset$, for all $A \in \mathcal{A}$
2: for each action $A \in \mathcal{A}$ do
3:     for $i \in \{i : c_i \le \xi, Y_i = 1, A_i = A, i \in [B]\}$ do
4:         $Y_i^{\text{obs}} \leftarrow \begin{cases} 1 & c_i + e_i \le \xi \\ 0 & \text{otherwise} \end{cases}$, $e_i^{\text{obs}} \leftarrow \min\{\xi - c_i, e_i\}$
5:         $\mathcal{D}_A \leftarrow \mathcal{D}_A \bigcup \{(S_i, Y_i^{\text{obs}}, e_i^{\text{obs}})\}$
6:     end for
7:     $\boldsymbol{\beta}_A \leftarrow \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \mathcal{L}_{\boldsymbol{\beta}}(\mathcal{D}_A)$ {Eq. (9)}
8: end for
9: $\mathcal{D}^{\text{mod}} \leftarrow \emptyset$
10: for $i \in [B]$ do
11:     $R_i^{\text{mod}} \leftarrow \lambda C_i + \frac{(1-\lambda) Y_i}{1 - \exp\{-\exp(\langle \boldsymbol{\beta}_{A_i}, S_i \rangle) e_i\}}$ {Eq. (10)}
12:     $\mathcal{D}^{\text{mod}} \leftarrow \mathcal{D}^{\text{mod}} \bigcup \{(S_i, A_i, R_i^{\text{mod}})\}$
13: end for
14: return $\mathcal{D}^{\text{mod}}$
```

action $A \in \mathcal{A}$. From the ridge regression, for each action $A \in \mathcal{A}$, we can obtain the closed-form solution for the *policy's parameter vector* as $\boldsymbol{\theta}_A^n = \left(\boldsymbol{\Phi}_A^n\right)^{-1} \boldsymbol{b}_A^n$, where $\boldsymbol{b}_A^n = \boldsymbol{b}_A^{n-1} + \boldsymbol{S}_A^{n-1\top} \boldsymbol{R}_A^{n-1}$. Finally, by generalizing the UCB policy to a batch version, we obtain the updated policy $\boldsymbol{\pi}_n : \mathcal{S} \to \mathcal{A}$ as

$$\boldsymbol{\pi}_n(S) = \arg\max_{A \in \mathcal{A}} \left\langle \left(\boldsymbol{\Phi}_A^n\right)^{-1} \boldsymbol{b}_A^n, S \right\rangle + \mu \sqrt{S^\top \left(\boldsymbol{\Phi}_A^n\right)^{-1} S}. \tag{11}$$

4.3.2 Online Recommendation. In the online recommendation of the n-th episode, the agent chooses actions based on the new contexts under the updated policy $\boldsymbol{\pi}_n(S)$ for B steps, that is, offering recommendations on items to users using the updated policy:

$$A = \arg\max_{A \in \mathcal{A}} \left\langle \boldsymbol{\theta}_A^n, S \right\rangle + \mu \sqrt{S^\top \left(\boldsymbol{\Phi}_A^n\right)^{-1} S}. \tag{12}$$

The system then receives the context-action pairs and immediate feedback after recommending the items, and collects the resultant delayed reward and following-up time till the end of the batch. Finally, these records are stored into a data buffer $\mathcal{D}_{n+1}$ for the updating in the next iteration.

5 DISCUSSIONS

Regret analysis. We prove that the proposed CBDF enjoys a sublinear regret bound competing with the optimal policy. For any context $S_i \in \mathcal{S} \subseteq \mathbb{R}^d$ and any action $A \in \mathcal{A}$, we assume that the expectation of true reward $R_{i,A}^{\text{true}}$ is linear: there exists an unknown parameter vector $\boldsymbol{\theta}_A^* \in \mathbb{R}^d$ such that $\mathbb{E}\left[R_{i,A}^{\text{true}} \mid S_i\right] = \left\langle \boldsymbol{\theta}_A^*, S_i \right\rangle$, where the true reward is defined by $R_{i,A}^{\text{true}} = \lambda C_{i,A} + (1-\lambda) V_{i,A}$ ($C_{i,A}$ and $V_{i,A}$ denote true binary variables of click and conversion when executing action A given context S_i). To measure the convergence of streaming recommendation, we define the *regret* as follow:

$$\mathcal{R}(\{A_{I_{n,b}}\}_{n \in [N], b \in [B]}) := \max_{A \in \mathcal{A}} \sum_{n \in [N], b \in [B]} \left\langle \boldsymbol{\theta}_A^* - \boldsymbol{\theta}_{A_{I_{n,b}}}^*, S_{n,b} \right\rangle,$$

where $I_{n,b}$ denotes the index of the executed action using policy π_n (parameterized by $\{\boldsymbol{\theta}_A^n\}_{A \in \mathcal{A}}$) at step b in the n-th episode. We demonstrate the regret bound of our CBDF (Algorithm 1) as follows.

