Large-scale Validation of Counterfactual Learning Methods: A Test-Bed

Damien Lefortier[*]
Facebook & University of Amsterdam
dlefortier@fb.com

Adith Swaminathan
Cornell University, Ithaca, NY
adith@cs.cornell.edu

Xiaotao Gu
Tsinghua University, Beijing, China
gxt13@mails.tsinghua.edu.cn

Thorsten Joachims
Cornell University, Ithaca, NY
tj@cs.cornell.edu

Maarten de Rijke
University of Amsterdam
derijke@uva.nl

Abstract

The ability to perform effective off-policy learning would revolutionize the process of building better interactive systems, such as search engines and recommendation systems for e-commerce, computational advertising and news. Recent approaches for off-policy evaluation and learning in these settings appear promising [1, 2]. With this paper, we provide real-world data and a standardized test-bed to systematically investigate these algorithms using data from display advertising. In particular, we consider the problem of filling a banner ad with an aggregate of multiple products the user may want to purchase. This paper presents our test-bed, the sanity checks we ran to ensure its validity, and shows results comparing state-of-the-art off-policy learning methods like doubly robust optimization [3], POEM [2], and reductions to supervised learning using regression baselines. Our results show experimental evidence that recent off-policy learning methods can improve upon state-of-the-art supervised learning techniques on a large-scale real-world data set.

1 Introduction

Effective learning methods for optimizing policies based on logged user-interaction data have the potential to revolutionize the process of building better interactive systems. Unlike the industry standard of using expert judgments for training, such learning methods could directly optimize user-centric performance measures, they would not require interactive experimental control like online algorithms, and they would not be subject to the data bottlenecks and latency inherent in A/B testing.

Recent approaches for off-policy evaluation and learning in these settings appear promising [1, 2, 4], but highlight the need for accurately logging propensities of the logged actions. With this paper, we provide the first public dataset that contains accurately logged propensities for the problem of Batch Learning from Bandit Feedback (BLBF). We use data from Criteo, a leader in the display advertising space. In addition to

[*]This work was done while working at Criteo.

providing the data, we propose an evaluation methodology for running BLBF learning experiments and a standardized test-bed that allows the research community to systematically investigate BLBF algorithms.

At a high level, a BLBF algorithm operates in the contextual bandit setting and solves the following learning task:

1. Take as input: $\{\pi_0, \langle x_i, y_i, \delta_i \rangle_{i=1}^n\}$. π_0 encodes the system from which the logs were collected, x denotes the input to the system, y denotes the output predicted by the system and δ is a number encoding the observed online metric for the output that was predicted;
2. Produce as output: π, a new policy that maps $x \mapsto y$; and
3. Such that π will perform well (according to the metric δ) *if it were* deployed online.

We elaborate on the definitions of x, y, δ, π_0 as logged in our dataset in the next section. Since past research on BLBF was limited due to the availability of an appropriate dataset, we hope that our test-bed will spur research on several aspects of BLBF and off-policy evaluation, including the following:

1. New training objectives, learning algorithms, and regularization mechanisms for BLBF;
2. Improved model selection procedures (analogous to cross-validation for supervised learning);
3. Effective and tractable policy classes $\pi \in \Pi$ for the specified task $x \mapsto y$; and
4. Algorithms that can scale to massive amounts of data.

The rest of this paper is organized as follows. In Section 2, we describe our standardized test-bed for the evaluation of off-policy learning methods. Then, in Section 3, we describe a set of sanity checks that we used on our dataset to ensure its validity and that can be applied generally when gathering data for off-policy learning and evaluation. Finally, in Section 4, we show results comparing state-of-the-art off-policy learning methods like doubly robust optimization [3], POEM [2], and reductions to supervised learning using regression baselines. Our results show, for the first time, experimental evidence that recent off-policy learning methods can improve upon state-of-the-art supervised learning techniques on a large-scale real-world data set.

2 Dataset

We create our test-bed using data from display advertising, similar to the Kaggle challenge hosted by Criteo in 2014 to compare CTR prediction algorithms.[1] However, in this paper, we do not aim to build clickthrough or conversion prediction models for bidding in real-time auctions [5, 6]. Instead, we consider the problem of filling a banner ad with an aggregate of multiple products the user may want to purchase. This part of the system takes place after the bidding agent has won the auction. In this context, each ad has one of many banner types, which differ in the number of products they contain and in their layout as shown in Figure 1. The task is to choose the products to display in the ad knowing the banner type in order to maximize the number of clicks. This task is thus very different from the Kaggle challenge.

