Unifying Offline Causal Inference and Online Bandit Learning for Data Driven Decision

Li Ye
The Chinese University of Hong Kong
Hong Kong, China

Hong Xie
College of Computer Science, Chongqing University
China

Yishi Lin
Tencent Inc.
China

John C.S. Lui
The Chinese University of Hong Kong
Hong Kong, China

ABSTRACT

A fundamental question for companies with large amount of logged data is: *How to use such logged data together with incoming streaming data to make good decisions?* Many companies currently make decisions via online A/B tests, but wrong decisions during testing hurt users' experiences and cause irreversible damage. A typical alternative is offline causal inference, which analyzes logged data alone to make decisions. However, these decisions are not adaptive to the new incoming data, and so a wrong decision will continuously hurt users' experiences. To overcome the aforementioned limitations, we propose a framework to unify offline causal inference algorithms (e.g., weighting, matching) and online learning algorithms (e.g., UCB, LinUCB). We propose novel algorithms and derive bounds on the decision accuracy via the notion of "regret". We derive the first upper regret bound for forest-based online bandit algorithms. Experiments on two real datasets show that our algorithms outperform other algorithms that use only logged data or online feedbacks, or algorithms that do not use the data properly.

ACM Reference Format:
Li Ye, Hong Xie, Yishi Lin, and John C.S. Lui. 2021. Unifying Offline Causal Inference and Online Bandit Learning for Data Driven Decision. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3442381.3449982

1 INTRODUCTION

How to make good decisions is a key challenge in many web applications, i.e., an Internet company such as Facebook that sells in-feeds advertisements (or "ads" for short) needs to decide whether to place an ad below videos or below images, as illustrated in Fig. 1.

It is common that Internet companies have archived lots of logged data which may assist decision making. For example, Internet companies which sell in-feeds advertisements have logs of advertisements' placement, as well as users' feedbacks to these ads as illustrated in Table 1. The question is: *how to use these logs to make a better decision?* To motivate this problem, consider Example 1.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449982

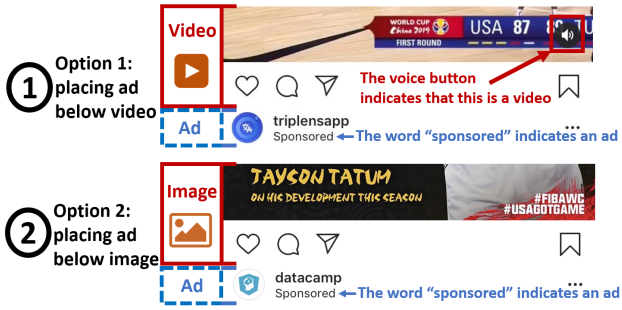

Figure 1: In-feeds ad placement of Instagram

Table 1: Logged data of a company that sells in-feeds ads

	action	contexts			outcome
ID	**Ad below video?**	**User likes videos?**	**Age**	**...**	**Click?**
1	no	no	30	...	no (0)
2	yes	yes	20	...	yes (1)
⋮	⋮	⋮	⋮	⋮	⋮

Example 1. *10,000 new users will arrive to see the advertisement. The Internet company needs to decide whether to place the advertisement (ad) below a video or below an image. The company's goal is to increase click-through rate from these 10,000 new users. Users are of two types — users who "like" or users who "dislike" videos. Assume 50% of these new user like (or dislike) videos. The "true click rates" for each types of user, which are unknown to the company, are summarized in Table 2. Furthermore, the company has a logged statistics of the past 400 users, half of whom like (or dislike) videos, as shown in Table 3.*

Table 2: True click rates of each type of user. Action 2 (*ad below image*, with "*") is better for both types of users.

Action # \ User type	Like videos	Dislike videos
1. Ad below video	*11%*	*1%*
2. Ad below image*	*14%*	*4%*

One may consider the following three strategies to make decisions.
Empirical Average. *The company chooses the action with the highest average click rate in the logged data to serve 10,000 incoming users.*

Table 3: Average click rate in logs of 400 users. In the logged data, users who like videos were more likely to see ads below videos, as they subscribed to more videos.

User type / Action #	Like videos (200 users)	Dislike videos (200 users)
1. Ad below video	*10% of 150 ads*	*2% of 50 ads*
2. Ad below image*	*12% of 50 ads*	*4% of 150 ads*

For logs in Table 3, the average click rate for "ad below video" is (10%×150+2%×50)/(150+50)=8%. Similarly, the average click rate is 6% for "ad below image". Thus, the company chooses to place "ad below video" for the 10,000 incoming users. But it is the wrong action implied by the true click rates in Table 2. This method fails because it ignores users' *preferences to videos*. This example is similar to the Simpson's Paradox [33] in the discussion of causality.

Offline causal inference. *First, the company computes the average click rates w.r.t. each user type (as in Table 3). Second, for each action, it computes the weighted average of such type-specific click rates where the weight is the fraction of users in each type.* For logs in Table 3, the weighted average click rate for action 1 (ad below video) is 10%×(200/400) + 2%×(200/400)=6%. Similarly, the weighted average click rate for action 2 is (12%+4%)/2=8%. Thus, the company chooses action 2 for logs in Table 3. However, the causal inference strategy has a risk of not finding the right action as the logs are only finite samples from the population. For example, in another sample statistics where the number of clicks for users who dislike videos and see ad below video (the upper right cell in Table 3) increases from 1 (i.e. 2%×50) to 4, the "offline causal inference" strategy will then choose the inferior action of "placing ad below video".

Online A/B testing. *Each of the first 4,000 incoming users is randomly assigned to group A or B with equal probability. Users in group A see ads below videos (action 1), while users in group B see ads below images (action 2). Then, the company selects the action with a higher average testing click rate for the remaining 6,000 users.* In this A/B test, 2,000 testing users in group A suffer from the inferior action.

The above three strategies have their own limitations. Taking the "empirical average" leads to a wrong decision by ignoring the important factor of users' preferences. "Offline causal inference" only uses the logged data and has a risk to make the wrong decision due to the incompleteness of the logged samples. "A/B testing" only uses the online data and pays a *high cost* of testing the inferior actions. In this paper, we propose a novel strategy which can use both the logged data and the online feedbacks.

Causal inference + online learning (our method). *The company applies offline causal inference to "judiciously" use the logged data to improve the efficiency of an online learning algorithm. For example, UCB is used [4] as the online learning algorithm in Table 4.*

Table 4 shows that our algorithm achieves the highest revenue for Example 1. The key is to choose the appropriate data from the logged data to improve our decision making.

Our contributions are:

• **A unified framework with novel algorithms.** We formulate a general online decision making problem, which utilizes logged data to improve both (1) *context-independent decisions*, and (2) *contextual decisions*. Our framework unifies offline causal inference and online bandit algorithms. Our framework is generic enough to combine different causal inference methods like matching and weighting [6], and bandit algorithms like UCB [4] and LinUCB [30]. This unification inspires us to extend the offline regression-forest to an *"ϵ-decreasing multi-action forest"* online learning algorithm.

• **Theoretical regret bounds.** We derive regret upper bounds for algorithms in our framework. We show how the logged data can reduce the regret of online decisions. Moreover, we derive an asymptotic regret bound for the *"ϵ-decreasing multi-action forest"* algorithm. To the best of our knowledge, this is the first regret analysis for a forest-based online bandit algorithm.

• **Extensive empirical evaluations.** Experiments on synthetic data and real web datasets from Weixin and Yahoo show that our algorithms that use both logged data and online feedbacks can make the right decision with the highest accuracy. On the Weixin's dataset, we reduce the regret of the decision maker by 37.1% (or 45.5%) compared to the online bandit learning algorithm (or offline causal inference algorithm). On the Yahoo's dataset, we reduce the regret by 21.1% compared to LinUCB of [30]. Moreover, we show our algorithms outperforms the heuristics that uses supervised learning algorithm to learn from offline data for decision making.

Table 4: The expected revenue($) of the four strategies over 10,000 users. Suppose each click yields a revenue of $1. The optimal expected revenue is $900 (where the optimal action is to "place videos below an image"). A strategy's *"regret"* is the difference between the optimal revenue and its revenue.

Strategy	Empirical average	Causal inference	A/B testing	Our method
Expected Revenue	674.4	847.7	839.9	**894.4**
Expected Regret	225.6	52.3	60.1	**5.6**

2 MODEL & PROBLEM FORMULATION

Our approach for the new online decision problem uses the logged data to improve online decision accuracy (more details in Section 3). Note that the observed logged data may have "selection bias" on the actions, while in the online environment actions are chosen by the decision maker. This is why we need to find a formal approach to "connect" the logged data and the online data for correct usage.

In this section, we first present the logged data model. Then we model the online environment. Finally, we present the online decision problem which aims to utilize both the logged data and online feedbacks to minimize the regret.

2.1 Model of Logged Data

We consider a tabular logged dataset (e.g., Table 1), which was collected before the running of online decision algorithms. The logged dataset has $I \in \mathbb{N}_+$ items, denoted by $\mathcal{L} \triangleq \{(a_i, \boldsymbol{x}_i, y_i) | i \in [-I]\}$, where $(a_i, \boldsymbol{x}_i, y_i)$ denotes the i^{th} recorded data item and $[-I] \triangleq \{-I, -I+1, \ldots, -1\}$. Here, we use *negative* indices to indicate that the logged data were collected in the past. The action for data item i is denoted as $a_i \in [K] \triangleq \{1, \ldots, K\}$, where $K \in \mathbb{N}_+$. The actions in the logged data can be generated according to the users' natural behaviors or by the company's interventions. For example, option 1 and 2 in Figure 1 are actions. The $y_i \in \mathcal{Y} \subseteq \mathbb{R}$ denotes the outcome (or reward). The $\boldsymbol{x}_i \triangleq (x_{i,1}, \ldots, x_{i,d}) \in \mathcal{X}$ denotes the

contexts (or features) of data item i, where $d \in \mathbb{N}_+$ and $\mathcal{X} \subseteq \mathbb{R}^d$. The contexts are also known as *"observed confounders"* [6]. We use $\boldsymbol{u}_i \triangleq (u_{i,1}, \ldots, u_{i,\ell}) \in \mathcal{U}$, where $\ell \in \mathbb{N}_+$ and $\mathcal{U} \subseteq \mathbb{R}^\ell$, to model the unobserved confounders. The $\boldsymbol{u}_i$ captures latent or hidden contexts, e.g., a user's monthly income.

Now we introduce the generating process of the logged data. For the i^{th} user with context $\boldsymbol{x}_i$, let A_i be the random variable for the action of the i^{th} user. To capture the randomness of the outcome, let the random variable $Y_i(k)$ denote the outcome for the i^{th} user if we had changed the action of the i^{th} user to k. When $k \neq a_i$, $Y_i(k)$ is also called a *"potential outcome"* in the causal model [35] and it is not recorded in the logged data. We have the following two assumptions, which are common for causal inference [35].

Assumption 1 (Stable unit for logged data). *The potential outcome of a data item is independent of the actions of other data items, i.e.* $\mathbb{P}[Y_i(k)=y|A_i=a_i, A_j=a_j] = \mathbb{P}[Y_i(k)=y|A_i=a_i], \forall i\in[-I], j\neq i$.

