


Large-scale Open Dataset and Pipeline for Bandit Algorithms



Yuta Saito¹, Shunsuke Aihara²,

Megumi Matsutani², and Yusuke Narita³

¹Tokyo Institute of Technology ²ZOZO Technologies, Inc. ³Yale University.

Outline

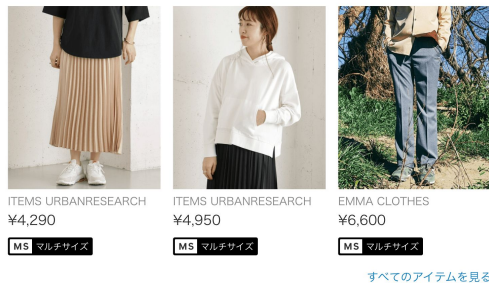
- overview of *off-policy evaluation* (just briefly)
- *open bandit project* (on-going)
 - open bandit dataset v1 (v2 will be released)
 - open bandit pipeline
 - example analysis with the data and pipeline
 - limitations and future work
- Q & A

Machine Learning for Decision Making (Bandit / RL)

We often use machine learning to make **decisions, not predictions**

decide which items to show

a coming user



observe reward (e.g., click)



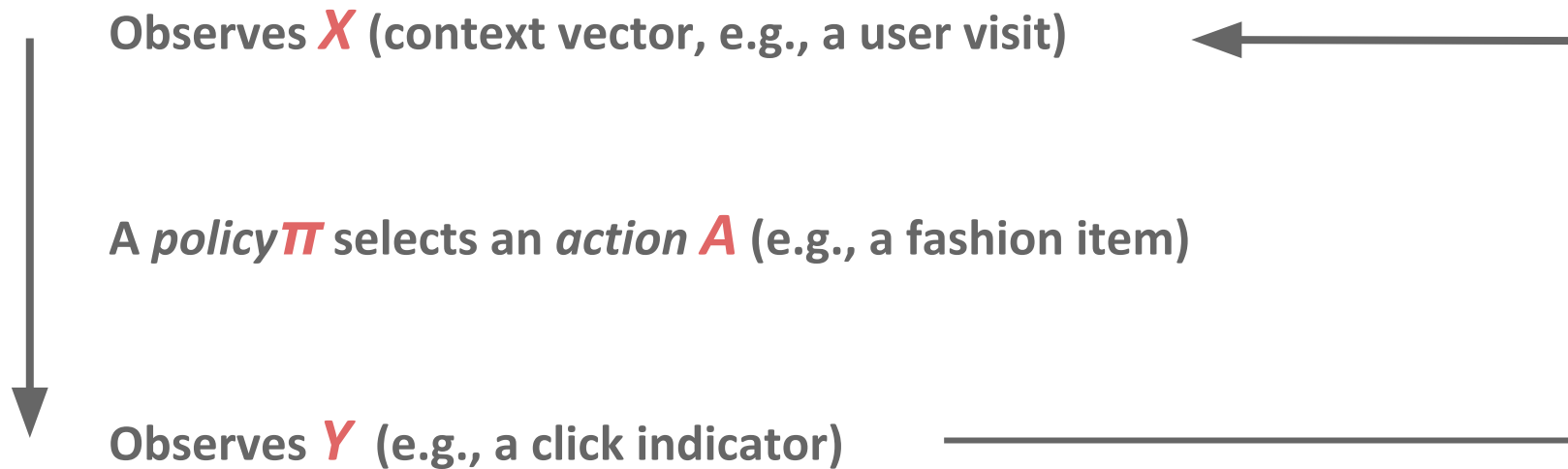
often multiple items are recommended at the same time

Many Applications of “Machine Decision Making”

- news recommendation (by Yahoo)
- music/playlist recommendation (by Spotify)
- artwork personalization (by Netflix)
- ad allocation optimization (by Criteo)
- medicine
- education

We want to evaluate the performance of a *new decision making policy* using data generated by a *behavior, past policy*

