Large-scale Open Dataset, Pipeline, and Benchmark for Off-Policy Evaluation

Yuta Saito[1,4]
saito.y.bj@m.titech.ac.jp

Shunsuke Aihara[2]
shunsuke.aihara@zozo.com

Megumi Matsutani[2]
megumi.matsutani@zozo.com

Yusuke Narita[3,4]
yusuke.narita@yale.edu

[1]Tokyo Institute of Technology, [2]ZOZO Technologies, Inc.,
[3]Yale University, [4]Hanjuku-kaso Co., Ltd.

Abstract

We build and publicize the *Open Bandit Dataset* to facilitate scalable and reproducible research on bandit algorithms. It is especially suitable for *off-policy evaluation* (OPE), which attempts to estimate the performance of hypothetical policies using data generated by a different policy. We construct the dataset based on experiments and implementations on a large-scale fashion e-commerce platform, ZOZOTOWN. The data contain the ground-truth about the performance of several bandit policies and enable fair comparisons of different OPE estimators. We also build a Python package called the *Open Bandit Pipeline* to streamline implementations of bandit algorithms and OPE estimators. Our open data and pipeline will allow researchers and practitioners to easily evaluate and compare their bandit algorithms and OPE estimators with others in a large, real-world setting. Using our data and pipeline, we provide extensive benchmark experiments of existing OPE estimators. The latest version of the paper can be found at **https://arxiv.org/abs/2008.07146**. Moreover, our pipeline and example data are available at **https://github.com/st-tech/zr-obp**.

1 Introduction

Interactive bandit and reinforcement learning systems (e.g. personalized medicine, ad/recommendation/search platforms) produce log data valuable for evaluating and redesigning the system. For example, the logs of a news recommendation system record which news article was presented and whether the user read it, giving the system designer a chance to make its recommendations more relevant. Exploiting log data is, however, more difficult than conventional supervised machine learning: the result is only observed for the action chosen by the system but not for all the other actions the system could have taken. The logs are also biased in that the logs over-represent the actions favored by the system. A potential solution to this problem is an A/B test that compares the performance of counterfactual systems in an online environment. However, A/B testing counterfactual systems is often difficult, since deploying a new policy is time- and money-consuming, and entails risks of failure. This leads us to the problem of *off-policy evaluation* (OPE), which aims to estimate the performance of a counterfactual (or evaluation) policy using only log data collected by a past (or behavior) policy. Such an evaluation allows us to compare the performance of candidate counterfactual policies without additional online experiments.This alternative approach thus solves the above problems with the A/B test approach. Applications range from contextual bandits [2, 39, 18, 19, 20, 27, 31, 33, 34, 35, 42]

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2020.

and reinforcement learning in the web industry [7, 13, 14, 15, 22, 28, 36, 37, 38, 43] to other social domains such as healthcare [26] and education [24].

Issues with current experimental procedures. While the research community has produced theoretical breakthroughs, the experimental evaluation of OPE remains primitive. Specifically, it lacks a public benchmark dataset for comparing the performance of different methods. Researchers often validate their methods using synthetic simulation environments [15, 39, 22, 41, 43]. A version of the synthetic approach is to modify multi-class classification datasets and treat supervised machine learning methods as bandit policies to evaluate off-policy estimators [5, 7, 40, 42]. An obvious problem with these studies is that there is no guarantee that their simulation environment is similar to real-world settings. To solve this issue, [8, 10, 27, 28] use proprietary real-world datasets. Since these datasets are not public, however, it remains challenging to reproduce the results, and compare their methods with new ideas in a fair manner. This is in contrast to other domains of machine learning, where large-scale open datasets, such as the ImageNet dataset [4], have been pivotal in driving objective progress [6, 12, 9, 11, 23].

Contributions. Our goal is to implement and evaluate OPE of bandit algorithms in realistic and reproducible ways. We release the *Open Bandit Dataset*, a logged bandit feedback collected on a large-scale fashion e-commerce (e-commerce) platform, ZOZOTOWN.[1] ZOZOTOWN is the largest fashion e-commerce platform in Japan with over 3 billion USD annual Gross Merchandise Value. When the platform produced the data, it used Bernoulli Thompson Sampling (Bernoulli TS) and uniform random (Random) policies to recommend fashion items to users. The dataset includes an A/B test of these policies and collected over 26 million records of users' clicks and the ground-truth about the performance of Bernoulli TS and Random. To streamline and standardize analysis of the Open Bandit Dataset, we also provide the *Open Bandit Pipeline*, a Python software including a series of implementations of dataset preprocessing, bandit policies, and OPE estimators.

In addition to building the data and pipeline, we also perform extensive benchmark experiments on existing estimators. Specifically, we can do this by using the log data of one of the behavior policies to estimate the policy value of the other policy with each estimator. We then assess the accuracy of the estimator by comparing its estimation with the ground truth obtained from the data. This is the first experimental study comparing a variety of OPE estimators in realistic and reproducible manner.

We summarize the key findings in our benchmark experiments below:

- The estimation performance of all OPE estimators drop significantly when they are applied to estimate the future (or out-sample) performance of a new policy.
- The estimation performances of OPE estimators heavily depend on experimental settings and hyperparameters.

These empirical findings highlight the following future research directions: (i) improving out-of-distribution estimation performance and (ii) developing methods to identify appropriate OPE estimators for various settings.

Our data, pipeline, and benchmark experiments are open-sourced to advance future OPE research. Our implementations help practitioners use logged bandit data to compare different estimators and find an appropriate one to improve their bandit systems.

2 Off-Policy Evaluation

2.1 Setup

We consider a general contextual bandit setting. Let $r \in [0, R_{\max}]$ denote a reward or outcome variable (e.g., whether a fashion item as an action results in a click). We let $x \in \mathcal{X}$ be a context vector (e.g., the user's demographic profile) that the decision maker observes when picking an action. Rewards and contexts are sampled from the unknown probability distributions $p(r \mid x, a)$ and $p(x)$, respectively. Let $\mathcal{A}$ be a finite set of actions. We call a function $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ a *policy*. It maps each context $x \in \mathcal{X}$ into a distribution over actions, where $\pi(a \mid x)$ is the probability of taking an action a given x. We describe some examples of such decision making policies in Appendix A.

[1] https://corp.zozo.com/en/service/

Let $\mathcal{D} := \{(x_t, a_t, r_t)\}_{t=1}^T$ be historical logged bandit feedback with T rounds of observations. a_t is a discrete variable indicating which action in $\mathcal{A}$ is chosen in round t. r_t and x_t denote the reward and the context observed in round t, respectively. We assume that a logged bandit feedback is generated by a *behavior policy* π_b as follows:

$$\{(x_t, a_t, r_t)\}_{t=1}^T \sim \prod_{t=1}^T p(x_t)\pi_b(a_t \mid x_t)p(r_t \mid x_t, a_t),$$

where each context-action-reward triplets are sampled independently from the product distribution. Note that we assume a_t is independent of r_t conditional on x_t.

We let $\pi(x, a, r) := p(x)\pi(a \mid x)p(r \mid x, a)$ be the product distribution given by a policy π. For a function $f(x, a, r)$, we use $\mathbb{E}_{\mathcal{D}}[f] := |\mathcal{D}|^{-1} \sum_{(x_t, a_t, r_t) \in \mathcal{D}} f(x_t, a_t, r_t)$ to denote its empirical expectation over T observations in $\mathcal{D}$. Then, for a function $g(x, a)$, we let $g(x, \pi) := \mathbb{E}_{a \sim \pi(a|x)}[g(x, a) \mid x]$. We also use $q(x, a) := \mathbb{E}_{r \sim p(r|x,a)}[r \mid x, a]$ to denote the mean reward function.