Assumption 2 (Ignorability). *The potential outcomes of a data item i are independent of the action a_i given the context $\boldsymbol{x}_i$ (so that we can ignore $\boldsymbol{u}_i$'s impacts), i.e.* $[Y_i(1), \ldots, Y_i(K)] \perp\!\!\!\perp A_i|\boldsymbol{x}_i, \forall i \in [-I]$.

Assumption 2 holds in Example 1 since the decision maker observes users' *preferences to videos* which determine the users' types. In Table 2, each type of users have a fixed click rates for the actions, which are independent of action.

2.2 Model of Online Decision Environment

Consider a discrete time system $t \in [T]$, where $T \in \mathbb{N}_+$ and $[T] \triangleq \{1, \ldots, T\}$. In time slot t, one new user arrives, and she is associated with the context $\boldsymbol{x}_t \in \mathcal{X}$ and unobserved confounders $\boldsymbol{u}_t \in \mathcal{U}$. Then, the decision maker chooses an action $a_t \in [K]$, and observes the outcome (or reward) y_t corresponding to this chosen action.

Consider that the confounders $(\boldsymbol{x}_t, \boldsymbol{u}_t)$ are independent and identically generated by a cumulative distribution function $F_{X,U}(\boldsymbol{x}, \boldsymbol{u}) \triangleq \mathbb{P}[\boldsymbol{X} \leq \boldsymbol{x}, \boldsymbol{U} \leq \boldsymbol{u}]$, where $\boldsymbol{X} \in \mathcal{X}$ and $\boldsymbol{U} \in \mathcal{U}$ denote two random variables. The distribution $F_{X,U}(\boldsymbol{x}, \boldsymbol{u})$ characterizes the joint distribution of the confounders over the whole user population. If we marginalize over $\boldsymbol{u}$, then the observed confounders $\boldsymbol{x}_t$ are independently identically generated from the marginal distribution $F_{\boldsymbol{X}}(\boldsymbol{x}) \triangleq \mathbb{P}[\boldsymbol{X} \leq \boldsymbol{x}]$. Let the random variable $Y_t(k)$ denote the outcome of taking action k in time slot t.

Assumption 3 (Stable unit for online model). *The outcome $Y_t(k)$ in time t is independent of the actions in other time slots, i.e.*

$$\mathbb{P}[Y_t(k)=y|A_t=a_t, A_s=a_s] = \mathbb{P}[Y_t(k)=y|A_t=a_t], \forall t\in[T], s\neq t. \quad (1)$$

In the online setting, before the decision maker chooses the action, the distributions of the "potential outcomes" $[Y_t(1), \cdots, Y_t(K)]$ are determined given the confounders $(\boldsymbol{x}_t, \boldsymbol{u}_t)$. Moreover, as the unobserved confounders $\boldsymbol{u}_t$ are i.i.d. in different time slots, the potential outcomes are independent of how we select the action, given the user's context $\boldsymbol{x}_t$. Formally, we have the following property.

Property 1. *The potential outcomes in time slot t satisfies*

$$[Y_t(1), \ldots, Y_t(K)] \perp\!\!\!\perp A_t|\boldsymbol{x}_t, \forall t \in [T]. \quad (2)$$

One can see that Assumption 1 and 2 for the logged data correspond to Assumption 3 and Property 1 for the online decision model. This way, we can *"connect"* the logged data with the online decision environment. Figure 2 summarizes our models of logged data and the online feedbacks.

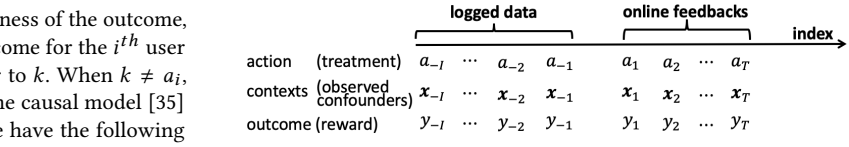

Figure 2: Summary of logged data and online feedbacks

2.3 Online Decision Problems

The decision maker selects an action in each time slot. We consider two kinds of online decision problems depending on whether users with different contexts can be treated differently or not.

• **Context-independent decision problem.** Consider the setting where a company makes a context-independent decision for all users. In causal inference, this setting corresponds to the estimation of "average treatment effect" [35]. In online learning, this setting corresponds to the "stochastic multi-armed bandit" problem [25]. In time slot t, the decision maker can use the logged data $\mathcal{L}$ and the feedback history $\mathcal{F}_t \triangleq \{(a_1, \boldsymbol{x}_1, y_1), \cdots, (a_{t-1}, \boldsymbol{x}_{t-1}, y_{t-1})\}$. Let $\mathcal{E}$ denote an *"offline evaluator"* (e.g., an offline causal inference algorithm), which synthesizes feedbacks from the logged data $\mathcal{L}$. Let O denote an online context-independent bandit learning algorithm. We defer the details of $\mathcal{E}$ and O to Section 4. Let $\mathcal{A}_{O+\mathcal{E}}(\cdot,\cdot)$ denote an algorithm that combines O and $\mathcal{E}$ to make online context-independent decisions, i.e., $a_t = \mathcal{A}_{O+\mathcal{E}}(\mathcal{L}, \mathcal{F}_t)$. The decision accuracy is quantified the following pseudo-regret:

$$R(T, \mathcal{A}_{O+\mathcal{E}}) \triangleq \sum_{t=1}^{T} \left(\mathbb{E}[y_t|a^*] - \mathbb{E}[y_t|a_t=\mathcal{A}_{O+\mathcal{E}}(\mathcal{L}, \mathcal{F}_t)]\right), \quad (3)$$

where $a^* \triangleq \arg\max_{a\in[K]} \mathbb{E}[y_t|a_t=a]$ denotes the optimal action.

• **Context-dependent decision problem.** Consider that a company can make different decisions for users coming with different contexts. Let O_c denote an online contextual bandit learning algorithm. Let $\mathcal{A}_{O_c+\mathcal{E}}(\cdot,\cdot,\cdot)$ denote an algorithm, that combines O_c and $\mathcal{E}$ to make online contextual decisions, i.e., $a_t = \mathcal{A}_{O_c+\mathcal{E}}(\mathcal{L}, \mathcal{F}_t, \boldsymbol{x}_t)$. Given $\boldsymbol{x}_t$, the *unknown* optimal action is $a_t^* \triangleq \max_{a\in[K]} \mathbb{E}[y_t|a, \boldsymbol{x}_t]$. The decision accuracy is quantified the following pseudo-regret:

$$R(T, \mathcal{A}_{O_c+\mathcal{E}}) \triangleq \sum_{t=1}^{T} \left(\mathbb{E}[y_t|a_t^*, \boldsymbol{x}_t] - \mathbb{E}[y_t|a_t=\mathcal{A}_{O_c+\mathcal{E}}(\mathcal{L}, \mathcal{F}_t, \boldsymbol{x}_t), \boldsymbol{x}_t]\right).$$

This paper aims to develop a generic framework to combine different bandit learning algorithms O, O_c, and offline evaluator $\mathcal{E}$ to make decisions with provable theoretical guarantee on the regret.

In the following sections, we explore the following questions: *(1) How to combine offline evaluator $\mathcal{E}$ with online bandit learning O or O_c? (2) How to prove bounds on the decision maker's regrets? (3) What are the advantages of our methods on real decision problems?*

3 GENERAL ALGORITHMIC FRAMEWORK

We first develop a general algorithmic framework to combine offline evaluators ($\mathcal{E}$) with online bandit learning algorithms ($\mathcal{O}$ and $\mathcal{O}_c$). Then, we present regret bounds for the proposed framework.

3.1 Algorithmic Framework

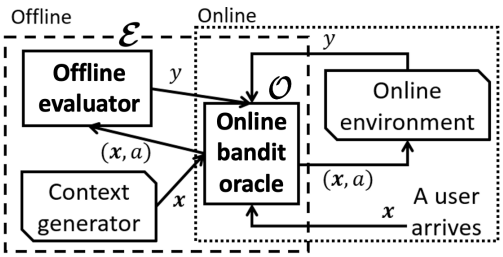

Figure 3: Illustration of algorithmic framework. *Online bandit oracle* **has two functions: function** *play*$(\boldsymbol{x})$ **returns an action** a **given a context** $\boldsymbol{x}$**; function** *update*$(\boldsymbol{x}, a, y)$ **updates the oracle with the feedback** y **w.r.t. action** a**, under the context** $\boldsymbol{x}$**.** *Offline evaluator* **has one function** *get_outcome*$(\boldsymbol{x}, a)$ **that searches the logged data and returns a "synthetic outcome"** y **given the pair** $(\boldsymbol{x}, a)$**, where the return value** y **= NULL if the offline evaluator is not able to synthesize a feedback.**

The key idea of our framework is to select "appropriate" data from the log to improve online learning. This is achieved via the idea of "virtual play". Figure 3 illustrates the workflow of our framework. The *"BanditOracle"* $\mathcal{O}$ denotes an online learning algorithm. The *"OfflineEvaluator"* $\mathcal{E}$ denotes an algorithm that synthesizes feedbacks from the log. Algorithm 1 shows how to coordinate these two components to make sequential decisions in T rounds. Each round has an offline phase and an online phase. In the offline phase (Line 4-11), we first generate a context according to the CDF $F_{\boldsymbol{X}}(\cdot)$[1]. Then, we get an action from the *BanditOracle*. The *OfflineEvaluator* returns a synthetic feedback to update the *BanditOracle*. We repeat such procedure until the *OfflineEvaluator* cannot synthesize a feedback. When this happens, we turn to the online phase (Line 12-14), where the same *BanditOracle* chooses the action, and updates itself with online feedbacks.

Unifying causal inference and online bandit learning. Both online bandit algorithms and causal inference algorithms are special cases of our framework. First, if there are no logged data, then the offline evaluator cannot synthesize feedbacks and always returns "NULL". We use $\mathcal{E}_\emptyset$ to denote such offline evaluator that always returns "NULL". Then, our framework always calls the online bandit oracle, and it reduces to an online bandit algorithm. Second, we consider a specific A/B test online learning oracle described in BanditOracle 0, and we let T=1. Then, after the offline phase, the estimated outcome $\bar{y}_a$ can be used to estimate the causal effect. In this case, our framework reduces to a causal inference algorithm.

[1]In practice, the CDF is usually *unknown* but can be estimated asymptotically ([23]). We discuss the impact of using empirical context distribution in our supplement [44].