In this setting of choosing the best products to fill the banner ad, we can easily gather exploration data where the placement of the products in the banner ad is randomized, without incurring a prohibitive cost unlike in Web search for which such exploration is much more costly (see, e.g., [7, 8]). Our logging policy uses randomization aggressively, while being very different from a uniformly random policy.

Each banner type corresponds to a different *look & feel* of the banner ad. Banner ads can differ in the number of products, size, geometry (vertical, horizontal, ...), background color and in the data shown

[1] `https://www.kaggle.com/c/criteo-display-ad-challenge`

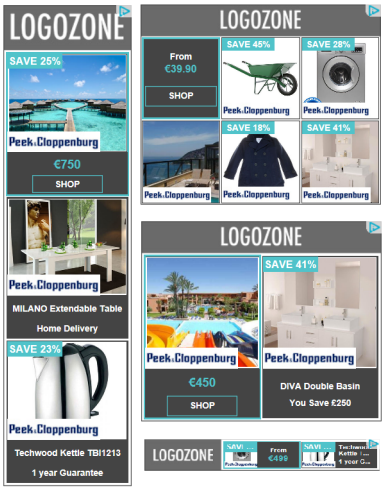

Figure 1: Four examples of ads used in display advertising: a vertical ad, a grid, and two horizontal ads (mock-ups).

(with or without a product description or a call to action); these we call the *fixed* attributes. Banner types may also have *dynamic aspects* such as some form of pagination (multiple pages of products) or an animation. Some examples are shown in Figure 1. Throughout the paper, we label positions in each banner type from 1 to N from left to right and from top to bottom. Thus 1 is the top left position.

For each user impression, we denote a user context by c, the number of slots in the banner type by l_c, and the candidate pool of products p by P_c. Each context c and product p pair is described by features $\phi(c, p)$. The input x to the system encodes $c, P_c, \{\phi(c, p) : p \in P_c\}$. The logging policy π_0 stochastically selects products to construct a banner by first computing non-negative scores f_p for all candidate products $p \in P_c$, and using a Plackett-Luce ranking model (i.e., sampling without replacement from the multinomial distribution defined by the f_p scores):

$$P(slot1 = p) = \frac{f_p}{\sum_{\{p' \in P_c\}} f_{p'}} \qquad P(slot2 = p' \mid slot1 = p) = \frac{f_{p'}}{\sum_{\{p^\dagger \in P_c \wedge p^\dagger \neq p\}} f_{p^\dagger}}, \quad \text{etc.} \tag{1}$$

The propensity of a chosen banner ad $\langle p_1, p_2, \ldots \rangle$ is $P(slot1 = p_1) \cdot P(slot2 = p_2 \mid slot1 = p_1) \cdot \ldots$. With these propensities in hand, we can counterfactually evaluate any banner-filling policy in an unbiased way using inverse propensity scoring [9].

The following was logged, committing to a single feature encoding $\phi(c, p)$ and a single π_0 that produces the scores f for the entire duration of data collection.

- Record the feature vector $\phi(c, p)$ for all products in the candidate set P_c;
- Record the selected products sampled from π_0 via the Plackett-Luce model and its propensity;
- Record the click/no click and their location(s) in the banner.

The format of this data is:

```
example ${exID}:  ${hashID} ${wasAdClicked} ${propensity} ${nbSlots} ${nbCandidates} ${displayFeat1}:${v_1} ...

${wasProduct1Clicked} exid:${exID} ${productFeat1_1}:${v1_1} ...
```

```
...
${wasProductMClicked} exid:${exID} ${productFeatM_1}:${vM_1} ...
```

