Offline Multi-Objective Optimization (OMOO) in Search Page Layout Optimization Using Off-policy Evaluation

Pratik Lahiri[1], Zhou Qin[1] and Wenyang Liu[1]

[1]*Amazon. 550 Terry Ave N, Seattle, Washington 98109*

Abstract

E-commerce stores typically test changes to ranking algorithms through rigorous A/B testing which requires a change to satisfy some predefined success criteria on multiple metrics. This problem of simultaneously optimization of multiple metrics is multi-objective-optimization (MOO). A common method for MOO is to choose a set of weights to scalarize the multiple metrics into one ranking objective. However, in practical settings, rather than simply improving all metrics, the experimenter might be interested in improving a few metrics significantly with negligible trade-off for others. We can refer to such requirements as a desired policy. An experimenter chooses weights to scalarize the objective such that it best approximates the desired policy. Repeated A/B testing to arrive at a set of weights that approximates the desired policy well enough is costly and inefficient. This problem lends itself to off-policy evaluation methods. In this paper, we develop a framework for approximate Offline Multi-Objective Optimization for $\pi_{,w}$ explore-exploit policies where a small ($\epsilon = 1 - 5\%$) traffic is reserved for exploration while the majority is served by exploiting the current best arm under the policy. Further, the metrics being optimized in our use case are highly skewed with zero-inflation. We then develop a simulator/ reward vector generator using a neural network that learns a distribution of rewards for a given context from exploration data. We empirically show that this reward vector generator is an unbiased estimator of such policies. Finally, we demonstrate empirical data that this estimator is able to correctly predict the order of treatments from an A/B test in an e-commerce page layout ranker across 4 different metrics.

Keywords

Multio-objective Optimization, Off-policy Evaluation

1. Introduction

Page layouts for e-commerce search determine the different kinds of content displayed on the search page. These layouts are aimed at providing excellent shopping experience to customers and improving various metrics such as clicks, purchases etc. Typically, these page layouts vary the number and type of item slots. Determining the most appropriate page layout can be thought of as a ranking problem which takes as input some customer and query features and chooses a layout from a set of well designed candidates [1].

To continuously learn from customer behaviour, explore-exploit policies such as contextual bandits are effective and widely used in production systems [2]. A page layout ranker (PLR)

SIGIR'24: Workshop on eCommerce, July 18, 2024, Washington DC

lahirip@amazon.com (P. Lahiri); qinzhouhit@hotmail.com (Z. Qin); lwenyang@amazon.com (W. Liu)
https://zhouqin.info (Z. Qin)
0000-0002-9664-4948 (P. Lahiri); 0000-0002-1641-772X (Z. Qin)

can be trained on contextual features to optimize for one or more metrics. Typically, in an e-commerce system, we are interested in optimizing multiple metrics- multi objective optimization (MOO). To make ranking decisions, we can then choose a set of weights to scalarize the multiple metrics into one ranking objective. To launch any changes to the ranking objective we follow the standard A/B testing approach. We embed this ranking objective in the PLR and evaluate it on a segment of live traffic for some sufficiently long time-period. If the new ranking objective outperforms the current one, it is accepted and either stored for future use or deployed right after. This approach has three major limitations. Firstly, each iteration of A/B testing takes a long time and many iterations are needed to find an ideal choice. Secondly, it is expensive; it requires substantial engineering effort (in deploying each treatment to the live traffic). Lastly and most importantly, it can have negative customer experience impact.

Given the multi-dimensional nature of MOO and its post-facto nature, it is desirable to have some systematic way of selecting a few good choices of ranking objectives that can then be deployed online for final A/B testing. Using Off Policy Evaluation Methods for solving MOO has been described in IMO^3[3]. However, in that paper the authors devised a method to learn the preferences of the experimenter with regards to tradeoffs between metrics and assumed that a good enough off policy estimator was available. In this work, we tackle the problem of devising a good enough estimator when the reward distributions are skewed, zero-inflated and multi-modal as in our applicationand formulate a framework for applying this in production.

2. Problem Formulation

In this section, we develop our notation and formulate the MOO problem as a contextual bandit problem with delayed vector-valued rewards.

Let us consider a decision-maker (in this case, PLR along with its prediction module), which interacts with an environment (the customer population) over a large number of time-steps $t = 1, 2, \dots, T$, with the goal of optimizing a fixed number of distinct objectives, say K. The interaction of the decision-maker with the environment is modeled as follows.