Algorithm 1: General Algorithmic Framework

```
1  Initialize the OfflineEvaluator with logged data ℒ
2  Initialize the BanditOracle
3  for t = 1 to T do
4      while True do
5          x ← context_generator() //from CDF F_X(·)
6          a ← BanditOracle.play(x) //virtual play
7          y ← OfflineEvaluator.get_outcome(x, a)
8          if y ≠ NULL then
9              BanditOracle.update(x, a, y)
10         else //offline evaluator cannot synthesize a feedback
11             break
12     a_t ← BanditOracle.play(x_t) //online play
13     y_t ← the outcome from the online environment
14     BanditOracle.update(x_t, a_t, y_t)
```

BanditOracle 0: A/B Testing

Member variables: the average outcome $\bar{y}_a$ of each action $a \in [K]$, and the number of times n_a that action a was played.
Function play$(\boldsymbol{x})$**:**
  return a with probability $1/K$ for each $a \in [K]$
Function update$(\boldsymbol{x}, a, y)$**:**
  $\bar{y}_a \leftarrow (n_a \bar{y}_a \quad y)/(n_a \quad 1)$, $n_a \leftarrow n_a \quad 1$

3.2 Regret Analysis Framework

We decompose the regret of Algorithm 1 as *"online regret = total regret - regret of virtual plays"*. The intuition is that among all the decisions of the online bandit oracle, there are "virtual plays" whose feedbacks are simulated from *the logged data*, and "online plays" whose feedbacks are from *the real online environment*. The online bandit oracle cannot distinguish the "virtual plays" from "online plays". Thus we can apply the theories of the online bandit oracles (e.g. [4][30][2]) to bound the *total regret*. By subtracting the *regret of virtual plays*, we get the bound for *online regret*.

Theorem 1 (General upper bound). *Suppose there exist* $g(T)$ *and* $g_c(T)$*, such that* $R(T, \mathcal{A}_{\mathcal{O}+\mathcal{E}_\emptyset}) \le g(T)$*, and* $R(T, \mathcal{A}_{\mathcal{O}_c+\mathcal{E}_\emptyset}) \le g_c(T), \forall T$*. Denote the returns of the offline evaluator till time* T *as* $\{\tilde{y}_j\}_{j=1}^N$ *w.r.t. input* $\{(\tilde{\boldsymbol{x}}_j, \tilde{a}_j)\}_{j=1}^N$*. If* $\mathcal{E}$ *satisfies* $\mathbb{E}[\mathcal{E}.get\_outcome(\boldsymbol{x}, a)] = \mathbb{E}[y|a]$*, then*

$$R(T, \mathcal{A}_{\mathcal{O}+\mathcal{E}}) \le g(T+N) - \sum_{j=1}^{N} \left(\max_{a' \in [K]} \mathbb{E}[y|a'] - \mathbb{E}[y|a = \tilde{a}_j] \right). \quad (4)$$

If $\mathcal{E}$ *satisfies* $\mathbb{E}[\mathcal{E}.get\_outcome(\boldsymbol{x}, a)] = \mathbb{E}[y|a, \boldsymbol{x}]$ *contextually, then*

$$R(T, \mathcal{A}_{\mathcal{O}_c+\mathcal{E}}) \le g_c(T+N) - \sum_{j=1}^{N} \left(\max_{a' \in [K]} \mathbb{E}[y|a', \tilde{\boldsymbol{x}}_j] - \mathbb{E}[y|a = \tilde{a}_j, \tilde{\boldsymbol{x}}_j] \right).$$

Due to page limit, all proofs are presented in the supplementary materials [44]. In Inequality (4), $g(T+N)$ is the upper bound of

total regret, and $\sum_{j=1}^{N}\left(\max_{a'\in[K]}\mathbb{E}[y|a']-\mathbb{E}[y|a=\tilde{a}_j]\right)$ is the *regret of virtual plays*. The condition $\mathbb{E}[\mathcal{E}.get\_outcome(\boldsymbol{x},a)]=\mathbb{E}[y|a]$ (or $\mathbb{E}[\mathcal{E}.get\_outcome(\boldsymbol{x},a)]=\mathbb{E}[y|a,\boldsymbol{x}]$) implies that the offline evaluator $\mathcal{E}$ returns unbiased context-independent (or contextual) outcomes. Using similar regret decomposition, we also derive a regret lower bound with logged data in our supplementary material [44].

4 CASE STUDY I: CONTEXT-INDEPENDENT DECISION

To demonstrate the versatility of our algorithmic framework for context-independent decisions, we start with a case of using UCB and exact matching in our framework. Then we extend the offline evaluator from exact matching to propensity score matching, and weighting method like inverse propensity score weighting. Finally, we study the case when Assumptions 1 and 2 do not hold.

4.1 Warm-up: UCB + Exact Matching

To illustrate Algorithm 1, let us start with an instance that uses UCB [4] (BanditOracle 1) as the online bandit oracle and the "exact matching" causal inference algorithm [38] (OfflineEvaluator 1) as the offline evaluator. We denote this instance of Algorithm 1 as $\mathcal{A}_{\text{UCB+EM}}$. In each round, BanditOracle 1selects an action with the maximum upper confidence bound defined as $\bar{y}_a+\beta\sqrt{2\ln(n)/n_a}$, where $\bar{y}_a$ is the average outcome, β is a constant, and n_a is the number of times that an action a was played. OfflineEvaluator 1searches for a data item in log $\mathcal{L}$ with the exact same context $\boldsymbol{x}$ and action a, and returns the outcome y of that data item. If it cannot find a matched data item for an action a, it stops the matching process for the action a. The stop of matching is to ensure that the synthetic feedbacks simulate the online feedbacks correctly.

BanditOracle 1: UCB [4]

Variables: the average outcome $\bar{y}_a$ of each action $a\in[K]$, number of times n_a action a was played.
Function play$(\boldsymbol{x})$**:**
  return $\arg\max_{a\in[K]}\ \bar{y}_a\ \ \beta\sqrt{\frac{2\ln(\sum_{a\in[K]}n_a)}{n_a}}$
Function update$(\boldsymbol{x},a,y)$**:**
  $\bar{y}_a\leftarrow(n_a\bar{y}_a\ \ y)/(n_a\ \ 1)$, $n_a\leftarrow n_a\ \ 1$

OfflineEvaluator 1: Exact Matching (EM) [38]

Member variables: $S_a\in\{False,True\}$ indicates whether we stop matching for action a, initially $S_a\leftarrow False, \forall a\in[K]$.
Function get_outcome$(\boldsymbol{x},a)$**:**
  if $S_a=False$ **then**
    $\mathcal{I}(\boldsymbol{x},a)\leftarrow\{i\mid\boldsymbol{x}_i=\boldsymbol{x},a_i=a\}$
    if $\mathcal{I}(\boldsymbol{x},a)\neq\emptyset$ **then**
      $i\leftarrow$ a random sample from $\mathcal{I}(\boldsymbol{x},a)$
      $\mathcal{L}\leftarrow\mathcal{L}\backslash\{(a_i,\boldsymbol{x}_i,y_i)\}$
      return y_i
  $S_a\leftarrow True$ `//If we can't find a sample for the action a, i.e. I(x,a)=∅, stop matching for a`
  return NULL

Applying Theorem 1, we present the regret upper bound of $\mathcal{A}_{\text{UCB+EM}}$ in the following theorem.

Theorem 2 (UCB+Exact matching). *Suppose there are $C\in\mathbb{N}_+$ possible categories of users' features denoted by $\boldsymbol{x}^1,\ldots,\boldsymbol{x}^C$. Denote $\mathbb{P}[\boldsymbol{x}^c]$ as the probability for an online user to have context $\boldsymbol{x}^c$. Recall $a^*=\arg\max_{\tilde{a}\in[K]}\mathbb{E}[y|\tilde{a}]$ and denote $\Delta_a\triangleq\mathbb{E}[y|a^*]-\mathbb{E}[y|a]$. Let $N(\boldsymbol{x}^c,a)\triangleq\sum_{i\in[-I]}\mathbb{1}_{\{\boldsymbol{x}_i=\boldsymbol{x}^c,a_i=a\}}$ be the number of samples with context $\boldsymbol{x}^c$ and action a. Suppose the reward $y\in[0,1]$. Then,*

$$R(T,\mathcal{A}_{UCB+EM})\leq\sum_{a\neq a^*}\Delta_a\left(1+\frac{\pi^2}{3}\right.$$
$$\left.+\sum_{c\in[C]}\max\left\{0,8\frac{\ln(T+A)}{\Delta_a^2}\mathbb{P}[\boldsymbol{x}^c]-\min_{\tilde{c}\in[C]}\frac{N(\boldsymbol{x}^{\tilde{c}},a)\mathbb{P}[\boldsymbol{x}^c]}{\mathbb{P}[\boldsymbol{x}^{\tilde{c}}]}\right\}\right),$$

where A is derived as:

$$A=N-\sum_{a\neq a^*}\sum_{c\in[C]}\max\left\{0,N(\boldsymbol{x}^c,a)-(8\frac{\ln(T+N)}{\Delta_a^2}+1+\frac{\pi^2}{3})\mathbb{P}[\boldsymbol{x}^c]\right\}.$$

Theorem 2 states how logged data reduces the regret. When there is no logged data, i.e., $N(\boldsymbol{x}_c,a)=0$ for $\forall\boldsymbol{x}_c,a$, the regret bound $O(\log(T))$ is the same as that of UCB. If the number of logged data $N(\boldsymbol{x}^c,a)$ is greater than a threshold $\mathbb{P}[\boldsymbol{x}^c]8\ln(T+A)/\Delta_a^2$ for each context $\boldsymbol{x}^c$ and action a, then the regret is smaller than a constant $\left(1+\frac{\pi^2}{3}\right)\sum_{a\neq a^*}\Delta_a$. Note that when we give all the data items the

OfflineEvaluator 2: Propensity Score Matching (PSM)[38]

1 **Variables**: initially $S_a \leftarrow False, \forall a \in [K]$. The pivot set $\mathcal{Q} \subset [0,1]$ with a finite number of elements.
2 **Function get_outcome**$(\boldsymbol{x}, a)$:
3 **if** $S_a = False$ **then**
4 $\boldsymbol{p} \leftarrow (\mathbb{P}[A=1|\boldsymbol{x}], \cdots, \mathbb{P}[A=K-1|\boldsymbol{x}])$ //here, $\boldsymbol{p} \in [0,1]^{K-1} = (p(1), \cdots, p(K-1))$ is a vector
5 $\mathcal{I}(\boldsymbol{p}, a) \leftarrow \{i \mid$ **stratify**$(\boldsymbol{p}_i)$=**stratify**$(\boldsymbol{p}), a_i = a\}$
6 **if** $\mathcal{I}(\boldsymbol{p}, a) \neq \emptyset$ **then**
7 $i \leftarrow$ a random sample from $\mathcal{I}(\boldsymbol{p}, a)$
8 $\mathcal{L} \leftarrow \mathcal{L} \backslash \{(\boldsymbol{x}_i, a_i, y_i)\}$ //delete item
9 **return** y_i
10 $S_a \leftarrow True$ //stop matching for a
11 **return** NULL
12 **Function stratify**$(\boldsymbol{p})$: //this is used by $\mathcal{E}_{\text{PSM}}$
13 **return** $\arg\min_{\boldsymbol{q} \in \mathcal{Q}} ||\boldsymbol{p} - \boldsymbol{q}||_2$ //round to the nearest pivot

same dummy context $\boldsymbol{x}_0$, our $\mathcal{A}_{\text{UCB+EM}}$ reduces to the "Historical UCB" (HUCB) algorithm in [36], as HUCB ignores the context and only matches the actions.

One limitation of the exact matching evaluator is that when $\boldsymbol{x}$ is continuous or has a high dimension, it will be difficult to find a sample in log-data with exactly the same context $\boldsymbol{x}$. To address this limitation, we consider the propensity score matching method [38].