Each impression is represented by $M + 1$ lines where M is the cardinality of P_c and the first line is a header containing summary information. Note that the first ${nbSlots} candidates correspond to the displayed products ordered by position (consequently, ${wasProductMClicked} information for all other candidates is irrelevant). There are 35 features. Display features are context features or banner type features, which are constant for all candidate products in a given impression. Each unique quadruplet of feature IDs $\langle 1, 2, 3, 5 \rangle$ correspond to a unique banner type. Product features are based on the similarity and/or complementarity of the candidate products with historical products seen by the user on the advertiser's website. We also included interaction terms between some of these features directly in the dataset to limit the amount of feature engineering required to get a good policy. Features 1 and 2 are numerical, while all other features are categorical. Some categorical features are multi-valued, which means that they can take more than one value for the same product (order does not matter). Note that the example ID is increasing with time, allowing temporal slices for evaluation [10], although we do not enforce this for our test-bed. Importantly, non-clicked examples were sub-sampled aggressively to reduce the dataset size and *we kept only a random* 10% *sub-sample of them*. So, one needs to account for this during learning and evaluation – the evaluator we provide with the test-bed accounts for this sub-sampling.

The result is a dataset of over 103 million ad impressions. In this dataset, we have:

- $8500+$ banner types with the top 10 banner types representing 30% of the total number of ad impressions, the top 50 about 65%, and the top 100 about 80%.
- The number of displayed products is between 1 and 6 included.
- There are over $21M$ impressions for 1-slot banners, over $35M$ for 2-slot, almost $23M$ for 3-slot, $7M$ for 4-slot, $3M$ for 5-slot and over $14M$ for 6-slot banners.
- The size of the candidate pool P_c is about 10 times (upper bound) larger than the number of products to display in the ad.

This dataset is hosted on Amazon AWS (35GB gzipped / 256GB raw). Details for accessing and processing the data are available at `http://www.cs.cornell.edu/~adith/Criteo/`.

3 Sanity Checks

The work-horse of counterfactual evaluation is Inverse Propensity Scoring (IPS) [11, 9]. IPS requires accurate propensities, and, to a crude approximation, produces estimates with variance that scales roughly with the range of the inverse propensities. In Table 1, we report the number of impressions and the average and largest inverse propensities, partitioned by ${nbSlots}. When constructing confidence intervals for importance weighted estimates like IPS, we often appeal to asymptotic normality of large sample averages [12]. However, if the inverse propensities are very large relative to the number of samples (as we can see for $\${nbSlots} \geq 4$), the asymptotic normality assumption will probably be violated.

There are some simple statistical tests that can be run to detect some issues with inaccurately logged propensities [13]. These *arithmetic* and *harmonic* tests, however, require that the candidate actions available for each impression are fixed a priori. In our scenario, we have a context-dependent candidate set that precludes running these tests, so we propose a more general class of diagnostics that can detect some systematic biases and issues in propensity-logged datasets.

Some notation: $x_i \overset{iid}{\sim} \Pr(X)$; $y_i \sim \pi_0(Y \mid x_i)$; $\delta_i \sim \Pr(\Delta \mid x_i, y_i)$. The propensity for the logging policy π_0 to take the logged action y_i in context x_i is denoted $q_i \equiv \pi_0(y_i \mid x_i)$. If the propensities

Table 1: Number of impressions and propensity statistics computed for slices of traffic with k-slot banners, $1 \leq k \leq 6$. Estimated sample size ($\hat{N}$) corrects for 10% sub-sampling of unclicked impressions.

#Slots	**1**	**2**	**3**	**4**	**5**	**6**
#Impressions	$2.13e+07$	$3.55e+07$	$2.27e+07$	$6.92e+06$	$2.95e+06$	$1.40e+07$
$\hat{N}$	$2.03e+08$	$3.39e+08$	$2.15e+08$	$6.14e+07$	$2.65e+07$	$1.30e+08$
Avg(InvPropensity)	11.96	$3.29e+02$	$1.87e+04$	$2.29e+06$	$2.62e+07$	$3.51e+09$
Max(InvPropensity)	$5.36e+05$	$3.38e+08$	$3.23e+10$	$9.78e+12$	$2.03e+12$	$2.34e+15$

are correctly logged, then the expected importance weight should be 1 for any new banner-filling policy $\pi(Y \mid x)$. Formally, we have the following:

$$\hat{C}(\pi) = \frac{1}{N} \sum_{i=1}^{N} \frac{\pi(y_i \mid x_i)}{q_i} \simeq 1. \tag{2}$$