THEOREM 5.1 (REGRET BOUND OF CBDF). *Let $T = BN$, the important weights $w_i \le w_{\max}, i \in [T]$, $\|\boldsymbol{\theta}_A^*\|_2 < 1$, and $\mu = 1 + \sqrt{2 w_{\max}^2 \ln(2MB/\delta)}$. Assume that the conditional independence assumption about rewards in Theorem 4.1 holds, $M = O(\text{poly}(d))$ and $T \ge d^2$. Then, for an arbitrary sequence of contexts $\{S_i\}_{i=1}^T$, with probability at least $1 - N\delta$,*

$$\mathcal{R}(\{A_{I_{n,b}}\}_{n \in [N], b \in [B]}) \le 2\mu\sqrt{10M} \ln(T+1)(\sqrt{dT} + dB).$$

REMARK 1. *Setting $B = O\left(\sqrt{T/d}\right)$ yields an upper bound of regret of order $\widetilde{O}\left(\sqrt{dT}\right)$ w.r.t. T and d, which matches the lower bound in the undelayed feedback case up to a logarithmic factor [11, 14].*

Regret bounds guarantee the global convergence of approximating the optimal policy in sequential decision making [7, 43, 44]. Besides, the probabilistic error bound Eq. (7) is of order $O(B^{-1/2})$ given constant N, indicating that a suitable batch size B needs to be set. A theoretical value of the batch size is $B = C_B\sqrt{T/d}$ (i.e., $B = C_B^2 N/d$), where the constant $C_B \in [75, 80]$ is a suitable choice that has been verified in the experiments in Section 6.5.

The detailed proofs can be found in Appendix A.2.

Comparison with batched bandit. Recently, batched bandit has become an important topic in statistics and learning theory [12, 14, 25, 36, 42]. Little efforts have been spent to apply the batched bandit to recommender systems. Compared to the Batched Bandit Framework (BBF), the proposed CBDF has several striking differences: (1) In BBF, the reward of each action can only be received at the end of the batch. But CBDF considers a more realistic scenario where each reward is composed of the immediate feedback and the delayed feedback; (2) BBF updates the policy directly using the rewards. But in real-world environments, the feedbacks are usually limited, resulting in biased rewards that hinder the effectiveness of policy updating. CBDF maintains modified rewards using a counterfactual approach, relieving the bias.

In recommendation with delayed feedback, it is hard to formulate the counterfactual approach [1, 17, 26, 29, 31–33, 45] in online settings due to the delayed user behaviors and the bias affected by delayed time. CBDF can be seen as an attempt to model a counterfactual online approach to recommendation with delayed feedback.

6 EXPERIMENTS

We conducted experiments to test the performance of CBDF using three datasets: the synthetic dataset, publicly available Criteo dataset, and dataset collected from Tencent's WeChat.

6.1 Experimental Settings

According to theoretical results in Remark 1, we set the batch size as $B = C_B^2 N/d$, and set the constant $C_B \in [75, 80]$, where

$C_B \approx 79, 77, 75, 76$ on the synthetic dataset, two Criteo datasets, and the Tencent's WeChat, respectively. We set the weighting coefficient in the reward as $\lambda = C_\lambda \times \frac{1}{100}$ to maintain the relative importance of clicks and conversions, where C_λ is the estimate of CVR, and $C_\lambda = 0.4, 0.5, 0.7$ on the synthetic dataset, Criteo dataset, and Tencent's WeChat, respectively. The regularization parameter μ in the UCB policy was tuned in $[0.5 : +0.1 : 2]$. For our CBDF, the counterfactual deadline parameter is set as $C_\xi = 50\%$ to trade-offs the sizes of observed set and hold-out set, which is analyzed in Section 6.5.2. Guided by the inequality in Theorem 4.1, the importance weights is truncated as $w_i = \min\{w_i, w_{\max}\}$ where the constant $w_{\max}$ is tuned in $[1 : +0.1 : 2]$.

As for baselines, CBDF was compared with several algorithms that directly use the unmodified delayed user feedbacks, including:

Sequential Batch UCB (SBUCB) [14] is a generalization of LinUCB [22] where it is fed with batched data continuously;

Batched version of EXP3 (EXP3-B) [5]: EXP3 is a state-of-the-art algorithm for adversarial bandits. The batched version of EXP3 is denoted by EXP3-B.

CBDF was compared to another baseline that directly employs the reward modification developed for batch learning, called **Sequential version of Delayed Feedback Model (DFM-S) [10]**.

CBDF was also compared to the baseline that directly discards the instances in the batch data whose labels may be incomplete, called **SBUCB with Discard (SBUCB-D)**. It is a variant of SBUCB, which discards the potentially incomplete instances (i.e., $C = 1, Y = 0$).