4.2 UCB + Propensity Score Matching

We replace the offline evaluator, i.e., exact matching, of $\mathcal{A}_{\text{UCB+EM}}$ with the *propensity score matching* stated in OfflineEvaluator 2. This replacement results in $\mathcal{A}_{\text{PSM+UCB}}$. The propensity score $p_i(a) \in [0,1]$ for action a is the probability of observing the action a given the context $\boldsymbol{x}_i$, i.e. $p_i(a) = \mathbb{P}[A_i = a | \boldsymbol{x}_i]$. For the context-independent case, Assumption 2 implies that one can ignore other contexts given the *propensity scores* ([34]), i.e. $[Y_i(1), \cdots, Y_i(K)] \perp\!\!\!\perp A_i | (p_i(1), \cdots, p_i(K))$. Since $\sum_{a=1}^{K} p(a) = 1$, we use a vector $\boldsymbol{p} \triangleq (p(1), \cdots, p(K-1))$ to represent the propensity scores on all actions. For any incoming context-action pair $(\boldsymbol{x}, a)$, OfflineEvaluator 2first finds a logged sample i with a similar propensity score vector $\boldsymbol{p}_i$ and the same action $a_i = a$, and returns the outcome y_i of that logged sample (Line 5-9). We use the stratification strategy [6] to find samples with similar propensity scores. Note that every time we find a matched sample, we delete it in Line 8. Thus the matching process will terminate as we have finite samples. Since we can get a random element and delete it in $O(1)$ time via a HashMap, the total time complexity of calling $\mathcal{E}_{\text{PSM}}$ is $O(I)$ where I is the number of logged samples.

Applying Theorem 1, we present the regret upper bound of $\mathcal{A}_{\text{UCB+PSM}}$ in the following theorem.

Theorem 3 (UCB+Propensity score matching). *Suppose the propensity scores are in a finite set $\boldsymbol{p}_i \in \mathcal{Q} \triangleq \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_Q\} \subseteq [0,1]^{K-1}$, for $\forall i \in [-I]$. Let $N(\boldsymbol{q}, a)$ be the number of data items whose $\boldsymbol{p}_i = \boldsymbol{q}$ and action $a_i = a$, and $N \triangleq \sum_{c \in [Q], a \in [K]} N(\boldsymbol{q}, a)$. Denote $\mathbb{P}[\boldsymbol{q}_c]$ as the probability for an online user to have propensity score $\boldsymbol{q}_c$. Suppose the reward $y \in [0,1]$. Then,*

$$R(T, \mathcal{A}_{UCB+PSM}) \leq \sum_{a \neq a^*} \Delta_a \left(1 + \frac{\pi^2}{3} + \sum_{c \in [Q]} \max\left\{0, 8\frac{\ln(T+A)}{\Delta_a^2}\mathbb{P}[\boldsymbol{q}_c] - \min_{\tilde{c} \in [Q]} \frac{N(\boldsymbol{q}_{\tilde{c}}, a)\mathbb{P}[\boldsymbol{q}_c]}{\mathbb{P}[\boldsymbol{q}_{\tilde{c}}]}\right\}\right), \quad (5)$$

where A is derived as:

$$A = N - \sum_{a \neq a^*} \sum_{c \in [Q]} \max\left\{0, \min_{\tilde{c} \in [Q]} \frac{N(\boldsymbol{q}_{\tilde{c}}, a)\mathbb{P}[\boldsymbol{q}_c]}{\mathbb{P}[\boldsymbol{q}_{\tilde{c}}]} - \left(\frac{8\ln(T+N)}{\Delta_a^2} + 1 + \frac{\pi^2}{3}\right)\mathbb{P}[\boldsymbol{q}_c]\right\}.$$

Theorem 3 is similar to Theorem 2 where we replace the context vector $\boldsymbol{x}^c$ with the propensity score vector $\boldsymbol{q}_c$. If the number of logged data $N(\boldsymbol{q}_c, a)$ is greater than $\mathbb{P}[\boldsymbol{q}_c] 8\ln(T+A)/\Delta_a^2$ for $\forall c \in [Q]$ and $a \in [K]$, then the regret is smaller than a constant $(1+\pi^2/3)\sum_{a \neq a^*} \Delta_a$. When we only have two actions, the propensity score vector $\boldsymbol{p}$ only has one dimension, and the *propensity score matching* do not have the problem of *exact matching* from the high-dimensional context $\boldsymbol{x}$. But when the number of actions $K > 2$, it is still difficult to find matched propensity score vector $\{p(1), \cdots, p(K-1)\}$. The following weighting algorithm can deal with more than two actions.

4.3 UCB + Inverse Propensity Score Weighting

To further demonstrate the versatility of our framework, we show how to use weighting methods [39][24] in causal inference. As shown in Line 4 in OfflineEvaluator 3, we use the inverse of the propensity score $1/p_i(a_i)$ as the weight. Here, we only need the propensity score for the chosen action a_i. We replace the offline evaluator with the IPS weighting OfflineEvaluator 3to get $\mathcal{A}_{\text{UCB+IPSW}}$.

OfflineEvaluator 3first estimates the outcome $\bar{y}_a$ as the weighted average of logged outcomes. The intuition of IPS weighting is as follows: if an action is applied to users in group A more often than users in other groups, then each sample for group A should have smaller weight so the total weights of each group is proportional to its population. In fact, the IPS weighting estimator is unbiased via *importance sampling*[35]. Then, we calculate the *effective sample size* (a.k.a. ESS) N_a of logged plays on the action a according to [22]. After such initialization, the offline evaluator returns $\bar{y}_a$ w.r.t. action a for $\lfloor N_a \rfloor$ times, and return NULL afterwards.

OfflineEvaluator 3: IPS Weighting (IPSW) [39]

1 **Member variables:** $\bar{y}_a, N_a (a \in [K])$ initialized in **__init__**$(\mathcal{L})$
2 **Function __init__**$(\mathcal{L})$:
3 **for** $a \in [K]$ **do**
4 $\bar{y}_a \leftarrow \frac{\sum_{i \in [-I], a_i = a} y_i / p_i(a_i)}{\sum_{i \in [-I], a_i = a} 1/p_i(a_i)}$, $N_a \leftarrow \frac{(\sum_{i \in [-I], a_i = a} 1/p_i(a_i))^2}{\sum_{i \in [-I], a_i = a} (1/p_i(a_i))^2}$
5 **Function get_outcome**$(\boldsymbol{x}, a)$:
6 **if** $N_a \geq 1$ **then**
7 $N_a \leftarrow N_a - 1$
8 **return** $\bar{y}_a$
9 **return** NULL

Theorem 4 (UCB + IPS weighting). *Suppose the reward $y \in [0,1]$, and the propensity score is bounded $p_i \geq \bar{s} > 0$ $\forall i \in [I]$, then*

$$R(T,\mathcal{A}_{UCB+IPSW}) \leq \sum_{a\neq a^*} \Delta_a \left(1 + \pi^2/3 + \max\left\{0, 8\Delta_a^{-2}\ln(T + \sum_{a=1}^{K}\lceil N_a\rceil) - \lfloor N_a\rfloor\right\}\right),$$

where $N_a = \left(\sum_{i\in[-I]} p_i(a_i)^{-1}\mathbb{1}_{\{a_i=a\}}\right)^2 / \sum_{i\in[-I]}\left(p_i(a_i)^{-1}\mathbb{1}_{\{a_i=a\}}\right)^2$.

Theorem 4 quantifies the impact of the logged data on the regret of the algorithm $\mathcal{A}_{\text{UCB+IPSW}}$. Recall that N_a is the *effective sample size* of feedbacks for action a. When there is no logged data, i.e. $N_a = 0$, the regret bound reduces to the $O(\log T)$ bound of UCB. A larger N_a indicates a lower regret bound. Notice that the number N_a depends on the distribution of logged data items' propensity scores. In particular, when all the propensity scores are a constant $\tilde{p}$, i.e. $p_i(a_i)=\tilde{p}$ for $\forall i$, the effective sample size is the actual number of samples with action a, i.e. $N_a = \sum_{i\in[-I]}\mathbb{1}_{\{a_i=a\}}$. When the propensity scores $\{p_i(a_i)\}_{i\in[-I]}$ have a more skewed distribution, the number N_a will be smaller, leading to a larger regret bound.

Note that our framework is not limited to the above instances. One can replace the online bandit oracle with ϵ-greedy [25], EXP3 [5] or Thompson sampling [2]. One can also replace the offline evaluator with balanced weighting [24] or supervised learning [45]. In Section 6, we will discuss more algorithms in the experiments.

4.4 Relaxation of Assumptions on Logged Data

The above theorems require the logged data to satisfy the stable-unit Assumption 1 and ignorability Assumption 2. To see the impact of removing the Assumption 2, consider Example 1. Let's say the logs do not record *users' preferences to video*. In this case, our *causal inference* strategy will calculate the *empirical average*. Then, it will select the wrong action of placing ad below videos. The following theorem gives the regret upper bound when the assumptions on the logged data do not hold.

Theorem 5 (Removing assumptions on logged data). *Suppose Assumptions 1 and 2 were removed. Suppose the offline evaluator $\mathcal{E}$ returns $\{y_j\}_{j=1}^N$ w.r.t. $\{(\boldsymbol{x}_j,a_j)\}_{j=1}^N$. The bias of the average outcome for action a is denoted as*

$$\delta_a \triangleq \left(\sum_{j=1}^N \mathbb{1}_{\{a_j=a\}} y_j\right)/\left(\sum_{j=1}^N \mathbb{1}_{\{a_j=a\}}\right) - \mathbb{E}[y|a].$$

Suppose the reward y is bounded in $[0,1]$. Denote the number of samples for action a as $N_a \triangleq \sum_{j=1}^N \mathbb{1}_{\{a_j=a\}}$. Then,

$$R(T,\mathcal{A}_{O+\mathcal{E}}) \leq \sum_{a\neq a^*}\Delta_a\left(16\Delta_a^{-2}\ln(N_a+T) - 2N_a(1-\Delta_a^{-1}\max\{0,\delta_a-\delta_{a^*}\}) + (1+\pi^2/3)\right)$$

Theorem 5 states the relationship between the bias of the offline evaluator (i.e. δ_a) and the algorithm's regret. When Assumptions 1 and 2 hold, the bias $\delta_a=0$. In this case, the bound in Theorem 5 is similar to the previous bounds in Theorem 3 except that we raise the constant from 8 to 16. When $\delta_a - \delta_{a^*} > 0$, i.e., the offline evaluator has a greater bias for an inferior action than the bias of the optimal action, the regret upper bound becomes larger compared to the case when the offline evaluator is unbiased. In Theorem 5, we also have a sufficient condition for "the logged data to reduce the regret upper bound", i.e. $1-\max\{0,\delta_a-\delta_{a^*}\}/\Delta_a > 0$, or, $\delta_a - \delta_{a^*} < \Delta_a$ for $\forall a \neq a^*$. The physical meaning is that when the estimated reward of the optimal action is greater than that of other actions, the logged data help to identify the optimal action and reduce the regret.