Data Generating Process (contextual bandit setting)




a *policy* interacts with the environments and produced the log data

Logged Bandit Feedback

We can use the *logged bandit feedback* collected by a *behavior (or past) policy* to estimate the policy value of a new policy

$$\mathcal{D} = \{ (X_i, A_i, Y_i) \}_{i=1}^n$$


$$A_i \sim \pi_b (a \mid X_i)$$

action choice by behavior policy


$$Y_i = Y_i (A_i)$$

observed reward

Estimation Target in Off-Policy Evaluation

In OPE, we aim to estimate the following *policy value*
of an *evaluation (or new)* policy

$$V(\pi_e) := \mathbb{E}_{(Y(\cdot), X)} \left[\sum_{a=0}^m Y(a) \pi_e(a \mid X) \right]$$



expected reward obtained by running π_e on a real system

Benefits of Off-Policy Evaluation

Accurately estimating the policy value of an evaluation policy

$$V(\pi_e) \approx \hat{V}(\pi_e; \mathcal{D})$$

an estimated policy value of π_e using historical data \mathcal{D}

- avoid deploying poor performing policies
- identify promising new policies among many candidates

Direct Method (DM)

DM first estimates the expected reward and uses it to estimate the policy value

$$\hat{V}_{DM}(\pi_e; \mathcal{D}) = \mathbb{E}_n \left[\sum_{a=0}^m \pi(a \mid X_i) \underbrace{\hat{\mu}(X_i, a)}_{\text{estimated expected reward}} \right]$$

- **High bias** when the model is mis-specified
- **Low variance**

$$\begin{aligned} \mathbb{E}[Y(a) \mid X = x] \\ \approx \hat{\mu}(x, a) \end{aligned}$$

Inverse Probability Weighting (IPW)

IPW re-weights observed rewards by importance weights

$$\hat{V}_{IPW}(\pi_e; \mathcal{D}) = \mathbb{E}_n \left[Y_i \frac{\pi_e(A_i | X_i)}{\pi_b(A_i | X_i)} \right]$$

importance weight

- **Consistent**
- **High variance** when old and new policies are largely different

Doubly Robust (DR)

DR uses DM as a baseline and applies IPW to shifted rewards

$$\hat{V}_{DR}(\pi_e; \mathcal{D}) = \underbrace{\hat{V}_{DM}(\pi_e; \mathcal{D})}_{\text{baseline}} + \mathbb{E}_n \left[\underbrace{(Y_i - \hat{\mu}(X_i, A_i)) \frac{\pi_e(A_i|X_i)}{\pi_b(A_i|X_i)}}_{\text{weighted shifted reward}} \right]$$

- **Consistent**
- **Locally Efficient**

$$\mathbb{E}[Y(a) \mid X = x] \approx \hat{\mu}(x, a)$$

Theoretical/Methodological Advances in OPE

- Self-Normalized IPW [[Swaminathan and Joachims 2015](#)]
- Switch Doubly Robust Estimator [[Wang+ 2017](#)]
- More Robust Doubly Robust Estimator [[Farajtabar+ 2018](#)]
- Hirano-Imbens-Ridder Estimator [[Narita+ 2019](#)]
- REG and EMP [[Kallus & Uehara 2019](#)]
- Doubly Robust with Shrinkage [[Su+ 2020](#)]

**It seems the OPE community
have made great progress
over the years!**

There are many other estimators in the reinforcement learning setting

Issues with the current experimental procedures

Experiments in every OPE paper rely on

- Synthetic or classification data (unrealistic)

or

- (Real, but) Unpublished data (irreproducible)

We need real-world data enabling the “*evaluation of OPE*”

Project's Goal and Components

We enable *realistic and reproducible* experiments on

- Bandit Algorithms
- Off-Policy Evaluation (OPE)