2.2 Estimation Target

We are interested in using the historical logged bandit data to estimate the following *policy value* of any given *evaluation policy* π_e which might be different from π_b:

$$V(\pi_e) := \mathbb{E}_{(x,a,r) \sim \pi_e(x,a,r)}[r].$$

Estimating $V(\pi_e)$ before implementing π_e in an online environment is valuable, because π_e may perform poorly and damage user satisfaction. Additionally, it is possible to select an evaluation policy that maximizes the policy value by comparing their estimated performances without having additional implementation costs.

2.3 Basic Estimators

Here, we summarize several standard OPE methods. We describe other advanced methods in Appendix B.

Direct Method (DM). A widely-used method, DM [1], first learns a supervised machine learning model, such as random forest, ridge regression, and gradient boosting, to estimate the mean reward function. DM then plugs it in to estimate the policy value as

$$\hat{V}_{\mathrm{DM}}(\pi_e; \mathcal{D}, \hat{q}) := \mathbb{E}_{\mathcal{D}}[\hat{q}(x_t, \pi_e)],$$

where $\hat{q}(x, a)$ is the estimated mean reward function. If $\hat{q}(x, a)$ is a good approximation to the mean reward function, this estimator accurately estimates the policy value of the evaluation policy. If $\hat{q}(x, a)$ fails to approximate the mean reward function well, however, the final estimator is no longer consistent. The model misspecification issue is problematic because the extent of misspecification cannot be easily quantified from data [7].

Inverse Probability Weighting (IPW). To alleviate the issue with DM, researchers often use another estimator called IPW [29, 31]. IPW re-weighs the rewards by the ratio of the evaluation policy and behavior policy (importance weight) as

$$\hat{V}_{\mathrm{IPW}}(\pi_e; \mathcal{D}) := \mathbb{E}_{\mathcal{D}}[w(x_t, a_t)r_t],$$

where $w(x, a) := \pi_e(a \mid x)/\pi_b(a \mid x)$ is the importance weight given x and a. When the behavior policy is known, IPW is unbiased and consistent for the policy value. However, it can have a large variance, especially when the evaluation policy significantly deviates from the behavior policy.

Doubly Robust (DR). DR [5] combines DM and IPW as

$$\hat{V}_{\mathrm{DR}}(\pi_e; \mathcal{D}, \hat{q}) := \mathbb{E}_{\mathcal{D}}[\hat{q}(x_t, \pi_e) + w(x_t, a_t)(r_t - \hat{q}(x_t, a_t))].$$

DR mimics IPW to use a weighted version of rewards, but it also uses the estimated mean reward function as a control variate to decrease the variance. It preserves the consistency of IPW if either the importance weight or the reward estimator is consistent (a property called *double robustness*). Moreover, DR is *semiparametric efficient* [27] when the reward estimator is correctly specified. On the other hand, when it is misspecified, this estimator can have larger asymptotic mean-squared-error than IPW [15] and perform poorly in practice [16].

Figure 1: Fashion items as actions displayed in ZOZOTOWN. Three fashion items are simultaneously presented to a user in each recommendation.

Table 1: Statistics of the Open Bandit Dataset

Campaigns	Behavior Policies	#Data	#Items	Average Age	CTR (V^{π}) ±95% CI	Relative-CTR
ALL	RANDOM	1,374,327	80	37.93	0.35% ±0.010	1.00
	BERNOULLI TS	12,168,084			0.50% ±0.004	1.43
MEN'S	RANDOM	452,949	34	37.68	0.51% ±0.021	1.48
	BERNOULLI TS	4,077,727			0.67% ±0.008	1.94
WOMEN'S	RANDOM	864,585	46	37.99	0.48% ±0.014	1.39
	BERNOULLI TS	7,765,497			0.64% ±0.056	1.84

Notes: Bernoulli TS stands for Bernoulli Thompson Sampling. **#Data** is the total number of user impressions observed during the 7-day experiment. **#Items** is the total number of items having a non-zero probability of being recommended by each behavior policy. **Average Age** is the average age of users in each campaign. **CTR** is the percentage of a click being observed in log data, and this is the ground-truth performance of behavior policies for each campaign. The 95% confidence interval (CI) of CTR is calculated based on a normal approximation of Bernoulli sampling. **Relative-CTR** is the CTR relative to that of the Random policy for the "All" campaign.

3 Open-Source Dataset and Pipeline

Motivated by the paucity of real-world data and implementation enabling the **evaluation of OPE**, we release the following dataset and pipeline package for research uses.

Open Bandit Dataset. Our open-source data is logged bandit feedback data provided by ZOZO, Inc.[2], the largest Japanese fashion e-commerce company with a market capitalization of over 5 billion USD (as of May 2020). The company recently started using context-free multi-armed bandit algorithms to recommend fashion items to users in their large-scale fashion e-commerce platform called ZOZOTOWN. We present examples of displayed fashion items in Figure 1. We collected the data in a 7-day experiment in late November 2019 on three "campaigns," corresponding to "all", "men's", and "women's" items, respectively. Each campaign randomly uses either the Random policy or the Bernoulli Thompson Sampling (Bernoulli TS) policy for each user impression.[3] These policies select three of the candidate fashion items to each user. Let $\mathcal{I}$ be a set of (fashion) *items* and $\mathcal{K}$ be a set of *positions*. Figure 1 shows that $|\mathcal{K}| = 3$ for our data. We assume that the reward (click indicator) depends only on the item and its position, which is a general assumption on the click generative model in the web industry [21]. Under the assumption, the action space is simply the product of the item set and the position set, i.e., $\mathcal{A} = \mathcal{I} \times \mathcal{K}$. Then, we can apply the OPE setup and estimators in Section 2 to our dataset. We describe some statistics of the dataset in Table 1. The data is large and contains many millions of recommendation instances. They also include the true action choice probabilities by behavior policies computed by Monte Carlo simulations based on the policy parameters (e.g.,

[2]https://corp.zozo.com/en/about/profile/

[3]Note that we pre-trained Bernoulli TS for over a month before the data collection process and the policy well converges to a fixed one. Thus, we suppose our data is generated by a fixed policy and apply the standard OPE formulation that assumes static behavior and evaluation policies.

parameters of the beta distribution used by Bernoulli TS) used during the data collection process. The number of actions is also sizable, so this setting is challenging for bandit algorithms and their OPE. We share the full version of our data at **https://research.zozo.com/data.html**.

Open Bandit Pipeline. To facilitate the usage of OPE, we also build a Python package called the *Open Bandit Pipeline*. Our pipeline contains dataset module, policy module, simulator module, and ope module. We describe the basic usage of the pipeline in its documentation page.[4] This pipeline allows researchers to focus on building their OPE estimator and easily compare it with other methods in realistic and reproducible ways.

To our knowledge, our real-world dataset and pipeline are the first to include logged bandit datasets collected by running *multiple* different policies, policy implementations used in production, and their ground-truth policy values. These features enable the "***realistic and reproducible evaluation of OPE***" for the first time. We share a part of our dataset and pipeline implementation at **https://github.com/st-tech/zr-obp**.

Table 2: Comparison of Currently Available Large-scale Bandit Datasets

	Criteo Data [17]	**Yahoo! Data** [19]	**Open Bandit Dataset** (ours)
Domain	Display Advertising	News Recommendation	Fashion E-Commerce
#Data	$\geq$ 103M	$\geq$ 40M	$\geq$ 26M (will increase)
#Behavior Policies	1	1	2 (will increase)
Random A/B Test Data	✗	✓	✓
Behavior Policy Code	✗	✗	✓
Evaluation of Bandit Algorithms	✓	✓	✓
Evaluation of OPE	✗	✗	✓
Pipeline Implementation	✗	✗	✓

Notes: **#Data** is the total number of samples included in the data. **#Behavior Policies** is the number of behavior policies that were used to collect the data. **Random A/B Test Data** is whether the data contain a subset of data generated by the uniform random policy. **Behavior Policy Code** is whether the code (production implementation) of behavior policies is publicized along with the data. **Evaluation of Bandit Algorithms** is whether it is possible to use the data to evaluate a new bandit algorithm. **Evaluation of OPE** is whether it is possible to use the data to evaluate a new OPE estimator. **Pipeline Implementation** is whether a pipeline tool to handle the data is available.