The IPS estimate for a new policy is simply:

$$\hat{R}(\pi) = \frac{1}{N} \sum_{i=1}^{N} \delta_i \frac{\pi(y_i \mid x_i)}{q_i}. \tag{3}$$

These equations are valid when π_0 has full support, as our logging system does: $\pi_0(y \mid x) > 0 \qquad \forall x, y$. The self-normalized estimator [14, 4] is:

$$\hat{R}_{snips}(\pi) = \frac{\hat{R}(\pi)}{\hat{C}(\pi)}. \tag{4}$$

Remember that we sub-sampled non-clicked impressions. Sub-sampling is indicated by the binary random variable o_i:

$$o_i \sim \Pr(O = 1 \mid \delta) = \begin{cases} 0.1 & \text{if } \delta = 0, \\ 1 & \text{otherwise.} \end{cases} \tag{5}$$

The IPS estimate and the diagnostic above are not computable in our case since they require all data-points before sub-sampling. So, we use the following straightforward modification to use only our N sub-sampled data-points instead.

First, we estimate the number of data-points before sub-sampling $\hat{N}$ only using samples where $o_i = 1$:

$$\hat{N} = \sum_{i=1}^{N} \frac{\mathbb{1}\{o_i = 1\}}{\Pr(O = 1 \mid \delta_i)} = \#\{\delta = 1\} + 10\#\{\delta = 0\}. \tag{6}$$

$\hat{N}$ is an unbiased estimate of $N = \sum_{i=1}^{N} 1$ since $\mathbb{E}_{(x_i, y_i, \delta_i)} \mathbb{E}_{o_i \sim \Pr(O \mid \delta_i)} \left[\frac{\mathbb{1}\{o_i = 1\}}{\Pr(O = 1 \mid \delta_i)} \right] = \mathbb{E}_{(x_i, y_i, \delta_i)} 1 = 1$.
Next, consider estimating $R(\pi) = \mathbb{E}_{(x_i, y_i, \delta_i)} \delta_i \frac{\pi(y_i \mid x_i)}{q_i}$ as:

$$\hat{R}(\pi) = \frac{1}{\hat{N}} \sum_{i=1}^{N} \delta_i \frac{\pi(y_i \mid x_i)}{q_i} \frac{\mathbb{1}\{o_i = 1\}}{\Pr(O = 1 \mid \delta_i)}. \tag{7}$$

Again, $\mathbb{E}_{(x_i,y_i,\delta_i)}\mathbb{E}_{o_i\sim\Pr(O|\delta_i)}\left[\delta_i\frac{\pi(y_i|x_i)}{q_i}\frac{\mathbb{1}\{o_i=1\}}{\Pr(O=1|\delta_i)}\right] = \mathbb{E}_{(x_i,y_i,\delta_i)}\delta_i\frac{\pi(y_i|x_i)}{q_i}$. Hence, the sum in the numerator of $\hat{R}(\pi)$ is, in expectation, $NR(\pi)$, while the normalizing constant $\hat{N}$ is, in expectation, N. Ratios of expectations are not equal to the expectation of a ratio, so we expect a small bias in this estimate but it is easy to show that this estimate is asymptotically consistent.

Finally consider estimating $C(\pi) = \mathbb{E}_{(x_i,y_i)}\frac{\pi(y_i|x_i)}{q_i} = 1$ as:

$$\hat{C}(\pi) = \frac{1}{\hat{N}}\sum_{i=1}^{N}\frac{\pi(y_i \mid x_i)}{q_i}\frac{\mathbb{1}\{o_i=1\}}{\Pr(O=1 \mid \delta_i)}. \tag{8}$$

Again, $\mathbb{E}_{(x_i,y_i,\delta_i)}\mathbb{E}_{o_i\sim\Pr(O|\delta_i)}\left[\frac{\pi(y_i|x_i)}{q_i}\frac{\mathbb{1}\{o_i=1\}}{\Pr(O=1|\delta_i)}\right] = \mathbb{E}_{(x_i,y_i,\delta_i)}\frac{\pi(y_i|x_i)}{q_i} = 1$. The sum in the numerator of $\hat{C}(\pi)$ is, in expectation, N as is the denominator. Again, we expect this estimate to have a small bias but to remain asymptotically consistent. The computable variant of the self-normalized IPS estimator simply uses the computable $\hat{R}(\pi)$ and $\hat{C}(\pi)$ in its definition: $\hat{R}_{snips}(\pi) = \hat{R}(\pi)/\hat{C}(\pi)$.