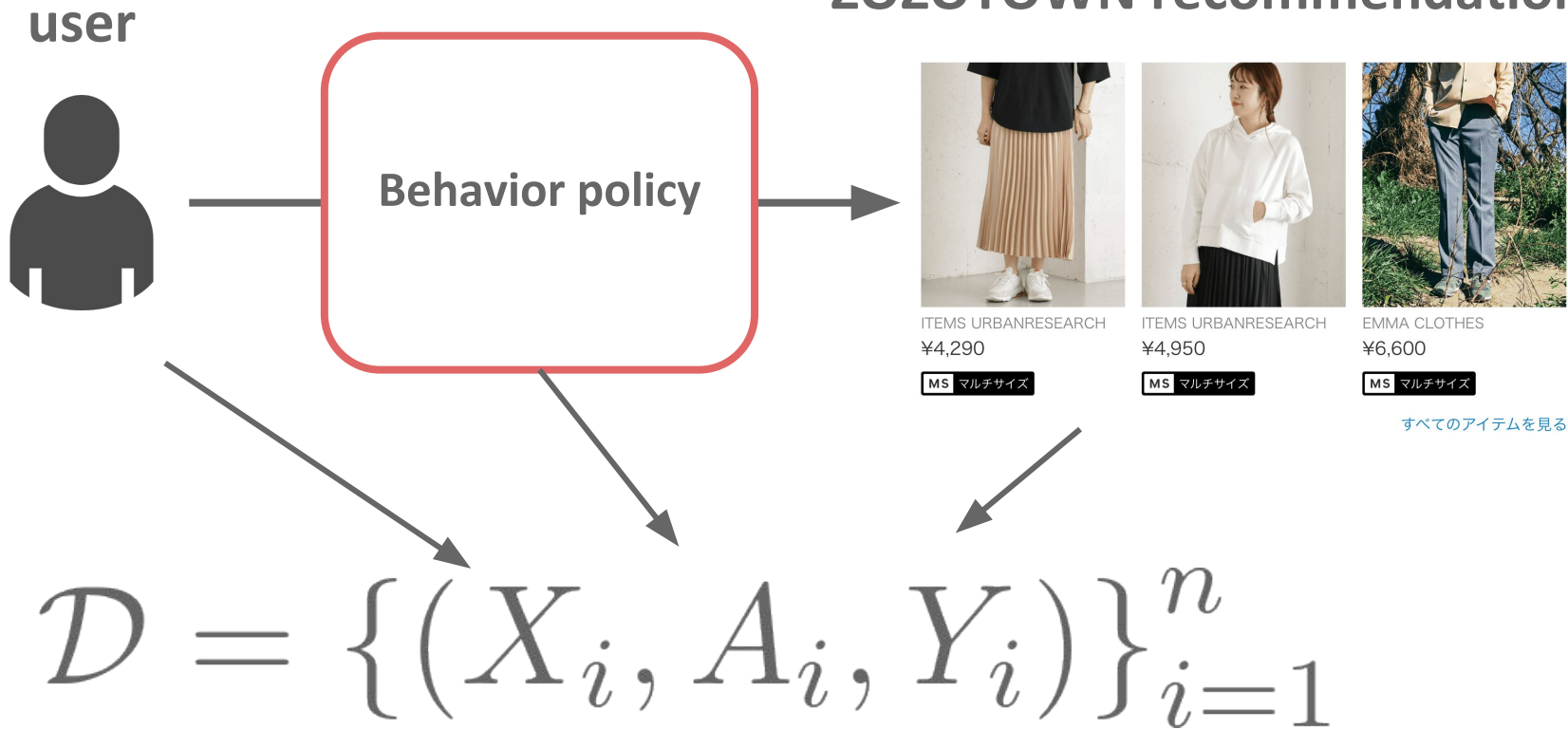


“Open Bandit Dataset”

and ***“Open Bandit Pipeline”***

Overview of Open Bandit Dataset

ZOZOTOWN recommendation

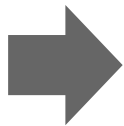


Schema of Open Bandit Dataset

	A		$\pi_b(\cdot X)$	Y	X	
timestamp	item_id	position	propensity_score	click indicator	features	...
2019-11-xx	25	1	0.0002	0	e2500f3f	...
2019-11-xx	32	2	0.043	1	7c414ef7	...
2019-11-xx	11	3	0.167	0	60bd4df9	...
2019-11-xx	40	1	0.0011	0	7c20d9b5	...
...

Essential Features of Open Bandit Dataset

- **over 25M records** collected by online experiments of bandit algorithms on a large-scale fashion e-commerce (*ZOZOTOWN*)
- **logged bandit feedback collected by *multiple* bandit policies**
 - *Uniform Random* (fixed)
 - *Bernoulli Thompson Sampling* (pre-trained before collection)



enabling realistic experiments on OPE for the first time

Protocol for the Evaluation of OPE with Open Bandit Dataset

1. Prepare **two** logged bandit feedback data collected by different policies

$$\mathcal{D}^{(1)} = \left\{ \left(X_i^{(1)}, A_i^{(1)}, Y_i^{(1)} \right) \right\}_{i=1}^n$$