Table 3: Comparison of Currently Available Packages of Bandit Algorithms

	contextualbandits [3]	**RecoGym** [30]	**Open Bandit Pipeline** (ours)
Synthetic Data Generator	✗	✓	✓
Support for Real-World Data	✗	✗	✓
Implementation of Bandit Algorithms	✓	✓	✓
Implementation of Basic Off-Policy Estimators	✓	✗	✓
Implementation of Advanced Off-Policy Estimators	✗	✗	✓
Evaluation of OPE	✗	✗	✓

Notes: **Synthetic Data Generator** is whether it is possible to create synthetic bandit datasets with the package. **Support for Real-World Data** is whether it is possible to handle real-world bandit datasets with the package. **Implementation of Bandit Algorithms** is whether the package includes implementations of online and offline bandit algorithms. **Implementation of Basic Off-Policy Estimators** is whether the package includes implementations of *basic* off-policy estimators such as DM, IPW, and DR described in Section 2.3. **Implementation of Advanced Off-Policy Estimators** is whether the package includes implementations of *advanced* off-policy estimators such as Switch Estimators and More Robust Doubly Robust. **Evaluation of OPE** is whether it is possible to evaluate the accuracy of off-policy estimators with the package.

4 Related Resources

In this section, we summarize existing related resources for bandit algorithms and off-poicy evaluation.

[4] https://zr-obp.readthedocs.io/en/latest/

Related Datasets. Our dataset is closely related to those of [17] and [19]. [17] introduces a large-scale logged bandit feedback data (Criteo data) from a leading company in display advertising, Criteo. The data contain context vectors of user impressions, advertisements (ads) as actions, and click indicators as reward. It also provides the ex ante probability of each ad being selected by the behavior policy. Therefore, this data can be used to compare different *off-policy learning* methods, which aim to learn a new bandit policy using only log data generated by a behavior policy. In contrast, [19] introduces a dataset (Yahoo! data) collected on a news recommendation interface of the the Yahoo! Today Module. The data contain context vectors of user impressions, presented news as actions, and click indicators as reward. It was collected by running the uniform random policy on the news recommendation platform, allowing researchers to evaluate their own bandit algorithms.

We summarize key differences between our data and existing ones in Table 2.

Related Packages. There are several existing Python packages related to our Open Bandit Pipeline. For example, *contextualbandits* package[5] contains implementations of several contextual bandit algorithms [3]. It aims to provide an easy procedure to compare bandit algorithms to reproduce research papers that do not provide easily-available implementations. In addition, *RecoGym*[6] focuses on providing simulation bandit environments imitating the e-commerce recommendation setting [30]. This package also implements an online bandit algorithm based on epsilon greedy and off-policy learning method based on IPW.

We summarize key differences between our pipeline and existing ones in Table 3.

5 Benchmark Experiments

We perform benchmark experiments of off-policy estimators using our Open Bandit Dataset and Pipeline. We first describe an experimental protocol to evaluate OPE estimators and use it to compare a wide variety of existing estimators. We then discuss our initial findings in the experiments and indicate future research directions. We share the code for running the benchmark experiments at **https://github.com/st-tech/zr-obp/tree/master/benchmark/ope**.

5.1 Experimental Protocol

We can empirically evaluate OPE estimators' performances by using two sources of logged bandit feedback collected by two different policies $\pi^{(he)}$ (**h**ypothetical **e**valuation policy) and $\pi^{(hb)}$ (**h**ypothetical **b**ehavior policy). We denote log data generated by $\pi^{(he)}$ and $\pi^{(hb)}$ as $\mathcal{D}^{(he)} := \{(x_t^{(he)}, a_t^{(he)}, r_t^{(he)})\}_{t=1}^T$ and $\mathcal{D}^{(hb)} := \{(x_t^{(hb)}, a_t^{(hb)}, r_t^{(hb)})\}_{t=1}^T$, respectively. By applying the following protocol to several different OPE estimators, we can compare their estimation performances:

1. Define the evaluation and test sets as
   - *in*-sample case: $\mathcal{D}_{\mathrm{ev}} := \mathcal{D}_{1:T}^{(hb)}$, $\mathcal{D}_{\mathrm{te}} := \mathcal{D}_{1:T}^{(he)}$
   - *out*-sample case: $\mathcal{D}_{\mathrm{ev}} := \mathcal{D}_{1:\tilde{t}}^{(hb)}$, $\mathcal{D}_{\mathrm{te}} := \mathcal{D}_{\tilde{t}+1:T}^{(he)}$

   where $\mathcal{D}_{a:b} := \{(x_t, a_t, r_t)\}_{t=a}^b$.
2. Estimate the policy value of $\pi^{(he)}$ using $\mathcal{D}_{\mathrm{ev}}$ by an estimator $\hat{V}$. We can represent an estimated policy value by $\hat{V}$ as $\hat{V}(\pi^{(he)}; \mathcal{D}_{\mathrm{ev}})$.
3. Estimate $V(\pi^{(he)})$ by the *on-policy estimation* and regard it as the ground-truth as[7]

$$V_{\mathrm{on}}(\pi^{(he)}; \mathcal{D}_{\mathrm{te}}) := \mathbb{E}_{\mathcal{D}_{\mathrm{te}}}[r_t^{(he)}].$$

4. Compare the off-policy estimate $\hat{V}(\pi^{(he)}; \mathcal{D}_{\mathrm{ev}})$ with its ground-truth $V_{\mathrm{on}}(\pi^{(he)}; \mathcal{D}_{\mathrm{te}})$. We can evaluate the estimation accuracy of $\hat{V}$ by the following *relative estimation error* (relative-

[5] https://github.com/david-cortes/contextualbandits

[6] https://github.com/criteo-research/reco-gym

[7] Note that Table 1 presents $V_{\mathrm{on}}(\pi^{(he)}; \mathcal{D}_{\mathrm{te}})$ for each pair of behavior policies and campaigns, and the small confidence intervals ensure that the on-policy estimation of the ground-truth is accurate.

EE):

$$\textit{relative-EE}(\hat{V}; \mathcal{D}_{\mathrm{ev}}) := \left| \frac{\hat{V}(\pi^{(he)}; \mathcal{D}_{\mathrm{ev}}) - V_{\mathrm{on}}(\pi^{(he)}; \mathcal{D}_{\mathrm{te}})}{V_{\mathrm{on}}(\pi^{(he)}; \mathcal{D}_{\mathrm{te}})} \right|.$$

5. To estimate standard deviation of relative-EE, repeat the above process several times with different bootstrap samples of the logged bandit data created by sampling data *with replacement* from $\mathcal{D}_{\mathrm{ev}}$.

We call the problem setting **without** the sample splitting by time series as *in*-sample case. In contrast, we call that **with** the sample splitting as *out*-sample case where OPE estimators aim to estimate the policy value of an evaluation policy in the test data. Algorithm 1 in Appendix C describes the detailed experimental protocol to evaluate OPE estimators.

5.2 Estimators Compared

We use our protocol and compare the following OPE estimators: Direct Method (DM) [1], Inverse Probability Weighting (IPW) [29], Self-Normalized Inverse Probability Weighting (SNIPW) [34], Doubly Robust (DR) [5], Self-Normalized Doubly Robust (SNDR), Switch Doubly Robust (Switch-DR) [42], and More Robust Doubly Robust (MRDR) [7]. We describe DM, IPW, and DR in Section 2.3. We define the other estimators in Appendix B. We test different values of hyperparameters for Switch-DR (the details about their hyperparameters are in Appendix B). These above estimators have not yet been compared in a large, realistic setting.

