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

Damien Lefortier1,2, Xiaotao Gu3, Adith Swaminathan4, Thorsten Joachims4, and Maarten de Rijke2

1 Facebook, Inc.
2 University of Amsterdam
3 Tsinghua University, Beijing, China
4 Cornell University, Ithaca, NY

Motivation

The ability to do effective offline off-policy learning that reliably optimizes online metrics would revolutionize the process of building better interactive systems, such as search engines and recommendation systems for e-commerce, computational advertising, ...

Recent approaches for off-policy evaluation and learning in these settings appear promising [1,2,3]. We provide a large-scale dataset and a standardized test-bed to systematically investigate off-policy learning algorithms using data from Criteo (a leader in the display advertising space).

Off-policy learning algorithm

- Take as input: \( \{ \pi_c(x, y, \delta)^{exID} \} \), where \( \pi_c \) 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.
- Produce as output: \( \pi \), a new policy that maps \( x \rightarrow y \).
- Such that \( \pi \) will perform well (according to the metric \( \delta \)) if it were deployed online.

Goals of the test-bed

- Good policy classes \( \pi \in \Pi \) for the specified task \( x \rightarrow y \).
- Good regularization mechanisms and training objectives for off-policy learning.
- Good model selection procedures (analogous to cross-validation for supervised learning).
- Algorithms that can scale to massive amounts of data.

Dataset

We create our test-bed using data from display advertising and consider the problem of filling a banner ad with an aggregate of multiple products the user may want to purchase (knowing the banner type). The goal is to maximize the number of clicks.

For each user impression, denote a user context (specifying banner type and look and feel) by \( c \), number of slots in the banner type by \( L \), and the candidate pool of products \( p \) by \( P \). Each context \( c \) and product \( p \) pair is described by features \( \phi(c, p) \). The input \( x \) to the system encodes \( c, P, \{\phi(c, p), p \in P\} \). The logging policy \( \pi_c \) stochastically selects products to construct a banner by first computing non-negative scores \( f_p \) for all candidate products \( p \in P \), and using a Plackett-Luce ranking model:

\[
P(\text{slot1} = p) = \frac{f_p}{\sum_{p \in P} f_p} \quad P(\text{slot2} = p' \mid \text{slot1} = p) = \frac{f_{p'}}{\sum_{p' \in P \setminus \{p\}} f_{p'}} \quad [\text{...}]
\]

The propensity of a chosen banner \( (p_1, p_2, \ldots) \) is \( P(\text{slot1} = p_1) = P(\text{slot2} = p_2) = P(\text{slot3} = p_3) \ldots \). With these propensities in hand, we can counterfactually evaluate any banner-filling policy in an unbiased way using inverse propensity scoring [4].

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

\[
\text{sample} \{\text{exID}\} \quad \text{context} \{\phi(c, p)\} \quad \text{propensity} \{\pi_c(x, y, \delta)^{exID}\} \quad \text{clicks} \{\text{slot1} = p_1, \text{slot2} = p_2, \ldots\} \quad \text{propensity} \{\pi_c(x, y, \delta)^{exID}\} \quad \text{estimated click count} \{\text{slot1} = p_1, \text{slot2} = p_2, \ldots\}
\]


Sanity Checks

If the propensities are correctly logged, then importance weights should on average be close to 1 for any candidate policy. Formally, we have the following:

\[
\frac{1}{\sum_{c,p} SW_{c,p}} \sum_{c,p} SW \cdot \left[ h_{\text{logging}}(a_{\text{logging}}(c, p)) \rightarrow 1 \right]
\]

where \( SW \) is the sub-sampling weight (10 for negative impressions and 1 otherwise as discussed above), \( a_{\text{logging}} \) is the action chosen by the logging policy, \( h_{\text{logging}}(a_{\text{logging}}(c, p)) \) is its propensity under the logging policy given context and candidate products, while \( h(a_{\text{logging}}(c, p)) \) is the same propensity under the candidate policy. We run this sanity check for the following candidate policies: deterministic and stochastic Uniform, deterministic and stochastic Regression, and POEM (see below). See paper for details.

Benchmark Learning Algorithms

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 0.02% 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) and self-normalized (SN) estimate [5] for the new policy (self-normalized, so that learnt policies cannot cheat by not having their importance weights sum to 1);
- Compute approximate confidence intervals, subtract two std. dev. from the SN estimate, and this is the number we report.

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

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. The following row represents the code accompanying this dataset release, and we report their performance on a random 50 – 50 train-test split of the data. All these methods use a linear policy class \( \pi \in \Pi_0 \) to map \( x \rightarrow y \), but differ in their training objectives.

Results for 1-slot task

- Random: A policy that picks \( p \) uniformly at random to display, has an estimated (lower bound) on CTR \( 10^3 \) of 30.829.
- Regression: A reduction to supervised learning that predicts \( \delta \) for every candidate action, achieves a conservative estimated CTR \( 10^3 \) of 48.907.
- Doubly Robust Optimization [3]: Uses a regression oracle as well as inverse propensity scoring to perform policy optimization via a reduction to weighted multi-class classification, achieves a conservative estimated CTR \( 10^3 \) of 50.339.
- POEM [2]: Learns a stochastic policy using a linear scorer \( \phi(c, p) \) while additionally reasoning about variance of off-policy estimates. This achieves estimated CTR \( 10^3 \) of 53.534.


Conclusion

Remember:

- We have introduced a standardized test-bed to systematically investigate off-policy learning algorithms using real-world data.
- We described sanity checks that we used on our dataset to ensure its validity.
- We show experimental evidence that recent off-policy learning methods can improve upon state-of-the-art supervised learning techniques on large-scale real-world data.

Our dataset & evaluation software are public (see paper for the links).