1. The environment reveals a (random) context c_t to the decision-maker as well as a fixed set of possible actions $A(c_t)$. Each such action is also referred to as an *arm permissible for context* c_t. Let us denote the set of all possible contexts by C.
2. The decision-maker chooses an arm a_t from $A(c_t)$ according to some behavioral policy π.
3. Our policy is policy space restricted to all $\pi_{\epsilon,w}$-policies where $\pi_{\epsilon,w}$ is defined as ($\pi_{\epsilon,w}$-Policy) For any $\epsilon[0, 1]$ and any $w \in \Delta_K$, by $\pi_{\epsilon,w} = \pi_{\epsilon,w}(\cdot|c_t, \phi_t)$, we denote a policy that, given a context c_t,
   i) with probability ϵ, chooses an arm from $A(c_t)$ randomly,
   ii) with probability $1 - \epsilon$, chooses an arm with the highest predicted scalar reward $w^T r_t(c_t, a, \phi_t)$.

   . Let Π be the set of all behavioral policies.
4. Once a_t is chosen, the environment generates a (random) vector-valued reward $r_t = (r_t^{(1)}, r_t^{(2)}, \dots, r_t^{(K)})^T \in R^K$ that is revealed to the decision-maker after a delay such as in the batched bandits setting[4].

Below are our assumptions for our problem setup.

1. **Assumption 1**: The contexts $\{(c_t)\}_{t=1}^{T}$ are drawn in an i.i.d (independent and identically distributed) manner from some fixed/time-invariant unknown distribution $D_c(\cdot)$.
2. **Assumption 2**: Given c_t and decision-maker's chosen action $a_t \in A(c_t)$, the reward r_t is drawn from some fixed/time-invariant unknown distribution $D_r(\cdot|c_t, a_t)$.[1]
3. **Assumption 3**: The reward-vector is almost-surely bounded, i.e., $\|r_t\|_2 \leq M < \infty$ for all $t = 1, 2, \ldots, T$.[2] Here, each $r_t^{(j)}$ denotes the reward obtained for the j^{th}-objective at time t.

In T rounds, the expected average reward of the decision-maker in objective $j \in [K]$[3], if they follow a behavioral policy π, is given by,

$$V^{(j)}(\pi; T)\frac{1}{T}[\sum_{t=1}^{T} r_t^{(j)}]. \tag{1}$$

The goal of the decision-maker is to learn a behavioral policy that optimizes over all the K objectives simultaneously, i.e., the decision-maker seeks a policy $\pi^\star$ that satisfies,

$$\pi^\star \in argmax_{\pi\in\Pi} \underbrace{(V^{(1)}(\pi; T), V^{(2)}(\pi; T), \ldots, V^{(K)}(\pi; T))^T}_{V(\pi;T)^T}. \tag{2}$$

With slight abuse of notation, assume that Π is the set of all policies that at time t take action by using the current context c_t and some statistic ϕ_t that is a summary/compression of the decision-maker's observations in the previous time-steps $1, 2, \ldots, t-1$. Also, we will call $V(\pi; T)$ as the value-function of policy π when the horizon is T time-steps.

If we consider a weight-vector

$$w \in \Delta_K\{w \in R^K : w \geq 0, w^T 1 = \sum_{j=1}^{K} w^{(j)} = 1\}, \tag{3}$$

then from the definition of Pareto-optimality, it is clear that a policy π that solves the scalarized optimization problem,

$$\pi^\star \in argmax_{\pi\in\Pi} w^T V(\pi; T), \tag{4}$$

is a Pareto-optimal policy. Thanks to linearity of expectation, this is equivalent to optimizing in a contextual bandit setting with delayed scalar rewards of the form $w^T r_t$.

In OPE, a target-policy π is evaluated using experience-data collected from some logging-policy μ that should preferably satisfy the below coverage(/absolute-continuity) criterion.
Assumption 4: For any context c and corresponding permissible action $a \in A(c)$, if $(\pi(c))(a) > 0$, then $(\mu(c))(a) > 0$.

Any soft explore-exploit based policy, μ, will satisfy Assumption 4 for all policies π. (However, this does not mean that it is a good candidate for generating experience-data for OPE).