For estimators except for DM, we use the true action choice probabilities by behavior policies contained in the Open Bandit Dataset. For estimators except for IPW and SNIPW, we need to obtain a reward estimator $\hat{q}$. We do so by using Logistic Regression (implemented in *scikit-learn*) and training it using 30% of $\mathcal{D}_{\mathrm{ev}}$. We then use the rest of the data to estimate the policy value of an evaluation policy.

Table 4: Comparing Relative-Estimation Errors of OPE Estimators (**All Campaign**)

	Random → Bernoulli TS		**Bernoulli TS → Random**	
Estimators Compared	*in*-sample	*out*-sample	*in*-sample	*out*-sample
DM	**0.23433** ±0.02131	**0.25730** ±0.02191	**0.34522** ±0.01020	**0.29422** ±0.01199
IPW	**0.05146** ±0.03418	0.09169 ±0.04086	**0.02341** ±0.02146	0.08255 ±0.03798
SNIPW	**0.05141** ±0.03374	**0.08899** ±0.04106	0.05233 ±0.02614	0.13374 ±0.04416
DR	0.05269 ±0.03460	0.09064 ±0.04105	0.06446 ±0.03001	0.14907 ±0.05097
SNDR	0.05269 ±0.03398	**0.09013** ±0.04122	0.04938 ±0.02645	0.12306 ±0.04481
Switch-DR ($\tau = 5$)	0.15350 ±0.02274	0.16918 ±0.02231	0.26811 ±0.00780	0.21945 ±0.00944
Switch-DR ($\tau = 10$)	0.09932 ±0.02459	0.12051 ±0.02203	0.21596 ±0.00907	0.16532 ±0.01127
Switch-DR ($\tau = 50$)	0.05269 ±0.03460	0.09064 ±0.04105	0.09769 ±0.01515	**0.04019** ±0.01349
Switch-DR ($\tau = 100$)	0.05269 ±0.03460	0.09064 ±0.04105	0.05938 ±0.01597	**0.01310** ±0.00988
Switch-DR ($\tau = 500$)	0.05269 ±0.03460	0.09064 ±0.04105	**0.02123** ±0.01386	0.06564 ±0.02132
Switch-DR ($\tau = 1000$)	0.05269 ±0.03460	0.09064 ±0.04105	0.02840 ±0.01929	0.05347 ±0.03330
MRDR	0.05458 ±0.03386	0.09232 ±0.04169	0.02511 ±0.01735	0.08768 ±0.03821

Notes: The averaged relative-estimation errors and their unbiased standard deviations estimated over 30 different bootstrapped iterations are reported. We describe the method to estimate the standard deviations in Appendix C. $\pi^{(hb)} \rightarrow \pi^{(he)}$ represents the OPE situation where the estimators aim to estimate the policy value of $\pi^{(he)}$ using logged bandit data collected by $\pi^{(hb)}$. The **red** and **green** fonts represent the best and the second best estimators. The **blue** fonts represent the worst estimator for each setting.

5.3 Results and Discussions

The results of the benchmark experiments on "All" campaign are given in Table 4 (We report experimental results on the other campaigns in Appendix C). We describe **Random → Bernoulli TS** to represent the OPE situation where we use Bernoulli TS as a hypothetical evaluation policy and Random as a hypothetical behavior policy. Similary, we use **Bernoulli TS → Random** to represent the situation where we use Random as a hypothetical evaluation policy and Bernoulli TS as a hypothetical behavior policy.

Performance comparisons. First, DM fails to estimate the policy values in all settings. The failure of DM likely comes from the bias of the reward estimator. We observe that the reward function estimations do not improve upon a naive estimation using the mean CTR for every estimation in the binary cross-entropy measure (We present the performance of the reward estimator in Appendix C). The problem with DM leads us to expect that the other estimators may perform better because they do not rely on correct specifications of the reward estimator. We confirm this expectation in Table 4, where the others drastically outperform DM. Among the other estimators, IPW, SNDR, and MRDR reveal stable estimation performances across different settings, and thus we can use these estimators safely. In the **Bernoulli TS** $\rightarrow$ **Random** situation, Switch-DR performs the best with a proper hyperparameter. Its performance, however, largely depends on the choice of hyperparameter, as we discuss later in detail. Note here that the performances of Switch-DR with some large hyperparameters are the same as that of DR. This is a natural observation, as their definitions are the same when the importance weights of all samples are lower than a given hyperparameter.

Out-sample generalization of OPE. Next, we compare the estimation accuracies of each estimator between the *in*-sample and *out*-sample situations. Table 4 shows that estimators' performances drop significantly in almost all situations when they attempt to generalize their OPE results to the out-sample or future data. The result suggests that the current OPE methods may fail to evaluate the performance of a new policy in the future environment, as they implicitly rely on the critical assumption of the same train-test distributions. Moreover, the results demonstrate that our Open Bandit Dataset is a suitable real-world dataset to evaluate the robustness of off-policy evaluation methods to the distributional changes.

Performance of each estimator across different settings. Finally, we compare the estimation accuracies of each estimator under different experimental conditions and with different hyperparameters. We observe in Table 4 that estimation accuracies can change significantly depending on the experimental conditions. In particular, we tested several values for the hyperparameter τ of Switch-DR. We observe that its estimation performance largely depends on the choice of τ. For example, the performance of Switch-DR is significantly better with large values of τ on our data. This observation suggests that practitioners have to choose an appropriate OPE estimator or to tune estimators' hyperparameters carefully for their specific application. It is thus necessary to develop a reliable method to choose and tune OPE estimators in a data-driven manner [41]. Specifically, in many cases, we have to tune estimators' hyperparameters (including the choice of the reward estimator) without the ground-truth policy value of the evaluation policy.

6 Conclusion and Future Work

To enable realistic and reproducible evaluation of off-policy evaluation, we have publicized the Open Bandit Dataset–a benchmark logged bandit dataset collected on a large-scale fashion e-commerce platform. The data comes with the Open Bandit Pipeline, a collection of implementations that makes it easy to evaluate and compare different OPE estimators. We expect them to facilitate the understanding of the empirical properties of the OPE techniques and address experimental inconsistencies in the literature. In addition to building the data and pipeline, we have presented extensive benchmark experiments on OPE estimators. Our experiments highlight that the current OPE methods are not accurate for estimating out-of-distribution performance of a new policy. It is also evident from the results that it is necessary to develop a data-driven method to tune or select an appropriate estimator for each environment.

As future work, we aim to constantly expand and improve the Open Bandit Dataset to include more data and tasks. For example, we will add additional log data generated by contextual policies on the platform (while the current open data contain only log data generated by the context-free policies). Moreover, we assume that the reward of an item at a position does not depend on other simultaneously presented items. This assumption might not hold, as an item's attractiveness can have a significant effect on the expected reward of another item in the same recommendation list [21]. Thus, it is valuable to compare the standard OPE estimators and those for other settings such as the slate recommendation [25, 35]. We plan to allow our pipeline to implement bandit policies and OPE estimators for the slate recommendation setting. You can follow the updates of the whole project at **https://groups.google.com/g/open-bandit-project**.

References

[1] Alina Beygelzimer and John Langford. The offset tree for learning with partial labels. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 129–138, 2009.

[2] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising. *Journal of Machine Learning Research*, 14(1):3207–3260, 2013.