We use a family of new policies π_ϵ, parametrized by $0 \leq \epsilon \leq 1$ to diagnose $\hat{C}(\pi)$ and the expected behavior of IPS estimates $\hat{R}(\pi)$. The policy π_ϵ behaves like a uniformly random ranking policy with probability ϵ, and with probability $1-\epsilon$, behaves like the logging policy. Formally, for an impression with context x_i, $|\mathcal{Y}|$ possible actions (e.g., rankings of candidate products), and logged action y_i, the probability for choosing y_i under the new policy π_ϵ is:

$$\pi_\epsilon(y_i \mid x_i) = \epsilon\frac{1}{|\mathcal{Y}|} + (1-\epsilon)\pi_0(y_i \mid x_i). \tag{9}$$

As we vary ϵ away from 0, the new policy looks more different than the logging policy π_0 on the logged impressions. In Tables 2,3,4 we report $\hat{C}(\pi_\epsilon)$ and a 99% confidence interval assuming asymptotic normality, for different choices of ϵ. We also report the IPS-estimated clickthrough rates for these policies $\hat{R}(\pi_\epsilon)$, their standard error (99% CI), and finally, their self-normalized IPS-estimates [14, 4].

As we pick policies that differ from the logging policy, we see that the estimated variance of the IPS estimates (as reflected in their approximate 99% confidence intervals) increases. Moreover, the control variate $\hat{C}(\pi_\epsilon)$ is systematically under-estimated. This should caution us to not rely on a single point-estimate (e.g. only IPS or SNIPS). SNIPS can often provide a better bias-variance trade-off in these estimates, but can fail catastrophically when the variance is very high due to systematic under-estimation of $\hat{C}(\pi)$. Moreover, in these very high-variance situations (e.g. when $k \geq 3$ and $\epsilon \geq 2^{-2}$), the constructed confidence intervals are not reliable — $C(\pi_\epsilon)$ clearly does not lie in the computed intervals. Based on these sanity checks, we focus the evaluation set-up in Section 4 on the 1-slot case.

4 Benchmarking Learning Algorithms

4.1 Evaluation

Estimates based on importance sampling have considerable variance when the number of slots increases. We would thus need tens of millions of impressions to estimate the CTR of slot-filling policies with high precision. To limit the risks of people "over-fitting to the variance" by querying far away from our logging policy, we propose the following estimates for any policy:

- Report the inverse propensity scoring (IPS) [9] $\hat{R}(\pi)$ as well as the self-normalized (SN) estimate [4] for the new policy $\hat{R}(\pi)/\hat{C}(\pi)$ (self-normalized, so that learnt policies cannot cheat by not having their importance weights sum to 1);

Table 2: Diagnostics and IPS-estimated clickthrough rates for different policies π_ϵ evaluated on slices of traffic with k-slot banners, $k \in \{1, 2\}$. ϵ interpolates between the logging policy ($\epsilon = 0$) and the uniform random policy ($\epsilon = 1$). Error bars are 99% confidence intervals under a normal distribution.

#Slots	**1**			**2**		
ϵ	$\hat{C}(\pi_\epsilon)$	$\hat{R}(\pi_\epsilon) \times 10^4$	$\frac{\hat{R}(\pi_\epsilon)\times 10^4}{\hat{C}(\pi_\epsilon)}$	$\hat{C}(\pi_\epsilon)$	$\hat{R}(\pi_\epsilon) \times 10^4$	$\frac{\hat{R}(\pi_\epsilon)\times 10^4}{\hat{C}(\pi_\epsilon)}$
0	1.000±0.000	53.604±0.129	53.604	1.000±0.000	52.554±0.099	52.554
2^{-10}	1.000±0.000	53.598±0.129	53.599	1.000±0.000	52.541±0.099	52.545
2^{-9}	1.000±0.000	53.593±0.130	53.595	1.000±0.000	52.529±0.101	52.536
2^{-8}	1.000±0.000	53.582±0.131	53.585	1.000±0.000	52.503±0.107	52.517
2^{-7}	1.000±0.000	53.560±0.138	53.567	0.999±0.000	52.453±0.129	52.481
2^{-6}	1.000±0.000	53.516±0.163	53.531	0.999±0.001	52.351±0.193	52.407
2^{-5}	0.999±0.000	53.428±0.236	53.457	0.998±0.002	52.148±0.346	52.260
2^{-4}	0.999±0.001	53.251±0.416	53.311	0.996±0.003	51.742±0.671	51.965
2^{-3}	0.998±0.001	52.899±0.802	53.017	0.991±0.006	50.929±1.331	51.370
2^{-2}	0.996±0.003	52.194±1.589	52.428	0.983±0.012	49.305±2.657	50.166
2^{-1}	0.991±0.006	50.785±3.171	51.241	0.966±0.024	46.056±5.312	47.693
1	0.982±0.012	47.966±6.338	48.836	0.931±0.048	39.557±10.623	42.473