collected by $\pi^{(1)}$

$$\mathcal{D}^{(2)} = \left\{ \left(X_i^{(2)}, A_i^{(2)}, Y_i^{(2)} \right) \right\}_{i=1}^n$$

collected by $\pi^{(2)}$

Protocol for the Evaluation of OPE with Open Bandit Dataset

2. Regard one policy as an **evaluation policy** and the other as a **behavior policy**. Then, estimate the performance of the evaluation policy by OPE

$$V(\pi^{(1)}) \approx \hat{V}(\pi^{(1)}; \mathcal{D}^{(2)})$$

$\pi^{(1)}$: **evaluation policy**

$\pi^{(2)}$: **behavior policy**

- The task here is to evaluate the accuracy of \hat{V}

Protocol for the Evaluation of OPE with Open Bandit Dataset

3. Regard the *on-policy estimation* of the policy value of the evaluation policy as the ground-truth policy value

$$V(\pi^{(1)}) = \mathbb{E}_{n^{(1)}} [Y^{(1)}]$$

we can do this on-policy estimation because we have $\mathcal{D}^{(1)}$ in our data

Protocol for the Evaluation of OPE with Open Bandit Dataset

4. Compare the estimated policy value with the ground-truth to evaluate the OPE estimator, for example, using the *relative estimation error*

$$\text{relative estimation error of } \hat{V} = \left| \frac{\hat{V} \left(\pi^{(1)}; \mathcal{D}^{(2)} \right) - V \left(\pi^{(1)} \right)}{V \left(\pi^{(1)} \right)} \right|$$

By applying this procedure to several estimators, we can compare them

Comparison with Existing Real-World Bandit Datasets

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

	Criteo Data (Lefortier et al. 2016)	Yahoo! Data (Li et al. 2010)	Open Bandit Dataset (ours)
Domain	Display Advertising	News Recommendation	Fashion E-Commerce
#Data	$\geq 103\text{M}$	$\geq 40\text{M}$	$\geq 26\text{M}$ (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	✗	✗	✓

Our Open Bandit Dataset

- contains *multiple behavior policies*
- enables *the evaluation of OPE for the first time*
- *comes with the pipeline implementations* (Open Bandit Pipeline)

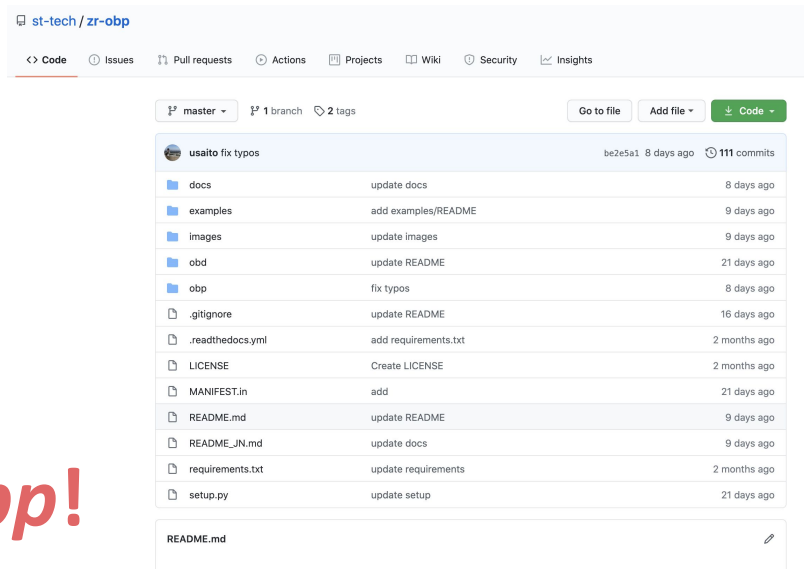
Open Bandit Pipeline (OBP)

We have implemented *Open Bandit Pipeline (OBP)*
to streamline and standardize experiments on OPE