[3] David Cortes. Adapting multi-armed bandits policies to contextual bandits scenarios. *arXiv preprint arXiv:1811.04383*, 2018.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Miroslav Dudík, Dumitru Erhan, John Langford, and Lihong Li. Doubly Robust Policy Evaluation and Optimization. *Statistical Science*, 29:485–511, 2014.

[6] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[7] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. More robust doubly robust off-policy evaluation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1447–1456, 2018.

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

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[10] Alois Gruson, Praveen Chandar, Christophe Charbuillet, James McInerney, Samantha Hansen, Damien Tardieu, and Ben Carterette. Offline evaluation to make decisions about playlist recommendation algorithms. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 420–428, 2019.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[13] Alex Irpan, Kanishka Rao, Konstantinos Bousmalis, Chris Harris, Julian Ibarz, and Sergey Levine. Off-policy evaluation via off-policy classification. In *Advances in Neural Information Processing Systems*, 2019.

[14] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 652–661, 2016.

[15] Nathan Kallus and Masatoshi Uehara. Intrinsically efficient, stable, and bounded off-policy evaluation for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

[16] Joseph DY Kang, Joseph L Schafer, et al. Demystifying double robustness: A comparison of alternative strategies for estimating a population mean from incomplete data. *Statistical science*, 22(4):523–539, 2007.

[17] Damien Lefortier, Adith Swaminathan, Xiaotao Gu, Thorsten Joachims, and Maarten de Rijke. Large-scale validation of counterfactual learning methods: A test-bed. *arXiv preprint arXiv:1612.00367*, 2016.

[18] Lihong Li, Wei Chu, John Langford, Taesup Moon, and Xuanhui Wang. An Unbiased Offline Evaluation of Contextual Bandit Algorithms with Generalized Linear Models. In *Journal of Machine Learning Research: Workshop and Conference Proceedings*, volume 26, pages 19–36, 2012.

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

[20] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased Offline Evaluation of Contextual-bandit-based News Article Recommendation Algorithms. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 297–306, 2011.

[21] Shuai Li, Yasin Abbasi-Yadkori, Branislav Kveton, S Muthukrishnan, Vishwa Vinay, and Zheng Wen. Offline evaluation of ranking policies with click models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1685–1694, 2018.

[22] Yao Liu, Omer Gottesman, Aniruddh Raghu, Matthieu Komorowski, Aldo A Faisal, Finale Doshi-Velez, and Emma Brunskill. Representation balancing mdps for off-policy policy evaluation. In *Advances in Neural Information Processing Systems*, pages 2644–2653, 2018.

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[24] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1077–1084, 2014.

[25] James McInerney, Brian Brost, Praveen Chandar, Rishabh Mehrotra, and Benjamin Carterette. Counterfactual evaluation of slate recommendations with sequential reward interactions. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1779–1788, 2020.

[26] Susan A Murphy, Mark J van der Laan, James M Robins, and Conduct Problems Prevention Research Group. Marginal mean models for dynamic regimes. *Journal of the American Statistical Association*, 96(456):1410–1423, 2001.

[27] Yusuke Narita, Shota Yasui, and Kohei Yata. Efficient counterfactual learning from bandit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4634–4641, 2019.

[28] Yusuke Narita, Shota Yasui, and Kohei Yata. Off-policy bandit and reinforcement learning. *arXiv preprint arXiv:2002.08536*, 2020.

[29] Doina Precup, Richard S. Sutton, and Satinder Singh. Eligibility Traces for Off-Policy Policy Evaluation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 759–766, 2000.

[30] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv preprint arXiv:1808.00720*, 2018.

[31] Alex Strehl, John Langford, Lihong Li, and Sham M Kakade. Learning from Logged Implicit Exploration Data. In *Advances in Neural Information Processing Systems*, pages 2217–2225, 2010.

[32] Yi Su, Maria Dimakopoulou, Akshay Krishnamurthy, and Miroslav Dudík. Doubly robust off-policy evaluation with shrinkage. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

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

[34] Adith Swaminathan and Thorsten Joachims. The Self-normalized Estimator for Counterfactual Learning. In *Advances in Neural Information Processing Systems*, pages 3231–3239, 2015.

[35] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miro Dudik, John Langford, Damien Jose, and Imed Zitouni. Off-policy Evaluation for Slate Recommendation. In *Advances in Neural Information Processing Systems*, pages 3635–3645, 2017.

[36] Philip Thomas and Emma Brunskill. Data-efficient Off-policy Policy Evaluation for Reinforcement Learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2139–2148, 2016.

[37] Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *Proceedings of the 32th International Conference on Machine Learning*, pages 2380–2388, 2015.

[38] Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-Confidence Off-Policy Evaluation. *AAAI*, pages 3000–3006, 2015.

[39] Masatoshi Uehara, Masahiro Kato, and Shota Yasui. Off-policy evaluation and learning for external validity under a covariate shift. *Advances in Neural Information Processing Systems*, 2020.

[40] Nikos Vlassis, Aurelien Bibaut, Maria Dimakopoulou, and Tony Jebara. On the design of estimators for bandit off-policy evaluation. In *International Conference on Machine Learning*, pages 6468–6476, 2019.

[41] Cameron Voloshin, Hoang M Le, Nan Jiang, and Yisong Yue. Empirical study of off-policy policy evaluation for reinforcement learning. *arXiv preprint arXiv:1911.06854*, 2019.

[42] Yu-Xiang Wang, Alekh Agarwal, and Miroslav Dudik. Optimal and Adaptive Off-policy Evaluation in Contextual Bandits. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3589–3597, 2017.

[43] Tengyang Xie, Yifei Ma, and Yu-Xiang Wang. Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling. In *Advances in Neural Information Processing Systems*, pages 9665–9675, 2019.

A Examples

Our setup allows for many popular multi-armed bandit algorithms, as the following examples illustrate.

Example 1 (Random A/B testing)**.** *We always choose each action uniformly at random:*

$$\pi_{\mathrm{Uniform}}(a \mid x) = \frac{1}{m+1}$$

always holds for any given $a \in \mathcal{A}$ and $x \in \mathcal{X}$.

Example 2 (Bernoulli Thompson Sampling)**.** *When x is given, we sample the potential reward $\tilde{r}(a)$ from the beta distribution $Beta(S_{ta} + \alpha, F_{ta} + \beta)$ for each action in $\mathcal{A}$, where $S_{ta} := \sum_{t'=1}^{t-1} r_t, F_{ta} := (t-1) - S_{ta}$. (α, β) are the parameters of the prior Beta distribution. We then choose the action with the highest sampled potential reward, $a :\in \operatorname*{argmax}_{a' \in \mathcal{A}} \tilde{r}(a')$ (ties are broken arbitrarily). As a result, this algorithm chooses actions with the following probabilities:*

$$\pi_{\mathrm{BernoulliTS}}(a \mid x) = \Pr\{a \in \operatorname*{argmax}_{a' \in \mathcal{A}} \tilde{r}(a')\}$$

for any given $a \in \mathcal{A}$ and $x \in \mathcal{X}$.

Example 3 (IPW Learner)**.** *When $\mathcal{D}$ is given, we can train a deterministic policy $\pi_{\mathrm{det}} : \mathcal{X} \to \mathcal{A}$ by maximizing the IPW estimator as follows:*

$$\begin{aligned}
\pi_{\mathrm{det}}(x) &\in \operatorname*{argmax}_{\pi \in \Pi} \hat{V}_{\mathrm{IPW}}(\pi; \mathcal{D}) \\
&= \operatorname*{argmax}_{\pi \in \Pi} \mathbb{E}_{\mathcal{D}} \left[\frac{\mathbb{I}\{\pi(x_t) = a_t\}}{\pi_b(a_t \mid x_t)} r_t \right] \\
&= \operatorname*{argmin}_{\pi \in \Pi} \mathbb{E}_{\mathcal{D}} \left[\frac{r_t}{\pi_b(a_t \mid x_t)} \mathbb{I}\{\pi(x_t) \neq a_t\} \right]
\end{aligned}$$

, which is equivalent to the cost-sensitive classification problem.