BanditOracle 2: LinUCB [30]

1 **Member variables:** a matrix $\boldsymbol{V}$ (initially $\boldsymbol{V}$ is a $d \times d$ matrix), a d-dimensional vector $\boldsymbol{b}$ (initially $\boldsymbol{b}=\boldsymbol{0}$ is zero), initial time t=1
2 **Function play**$(\boldsymbol{x})$**:**
3 $\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{V}^{-1}\boldsymbol{b}$
4 **for** $a \in [K]$ **do**
5 $\hat{y}_a \leftarrow \hat{\boldsymbol{\theta}}^T\phi(\boldsymbol{x},\boldsymbol{a}) \quad \beta_t\sqrt{\phi(\boldsymbol{x},a)^T\boldsymbol{V}^{-1}\phi(\boldsymbol{x},a)}$
6 **return** $\arg\max_{a\in[K]}\hat{y}_a$
7 **Function update**$(\boldsymbol{x},a,y)$**:**
8 $\boldsymbol{V} \leftarrow \boldsymbol{V} \quad \phi(\boldsymbol{x},a)\phi(\boldsymbol{x},a)^T, \quad \boldsymbol{b} \leftarrow \boldsymbol{b} \quad y\boldsymbol{x}, \quad t \leftarrow t \quad 1$

5 CASE STUDY II: CONTEXTUAL DECISION

We first consider the case that the mean of the outcome is parametrized by a linear function. Then, we generalize it to non-parametric functions, where we design a forest-based online bandit algorithm and prove its regret upper bound. To the best of our knowledge, it is the first regret upper bound for forest-based online bandit algorithms.

5.1 Linear Regression + LinUCB

We consider that the mean of outcome follows a linear function:

$$y_t = \boldsymbol{\theta}^T\phi(\boldsymbol{x}_t,a_t) + \epsilon_t \qquad \forall t \in [T], \tag{6}$$

where $\phi(\boldsymbol{x},a) \in \mathbb{R}^d$ is an d-dimensional *known* feature vector. The $\boldsymbol{\theta}$ is an d-dimensional *unknown* parameter to be learned, and ϵ_t is a stochastic noise with $\mathbb{E}[\epsilon_t]=0$. We consider the case that Algorithm 1 uses "LinUCB" (outlined in BanditOracle 2) as the online bandit oracle and "linear regression" (outlined in OfflineEvaluator 4) as the offline evaluator. We denote this instance of Algorithm 1 as $\mathcal{A}_{\text{LinUCB+LR}}$. BanditOracle 2uses the LinUCB (Linear Upper Confidence Bound [30]) to make contextual online decisions. It estimates the unknown parameter $\hat{\boldsymbol{\theta}}$ based on the feedbacks. The $\hat{y}_a \triangleq \hat{\boldsymbol{\theta}}^T\phi(\boldsymbol{x},a) + \beta_t\sqrt{\phi(\boldsymbol{x},a)^T\boldsymbol{V}^{-1}\phi(\boldsymbol{x},a)}$ is the upper confidence bound of reward, where $\{\beta_t\}_{t=1}^T$ are parameters. The oracle always plays the action with the largest upper confidence bound.
OfflineEvaluator 4uses linear regression to synthesize feedbacks from the logged data. From the logged data, it estimates the parameter $\hat{\boldsymbol{V}}$ (Line 3), and the parameter $\hat{\boldsymbol{\theta}}$ (Line 4). It returns the estimated outcome $\phi(\boldsymbol{x},a)^T\hat{\boldsymbol{\theta}}$ according to a linear model. It stops returning outcomes when the logged data cannot provide a tighter confidence bound than that of the online bandit oracle (Line 6 - 9).

Suppose for any context $\boldsymbol{x}_t$, the difference of expected rewards between the best and the *"second best"* actions is at least $\Delta_{\min}$. This is the settings of section 5.2 in the paper [1]. In the following theorem, we derive a regret upper bound for $\mathcal{A}_{\text{LinUCB+LR}}$.

Theorem 6 (LinUCB+Linear regression). *Suppose the rewards satisfy the linear model in Equation (6). Suppose offline evaluator returns a sequence $\{y_i\}_{i=1}^N$ w.r.t. $\{(\boldsymbol{x}_i,a_i)\}_{i=1}^N$. Let $\boldsymbol{V}_N \triangleq \sum_{i\in[N]}\boldsymbol{x}_i\boldsymbol{x}_i^T$,*

OfflineEvaluator 4: Linear Regression (LR)

Member variables: $\boldsymbol{V}, \hat{\boldsymbol{V}}$ are $d \times d$ matrices, where $\boldsymbol{V}$ ($\hat{\boldsymbol{V}}$) is for the online (offline) confidence bounds. $\hat{\boldsymbol{\theta}}$ is the estimated parameters. The $\boldsymbol{V}$ is shared with LinUCB oracle.
Function __init__($\mathcal{L}$):
  $\hat{\boldsymbol{V}} \leftarrow \boldsymbol{I}_d \quad \sum_{i\in[-I]} \phi(\boldsymbol{x}_i, a_i) \cdot \phi(\boldsymbol{x}_i, a_i)^T$ // $\boldsymbol{I}_d$ is the $d\times d$ identity matrix
  $\boldsymbol{b} \leftarrow \sum_{i\in[-I]} y_i \cdot \phi(\boldsymbol{x}_i, a_i)$, $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{V}}^{-1}\boldsymbol{b}$
Function get_outcome($\boldsymbol{x}, a$):
  if $||\phi(\boldsymbol{x}, a)||_{\boldsymbol{V}\ \phi(\boldsymbol{x}_i,a_i)\cdot\phi(\boldsymbol{x}_i,a_i)^T} > ||\phi(\boldsymbol{x}, a)||_{\hat{\boldsymbol{V}}}$ **then**
    $\boldsymbol{V} \leftarrow \boldsymbol{V} \quad \phi(\boldsymbol{x}_i, a_i) \cdot \phi(\boldsymbol{x}_i, a_i)^T$
    return $\phi(\boldsymbol{x}, a) \cdot \hat{\boldsymbol{\theta}}$
  return NULL

$L \triangleq \max_{t\le T}\{||\boldsymbol{x}_t||_2\}$. *Moreover, the random noise is* 1*-sub-Gaussian, i.e.* $\mathbb{E}[e^{\alpha\epsilon_t}] \le \exp(\alpha^2/2), \forall\alpha \in \mathbb{R}$. *Then*

$$R(T, \mathcal{A}_{LinUCB+LR}) \le \frac{8d^2(1+2\ln(T))}{\Delta_{\min}} \log\left(1 + \frac{TL^2}{\lambda_{\min}(\boldsymbol{V}_N)}\right) + 1.$$

When the smallest eigenvalue $\lambda_{\min}(\boldsymbol{V}_N)$ *is greater than a threshold* $(1/2+\ln(T))TL^2$*, the regret is bounded by a constant* $16d^2/\Delta_{\min}+1$.

Denote $\kappa = TL^2/\lambda_{\min}(\boldsymbol{V}_N)$ as the condition number. Theorem 6 implies that for a fixed κ, the regret in T time slots is $O(\log(T))$. Moreover, when the logged data contain enough samples, i.e., $\lambda_{\min}(\boldsymbol{V}_N)$ is greater than $(1/2+\ln(T))TL^2$, regret is upper bounded by a constant. Using our analytic framework, we observe a similar thresholding phenomena in [11] which focuses on the linear model.

5.2 Non-parametric Forest-based Online Decision Making

We generalize the linear outcome model (in Equation (6)) to the case that the mean of the outcome y_t is a nonparametric function of $\boldsymbol{x}_t$. We use the non-parametric forest estimator to generalize algorithm $\mathcal{A}_{\text{LR+LinUCB}}$ in two aspects: (1) replace the *LinUCB* with our forest-based online learning algorithm *ϵ-Decreasing Multi-action Forest* (abbr. Fst) outlined in BanditOracle 3; (2) replace *linear regression* with *Matching on Forest* (abbr. MoF) outlined in OfflineEvaluator 5. We denote the new contextual decision algorithm as $\mathcal{A}_{\text{Fst+MoF}}$.

ϵ-decreasing multi-action forest (Fst). A multi-action forest $\mathcal{F}$ is a set of B multi-action decision trees. It extends the regression forest of [40] to consider *multiple actions* in a leaf. Each context $\boldsymbol{x}$ belongs to a leaf $L_b(\boldsymbol{x})$ in a tree $b\in[B]$, and each leaf has multiple actions $a\in[K]$. Given the dataset $\mathcal{D}=\{(\boldsymbol{x}_i, a_i, y_i)\}_{i=1}^D$, tree b estimates the outcome of an action a under a context $\boldsymbol{x}$ as

$$\hat{L}_b(\boldsymbol{x}, a) \triangleq \frac{\sum_{i\in[D]} \mathbb{1}_{\{L_b(\boldsymbol{x}_i)=L_b(\boldsymbol{x})\}} \mathbb{1}_{\{a_i=a\}} y_i}{\sum_{i\in[D]} \mathbb{1}_{\{L_b(\boldsymbol{x}_i)=L_b(\boldsymbol{x})\}} \mathbb{1}_{\{a_i=a\}}}. \tag{7}$$

BanditOracle 3is the *ϵ-decreasing multi-action forest* algorithm. For a context $\boldsymbol{x}$, the algorithm first uses the average of all trees as the estimated outcome (Line 4). In the time slot t, with probability $1-\epsilon_t$, the algorithm chooses the action with the largest estimated outcome. Otherwise, the algorithm randomly selects an action to explore its outcome. The oracle will update the data $\mathcal{D}$ using the feedback (Line 8), and update the leaf functions $\{L_b(\cdot)\}_{b=1}^B$ of the forest $\mathcal{F}$ using the training algorithm in the paper [40] (Line 9).

BanditOracle 3: ϵ-DecreasingMulti-action Forest (Fst)

1 **Variables:** the multi-action forest $\mathcal{F}$ of B trees, data $\mathcal{D}$ with initial value $\emptyset$, t with initial value 1
2 **Function play**($\boldsymbol{x}$):
3 **for** $a \in [K]$ **do**
4 $\hat{y}_a \leftarrow \frac{1}{B}\sum_{b\in[B]} \hat{L}_b(\boldsymbol{x}, a)$
5 $a_t \leftarrow \begin{cases} \arg\max_{a\in[K]} \hat{y}_a & \text{w.p. } 1-\epsilon_t, \\ \text{a random action in } [K] & \text{w.p. } \epsilon_t. \end{cases}$
6 **return** a_t
7 **Function update**($\boldsymbol{x}, a, y$):
8 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}, a, y)\}$ and $t \leftarrow t \quad 1$
9 $\mathcal{F} \leftarrow$train_forest($\mathcal{D}$)//learn tree splits. In practice, one can re-train the forest every T_0 time slots

To analyze the regret of BanditOracle 3, we need the following two definitions, which are adapted from Definition 2b and 4b of [40].