The average reward was used to evaluate the accuracy of algorithms, which is computed by $\frac{1}{nB}\sum_{k=1}^{n}\sum_{b=1}^{B} R_{k,b}^{\text{true}}$ for the first n episodes, where $R_{k,b}^{\text{true}}$ is the true reward at step b in the k-th episode, defined by $R_{k,b}^{\text{true}} = \lambda\widehat{R}_{k,b} + (1-\lambda)\widetilde{R}(S_{k,b}, V_{k,b})$. λ was set as that in training. In the experiments, the algorithms were run 20 times and the average performances were reported.

6.2 Experiments on Synthetic Dataset

Following the practices in [27], we first conducted experiments on a synthetic dataset which simulates the online recommendation environment under delayed feedback. The synthetic data generation procedure was set as follows. Number of episodes: $N = 40$; batch size: $B = 10,000$; number of actions $M = 5$; context $S_i \in \mathbb{R}^d$: we drew elements of S_i independently from a Gaussian distribution $\mathcal{N}(0.1, 0.2^2)$, where $d = 10$; Click-Through-Rate (CTR): the CTRs for the 5 actions were respectively set as $\{30\%, 35\%, 40\%, 45\%, 50\%\}$; Conversion Rate (CVR) in context S_i: when $C_i = 1$, $\text{CVR}(\boldsymbol{S}_i) := \text{sigmoid}(\langle \boldsymbol{w}_\text{c}, S_i\rangle)$, where the coefficient vector $\boldsymbol{w}_\text{c} \in \mathbb{R}^d$ is sampled according to a Gaussian distribution as $\boldsymbol{w}_\text{c} \sim \mathcal{N}(\kappa_\text{c}\mathbf{1}_d, \sigma_\text{c}^2\boldsymbol{I}_d)$, and we set different means and standard deviations for different action with $\kappa_\text{c} \in [0 : -0.2 : -0.8]$ and $\sigma_\text{c} \in [0.01 : +0.01 : 0.05]$; delay time between the click and the conversion γ_i: when $C_i = 1$, the delay time γ_i is sampled according to an exponential distribution as $\gamma_i \sim \mathcal{E}(\lambda(S_i))$, where $\lambda(S_i) = \exp(\langle \boldsymbol{w}_\text{d}, S_i\rangle)$ and $\boldsymbol{w}_\text{d} \sim \mathcal{N}(\kappa_\text{d}\mathbf{1}_d, \sigma_\text{d}^2\boldsymbol{I}_d)$, and we set different means and standard deviations for different action with $\kappa_\text{d} \in [0 : +0.2 : 0.8]$ and $\sigma_\text{d} \in [0.01 : +0.01 : 0.05]$.

Figure 6(a) reports the average reward curves of CBDF and the baselines DFM-S, SBUCB, EXP3-B, and SBUCB-D. We observed that CBDF achieved the highest average reward after running about 10 episodes. Moreover, CBDF converged faster than SBUCB and EXP3-B that directly use the unmodified rewards, demonstrating the effectiveness of the reward modification in CBDF. CBDF also outperformed the discard approach SBUCB-D, while the discard trick in SBUCB-D could help improve the performance of SBUCB in a simple synthetic environment. Besides, DFM-S obtained much worse because its reward modification method was originally developed for batch learning, and resulting in low sample efficiency when being applied to sequential decision making problems.

6.3 Experiments on Criteo Dataset

In these experiments, we made use of the publicly available Criteo dataset[3], which consists of a portion of Criteo's traffic on display ads over a period of two months. Each context in a Criteo record consists of 8 integer features and 9 categorical features. Following the practices in [41], the categorical features were represented as one-hot vectors, which were then concatenated to the integer features. The dimensionality of the feature vectors was reduced to 50 by conducting principal component analysis (PCA).