B Definitions of Advanced Off-Policy Estimators

Here we define advanced OPE estimators other than the basic ones, including DM, IPW, and DR, which we describe in Section 2.3.

Self-Normalized Estimators. Self-Normalized Inverse Probability Weighting (SNIPW) is an approach to address the variance issue with the original IPW. It estimates the policy value by dividing the sum of weighted rewards by the sum of importance weights as:

$$\hat{V}_{\mathrm{SNIPW}}(\pi_e; \mathcal{D}) := \frac{\mathbb{E}_{\mathcal{D}}[w(x_t, a_t) r_t]}{\mathbb{E}_{\mathcal{D}}[w(x_t, a_t)]}.$$

SNIPW is more stable than IPW, because policy value estimated by SNIPW is bounded in the support of rewards and its conditional variance given action and context is bounded by the conditional variance of the rewards [15]. IPW does not have these properties. We can define Self-Normalized Doubly Robust (SNDR) in a similar manner as follows.

$$\hat{V}_{\mathrm{SNDR}}(\pi_e; \mathcal{D}) := \mathbb{E}_{\mathcal{D}} \left[\hat{q}(x_t, \pi_e) + \frac{w(x_t, a_t)(r_t - \hat{q}(x_t, a_t))]}{\mathbb{E}_{\mathcal{D}}[w(x_t, a_t)]} \right].$$

Switch Estimators. The DR estimator can still be subject to the variance issue, particularly when the importance weights are large due to low overlap. Switch-DR aims to reduce the effect of the variance issue by using DM where importance weights are large as:

$$\hat{V}_{\mathrm{SwitchDR}}(\pi_e; \mathcal{D}, \hat{q}, \tau) := \mathbb{E}_{\mathcal{D}} \left[\hat{q}(x_t, \pi_e) + w(x_t, a_t)(r_t - \hat{q}(x_t, a_t)) \mathbb{I}\{w(x_t, a_t) \leq \tau\} \right],$$

where $\mathbb{I}\{\cdot\}$ is the indicator function and $\tau \geq 0$ is a hyperparameter. Switch-DR interpolates between DM and DR. When $\tau = 0$, it coincides with DM, while $\tau \to \infty$ yields DR. This estimator is minimax optimal when τ is appropriately chosen [42].

We can define the Switch-IPW estimator in a similar manner as

$$\hat{V}_{\mathrm{SwitchIPW}}(\pi_e; \mathcal{D}, \hat{q}, \tau) := \mathbb{E}_{\mathcal{D}} \left[(\sum_{a \in \mathcal{A}} \hat{q}(x_t, a) \pi_e(a \mid x_t) \mathbb{I}\{w(x_t, a) > \tau\}) + w(x_t, a_t) r_t \mathbb{I}\{w(x_t, a_t) \leq \tau\} \right],$$

which interpolates between DM and IPW.

More Robust Doubly Robust (MRDR). MRDR uses a specialized reward estimator ($\hat{q}_{\mathrm{MRDR}}$) that minimizes the variance of the resulting policy value estimator [7]. This estimator estimates the policy value as:

$$\hat{V}_{\mathrm{MRDR}}(\pi_e; \mathcal{D}, \hat{q}_{\mathrm{MRDR}}) := \hat{V}_{\mathrm{DR}}(\pi_e; \mathcal{D}, \hat{q}_{\mathrm{MRDR}}),$$

where $\hat{q}_{\mathrm{MRDR}}$ is derived by minimizing the (empirical) variance objective:

$$\hat{q}_{\mathrm{MRDR}} := \operatorname*{argmin}_{\hat{q} \in \mathcal{Q}} \mathbb{V}_{\mathcal{D}}(\hat{V}_{\mathrm{DR}}(\pi_e; \mathcal{D}, \hat{q})),$$

where $\mathcal{Q}$ is a function class for the reward estimator. When $\mathcal{Q}$ is well-specified, then $\hat{q}_{\mathrm{MRDR}} = q$. Here, even if $\mathcal{Q}$ is misspecified, the derived reward estimator is expected to behave well since the target function is the resulting variance.

Doubly Robust with Shrinkage (DRs). [32] proposes DRs based on a new weight function $w_o : \mathcal{X} \times \mathcal{A} \to \mathbb{R}_+$ that directly minimizes sharp bounds on the MSE of the resulting estimator. DRs is defined as

$$\hat{V}_{\mathrm{DRs}}(\pi_e; \mathcal{D}, \hat{q}, \lambda) := \mathbb{E}_{\mathcal{D}}[\hat{q}(x_t, \pi_e) + w_o(x_t, a_t; \lambda)(r_t - \hat{q}(x_t, a_t))],$$

where $\lambda \geq 0$ is a hyperparameter and the new weight is

$$w_o(x, a; \lambda) := \frac{\lambda}{w^2(x, a) + \lambda} w(x, a).$$

When $\lambda = 0$, $w_o(x, a; \lambda) = 0$ leading to the standard DM. On the other hand, as $\lambda \to \infty$, $w_o(x, a; \lambda) = w(x, a)$ leading to the original DR.

Algorithm 1 Experimental Protocol for Evaluating Off-Policy Estimators

Input: a policy $\pi^{(he)}$; two different logged bandit feedback datasets $\mathcal{D}^{(he)} = \{(x_t^{(he)}, a_t^{(he)}, r_t^{(he)})\}_{t=1}^T$ and $\mathcal{D}^{(hb)} = \{(x_t^{(hb)}, a_t^{(hb)}, r_t^{(hb)})\}_{t=1}^T$ where $\mathcal{D}^{(he)}$ is collected by $\pi^{(he)}$ and $\mathcal{D}^{(hb)}$ is collected by a different one $\pi^{(hb)}$; an off-policy estimator to be evaluated $\hat{V}$; *split-point* $\tilde{t}$; a number of bootstrap iterations B

Output: the mean and standard deviations of *relative-EE*$(\hat{V})$

$\mathcal{S} \leftarrow \emptyset$
Define the evaluation set: $\mathcal{D}_{\mathrm{ev}} := \mathcal{D}_{1:T}^{(hb)}$ (*in*-sample case), $\mathcal{D}_{\mathrm{ev}} := \mathcal{D}_{1:\tilde{t}}^{(hb)}$ (*out*-sample case)
Define the test set: $\mathcal{D}_{\mathrm{te}} := \mathcal{D}_{1:T}^{(he)}$ (*in*-sample case), $\mathcal{D}_{\mathrm{te}} := \mathcal{D}_{\tilde{t}+1:T}^{(he)}$ (*out*-sample case)
Approximate $V(\pi^{(he)})$ by its on-policy estimation using $\mathcal{D}_{\mathrm{te}}$, i.e, $V_{\mathrm{on}}(\pi^{(he)}; \mathcal{D}_{\mathrm{te}}) = \mathbb{E}_{\mathcal{D}_{\mathrm{te}}}[r_t^{(he)}]$
for $b = 1, \ldots, B$ **do**
    Sample data from $\mathcal{D}_{\mathrm{ev}}$ with *replacement* and construct b-th bootstrapped samples $\mathcal{D}_{\mathrm{ev}}^{(b,*)}$
    Estimate the policy value of $\pi^{(he)}$ by $\hat{V}(\pi^{(he)}; \mathcal{D}_{\mathrm{ev}}^{(b,*)})$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{$*relative-EE* $(\hat{V}; \mathcal{D}_{\mathrm{ev}}^{(b,*)})\}$
end for
Estimate the mean and standard deviations of *relative-EE*$(\hat{V})$ using $\mathcal{S}$