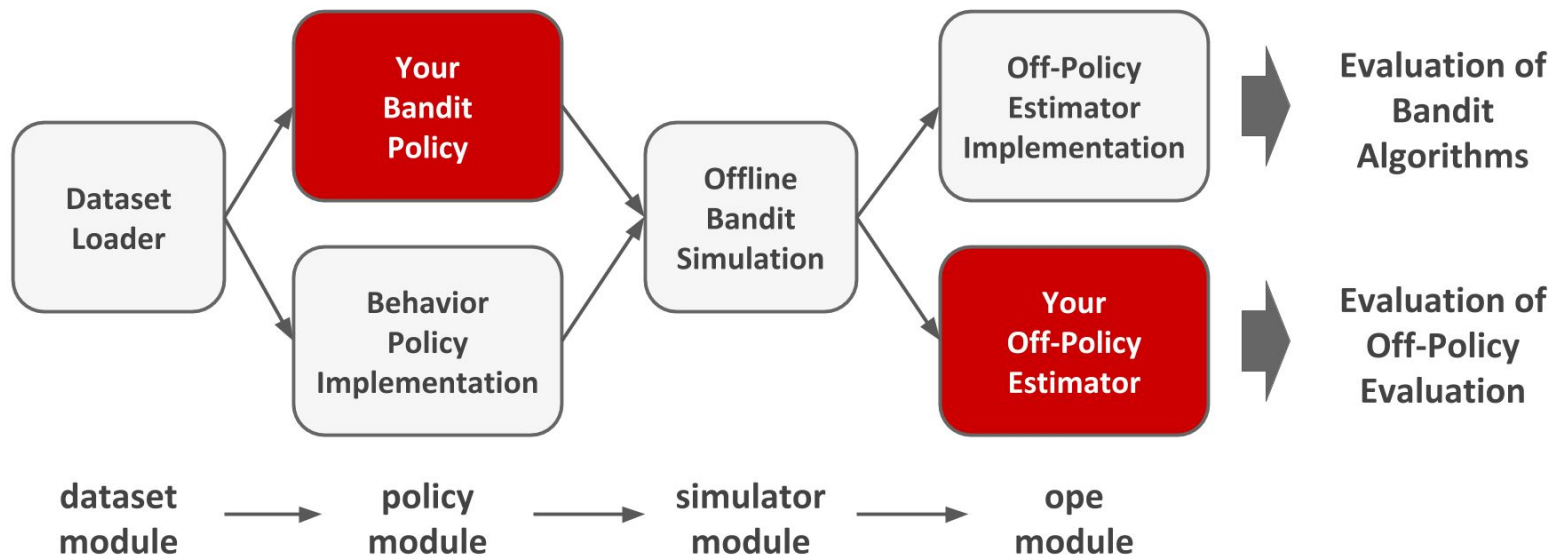
**OPEN
BANDIT
PIPELINE™**

*find out **zr-obp**!*

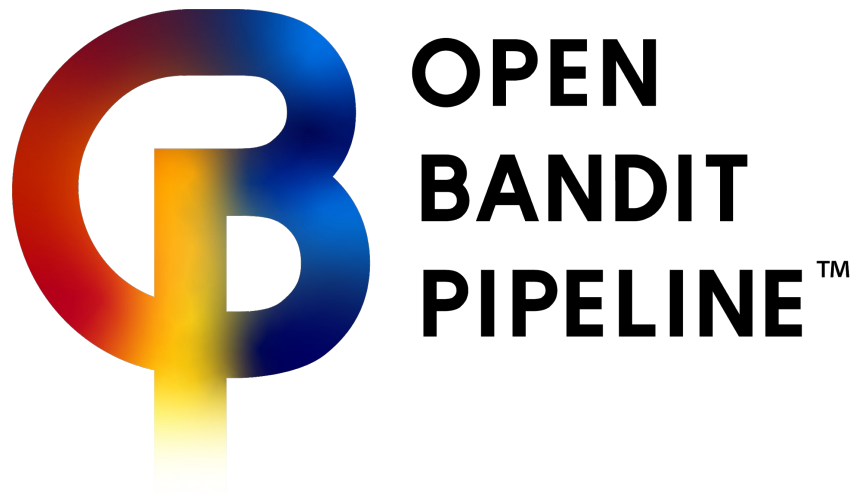


Structure of Open Bandit Pipeline

OBP consists of **four main modules** (dataset, policy, simulator, and ope)