Table 3: Diagnostics for different policies π_ϵ evaluated on slices of traffic with k-slot banners, $k \in \{3, 4\}$. Error bars are 99% confidence intervals under a normal distribution.

#Slots	**3**			**4**		
ϵ	$\hat{C}(\pi_\epsilon)$	$\hat{R}(\pi_\epsilon) \times 10^4$	$\frac{\hat{R}(\pi_\epsilon)\times 10^4}{\hat{C}(\pi_\epsilon)}$	$\hat{C}(\pi_\epsilon)$	$\hat{R}(\pi_\epsilon) \times 10^4$	$\frac{\hat{R}(\pi_\epsilon)\times 10^4}{\hat{C}(\pi_\epsilon)}$
0	1.000±0.000	64.298±0.137	64.298	1.000±0.000	141.114±0.366	141.114
2^{-10}	1.000±0.000	64.296±0.148	64.305	1.000±0.001	141.065±0.366	141.082
2^{-9}	1.000±0.000	64.294±0.179	64.312	1.000±0.001	141.015±0.366	141.049
2^{-8}	0.999±0.000	64.291±0.268	64.326	1.000±0.002	140.916±0.368	140.984
2^{-7}	0.999±0.001	64.284±0.480	64.354	0.999±0.003	140.717±0.378	140.853
2^{-6}	0.998±0.001	64.269±0.930	64.410	0.998±0.006	140.320±0.413	140.590
2^{-5}	0.996±0.003	64.240±1.844	64.523	0.996±0.012	139.526±0.534	140.065
2^{-4}	0.991±0.006	64.182±3.681	64.750	0.992±0.024	137.937±0.863	139.007
2^{-3}	0.982±0.011	64.066±7.359	65.211	0.985±0.049	134.761±1.610	136.867
2^{-2}	0.965±0.023	63.834±14.716	66.157	0.969±0.097	128.407±3.161	132.484
2^{-1}	0.930±0.045	63.370±29.430	68.156	0.938±0.194	115.700±6.295	123.288
1	0.860±0.090	62.443±58.860	72.643	0.877±0.389	90.285±12.577	102.960

- Compute the standard error of the IPS estimate (appealing to asymptotic normality), and report this error as an "approximate confidence interval".

This is provided in our evaluation software alongside the dataset online. In this way, learning algorithms must reason about bias/variance explicitly to reliably achieve better estimated CTR.

Table 4: Diagnostics for different policies π_ϵ evaluated on slices of traffic with k-slot banners, $k \in \{5, 6\}$. Error bars are 99% confidence intervals under a normal distribution.

