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

Damien Lefortier∗
Facebook & University of Amsterdam
dlefortier@fb.com

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

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

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: \( \pi_0, \{x_i, y_i, \delta_i\}_{i=1}^n \). \( \pi_0 \) 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 \( \delta \) is a number encoding the observed online metric for the output that was predicted;
2. Produce as output: \( \pi \), a new policy that maps \( x \mapsto y \); and
3. Such that \( \pi \) will perform well (according to the metric \( \delta \)) if it were deployed online.

We elaborate on the definitions of \( x, y, \delta, \pi_0 \) 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. 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.

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 \(\pi_0\) 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):

\[
P(slot1 = p) = \frac{f_p}{\sum_{p' \in P_c} f_{p'}} \quad P(slot2 = p' | slot1 = p) = \frac{f_{p'}}{\sum_{p'' \in P_c \land p'' \neq p} f_{p''}}, \quad \text{etc.} \tag{1}
\]

The propensity of a chosen banner ad \(\langle p_1, p_2, \ldots \rangle\) is \(P(slot1 = p_1) \cdot P(slot2 = p_2 | slot1 = p_1) \cdot \ldots\). 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 \(\pi_0\) 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 \(\pi_0\) 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:

```plaintext
example ${exID}: ${hashID} ${wasAdClicked} ${propensity} ${nbSlots} ${nbCandidates} ${displayFeat1}:${v1} ... ${wasProduct1Clicked} exid:${exID} ${productFeat1}://${v1} ...```

3
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 $\{nbCandidates\}$ candidates correspond to the displayed products ordered by position. 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 $(1, 2, 3, 5)$ 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, $35M$ for 2-slot, $23M$ for 3-slot, $7M$ for 4-slot, $3M$ for 5-slot and $14M$ 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/](http://www.cs.cornell.edu/~adith/Criteo/)

### 3 Sanity Checks

If the propensities are correctly logged, then the expected importance weight should be 1 for any candidate policy. Formally, we have the following:

$$\frac{1}{\sum_{i=1}^{N} SW_i} \sum_{i=1}^{N} SW_i \cdot \frac{h(a_{0i} | c_i, p_i)}{h_0(a_{0i} | c_i, p_i)} \rightarrow 1$$

where $SW_i$ is the sub-sampling weight (10 for negative impressions and 1 otherwise as discussed above), $a_{0i}$ is the action chosen by the logging policy, $h_0(a_{0i} | c_i, p_i)$ is the propensity of $a_{0i}$ under the logging policy given context and candidate products, while $h(a_{0i} | c, p)$ is the same propensity under the candidate policy. A simple but useful sanity check derived from the equation above and which does not require training any candidate policy is to use a uniform random policy. In this case, we obtain:

$$\frac{1}{\sum_{i=1}^{N} SW_i} \sum_{i=1}^{N} SW_i \cdot \frac{1}{h_0(a_{0i} | c_i, p_i)} \cdot \frac{1}{M} \rightarrow 1,$$
where \( M \) is the cardinality of \( P_c \).

In Table 1, we show average importance weights on 10M 1-slot banners computed using 3-fold cross-validation for different candidate policies (uniform and regression; see below for details). Policies prefixed by "S-" are stochastic, while the ones prefixed by "D-" are deterministic. Deterministic uniform policy uniformly selects a random candidate among all with propensity 1, and assigns 0 propensity for others. We see that the results are very close to 1 as expected even without using the full dataset. Then, in Table 2, we report average importance weights on multi-slot banners under the stochastic uniform policy. The confidence intervals are computed as two standard errors here.

Table 1: Average importance weights (IW) on 10M 1-slot banners for different candidate policies. Policies prefixed by "S-" are stochastic, while the ones prefixed by "D-" are deterministic.

<table>
<thead>
<tr>
<th>Policy</th>
<th>S-Uniform</th>
<th>S-Regression</th>
<th>D-Uniform</th>
<th>D-Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>IW</td>
<td>0.98 ± 0.04</td>
<td>0.98 ± 0.04</td>
<td>0.96 ± 0.08</td>
<td>0.99 ± 0.01</td>
</tr>
</tbody>
</table>

Table 2: Average importance weights (IW) on multi-slot banners for the stochastic uniform policy. See below why there is no CI.