Proof of Concept Demo with Our Data and Pipeline



Let me now run a quickstart example of OBP

Other Nice Features

We can easily implement experiments on OPE
or OPE itself with our OBP

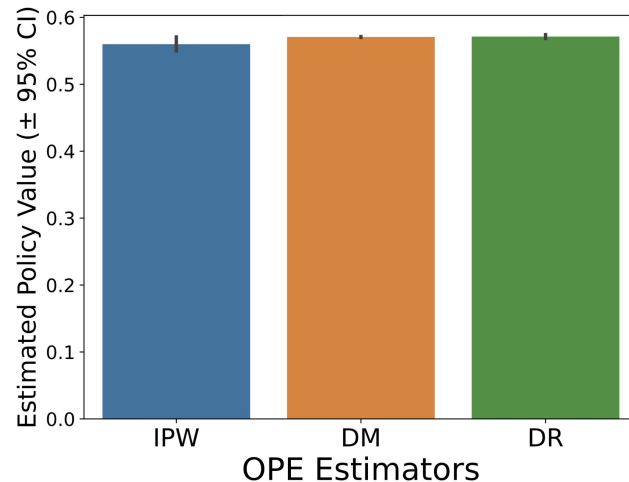
```
# a case for implementing OPE of the BernoulliTS policy using log data generated by the Random policy
from obp.dataset import OpenBanditDataset
from obp.policy import BernoulliTS
from obp.simulator import run_bandit_simulation
from obp.ope import OffPolicyEvaluation, ReplayMethod

# (1) Data loading and preprocessing
dataset = OpenBanditDataset(behavior_policy='random', campaign='women')
bandit_feedback = dataset.obtain_batch_bandit_feedback()

# (2) Offline Bandit Simulation
counterfactual_policy = BernoulliTS(n_actions=dataset.n_actions, len_list=dataset.len_list)
selected_actions = run_bandit_simulation(bandit_feedback=bandit_feedback, policy=counterfactual_policy)

# (3) Off-Policy Evaluation
ope = OffPolicyEvaluation(bandit_feedback=bandit_feedback, ope_estimators=[ReplayMethod()])
estimated_policy_value = ope.estimate_policy_values(selected_actions=selected_actions)

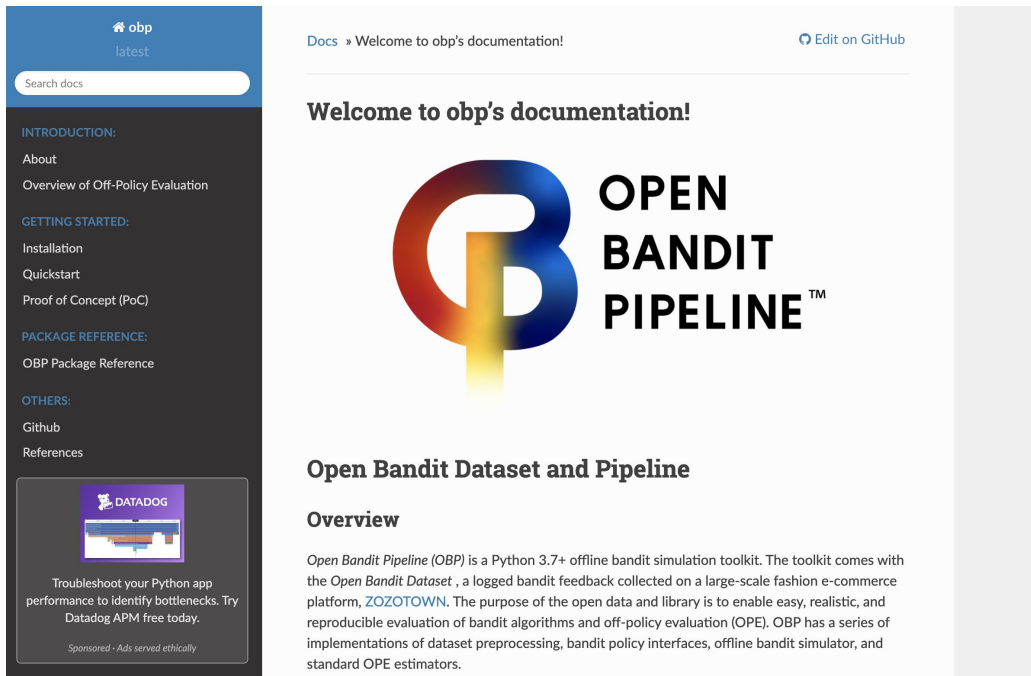
# estimated performance of BernoulliTS relative to the ground-truth performance of Random
relative_policy_value_of_bernoulli_ts = estimated_policy_value['rm'] / bandit_feedback['reward'].mean()
print(relative_policy_value_of_bernoulli_ts) # 1.120574...
```