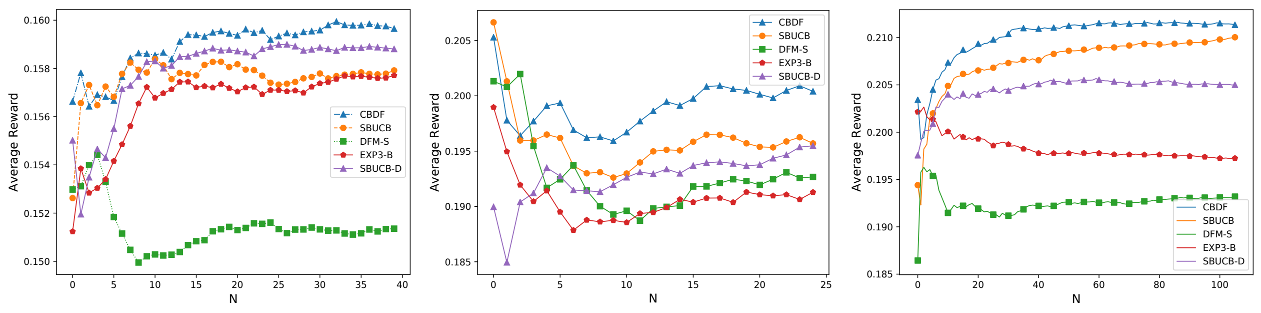
All of the algorithms were tested in a simulated online environment that was trained on users' logs in the Criteo dataset. Specifically, we chose some campaigns from the Criteo dataset, where each campaign represents a type of items and corresponds to an action. The online environment consists of a model for the delay time and another model for the CVR, both were well trained by applying DFM on each chosen campaign (with true user feedbacks). The models' AUCs are ranging from 70% to 90%, assuring that the online environment can provide nearly realistic feedbacks. To simulate the uncertainty of user behaviors, Gaussian noises with zero-mean were added to the model parameters. At each step, the online environment randomly selected a campaign and samples one context from this campaign, and revealed the context to the agent with a preset CTR. To generate a reasonable sequence of instances, the environment kept the order of timestamps for each campaign.

We tested CBDF and the baselines with two online environments on the Criteo dataset: in `Criteo-recent-5actions`, 5 campaigns (75, 021 instances, batch size $B = 3,000$) were chosen from the recent campaigns, corresponding to 5 actions; in `Criteo-all-15actions`, 15 campaigns (1, 278, 556 instances, batch size $B = 12,000$) were chosen from all the campaigns, corresponding to 15 actions. Figure 6(b) and (c) respectively depicts the curves in terms of average rewards of CBDF and the baselines on these two online environments. We can conclude that CBDF persistently performed better than the baselines. We noticed that all the algorithms achieve decreasing average rewards at the beginning of the testing, due to the influence of the biased feedbacks. Also noted that CBDF can correct the biases more effectively and approximate the optimal policy after several episodes, indicating the superiority of our counterfactual reward modification in streaming recommendation with delayed feedback.

6.4 Experiments on Real Commercial Product

We also tested the performance of CBDF on a real dataset collected from Tencent's WeChat app, to verify its effectiveness on real products. In the app, after clicking a recommended coupon, a user may convert the coupon after some time, or just leave it there. The dataset was collected during a 1-month period (with a

[3]https://labs.criteo.com/2013/12/conversion-logs-dataset/



(a) synthetic dataset (b) Criteo-recent-5actions (c) Criteo-all-15actions

Figure 6: Average rewards of DFM-S, SBUCB, EXP3-B, SBUCB-D, and CBDF on the synthetic dataset and the Criteo dataset.

Table 2: Performance comparison of coupon recommendation on a real commercial dataset from Tencent's WeChat.

Algorithm	CVR	CTCVR	Running time (s)
DFM-S	0.7389 ± 0.0090	0.2794 ± 0.0065	311.5 ± 10.5853
SBUCB	0.7307 ± 0.0015	0.2807 ± 0.0008	60.9 ± 3.2078
EXP3-B	0.4809 ± 0.0029	0.1792 ± 0.0050	26.9 ± 0.5385
SBUCB-D	0.6831 ± 0.0026	0.2643 ± 0.0012	37.0 ± 0.4472
CBDF	**0.7775** ± 0.0056	**0.3046** ± 0.0025	66.5 ± 6.9606

subsampling), and consists of 216, 568 instances from 5 categories of coupons. Each context is described by 86 numerical features and 16 categorical features. The timestamps of clicks and conversions were also recorded.

Following the previous settings on the Criteo dataset, we represented the categorical features as a one-hot vector, reduced the dimensionality of the feature vectors to 50 by PCA. We set the batch size $B = 5,000$ (yielding that $C_B \approx 76$). The action space contains 5 actions, each corresponding to one coupon category. Due to the limitation of real online experiments, in this experiment we still trained DFM (with true user feedbacks) as the online environment (AUCs ranging from 75% to 90%). Table 2 reports the performance of CBDF and the baselines, in terms of CVR (mean ± std), CTCVR (mean ± std), and total running time (mean ± std) over all the episodes, where CTCVR = CTR × CVR [23]. The results indicated that CBDF performed better than the baselines with the improvements of 3.86% CVR and 2.39% CTCVR over the second-best algorithm. Besides, CBDF was much more efficient than DFM-S, and had time costs that comparable to the online algorithm SBUCB. The results demonstrate the efficiency of the counterfactual reward modification adopted by CBDF. Although DFM-S also used the exponential distribution of delay time as the prior knowledge, its reward modification originally developed for batch learning still had much worse performance. From the reward curves in Figure 7(a), we found that CBDF outperformed all of the baselines after several episodes, indicating its effectiveness in real commercial products.