#Slots	**5**			**6**		
ϵ	$\hat{C}(\pi_\epsilon)$	$\hat{R}(\pi_\epsilon) \times 10^4$	$\frac{\hat{R}(\pi_\epsilon) \times 10^4}{\hat{C}(\pi_\epsilon)}$	$\hat{C}(\pi_\epsilon)$	$\hat{R}(\pi_\epsilon) \times 10^4$	$\frac{\hat{R}(\pi_\epsilon) \times 10^4}{\hat{C}(\pi_\epsilon)}$
0	1.000 ± 0.000	125.965 ± 0.530	125.965	1.000 ± 0.000	90.620 ± 0.206	90.620
2^{-10}	0.999 ± 0.000	125.899 ± 0.532	125.976	1.000 ± 0.000	90.579 ± 0.207	90.622
2^{-9}	0.999 ± 0.001	125.833 ± 0.538	125.988	0.999 ± 0.000	90.537 ± 0.210	90.625
2^{-8}	0.998 ± 0.001	125.702 ± 0.563	126.011	0.998 ± 0.000	90.454 ± 0.222	90.629
2^{-7}	0.995 ± 0.001	125.439 ± 0.653	126.057	0.996 ± 0.001	90.289 ± 0.264	90.638
2^{-6}	0.990 ± 0.002	124.913 ± 0.931	126.149	0.992 ± 0.001	89.957 ± 0.389	90.657
2^{-5}	0.980 ± 0.004	123.861 ± 1.624	126.337	0.985 ± 0.002	89.293 ± 0.691	90.694
2^{-4}	0.961 ± 0.007	121.756 ± 3.119	126.725	0.969 ± 0.004	87.967 ± 1.336	90.769
2^{-3}	0.922 ± 0.014	117.548 ± 6.172	127.549	0.938 ± 0.008	85.313 ± 2.649	90.928
2^{-2}	0.843 ± 0.029	109.131 ± 12.314	129.428	0.877 ± 0.017	80.006 ± 5.287	91.279
2^{-1}	0.686 ± 0.057	92.298 ± 24.613	134.475	0.753 ± 0.033	69.392 ± 10.568	92.154
1	0.373 ± 0.115	58.631 ± 49.221	157.307	0.506 ± 0.066	48.164 ± 21.135	95.185

4.2 Methods

Consider a 1-slot banner filling task defined using our dataset. This $21M$ slice of traffic can be modeled as a logged contextual bandit problem with a small number of arms. This slice is further randomly divided into a $33 - 33 - 33\%$ train-validate-test split. The following methods are benchmarked in the code accompanying this dataset release. All these methods use a linear policy class $\pi \in \Pi_{lin}$ to map $x \mapsto y$ (i.e., score candidates using a linear scorer $w \cdot \phi(c, p)$), but differ in their training objectives. Their hyper-parameters are chosen to maximize $\hat{R}(\pi)$ on the validation set and their test-set estimates are reported in Table 5.

1. **Random**: A policy that picks $p \in P_c$ uniformly at random to display.
2. **Regression**: A reduction to supervised learning that predicts δ for every candidate action. The number of training epochs (ranging from $1 \ldots 40$), regularization for Lasso (ranging from $10^{-8} \ldots 10^{-4}$), and learning rate for SGD $(0.1, 1, 10)$ are the hyper-parameters.
3. **IPS**: Directly optimizes $\hat{R}(\pi)$ evaluated on the *training* split. This implementation uses a reduction to weighted one-against-all multi-class classification as employed in [3]. The hyper-parameters are the same as in the Regression approach.
4. **DRO** [3]: Combines the Regression method with IPS using the doubly robust estimator to perform policy optimization. Again uses a reduction to weighted one-against-all multi-class classification, and uses the same set of hyper-parameters.
5. **POEM** [2]: Directly trains a stochastic policy following the counterfactual risk minimization principle, thus reasoning about differences in the variance of the IPS estimate $\hat{R}(\pi)$. Hyper-parameters are variance regularization, $L2$ regularization, propensity clipping and number of training epochs.

The results of the learning experiments are summarized in Table 5. For more details and the specifics of the experiment setup, visit the dataset website. Differences in Random and π_0 numbers compared to Table 2

Approach	Test set estimates $\hat{R}(\pi_\epsilon) \times 10^4$	$\hat{R}(\pi_\epsilon) \times 10^4 / \hat{C}(\pi_\epsilon)$	$\hat{C}(\pi_\epsilon)$
Random	44.676 ± 2.112	45.446 ± 0.001	0.983 ± 0.021
π_0	53.540 ± 0.224	53.540 ± 0.000	1.000 ± 0.000
Regression	48.353 ± 3.253	48.162 ± 0.001	1.004 ± 0.041
IPS	54.125 ± 2.517	53.672 ± 0.001	1.008 ± 0.016
DRO	57.356 ± 14.008	57.086 ± 0.005	1.005 ± 0.025
POEM	58.040 ± 3.407	57.480 ± 0.001	1.010 ± 0.018