Potential Users

- Researchers can compare their own method with others in an easy, realistic, and reproducible manner
- Practitioners (engineers, data scientists) can evaluate candidate policies and identify the best one immediately using our pipeline and their own data

Other Nice Features



obp
latest

Search docs

INTRODUCTION:

- About
- Overview of Off-Policy Evaluation

GETTING STARTED:


- Installation
- Quickstart
- Proof of Concept (PoC)

PACKAGE REFERENCE:

- OBP Package Reference

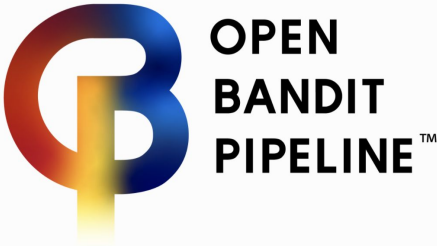
OTHERS:

- Github
- References


Troubleshoot your Python app performance to identify bottlenecks. Try Datadog APM free today.
Sponsored - Ads served ethically

Docs » Welcome to obp's documentation! [Edit on GitHub](#)

Welcome to obp's documentation!



Open Bandit Dataset and Pipeline

Overview

Open Bandit Pipeline (OBP) is a Python 3.7+ offline bandit simulation toolkit. The toolkit comes with the *Open Bandit Dataset*, a logged bandit feedback collected on a large-scale fashion e-commerce platform, [ZOZOTOWN](#). The purpose of the open data and library is to enable easy, realistic, and reproducible evaluation of bandit algorithms and off-policy evaluation (OPE). OBP has a series of implementations of dataset preprocessing, bandit policy interfaces, offline bandit simulator, and standard OPE estimators.

We built a detailed [documentation](#) of Open Bandit Pipeline

Comparison with Existing Bandit Packages

Table 3: Comparison of Currently Available Packages of Bandit Algorithms

	contextualbandits	RecoGym (Rohde et al. 2018)	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	✗	✗	✓

Our Open Bandit Pipeline

- can *handle real-world bandit data* (including ours)
- implements *advanced OPE estimators* (SNIPW, Switch, MRDR, and DML)
- streamline *the evaluation of OPE*

Benchmark Results of Some OPE Methods

We performed benchmark comparison on basic OPE estimators

Table 4: Comparing Relative-Estimation Errors of OPE Estimators (**Random** → **Bernoulli TS**)

Estimators	Campaigns		
	All	Men's	Women's
DM	0.23879 [0.22998, 0.24988]	0.24155 [0.22656, 0.25592]	0.22884 [0.22224, 0.23423]
IPW	0.03477 [0.01147, 0.06592]	0.09806 [0.07485, 0.12151]	0.03252 [0.01708, 0.04912]
SNIPW	0.03381 [0.01005, 0.06662]	0.08153 [0.05677, 0.10592]	0.03179 [0.01562, 0.04825]
DR	0.03487 [0.01094, 0.06784]	0.08528 [0.06186, 0.10876]	0.03224 [0.01605, 0.04843]

IPW, SNIPW, and DR seem accurate, but this is a very simple task

Some Limitations...

We now assume that the **click of an item (reward)**
depends only on that item and position

$$\mathcal{A} = \mathcal{I} \times \mathcal{K}$$

action set

item set
(total of 80 items)

position set
(total of 3 positions)

Some Limitations...

- assumption on the action decomposition might be too strict

$$\mathcal{A} = \mathcal{I} \times \mathcal{K}$$

because it ignores the interactions among items in the same list

→ We will compare other estimators relying on other reasonable assumptions (e.g., estimators for the slate setting)

Some Limitations...

- the current data contains only context-free policies
→ we will run another experiment to collect more useful bandit datasets (open bandit dataset v2)
- current benchmark analysis is simple, primitive
→ we are now conducting extensive experiments to answer
 - Can OPE estimate the performance of a new policy in the future environment (out-sample policy value)?
 - How does the performance of OPE change with different settings?

Thank you!

Please come to our poster for discussions!



github: <https://github.com/st-tech/zr-obp>

docs: <https://zr-obp.readthedocs.io/en/latest/>

dataset: <https://research.zozo.com/data.html>