6.5 Analysis

6.5.1 Empirical Verification of Remark 1. Section 5 gives a theoretical optimal value $B = C_B^2 N/d$. We empirically verified the correctness of this conclusion. We fixed the number of instances (i.e., $B \times N$) on synthetic data (160, 000 instances) and `Criteo-recent-5actions` (75, 021 instances), and tested CBDF with different B values (resulting in different number of episodes N). Figure 7(b) and (c) show the performance curves w.r.t. proportion of episodes that had passed (denoted as '$x\%N$'). From the curves in Figure 7(b), we can observe that CBDF with $B = 10,000$ achieved persistently higher rewards than the baselines on the synthetic dataset, yielding that $C_B = \sqrt{Bd/N} \approx 79$. Similar phenomenon can also be observed on the experiments of `Criteo-recent-5actions`. As shown in Figure 7(c), CBDF with $B = 3,000$ achieved the highest average rewards in which $C_B \approx 77$. In the previous experiments in Section 6.3, choosing $C_B \approx 75$ on the larger setting `Criteo-all-15actions` and $C_B \approx 76$ on Tencent's WeChat app also achieved the best performance. All of the results verified the theoretical conclusion: $B = C_B^2 N/d$ is a nearly optimal choice when setting $C_B \in [75, 80]$.

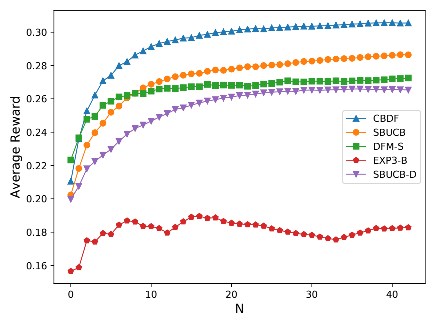
Table 3: Performance comparison of CBDF with different counterfactual deadlines ξ on the synthetic dataset, where $\xi = C_\xi \times (t^{\text{end}} - t^{\text{str}}) + t^{\text{str}}$ and $C_\xi \in (0, 100\%)$.

C_ξ	CVR	CTCVR	Average Reward
30%	0.3835 ± 0.0044	0.1580 ± 0.0005	0.1590 ± 0.0006
40%	0.3855 ± 0.0071	0.1580 ±0.0006	0.1590 ± 0.0006
50%	**0.3891** ± 0.0052	0.1581 ± 0.0005	0.1591 ± 0.0005
60%	0.3881 ± 0.0043	0.1583 ± 0.0004	0.1593 ± 0.0005
70%	0.3848 ± 0.0064	**0.1584** ± 0.0005	**0.1594** ± 0.0006
80%	0.3843 ± 0.0057	0.1578 ± 0.0005	0.1588 ± 0.0006

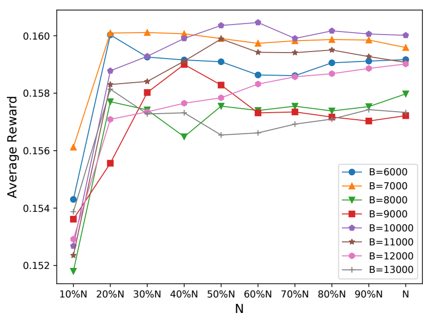
6.5.2 Influence of Counterfactual Deadline. The counterfactual deadline ξ trade-offs the sizes of observed set and hold-out set. Larger observed set (larger C_ξ) means more instances for estimating $\boldsymbol{\beta}$'s while hurts the deriving of Y^{obs} and e^{obs}, and vise versa. We conducted experiments to test the performance of CBDF with different C_ξ values. The results showed in Table 3 indicate that CBDF is not sensitive to parameter C_ξ, and setting $C_\xi \in [50\%, 70\%]$ achieved best performance.

7 CONCLUSION

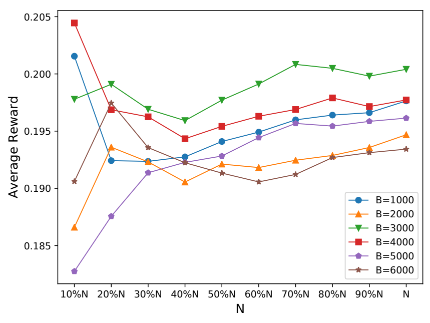
In real-world streaming recommender systems, the data are fed continuously and the models need to be updated frequently, and



(a) Dataset from Tencent's WeChat



(b) Influence of B on synthetic data



(c) Influence of B on `Criteo-recent-5actions`

Figure 7: (a) Average rewards of DFM-S, SBUCB, EXP3-B, SBUCB-D, and the proposed CBDF on real commercial product data from WeChat; (b) CBDF with different batch sizes on the synthetic dataset; (c) CBDF with different batch sizes on the Criteo dataset.