Table 5: Estimation Performances of Reward Estimator ($\hat{q}$)

		Random → Bernoulli TS		**Bernoulli TS → Random**	
Campaigns	**Metrics**	*in*-sample	*out*-sample	*in*-sample	*out*-sample
All	**AUC**	0.56380 ±0.00579	0.53103 ±0.00696	0.57139 ±0.00176	0.51900 ±0.00706
	RCE	0.00217 ±0.00133	-0.00853 ±0.00272	0.00588 ±0.00026	-0.01162 ±0.00271
Men's	**AUC**	0.58068 ±0.00751	0.54411 ±0.01025	0.57569 ±0.00264	0.56528 ±0.00272
	RCE	-0.00019 ±0.00316	-0.01767 ±0.00600	0.00588 ±0.00038	0.00329 ±0.00084
Women's	**AUC**	0.55245 ±0.00588	0.51900 ±0.00706	0.54642 ±0.00157	0.53387 ±0.00249
	RCE	-0.00100 ±0.00196	-0.01162 ±0.00271	0.00307 ±0.00018	0.00140 ±0.00031

Notes: This table presents the area under the ROC curve (AUC) and relative cross-entropy (RCE) of the reward estimator on a validation set for each campaign. The averaged results and their unbiased standard deviations estimated using 30 different bootstrapped samples are reported. $\pi^{(hb)} \rightarrow \pi^{(he)}$ represents the OPE situation where the estimators aim to estimate the policy value of $\pi^{(he)}$ using logged bandit data collected by $\pi^{(hb)}$, meaning that $\hat{q}$ is trained on data collected by $\pi^{(hb)}$.

C Additional Experimental Settings and Results

We describe detailed protocols for evaluating OPE estimators in Algorithm 1. Table 5 reports the estimation accuracies of the reward estimator. Table 6 and 7 show the results of the benchmark experiments on Men's and Women's compaigns.

C.1 Estimation Performance of Reward Estimator

We evaluate the performance of the reward estimator by using the following two evaluation metrics in classification.

Relative Cross Entropy (RCE). RCE is defined as the improvement of an estimation performance relative to the naive estimation, which predicts the mean CTR for every data. We calculate this metric using a size n of validation samples $\{(x_t, y_t)\}_{t=1}^n$ as:

$$\textit{RCE}\ (\hat{q}) := 1 - \frac{\sum_{t=1}^n y_t \log(\hat{q}(x_t)) + (1 - y_t) \log(1 - \hat{q}(x_t))}{\sum_{t=1}^n y_t \log(\hat{q}_{\mathrm{naive}}) + (1 - y_t) \log(1 - \hat{q}_{\mathrm{naive}})}$$

where $\hat{q}_{\mathrm{naive}} := n^{-1} \sum_{t=1}^n y_t$ is the naive estimation. A larger value of RCE means better performance of a predictor.

Table 6: Comparing Relative-Estimation Errors of OPE Estimators (**Men's Campaign**)

	Random → Bernoulli TS		**Bernoulli TS → Random**	
Estimators Compared	*in*-sample	*out*-sample	*in*-sample	*out*-sample
DM	**0.24311** ±0.03128	**0.29088** ±0.03440	**0.24332** ±0.01661	0.12275 ±0.01791
IPW	0.11060 ±0.04173	0.19521 ±0.04533	**0.02908** ±0.02413	0.08407 ±0.02471
SNIPW	**0.09343** ±0.04170	**0.17499** ±0.04611	0.07301 ±0.03406	0.19564 ±0.04117
DR	0.09727 ±0.04091	0.18073 ±0.04519	0.14994 ±0.05710	**0.28765** ±0.07703
SNDR	0.09447 ±0.04139	0.17794 ±0.04629	0.11218 ±0.04287	0.23546 ±0.05585
Switch-DR ($\tau = 5$)	0.23820 ±0.01950	0.27584 ±0.02035	0.17478 ±0.01145	0.06573 ±0.01204
Switch-DR ($\tau = 10$)	0.16504 ±0.02665	0.20912 ±0.03873	0.17381 ±0.01215	0.05575 ±0.01489
Switch-DR ($\tau = 50$)	0.22290 ±0.04091	0.18073 ±0.04519	0.13706 ±0.02529	0.02666 ±0.01919
Switch-DR ($\tau = 100$)	0.09727 ±0.04091	0.18073 ±0.04519	0.11114 ±0.02864	**0.02139** ±0.01596
Switch-DR ($\tau = 500$)	0.09727 ±0.04091	0.18073 ±0.04519	0.05424 ±0.03006	0.05825 ±0.02440
Switch-DR ($\tau = 1000$)	0.09727 ±0.04091	0.18073 ±0.04519	0.05199 ±0.02997	0.06140 ±0.02461
MRDR	**0.09173** ±0.04145	**0.17754** ±0.04673	**0.04385** ±0.03299	0.07649 ±0.02900

Notes: The averaged relative-estimation errors and their unbiased standard deviations estimated over 30 different bootstrapped iterations are reported. $\pi^{(hb)} \rightarrow \pi^{(he)}$ represents the OPE situation where the estimators aim to estimate the policy value of $\pi^{(he)}$ using logged bandit data collected by $\pi^{(hb)}$. The **red** and **green** fonts represent the best and the second best estimators. The **blue** fonts represent the worst estimator for each setting.

Table 7: Comparing Relative-Estimation Errors of OPE Estimators (**Women's Campaign**)

	Random → Bernoulli TS		**Bernoulli TS → Random**	
Estimators Compared	*in*-sample	*out*-sample	*in*-sample	*out*-sample
DM	**0.21719** ±0.03274	**0.25428** ±0.02940	**0.31762** ±0.01011	**0.21892** ±0.01346
IPW	0.02827 ±0.02418	**0.03957** ±0.02779	0.03992 ±0.01997	0.09295 ±0.02527
SNIPW	**0.02827** ±0.02383	0.04221 ±0.02976	0.07564 ±0.02578	0.11461 ±0.02646
DR	0.02835 ±0.02420	**0.04200** ±0.02952	0.09244 ±0.03063	0.12652 ±0.02904
SNDR	0.02833 ±0.02415	0.04280 ±0.02973	0.07659 ±0.02582	0.11809 ±0.02661
Switch-DR ($\tau = 5$)	0.15483 ±0.02355	0.20191 ±0.02660	0.24993 ±0.00614	0.16243 ±0.00919
Switch-DR ($\tau = 10$)	0.05966 ±0.03183	0.10547 ±0.03843	0.21151 ±0.00827	0.12292 ±0.00950
Switch-DR ($\tau = 50$)	0.02835 ±0.02420	**0.04200** ±0.02952	0.12182 ±0.01416	**0.02639** ±0.01515
Switch-DR ($\tau = 100$)	0.02835 ±0.02420	**0.04200** ±0.02952	0.08990 ±0.01381	**0.01129** ±0.00921
Switch-DR ($\tau = 500$)	0.02835 ±0.02420	**0.04200** ±0.02952	**0.01838** ±0.01793	0.05898 ±0.02007
Switch-DR ($\tau = 1000$)	0.02835 ±0.02420	**0.04200** ±0.02952	**0.01644** ±0.01352	0.07120 ±0.02171
MRDR	**0.02809** ±0.02388	0.04354 ±0.03060	0.02800 ±0.01758	0.08990 ±0.01898

Notes: The averaged relative-estimation errors and their unbiased standard deviations estimated over 30 different bootstrapped iterations are reported. $\pi^{(hb)} \rightarrow \pi^{(he)}$ represents the OPE situation where the estimators aim to estimate the policy value of $\pi^{(he)}$ using logged bandit data collected by $\pi^{(hb)}$. The **red** and **green** fonts represent the best and the second best estimators. The **blue** fonts represent the worst estimator for each setting.