Definition 1 (honest). *A multi-action tree on training samples* $\{(\boldsymbol{x}_1, y_1, a_1), \ldots, (\boldsymbol{x}_s, y_s, a_s)\}$ *is honest if (a) (standard-case) the tree does not use the responses* $y_1, \ldots, y_s$ *in choosing where to replace its splits; or (b) (double sample case) the tree does not use the responses in a subset of data called "*$\mathcal{I}$*-sample" to place splits, where* "double sample" *and "*$\mathcal{I}$-sample" *are defined in Section 2.4 of [40].*

Definition 2 (α-regular). *A multi-action tree grown by recursive partitioning is* α*-regular for some* $\alpha > 0$ *if either: (a) (standard case) (1) each split leaves at least a fraction* α *of training samples on each side of the split, (2) the leaf containing* $\boldsymbol{x}$ *has at least* m *samples from each action* $a \in [K]$ *for some* $m\in\mathbb{N}$*, and (3) the leaf containing* $\boldsymbol{x}$ *has less than* $2m-1$ *samples for some action* $a \in [K]$ *or (b) (double-sample case) for a double-sample tree, (a) holds for the* $\mathcal{I}$ *sample.*

Theorem 7 (Asymptotic regret of Fst). *Suppose that all potential outcome distributions* $(\boldsymbol{x}_i, Y_i(a))$ *for* $\forall a \in [K]$ *satisfy the same regularity assumptions as the pair* $(\boldsymbol{x}_i, Y_i)$ *did in Theorem 3.1 in [40][2]. Suppose the trees in* $\mathcal{F}$ *(Line 9) is honest,* α*-regular with* $\alpha \le 0.2$ *in the sense of Definition 1 and 2, and symmetric random-split (in the sense of Definition 3 and 5 in [40]). Denote* $A \triangleq \frac{\pi'}{d}\frac{\log((1-\alpha)^{-1})}{\log(\alpha^{-1})}$ *where* $\pi' \in [0, 1]$ *is the constant "*π*" in Definition 3 of [40]. Let* $\beta = 1 - \frac{2A}{(2+3A)}$ *and let the exploration rate to be* $\epsilon_t = t^{-1/2(1-\beta)}$*. Then for any small* $\omega > 0$*, the asymptotic regret of Fst (do not use logged data) satisfies*

$$\lim_{T\to+\infty} \frac{R(T, \mathcal{A}_{Fst+\mathcal{E}_\emptyset})}{T^{(1+\beta+\omega)/2}} = 0, \quad \textit{hence} \lim_{T\to+\infty} \frac{R(T, \mathcal{A}_{Fst+\mathcal{E}_\emptyset})}{T} = 0.$$

Theorem 7 states that our online forest-based bandit algorithm *Fst* achieves a sub-linear regret w.r.t. T. Note that our estimator can be biased. We see by appropriate choices of the exploration rate ϵ_t,

[2]The condition is: $\mu(\boldsymbol{x}, a) = \mathbb{E}[Y(a)|X = \boldsymbol{x}]$ and $\mu_2(\boldsymbol{x}, a) = \mathbb{E}[Y(a)^2|X = \boldsymbol{x}]$ are Lipschitz-continuous, and finally that $\text{Var}[Y(a)|X = \boldsymbol{x}] > 0$ and $\mathbb{E}[|Y(a) - \mathbb{E}[Y(a)|X = \boldsymbol{x}]|^{2+\delta}|X = \boldsymbol{x}]$ for some constants $\delta, M > 0$ and for $\delta = 1$, uniformly over all $\boldsymbol{x} \in [0, 1]^d$. Here, we slightly modify the condition to add the case $\delta = 1$.

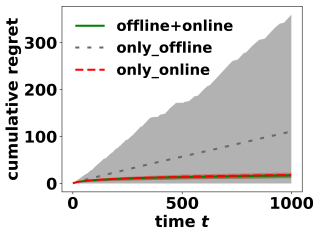

Figure 4: Cumulative regrets of $\mathcal{A}_{\text{UCB+EM}}$ & variants (K=2)

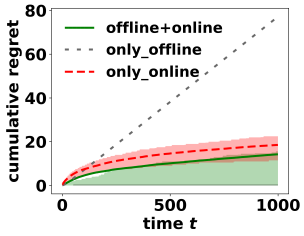

Figure 5: Cumulative regrets of $\mathcal{A}_{\text{UCB+PSM}}$ & variants (K=2)

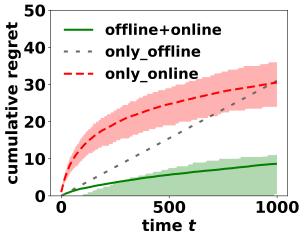

Figure 6: Cumulative regrets of $\mathcal{A}_{\text{UCB+IPSW}}$ and its variants

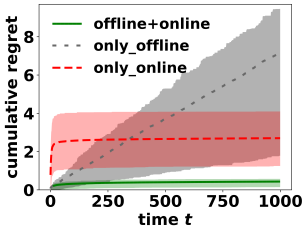

Figure 7: Cumulative regrets of $\mathcal{A}_{\text{LinUCB+LR}}$, linear f

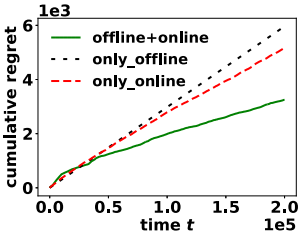

Figure 8: Total regrets of $\mathcal{A}_{\text{UCB+PSM}}$ and its variants [Weixin, context-independent]

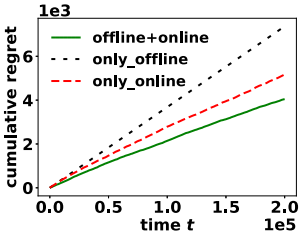

Figure 9: Total regrets of $\mathcal{A}_{\text{UCB+IPSW}}$ and its variants [Weixin, context-independent]

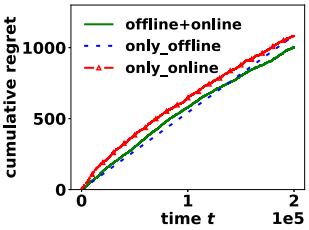

Figure 10: Cumulative regret of $\mathcal{A}_{\text{UCB+IPSW}}$ [Yahoo, context-independent]

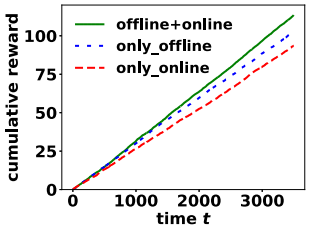

Figure 11: Reward of $\mathcal{A}_{\text{LinUCB+LR}}$ and its variants [Yahoo, contextual]

our algorithm *Fst* balances both the bias-variance tradeoff and the exploration-exploitation tradeoffs. For readers who study causal inference, note that we do not need the "overlap" assumption [40] on the logged data. This is because our exploration probability ϵ_t ensures that each action is played with a non-zero probability.

Matching-on-forest offline evaluator (MoF). OfflineEvaluator 5describes the *Matching-on-Forest* offline evaluator. It finds a (*weighted*) random "nearest neighbor" in the logs for the context-action pair $(\boldsymbol{x}, a)$. For a decision tree $b \in [B]$, the "nearest neighbors" of $(\boldsymbol{x}, a)$ is the data items in the same leaf $L_b(\boldsymbol{x})$ which have the same action a. If a data sample belongs to the nearest neighbors of $(\boldsymbol{x}, a)$ in more trees, then it will be returned by *MoF* with a higher probability.

OfflineEvaluator 5: Matching on Forest (MoF)

Input: a multi-action forest $\mathcal{F}$ with leaf functions $\{L_b(\cdot)\}_{b=1}^{B}$, and the logged data $\mathcal{L}$
Function get_outcome$(\boldsymbol{x}, a)$**:**
  $b \leftarrow$ a uniformly random number in $\{1, 2, \cdots, B\}$
  $\mathcal{I}_{\text{matched}} \leftarrow \{i \mid L_b(\boldsymbol{x}_i)=L_b(\boldsymbol{x}), a_i=a\}$
  if $\mathcal{I} \neq \emptyset$ **then**
    $i \leftarrow$ a random sample from $\mathcal{I}_{\text{matched}}$
    $\mathcal{L} \leftarrow \mathcal{L} \backslash \{(\boldsymbol{x}_i, a_i, y_i)\}$ `//delete item`
    return y_i
  return NULL

6 EXPERIMENTS

We use two real datasets of Weixin and Yahoo, as well as synthetic data to carry out our experiments[3]. First, we show that it is better to use both the logged data and the online feedbacks to make decisions, compared with using just one of the data sources. Second, we show why we need to judiciously use the logged data via our proposed method. Third, we discuss the practicability of our algorithms.

6.1 Datasets and Experiment Settings

Synthetic dataset. Each user's context $\boldsymbol{x}$ is drawn from $[-1, 1]^d$ uniformly at random. Consider propensity scores $\mathbb{P}[\text{action} = a|\boldsymbol{x}] = ps(\boldsymbol{x}, a)$ for all actions $a \in \{0, \cdots, K-1\}$. Unless we vary it explicitly, we set the propensity score $ps(\boldsymbol{x}, a) = \exp(s_a)/(\sum_{a=0}^{K-1} \exp(s_a))$ by default, where $s_a = \exp(-\boldsymbol{x}^T \boldsymbol{\theta}_a(\mathbb{E}[y|a] - \mathbb{E}[y|(a+1) \mod K]))$. We generate the action $a \in \{0, \cdots, K-1\}$ according to the propensity scores. We consider a reward function $y=f(\boldsymbol{x}, a)$ for each $(\boldsymbol{x}, a)$ pair. Unless we vary it explicitly, we set $f(\boldsymbol{x}, a) = \boldsymbol{x}^T \boldsymbol{\theta}_a + b_a$ for some parameter $\boldsymbol{\theta}_a \in \mathbb{R}^d$ and bias $b_a = 0.5 \times a$. For the contextual-independent cases, the expected reward for an action a is $\mathbb{E}[y|a]=\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x}, a)|a]$ by marginalizing over the context $\boldsymbol{x}$. By default, we set the number of arms as $K = 3$. We present experiment results under other settings in our supplementary materials [44].

Weixin A/B testing data and associated logs. Weixin pushes billions of notifications to users everyday. The Weixin's anonymized data contain A/B tests on the notifications strategies and their associated logs. The A/B test has two actions. We use the estimations of A/B tests as the ground truth values [8]. Each dataset has 100,000 rows of logged data and 500,000 rows of online A/B test's data, where each row has 51 contexts, an action and an outcome. Here, the action is the notification strategy. The outcome is the company's

[3]Code and Yahoo's data are in https://github.com/lonyle/causal_bandit.