some conversions have not yet occurred when the training data are collected, resulting in the issue of delayed feedback. In this paper, we propose a counterfactual approach to modifying the rewards and learning batched bandit recommendation models under delayed feedback, called CBDF. We proved that CBDF is statistically unbiased for reward estimation and enjoys a sublinear regret bound against the optimal policy. Experiments on synthetic data, public benchmark, and commercial product data from WeChat showed that CBDF can outperform the state-of-the-art baselines.

ACKNOWLEDGMENTS

This work was funded by the National Key R&D Program of China (2019YFE0198200), the National Natural Science Foundation of China (No. 62006234, No. 61872338, No. 61832017), the Tencent WeChat Rhino-Bird Focused Research Program, Beijing Academy of Artificial Intelligence (BAAI2019ZD0305), China Association for Science and Technology "Youth Project of High-End Technology Innovation and Think Tank", and Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098.

A PROOF OF THEOREMS

A.1 Proof of Theorem 4.1

Proof of Theorem 4.1. Combining the definition of $\widetilde{R}^{\text{mod}}$ with Eq. (2), Eq. (3) and Eq. (4), we have

$$\mathbb{E}_Y\left[\widetilde{R}^{\text{mod}}(S,Y)\right]=\mathbb{E}_Y\left[\frac{\Pr\{V=1\mid S\}}{\Pr\{Y=1\mid S\}}\widetilde{R}(S,Y)\right]=\mathbb{E}_V\left[\widetilde{R}(S,V)\right].$$

Since $S_i, i\in[T]$ are i.i.d., from Eq. (6) we have

$$\mathbb{E}_{S,Y}\left[\widetilde{R}^{\text{mod}}(S,Y)\right]=\mathbb{E}_{S,V}\left[\widetilde{R}(S,V)\right]. \tag{13}$$

From $w_i\le w_{\max}, i\in[T]$ we can obtain that $\widetilde{R}^{\text{mod}}(S,Y)\in[0,w_{\max}]$. Then combing Eq. (13) with Hoeffding's inequality yileds that

$$\Pr\left\{\left|\sum_{i=1}^{T}\frac{\widetilde{R}^{\text{mod}}(S_i,Y_i)}{T}-\mathbb{E}_{S,V}\left[\widetilde{R}(S,V)\right]\right|\ge\varepsilon\right\}\le 2\exp\left(\frac{-2\varepsilon^2T}{w_{\max}^2}\right).$$

Letting $\delta=2\exp\left(-2\varepsilon^2T/w_{\max}^2\right)$ yields the final result. □

A.2 Proof of Theorem 5.1

Proof of Theorem 5.1. We first demonstrate the instantaneous regret bound of CBDF at some step in one episode. For convenience we drop all the superscripts and subscripts about n and b. We introduce the notation $\boldsymbol{L}_A\in\mathbb{R}^{N_A\times d}$ and $\boldsymbol{T}_A\in\mathbb{R}^{N_A}$, representing the matrix that stores all the received context vectors corresponding to action A, and the reward vector that stores all the modified rewards of action A, respectively. By the formulation of $\boldsymbol{\theta}_A$ and the triangular inequality, we first obtain that

$$\begin{aligned}&\left|\langle\boldsymbol{\theta}_A,S\rangle-\langle\boldsymbol{\theta}_A^*,S\rangle\right|=\left|S^\top(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}\boldsymbol{L}_A^\top\boldsymbol{T}_A-S^\top\boldsymbol{\theta}_A^*\right|\\&\le\underbrace{\left|S^\top(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}\boldsymbol{L}_A^\top(\boldsymbol{T}_A-\boldsymbol{L}_A\boldsymbol{\theta}_A^*)\right|}_{Z_A^{(1)}}+\underbrace{\left|S^\top(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}\boldsymbol{\theta}_A^*\right|}_{Z_A^{(2)}}.\end{aligned}$$

Next, we bound $Z_A^{(1)}$. Let $R_A^{\text{mod}}(S,Y)$ be the modified reward that is obtained when action A is executed. For $\forall S\in\mathcal{S}$, from Eq. (6) in Theorem 4.1, we have $\mathbb{E}_Y\left[R_A^{\text{mod}}(S,Y)\right]=\mathbb{E}\left[R_A^{\text{true}}\mid S\right]=\langle\boldsymbol{\theta}_A^*,S\rangle$, yielding that $\mathrm{E}[\boldsymbol{T}_A]=\boldsymbol{L}_A\boldsymbol{\theta}_A^*$. Letting $\nu>0$ be some constant, since $R_A^{\text{mod}}(S,Y)\le w_{\max}$, by Azuma-Hoeffding bound and the conditional independence assumption of rewards, we get

$$\begin{aligned}&\Pr\left\{\left|Z_A^{(1)}\right|\ge\nu\sqrt{S^\top(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}S}\right\}\\&\le 2\exp\left\{-\frac{\nu^2S^\top(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}S}{2w_{\max}^2\|\boldsymbol{L}_A(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}S\|_2^2}\right\}.\end{aligned}\tag{14}$$

Since $\|\boldsymbol{L}_A(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}S\|_2^2\le S^\top(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}S$, combing Eq. (14) with the union bound, yields that, with probability at least $1-\delta$, for $\forall A\in\mathcal{A}$ at arbitrary step $\forall b\in[B]$, $\left|Z_A^{(1)}\right|\le\nu\sqrt{S^\top(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}S}$, where $\nu=\sqrt{2w_{\max}^2\ln(2MB/\delta)}$. Since $\boldsymbol{\Phi}_A$ is positive semi-definite, we can obtain that $|Z_A^{(2)}|\le\sqrt{S^\top(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}S}$, combined with the upper bound of $|Z_A^{(1)}|$ yielding that

$$\left|\langle\boldsymbol{\theta}_A,S\rangle-\langle\boldsymbol{\theta}_A^*,S\rangle\right|\le(1+\nu)\sqrt{S^\top(\boldsymbol{I}_d+\boldsymbol{\Phi}_A)^{-1}S}. \tag{15}$$

Let $\mu=1+\nu$, combining Eq. (15) with lemma 3 in [14] gives the final bound. □

REFERENCES

[1] Qingyao Ai, Tao Yang, Huazheng Wang, and Jiaxin Mao. 2020. Unbiased Learning to Rank: Online or Offline? *CoRR* abs/2004.13574 (2020).