Area Under the ROC Curve (AUC). AUC is defined as the probability that positive samples are ranked higher than negative items by a classifier under consideration.

$$AUC\ (\hat{q}) := \frac{1}{n^{\mathrm{pos}} n^{\mathrm{neg}}} \sum_{t=1}^{n^{\mathrm{pos}}} \sum_{j=1}^{n^{\mathrm{neg}}} \mathbb{I}\{\hat{q}(x_t^{\mathrm{pos}}) > \hat{q}(x_j^{\mathrm{neg}})\}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function. $\{x_t^{\mathrm{pos}}\}_{t=1}^{n^{\mathrm{pos}}}$ and $\{x_j^{\mathrm{neg}}\}_{j=1}^{n^{\mathrm{neg}}}$ are sets of positive and negative samples in the validation set, respectively. A larger value of AUC means better performance of a predictor.

C.2 Estimating Mean and Standard Deviation of Performance Measures

To estimate means and standard deviations of relative-EE in the benchmark experiment, we first construct an empirical cumulative distribution function $\hat{F}_K$ of the evaluation set of the logged bandit feedback ($\mathcal{D}_{\mathrm{ev}}$). Then, we draw bootstrap samples $\mathcal{D}_{\mathrm{ev}}^{(1,*)}, \dots, \mathcal{D}_{\mathrm{ev}}^{(B,*)}$ from $\hat{F}_K$ and compute the relative-EE of a given estimator $\hat{V}$ with each set. Finally, we estimate the mean and its standard deviation (Std) of the $\hat{V}$'s relative-EE by

$$\mathrm{Mean}(\textit{relative-EE}(\hat{V}; \mathcal{D}_{\mathrm{ev}})) := \frac{1}{B} \sum_{b=1}^{B} \textit{relative-EE}(\hat{V}; \mathcal{D}_{\mathrm{ev}}^{(b,*)}),$$

$$\mathrm{Std}(\textit{relative-EE}(\hat{V}; \mathcal{D}_{\mathrm{ev}})) := \sqrt{\frac{1}{B-1} \sum_{b=1}^{B} (\textit{relative-EE}(\hat{V}; \mathcal{D}_{\mathrm{ev}}^{(b,*)}) - \mathrm{Mean}(\textit{relative-EE}(\hat{V})))^2},$$

where we use $B = 30$ for all experiments.

D Open Bandit Pipeline (OBP) Package

As described in Section 3, Open Bandit Pipeline contains implementations of dataset preprocessing, several policy learning methods, and a variety of OPE estimators.

Below, we show an example of conducting an offline evaluation of the performance of BernoulliTS using IPW as an OPE estimator and the Random policy as a behavior policy. We see that only ten lines of code are sufficient to complete the standard OPE procedure from scratch (Code Snippet 1).

```
# a case for implementing OPE of BernoulliTS using log data generated by the Random
    policy
>>> from obp.dataset import OpenBanditDataset
>>> from obp.policy import BernoulliTS
>>> from obp.ope import OffPolicyEvaluation, InverseProbabilityWeighting as IPW

# (1) Data loading and preprocessing
>>> dataset = OpenBanditDataset(behavior_policy="random", campaign="all")
>>> bandit_feedback = dataset.obtain_batch_bandit_feedback()

# (2) Off-Policy Learning
>>> evaluation_policy = BernoulliTS(
        n_actions=dataset.n_actions,
        len_list=dataset.len_list,
        is_zozotown_prior=True, # replicate the policy in the ZOZOTOWN production
        campaign="all",
        random_state=12345
    )
>>> action_dist = evaluation_policy.compute_batch_action_dist(
        n_sim=100000, n_rounds=bandit_feedback["n_rounds"]
    )

# (3) Off-Policy Evaluation
>>> ope = OffPolicyEvaluation(bandit_feedback=bandit_feedback,
    ope_estimators=[IPW()])
>>> estimated_policy_value = ope.estimate_policy_values(action_dist=action_dist)

# estimate the performance improvement of BernoulliTS over the Random policy
>>> ground_truth_random = bandit_feedback["reward"].mean()
>>> print(estimated_policy_value["ipw"] / ground_truth_random)
1.198126...
```

Code Snippet 1: **Overall Flow of Off-Policy Evaluation using Open Bandit Pipeline**

In the following subsections, we explain some important features in the example flow.

D.1 Data Loading and Preprocessing

We prepare easy-to-use data loader for Open Bandit Dataset. The `obp.dataset.OpenBanditDataset` class will download and preprocess the original Open Bandit Dataset.

```
# load and preprocess raw data in "All" campaign collected by the Random policy
>>> dataset = OpenBanditDataset(behavior_policy="random", campaign="all")
# obtain logged bandit feedback generated by the behavior policy
>>> bandit_feedback = dataset.obtain_batch_bandit_feedback()
```

Code Snippet 2: **Data Loading and Preprcessing**

Users can implement their own feature engineering in the `pre_process` method of `OpenBanditDataset` class. Moreover, by following the interface of `BaseBanditDataset` in the

dataset module, one can handle future open datasets for bandit algorithms. The dataset module also provide a class to generate synthetic bandit datasets.

D.2 Off-Policy Learning

After preparing the logged bandit data, we now compute the action choice probability by BernoulliTS in the ZOZOTOWN production. Then, we can use it as the evaluation policy.

```
# define evaluation policy (the Bernoulli TS policy here)
>>> evaluation_policy = BernoulliTS(
        n_actions=dataset.n_actions,
        len_list=dataset.len_list,
        is_zozotown_prior=True, # replicate BernoulliTS in the ZOZOTOWN production
        campaign="all",
        random_state=12345
    )
# compute the action choice probability by the evaluation policy by running
    simulation
# action_dist is an array of shape (n_rounds, n_actions, len_list)
# representing the action choice probability made by the evaluation policy
>>> action_dist = evaluation_policy.compute_batch_action_dist(
        n_sim=100000, n_rounds=bandit_feedback["n_rounds"]
    )
```

Code Snippet 3: **Off-Policy Learning**

The `compute_batch_action_dist` method of BernoulliTS computes the action choice probabilities based on given hyperparameters of the beta distribution. By activating the `is_zozotown_prior` argument of BernoulliTS, one can replicate BernoulliTS used in ZOZOTOWN production. `action_dist` is an array representing the distribution over actions made by the evaluation policy.

D.3 Off-Policy Evaluation

Our final step is OPE, which attempts to estimate the performance of bandit algorithms using log data generated by a behavior policy. Our pipeline also provides an easy procedure for doing OPE as follows.

```
# estimate the policy value of BernoulliTS using its action choice probability
# it is possible to set multiple OPE estimators to the 'ope_estimators' argument
>>> ope = OffPolicyEvaluation(bandit_feedback=bandit_feedback,
    ope_estimators=[IPW()])
>>> estimated_policy_value = ope.estimate_policy_values(action_dist=action_dist)
>>> print(estimated_policy_value)
{"ipw": 0.004553...} # dictionary containing policy values estimated by each
    estimator

# compare the estimated performance of BernoulliTS
# with the ground-truth performance of the Random policy
# our OPE procedure suggests that BernoulliTS improves the Random policy by 19.81%
>>> ground_truth_random = bandit_feedback["reward"].mean()
>>> print(estimated_policy_value["ipw"] / ground_truth_random)
1.198126...
```

Code Snippet 4: **Off-Policy Evaluation**

Users can implement their own OPE estimator by following the interface of `BaseOffPolicyEstimator` class. `OffPolicyEvaluation` class summarizes and compares the policy values estimated by several off-policy estimators. `bandit_feedback["reward"].mean()` is the empirical mean of factual rewards (on-policy estimate of the policy value) in the log and thus is the ground-truth performance of the behavior policy (the Random policy).