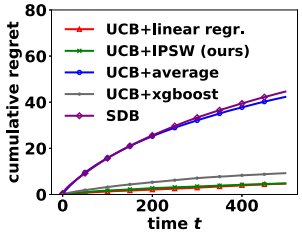

Figure 12: Different algorithms on synthetic data, linear function f

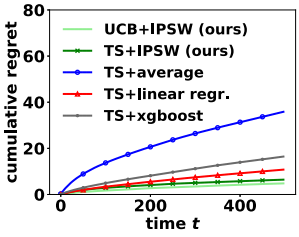

Figure 13: Different algorithms using Thompson Sampling, linear function f

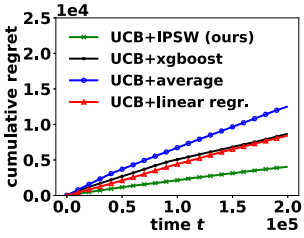

Figure 14: Total regrets of different algorithms [Weixin, context-independent]

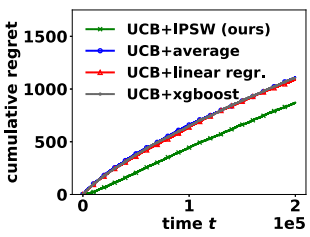

Figure 15: Regrets of different algorithms [Yahoo, context-independent]

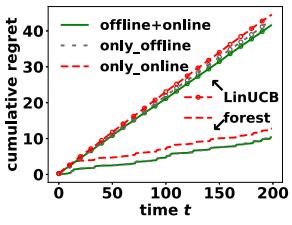

Figure 16: Regrets of $\mathcal{A}_{\text{Fst+MoF}}$, $\mathcal{A}_{\text{LinUCB+LR}}$ and their variants, non-linear $\tilde{f}$

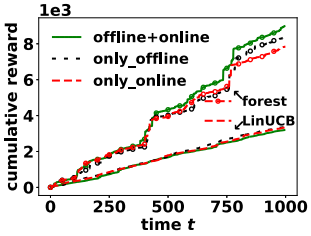

Figure 17: Total rewards of $\mathcal{A}_{\text{Fst+MoF}}$, $\mathcal{A}_{\text{LinUCB+LR}}$ and variants [Weixin, contextual]

metric to evaluate a strategy's performance. We use the protocol in [30] to evaluate an algorithm's performance.

The Weixin's observational datasets do not contain the propensity score. Therefore, we estimate the propensity scores and the inverse propensity score weights. For the offline evaluator PSM, we use generalized linear model to estimate the propensity score. For the inverse propensity score weighting evaluator IPSW, we use the *WeightIt* R package to estimate the propensity score and get the weight. We also use the differences-in-differences[9] method to pre-process the outcomes in Weixin's observational data.

Yahoo's news recommendation data. The publicly available Yahoo's news recommendation dataset [43] contains 100,000 rows of logs, where we split 20% of them as the logged data and 80% of them as the online feedbacks. Each row contains: (1) six user features, (2) candidate news IDs, (3) the selected news ID, (4) whether the user clicks the news. Since the user features in this dataset were learned via a linear model [43], the Yahoo's data favors LinUCB [30] for contextual decisions. We use the evaluation protocol of [30] and run the algorithms for 50 times to take the average.

Table 5: Summary Table of Experiments

Experiment	Question to answer
Exp1 - Exp3	The benefits to use both logged and online data?
Exp4 - Exp6	Why we need to judiciously use the logged data?
Exp7	Forest vs. linear models for contextual decisions?

6.2 Using Both Offline and Online Data

We compare the performance of algorithm $\mathcal{A}_{O+\mathcal{E}}$ (or $\mathcal{A}_{O_c+\mathcal{E}}$) with its two variants that do not combine offline and online data: (1) online bandit algorithm O (or O_c) that only uses online feedbacks; (2) offline causal inference algorithm $\mathcal{E}$ that only uses logged data.

Exp1: Synthetic data. We run each algorithm 500 times to get the average regret. We also plot the 20-80 percentiles as the confidence interval. In Figure 4, 5 and 6, we have 100 logged data points. We observe that our "offline+online" algorithms always have smaller regrets than the "only_online" variants. This is because using logged data to warm-start reduces the cost of online exploration. The regret for the "only_offline" version increases linearly in time, with a large variance. This is because the decisions can be either always right or always wrong depending on the initial decision. In particular, in Figure 5 and 6, the 80-percentile of the regrets for the "only_offline" variants are always zero, although the average regret is high. We set $K = 2$ for $\mathcal{A}_{\text{UCB+EM}}$ and $\mathcal{A}_{\text{UCB+PSM}}$ because they cannot work well for more actions [44]. We also set the context dimensions $d = 2K$. Figure 4 shows that using the offline data does not reduce the regret under the offline evaluator *EM*, because it is difficult to find exactly matched logged data point for contexts in high dimensions. In Figure 5, algorithm $\mathcal{A}_{\text{UCB+PSM}}$ improves the efficiency to use the logged data, and reduces the regret. Algorithm $\mathcal{A}_{\text{UCB+IPSW}}$ can work for $K = 3$ and further reduces the regret, as shown in Figure 6.

We also investigate the contextual decision case. In Figure 7, recall that by default our outcome function $\mathbb{E}[y] = f(\boldsymbol{x}, a) = \boldsymbol{\theta}_a^T \cdot \boldsymbol{x}$ is linear w.r.t. the contexts $\boldsymbol{x}$. We see our "offline+online" algorithm $\mathcal{A}_{\text{LinUCB+LR}}$ has the smallest regret which is nearly zero, because it uses the logged data to reduce the cost of online exploration.

Exp 2: Weixin data. We use the *total regret* as the performance metric. Results on each dataset are in [44]. Figure 8 shows that after 200,000 rounds, our $\mathcal{A}_{\text{UCB+PSM}}$ reduces the total regret by 37.1% (or 45.5%) compared to UCB (or PSM). Our $\mathcal{A}_{\text{UCB+IPSW}}$ reduces the total regret by 45.1% (or 21.7%) compared to UCB (or IPSW). For contextual decisions, Figure 17 shows that our "offline+online" algorithm $\mathcal{A}_{\text{Fst+MoF}}$ has the highest total reward, where we re-train the forest every 50 time slots. All three variants of $\mathcal{A}_{\text{LinUCB+LR}}$ have similar rewards which is samller than that of $\mathcal{A}_{\text{Fst+MoF}}$, because the linear model is probably not correct for Weixin's data.

Exp 3: Yahoo's dataset. Figure 11 shows that our "offline+online" $\mathcal{A}_{\text{LinUCB+LR}}$ improves the rewards by 21.1% (or 10.0%) compared to the "only_online" *LinUCB* (or the "only_offline" *LR* algorithm).

Although Yahoo's data were prepared to evaluate contextual decisions [30], in Figure 10 we restrict the decisions to be context-independent. Our "offline+online" $\mathcal{A}_{\text{UCB+IPSW}}$ has a lower regret than the "only_online" UCB algorithm. Our $\mathcal{A}_{\text{UCB+IPSW}}$ has a lower regret than the "only_offline" IPSW algorithm when T is large.

Lessons learned. Our algorithms that use both data sources achieve the largest rewards or the smallest regret on both real and synthetic datasets, for both context-independent and contextual decisions.

6.3 Proper Usage of the Offline Data

Besides our causal inference approach to use the offline logged data, there are other heuristic methods which can use both data sources. We will show that our proposed method has a superior performance over the following heuristics.

(1) **Historical average in data (historicalUCB [36]).** This method uses the empirical averages of each action in the logged data as the initial values for the online bandit oracle.
(2) **Linear regression.** Instead of simply calculating the average, another way is to use supervised learning algorithm to *"learn"* from offline data. The linear regression method learns a total number of K linear models for each actions where features are the contexts and labels are outcomes.
(3) **Xgboost.** Xgboost [12] is another supervised learning algorithm that often performs well for tabular data. The Xgboost method learns a total number of K models for the K actions.
(4) **Stochastic Delayed Bandits (SDB [31]).** Stochastic delayed bandit is a method proposed for bandit problem with delayed feedback. It can deal with bandit with logged data when we treat the logged data as the delayed feedbacks.
(5) **Thompson sampling with informed prior.** Thompson sampling [2] is a Bayesian online decision algorithm. With logged data, one can use the historical data to give a prior distribution for each action. For example, one can use the average reward for each action to calculate the prior.

All the above heuristics fall within our framework where different heuristics to use the offline data are different offline evaluators.

Exp 4: Our method vs. others on synthetic data. Figure 12 compares our algorithm and the baseline heuristics (1)-(4) on the synthetic data. Recall that by default, the outcome $y = \boldsymbol{x}^T\boldsymbol{\theta}_a + b_a$ is the linear function w.r.t. the context $\boldsymbol{x}$. We observe that our algorithm $\mathcal{A}_{\text{UCB+IPSW}}$ and the linear regression method have the smallest cumulative regret. The linear regression method performs comparatively well because linear regression is unbiased when the reward is a linear function [37]. Xgboost performs worse than our algorithm, because it cannot guarantee to unbiasedly estimate the rewards. Using historical average to initialize UCB (i.e. historicalUCB [36]) or using the stochastic delayed bandit result in the highest regrets, because they ignore the impacts of the confounders.

Figure 13 compares different heuristics to get the informed prior for the Thompson Sampling (TS) algorithm [2]. All these heuristics are instances in our framework where the online learning oracle is Thompson Sampling. Our algorithms $\mathcal{A}_{\text{TS+IPSW}}$ and $\mathcal{A}_{\text{UCB+IPSW}}$ that use the causal inference algorithm IPSW has the lowest regret.

Exp5: Our method vs. others on Weixin data. In Figure 14, we have similar observations for Weixin dataset. Our algorithm combining causal inference algorithm IPSW and UCB has the smallest regret, while the historicalUCB algorithm that uses the historical average to initialize the UCB algorithm has the highest regret. This is because in real A/B tests, the expected rewards for the two decisions A and B are often close to each other. Therefore even a small bias on the estimated reward can lead to the wrong decision.

Exp6: Our method vs. others on Yahoo's data. In Figure 15, we compare different algorithms' regrets on Yahoo's data. Here, we randomly delete some data rows to simulate the selection bias in the logged data. In particular, we delete a logged row with a probability of 0.9 if the average reward for the chosen article is ranked among the top-3 and the reward is 1, or if the average reward for the chosen article is not among the top-3 and the reward is 0. We see that our algorithm $\mathcal{A}_{\text{UCB+IPSW}}$ achieves the lowest regret under this setting. The linear regression does not perform well because the reward in Yahoo's data is not a perfectly linear function of the contexts [41].

Exp7: Linear vs. forest models for contextual decision. In Figure 16, we conduct experiments on synthetic data. We set the reward $y = \tilde{f}(\boldsymbol{x}, a) \triangleq (\sum_{j=1}^{d} \mathbb{1}_{\{\boldsymbol{x} \geq (\boldsymbol{\theta}_a)_j\}})/d + 0.5 \times \mathbb{1}_{\{a=1\}}$ to be a nonlinear function of the context $\boldsymbol{x}$, where $d = 10$. We see our non-parametric forest-based algorithm $\mathcal{A}_{\text{Fst+MoF}}$ can reduce the regrets of by over 75% (from around 40 to less than 10) compared to $\mathcal{A}_{\text{LR+LinUCB}}$.

Figure 17 shows that in Weixin's data, our forest-based algorithms (with circle marks) yields twice as many rewards as the LinUCB-based algorithms, since the reward is probably not a linear function of the contexts in Weixin's data.

The features in Yahoo's dataset were learned using a linear model, and we compare the linear and forest models in the supplement [44].

Lessons learned. One needs to use the offline data properly to reduce the regret in decisions. Our methods that combine causal inference and online bandit learning achieve the smallest regret. For contextual decisions, when the reward is not a linear function of the context, the forest-based model outperforms the linear model.