[2] Peter Auer. 2002. Using Confidence Bounds for Exploitation-Exploration Trade-offs. *Journal of Machine Learning Research* 3 (2002), 397–422.

[3] Santiago Balseiro, Negin Golrezaei, Mohammad Mahdian, Vahab Mirrokni, and Jon Schneider. 2019. Contextual Bandits with Cross-Learning. In *Advances in Neural Information Processing Systems 32*. 9679–9688.

[4] Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert E. Schapire. 2011. Contextual Bandit Algorithms with Supervised Learning Guarantees. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, Vol. 15. 19–26.

[5] Ilai Bistritz, Zhengyuan Zhou, Xi Chen, Nicholas Bambos, and Jose Blanchet. 2019. Online EXP3 Learning in Adversarial Bandits with Delayed Feedback. In *Advances in Neural Information Processing Systems 32*. 11349–11358.

[6] Léon Bottou, Jonas Peters, Joaquin Quiñonero Candela, Denis Xavier Charles, Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Y. Simard, and Ed Snelson. 2013. Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising. *Journal of Machine Learning Research* 14 (2013), 3207–3260.

[7] Sébastien Bubeck and Nicolò Cesa-Bianchi. 2012. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends in Machine Learning* 5, 1 (2012), 1–122.

[8] Badrish Chandramouli, Justin J. Levandoski, Ahmed Eldawy, and Mohamed F. Mokbel. 2011. StreamRec: A Real-time Recommender System. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1243–1246.

[9] Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A Hasegawa-Johnson, and Thomas S Huang. 2017. Streaming Recommender Systems. In *Proceedings of the 26th International Conference on World Wide Web*. 381–389.

[10] Olivier Chapelle. 2014. Modeling Delayed Feedback in Display Advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1097–1105.

[11] Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. 2011. Contextual Bandits with Linear Payoff Functions. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. 208–214.

[12] Zijun Gao, Yanjun Han, Zhimei Ren, and Zhengqing Zhou. 2019. Batched Multi-armed Bandits Problem. In *Advances in Neural Information Processing Systems 32*. 501–511.

[13] Aditya Grover, Todor M. Markov, Peter M. Attia, Norman Jin, Nicolas Perkins, Bryan Cheong, Michael H. Chen, Zi Yang, Stephen J. Harris, William C. Chueh, and Stefano Ermon. 2018. Best Arm Identification in Multi-armed Bandits with Delayed Feedback. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*. 833–842.

[14] Yanjun Han, Zhengqing Zhou, Zhengyuan Zhou, Jose H. Blanchet, Peter W. Glynn, and Yinyu Ye. 2020. Sequential Batch Learning in Finite-Action Linear Contextual Bandits. *CoRR* abs/2004.06321 (2020).

[15] 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 *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. 5:1–5:9.

[16] Amélie Héliou, Panayotis Mertikopoulos, and Zhengyuan Zhou. 2020. Gradient-free Online Learning in Games with Delayed Rewards. In *Proceedings of the 37th International Conference on Machine Learning*.

[17] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 15–24.

[18] Martin Jakomin, Zoran Bosnic, and Tomaz Curk. 2020. Simultaneous Incremental Matrix Factorization for Streaming Recommender Systems. *Expert Systems with Applications* 160 (2020), 113685.

[19] J. D. Kalbfleisch and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data (2nd Edition)*. John Wiley & Sons.