<table>
<thead>
<tr>
<th>#Slots</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>IW</td>
<td>0.93 ± 0.04</td>
<td>0.86 ± 0.07</td>
<td>0.88 ± 0.30</td>
<td>0.37 ± 0.08</td>
<td>0.51 ± 0.05</td>
</tr>
</tbody>
</table>

The estimated confidence intervals we computed for values in Table 2 are smaller than expected, especially for 5 and 6-slot banners. We conjecture that this is due to violations of the normality assumption underlying the standard-error confidence intervals. In particular, note that the uniform policy assigns very different probabilities to actions compared to the logging policy, since the logging policy strongly favors the best products (using a temperature parameter). This means that the propensity of some actions that have substantial probability under the uniform policy is very small, making these actions extremely rare in the dataset and their weights extremely large. Since we have far fewer impressions than the average number of actions for 5-slots and 6-slots banners, these actions can become unpredictably rare and thus lead to small average importance weights with a misleadingly small variance estimate. We reproduced this problem also on 2-slot banners when using a smaller number of impressions. Overall, we conjecture that the task with 5 or more slots are quite challenging, and that reasoning about variance and its estimate will be of prime importance.
4 Benchmarking Learning Algorithms

4.1 Evaluation

Estimates based on importance sampling have considerable variance when the number of slots increases. Indeed, with 6 slots, the mode, i.e., the most probable action, of \( \pi_0 \) has a propensity of only \( 2 \times 10^{-4} \) in our dataset. 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 estimate for any policy:

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

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. Click/No-click on the banner is encoded as \( \delta = 0/1 \), and The following methods are efficiently implemented in the code accompanying this dataset release, and we report their performance on a random 33\% train-validate-test split of the data below. 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.

1. **Random**: A policy that picks \( p \in P_c \) uniformly at random to display.
2. **Regression**: A reduction to supervised learning that predicts \( \delta \) for every candidate action. The number of training epochs and regression hyper-parameters are validated by evaluating the IPS estimate on the validation split.
3. **IPS**: Directly optimizes the inverse propensity scoring estimator \cite{9} evaluated on the training split. The number of training epochs is chosen using the validation split.
4. **DRO** \cite{3}: Combines the regression method with inverse propensity scoring using the doubly robust estimator to perform policy optimization via a reduction to weighted multi-class classification.
5. **POEM** \cite{2}: Directly trains a stochastic policy following the counterfactual risk minimization principle, thus reasoning about differences in the variance of the risk estimates. Hyper-parameters like variance regularization, propensity clipping and training epochs are chosen using the validation split.

The results of the learning experiments are summarized in Table 3. For more details and the specifics of the experiment setup, visit the dataset website.

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.

### References


<table>
<thead>
<tr>
<th>Approach</th>
<th>Test Set Performance ($\times 10^4$)</th>
<th>SNIPS</th>
<th>IPS</th>
<th>2*StdErr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logging $\pi_0$</td>
<td>53.540</td>
<td>53.540</td>
<td>0.087</td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>45.446</td>
<td>44.676</td>
<td>0.819</td>
<td></td>
</tr>
<tr>
<td>Regression</td>
<td>48.162</td>
<td>48.353</td>
<td>1.261</td>
<td></td>
</tr>
<tr>
<td>IPS</td>
<td>53.672</td>
<td>54.125</td>
<td>0.975</td>
<td></td>
</tr>
<tr>
<td>DRO</td>
<td>57.086</td>
<td>57.356</td>
<td>5.429</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Test set performance of baseline counterfactual learning methods on the 1-slot banner filling task. Performance estimates are computed using the IPS and the Self-Normalized IPS estimator [4] on the test set.