7 RELATED WORKS

Offline causal inference (e.g. [35][38][33]) focuses on observational logged data and asks "what the outcome would be if we had done another action?". Pearl formulated a Structural Causal Model (SCM) framework to model and infer causal effects[33]. Rubin proposed another alternative,i.e., Potential Outcome (PO) framework[35]. Researchers propose various techniques for causal inference. Matching (e.g. [32][38]) and weighting (e.g. [6][24][21]) are techniques that deal with the imbalance of action's distributions in offline data. Other techniques include "doubly robust" [17] that combines regression and causal inference, and "differences-in-differences" [9]. Recently, several works studied the individualized treatment effects [40][3]. Offline policy evaluation is closely related to offline causal inference. It estimates the performance (or "outcomes") of a policy, which prescribes an action for each context [39][28]. We also use offline policy evaluation to evaluate the performances of contextual bandit algorithms[29]. The offline policy evaluators can be used as the "offline evaluator" in our framework. For example, the Inverse Propensity Score Weighting method in this paper is commonly used in offline policy evaluation [39]. Our paper is orthogonal to the above works in that we focus on combining (or unifying) offline causal inference with online bandit learning algorithms to improve the online decision accuracy. Our work points out if we ignore the online feedbacks, these offline approaches can have a poor decision performance. Offline causal inference algorithms can be seen as special cases of our framework.

Many works studied the stochastic multi-armed bandit problem. Two typical algorithms are UCB [4] and Thompson sampling [16]. LinUCB is a parametric variants of UCB [14] tuning for linear reward functions. For the contextual bandit problem, LinUCB algorithm has a regret of $O(\sqrt{T\log(T)})$ [13][1] and was applied to news article recommendation [30]. The *Thompson sampling causal forest* by [15] and *random-forest bandit* by [18] were non-parametric contextual bandit algorithms, but these works did not provide regret bound. Guan *et al.* proposed a non-parametric online bandit algorithm using *k-Nearest-Neighbor* [20]. Our causal-forest based algorithm improves their bounds in a high-dimensional setting. Lattimore *et al.* used the causal structure of a problem to find online interventions [26]. Our paper is orthogonal to the above works in that we focus on developing a generic framework to combine offline causal inference with these online bandit learning algorithms such that offline logged data can be used to speed up theses bandit algorithms with provable regret bounds. In addition, we propose a novel *ϵ-greedy causal forest* algorithm, and prove regret upper bound for it (to the best of our knowledge, this is the first regret bound for forest based online bandit algorithms).

Several works aimed at using logged data to help online decision making. The historicalUCB algorithm [36] is a special case of our framework, while they ignored users' contexts. Bareinboim *et al.* [7] and Forney *et al.* [19] combined the observational data, experimental data and counterfactual data, to solve the MAB problem with unobserved confounders. They considered a different problem of maximizing the "intent-specific reward", and they did not analyze the regret bound. Ang Li and Judea Pearl [27] use conterfactual logic to integrate experimental and observational data. Zhang *et al.* [45] used adaptive weighting to robustly combine supervised learning and online learning. They focused on correcting the bias of supervised learning via online feedbacks, while we use causal inference methods to synthesize unbiased feedbacks to speed up online bandit algorithms. Our experiments in Section 6.3 show that using historicalUCB [36], SDB [31] or the supervised learning algorithm [45] to initialize the online learning algorithms can result in higher regrets than our method.

8 CONCLUSIONS

This paper studies how to use the logged data to make better online decisions. We unify the offline causal inference and online bandit algorithms into a single framework, and consider both context-independent and contextual decisions. We introduce five novel algorithm instances that incorporate causal inference algorithms including matching, weighting, causal forest, and bandit algorithms including UCB and LinUCB. For these algorithms, we present regret bounds under our framework. In particular, we give the first regret analysis for a forest-based bandit algorithm. Experiments on two real datasets and synthetic data show that our algorithms that can use both logged data and online feedbacks outperform algorithms that only use either of the data sources. We also show the importance to judiciously use the offline data via our methods.

Our framework can alleviate the cold-start problem of online learning, and we show how to use the results of offline causal inference to make online decisions. Our unified framework can be applied to all previous applications of offline causal inference and online bandit learning, such as A/B testing with logged data, recommendation systems [42][30] and online advertising [10].

ACKNOWLEDGMENTS

This work of Hong Xie was supported in part by Chongqing Natural Science Foundation (cstc2020jcyj-msxmX0652) and the Fundamental Research Funds for the Central Universities (2020CDJ-LHZZ-057). John C.S. Lui is supported in part by the GRF 14200420. (Hong Xie is the corresponding author)

REFERENCES

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*. 2312–2320.
[2] Shipra Agrawal and Navin Goyal. 2012. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*. 39–1.
[3] Susan Athey, Julie Tibshirani, Stefan Wager, et al. 2019. Generalized random forests. *The Annals of Statistics* 47, 2 (2019), 1148–1178.
[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2-3 (2002), 235–256.
[5] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multiarmed bandit problem. *SIAM journal on computing* 32, 1 (2002), 48–77.
[6] Peter C Austin. 2011. An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate behavioral research* 46, 3 (2011), 399–424.
[7] Elias Bareinboim, Andrew Forney, and Judea Pearl. 2015. Bandits with unobserved confounders: A causal approach. In *Advances in Neural Information Processing Systems*. 1342–1350.
[8] Kjell Benson and Arthur J Hartz. 2000. A comparison of observational studies and randomized, controlled trials. *New England Journal of Medicine* 342, 25 (2000), 1878–1886.
[9] Marianne Bertrand, Esther Duflo, and Sendhil Mullainathan. 2004. How much should we trust differences-in-differences estimates? *The Quarterly journal of economics* 119, 1 (2004), 249–275.
[10] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research* 14, 1 (2013), 3207–3260.
[11] Jinzhi Bu, David Simchi-Levi, and Yunzong Xu. 2019. Online pricing with offline data: Phase transition and inverse square law. *arXiv preprint arXiv:1910.08693* (2019).
[12] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.
[13] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. 2011. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 208–214.
[14] Varsha Dani, Thomas P Hayes, and Sham M Kakade. 2008. Stochastic linear optimization under bandit feedback. In *COLT*.
[15] Maria Dimakopoulou, Susan Athey, and Guido Imbens. 2017. Estimation considerations in contextual bandits. *arXiv preprint arXiv:1711.07077* (2017).
[16] Shi Dong and Benjamin Van Roy. 2018. An information-theoretic analysis for Thompson sampling with many actions. In *Advances in Neural Information Processing Systems*. 4157–4165.
[17] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601* (2011).
[18] Raphaël Féraud, Robin Allesiardo, Tanguy Urvoy, and Fabrice Clérot. 2016. Random forest for the contextual bandit problem. In *Artificial Intelligence and Statistics*.
[19] Andrew Forney, Judea Pearl, and Elias Bareinboim. 2017. Counterfactual data-fusion for online reinforcement learners. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1156–1164.
[20] Melody Y Guan and Heinrich Jiang. 2018. Nonparametric stochastic contextual bandits. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[21] Lars Peter Hansen. 1982. Large sample properties of generalized method of moments estimators. *Econometrica: Journal of the Econometric Society* (1982), 1029–1054.
[22] Wassily Hoeffding. 1994. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*. Springer, 409–426.
[23] Heinrich Jiang. 2017. Uniform convergence rates for kernel density estimation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1694–1703.

[24] Nathan Kallus. 2018. Balanced policy evaluation and learning. In *Advances in Neural Information Processing Systems*. 8895–8906.
[25] Volodymyr Kuleshov and Doina Precup. 2014. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028* (2014).
[26] Finnian Lattimore, Tor Lattimore, and Mark D Reid. 2016. Causal bandits: Learning good interventions via causal inference. In *Advances in Neural Information Processing Systems*. 1181–1189.
[27] Ang Li and Judea Pearl. 2019. Unit selection based on counterfactual logic. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*.
[28] Lihong Li. 2015. Offline evaluation and optimization for interactive systems. (2015).
[29] Lihong Li, Wei Chu, John Langford, Taesup Moon, and Xuanhui Wang. 2012. An unbiased offline evaluation of contextual bandit algorithms with generalized linear models. In *Proceedings of the Workshop on On-line Trading of Exploration and Exploitation 2*. 19–36.
[30] 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*. ACM, 661–670.
[31] Travis Mandel, Yun-En Liu, Emma Brunskill, and Zoran Popovic. 2015. The Queue Method: Handling Delay, Heuristics, Prior Data, and Evaluation in Bandits.. In *AAAI*. 2849–2856.
[32] Daniel F McCaffrey, Greg Ridgeway, and Andrew R Morral. 2004. Propensity score estimation with boosted regression for evaluating causal effects in observational studies. *Psychological methods* 9, 4 (2004), 403.
[33] Judea Pearl. 2000. *Causality: models, reasoning and inference*. Vol. 29. Springer.
[34] Paul R Rosenbaum and Donald B Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70, 1 (1983), 41–55.
[35] Donald B Rubin. 2005. Causal inference using potential outcomes: Design, modeling, decisions. *J. Amer. Statist. Assoc.* 100, 469 (2005), 322–331.
[36] Pannagadatta Shivaswamy and Thorsten Joachims. 2012. Multi-armed bandit problems with history. In *Artificial Intelligence and Statistics*. 1046–1054.
[37] Brandon Stewart. 2016. Causality with Measured Confounding. https://scholar.princeton.edu/sites/default/files/bstewart/files/lecture10handout.pdf
[38] Elizabeth A Stuart. 2010. Matching methods for causal inference: A review and a look forward. *Statistical science: a review journal of the Institute of Mathematical Statistics* 25, 1 (2010), 1.
[39] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual risk minimization: Learning from logged bandit feedback. In *International Conference on Machine Learning*. 814–823.
[40] Stefan Wager and Susan Athey. 2018. Estimation and inference of heterogeneous treatment effects using random forests. *J. Amer. Statist. Assoc.* 113, 523 (2018), 1228–1242.
[41] Huazheng Wang, Qingyun Wu, and Hongning Wang. 2016. Learning hidden features for contextual bandits. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 1633–1642.
[42] Yixin Wang, Dawen Liang, Laurent Charlin, and David M Blei. 2018. The deconfounded recommender: A causal inference approach to recommendation. *arXiv preprint arXiv:1808.06581* (2018).
[43] Yahoo. 2020. Yahoo! Front Page Today Module User Click Log Dataset, version 1.0, Link: webscope.sandbox.yahoo.com/catalog.php?datatype=r&did=49.
[44] Li Ye, Hong Xie, Yishi Lin, and John C.S. Lui. 2021. Supplementary material, Code and Data for "Unifying Offline Causal Inference and Online Bandit Learning for Data Driven Decision, Link: https://github.com/lonyle/causal_bandit.
[45] Chicheng Zhang, Alekh Agarwal, Hal Daumé Iii, John Langford, and Sahand Negahban. 2019. Warm-starting Contextual Bandits: Robustly Combining Supervised and Bandit Feedback. In *International Conference on Machine Learning*. 7335–7344.