[20] Konstantinos V Katsikopoulos and Sascha E Engelbrecht. 2003. Markov Decision Processes with Delays and Asynchronous Cost Collection. *IEEE Trans. Automat. Control* 48, 4 (2003), 568–574.

[21] Sofia Ira Ktena, Alykhan Tejani, Lucas Theis, Pranay Kumar Myana, Deepak Dilipkumar, Ferenc Huszár, Steven Yoo, and Wenzhe Shi. 2019. Addressing Delayed Feedback for Continuous Training with Neural Networks in CTR Prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 187–195.

[22] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A Contextual-bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. 661–670.

[23] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1137–1140.

[24] Gergely Neu and Julia Olkhovskaya. 2020. Efficient and Robust Algorithms for Adversarial Linear Contextual Bandits. In *Proceedings of the 33rd Conference on Learning Theory*. 3049–3068.

[25] Vianney Perchet, Philippe Rigollet, Sylvain Chassang, and Erik Snowberg. 2016. Batched Bandit Problems. *The Annals of Statistics* 44, 2 (2016), 660–681.

[26] Noveen Sachdeva, Yi Su, and Thorsten Joachims. 2020. Off-policy Bandits with Deficient Support. *CoRR* abs/2006.09438 (2020). arXiv:2006.09438 `https://arxiv.org/abs/2006.09438`

[27] Yuta Saito, Gota Morishita, and Shota Yasui. 2020. Dual Learning Algorithm for Delayed Conversions. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1849–1852.

[28] Eren Sezener, Marcus Hutter, David Budden, Jianan Wang, and Joel Veness. 2020. Online Learning in Contextual Bandits using Gated Linear Networks. In *Advances in Neural Information Processing Systems 33*.

[29] Yi Su, Lequn Wang, Michele Santacatterina, and Thorsten Joachims. 2019. CAB: Continuous Adaptive Blending for Policy Evaluation and Learning. In *Proceedings of the 36th International Conference on Machine Learning*. 6005–6014.

[30] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. 2007. Covariate Shift Adaptation by Importance Weighted Cross Validation. *Journal of Machine Learning Research* 8 (2007), 985–1005.

[31] Adith Swaminathan and Thorsten Joachims. 2015. Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization. *Journal of Machine Learning Research* 16 (2015), 1731–1755.

[32] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual Risk Minimization: Learning from Logged Bandit Feedback. In *Proceedings of the 32nd International Conference on Machine Learning*. 814–823.

[33] Adith Swaminathan and Thorsten Joachims. 2015. The Self-Normalized Estimator for Counterfactual Learning. In *Advances in Neural Information Processing Systems 28*. 3231–3239.

[34] Claire Vernade, Olivier Cappé, and Vianney Perchet. 2017. Stochastic Bandit Models for Delayed Conversions. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*.

[35] Thomas J Walsh, Ali Nouri, Lihong Li, and Michael L Littman. 2009. Learning and Planning in Environments with Delayed Feedback. *Autonomous Agents and Multi-Agent Systems* 18, 1 (2009), 83.

[36] Chi-Hua Wang and Guang Cheng. 2020. Online Batch Decision-Making with High-Dimensional Covariates. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. 3848–3857.

[37] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. 2018. Neural Memory Streaming Recommender Networks with Adversarial Training. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2467–2475.

[38] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. 2018. Streaming Ranking Based Recommender Systems. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 525–534.

[39] Qingyun Wu, Hongning Wang, Liangjie Hong, and Yue Shi. 2017. Returning is Believing: Optimizing Long-term User Engagement in Recommender Systems. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1927–1936.

[40] Shota Yasui, Gota Morishita, Komei Fujita, and Masashi Shibata. 2020. A Feedback Shift Correction in Predicting Conversion Rates under Delayed Feedback. In *Proceedings of the Web Conference 2020*. 2740–2746.

[41] Yuya Yoshikawa and Yusaku Imai. 2018. A Nonparametric Delayed Feedback Model for Conversion Rate Prediction. *arXiv:1802.00255v1* (2018).

[42] Kelly W. Zhang, Lucas Janson, and Susan A. Murphy. 2020. Inference for Batched Bandits. In *Advances in Neural Information Processing Systems 33*.

[43] Xiao Zhang and Shizhong Liao. 2019. Incremental Randomized Sketching for Online Kernel Learning. In *Proceedings of the 36th International Conference on Machine Learning*. 7394–7403.

[44] Xiao Zhang, Shizhong Liao, Jun Xu, and Ji-Rong Wen. 2021. Regret Bounds of Online Kernel Selection in Continuous Kernel Space. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

[45] Shengyao Zhuang and Guido Zuccon. 2020. Counterfactual Online Learning to Rank. In *Advances in Information Retrieval 42nd European Conference on IR Research*. 415–430.