Table 5: Test set performance of policies learnt using different counterfactual learning baselines. Errors bars are 99% confidence intervals under a normal distribution. Confidence interval for SNIPS is constructed using the delta method [12].

are because they are computed on a 33% subset — we do expect their confidence intervals to overlap. We see that the *Regression* approach, which loosely corresponds to predicting CTR for each candidate using supervised machine learning, can be substantially improved using many recent off-policy learning algorithms that effectively use the logged propensities. We also note that very limited hyper-parameter tuning was performed for methods like *POEM* and *DRO* — for instance, POEM can conceivably be improved by employing the doubly robust estimator. We leave such algorithm-tuning to future work.

5 Conclusions

In this paper, we have introduced a standardized test-bed to systematically investigate off-policy learning algorithms using real-word data. We presented this test-bed, the sanity checks we ran to ensure its validity, and showed results comparing state-of-the-art off-policy learning methods (doubly robust optimization [3] and POEM [2]) to regression baselines on a 1-slot banner filling task. Our results show experimental evidence that recent off-policy learning methods can improve upon state-of-the-art supervised learning techniques on a large-scale real-world data set.

These results we presented are for the 1-slot banner filling tasks. There are several dimensions in setting up challenging, interesting, relevant off-policy learning problems on the data collected for future work.

Size of the action space: Increase the size of the action space, i.e. of the number of slots in the banner.

Feedback granularity: We can use global feedback (was there a click somewhere in the banner), or per item feedback (which item in the banner was clicked).

Contextualization: We can learn a separate model for each banner type or learn a contextualized model across multiple banner types.

Acknowledgments

We thank Alexandre Gilotte and Thomas Nedelec at Criteo for their help in creating the dataset. This work was funded in part through NSF Awards IIS-1247637, IIS-1615706, IIS-1513692.

References

[1] L. Bottou, J. Peters, J. Q. Candela, D. X. Charles, M. Chickering, E. Portugaly, D. Ray, P. Y. Simard, and E. Snelson, "Counterfactual reasoning and learning systems: the example of computational advertising.," *Journal of Machine Learning Research*, pp. 3207–3260, 2013.

[2] A. Swaminathan and T. Joachims, "Batch learning from logged bandit feedback through counterfactual risk minimization," *Journal of Machine Learning Research*, pp. 1731–1755, 2015.

[3] M. Dudík, J. Langford, and L. Li, "Doubly robust policy evaluation and learning," in *ICML*, pp. 1097–1104, 2011.

[4] A. Swaminathan and T. Joachims, "The self-normalized estimator for counterfactual learning," in *NIPS*, pp. 3231–3239, 2015.

[5] O. Chapelle, E. Manavoglu, and R. Rosales, "Simple and scalable response prediction for display advertising," *Transactions on Intelligent Systems and Technology*, p. Article 61, 2014.

[6] F. Vasile, D. Lefortier, and O. Chapelle, "Cost-sensitive learning for utility optimization in online advertising auctions," *arXiv preprint arXiv:1603.03713*, 2016.

[7] A. Vorobev, D. Lefortier, G. Gusev, and P. Serdyukov, "Gathering additional feedback on search results by multi-armed bandits with respect to production ranking," in *WWW*, pp. 1177–1187, 2015.

[8] D. Lefortier, P. Serdyukov, and M. de Rijke, "Online exploration for detecting shifts in fresh intent," in *CIKM*, pp. 589–598, 2014.

[9] L. Li, W. Chu, J. Langford, and X. Wang, "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms," in *WSDM*, pp. 297–306, 2011.

[10] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, *et al.*, "Ad click prediction: a view from the trenches," in *KDD*, pp. 1222–1230, 2013.

[11] P. R. Rosenbaum and D. B. Rubin, "The central role of the propensity score in observational studies for causal effects," *Biometrika*, pp. 41–55, 1983.

[12] A. B. Owen, *Monte Carlo theory, methods and examples*. 2013.

[13] L. Li, S. Chen, J. Kleban, and A. Gupta, "Counterfactual estimation and optimization of click metrics in search engines: A case study," in *WWW*, pp. 929–934, 2015.

[14] T. Hesterberg, "Weighted average importance sampling and defensive mixture distributions," *Technometrics*, pp. 185–194, 1995.