Now, in principle, the general scheme to explore all choices of w could work as follows

[1]Here, we do not assume that the components of r_t are conditionally-independent given c_t and a_t.
[2]This assumption is useful when $T = \infty$.
[3]For all $b \in N$, we define $[b]\{1, 2, \ldots, b\}$.

1. Sample a weight-vector w from the weight-simplex Δ_K.
2. Find the estimate $V(\pi_{\epsilon,w};T)$ according to one of the algorithms presented in Section 3.
3. Save $(\epsilon, w, V(\pi_{\epsilon,w};T))$.

We refer to this scheme as *pseudo-Pareto-front generation.* The usage of the term *pseudo* highlights that we are only exploring $\pi_{,w}$-policies. One way to explore the set $\bigcup_{w\in\Delta_K}\{V(\pi_{,w};T)\}$ rather quickly would be to simulate simultaneous (possibly biased) random-walks in the weight-simplex Δ_K through multiple worker-nodes. We do not address methods of random walks in this paper.

3. Offline Policy Evaluation (OPE) Methods

Suppose we are given a $\pi_{\epsilon,w}$-policy whose value-function we would like to estimate using the experience data, $S_{explore}$. There are off-the-shelf algorithms, mainly including Simulation-method/Reward-vector-generator (described here), Direct-method, Vanilla Estimator [5], Inverse-Propensity-Score (IPS) Based Estimator [6], Doubly-Robust (DR) Estimator [7].

3.1. Our Recommendation of OPE Method

Distributions D_c and $\{D_r(\cdot|*)\}$ on contexts and rewards can change over time. But they should remain more or less time-invariant for some small time-frame. Performing off-policy evaluation on fresh experience-data helps ensure that value-function estimates of a given policy are reasonably accurate. Using experience-data from far off in the past may produce bad estimates if there has been a distribution-shift in the contexts and/or rewards (seasonal variations in customer's preferences of a specific locale, change in national economies etc.). In the limit of large-data, i.e., $|S_{exp}| = n \to \infty$, the Vanilla, IPS, and DR estimators provide $(1-\delta)$-type probabilistic guarantees for reliable accuracy. Amongst these algorithms, DR is preferable. However, its accuracy guarantees in the case of a finite data-set and for non-stationary policies (such as in our case) are different. Even for stationary policies, the accuracy of the DR estimator depends on the the accuracies of the IPS and the DM parts. Most of the work on OPE has been focused on improving the IPS part [8, 9]. One work [10] addressed this gap by designing the loss function of the DM part of the DR estimator to minimize the variance of the DR estimator. However, they assume that the target policy is known beforhand to design the loss function.

To use the DR estimator, either we require a mean-reward-vector predictor

$$\overline{R} : (c, a) \mapsto \overline{r} \qquad (c \in C, a \in A(c), \overline{r} \in R^K). \tag{5}$$

or a (possibly random) reward-vector generator,

$$R : (c, a) \mapsto r \qquad (c \in C, a \in A(c), r \in R^K). \tag{6}$$

Typically, the mean-reward-vector predictor is obtained by taking a dataset other than S_{exp} and then performing supervised learning on it. The performance of DR estimator relies greatly on the performance of the mean-reward-vector predictor (or the reward-vector generator). In succeeding sections we empirically show that the reward-vector generator approach achieves a lower bias than the mean-reward-vector predictor and describe how we construct the reward-vector-generator. We also emperically estimate its variance and compare against the variance of SNIPS [9].

4. Datasets

We used two datasets for our experiments. The **small dataset** we used for our experiments was obtained by 5% random sampling of `PLR model training` dataset from a policy deployed in production. This base data-set consists of 25MM samples and has a total of 13 fields. The **online experiment dataset** we used for our experiments was obtained by 10% random sampling of `PLR model training` datasets for treatments during an online A/B test of different policies. We used one policy as the logging policy and two (T1, T2) for testing.

The first eight features make up the context, the ninth feature is the action, and the remaining four are the corresponding rewards which are Reward 1, Reward 2, Reward 3, and Reward 4 respectively. In practice, there is a tension between these metrics.

We split the training data-sets into **train**, **validation**, and **test** data-sets using a split of 70%, 15%, and 15% respectively. A few important statistics of the unlogged versions of the target-labels are provided in Table 6 where we observe the below three properties.

1. The metrics Reward 1 (log), Reward 2 (log), and Reward 3 (log) are **zero-inflated**.
2. Conditioned on being non-zero, the histograms of Reward 1 (log) and Reward 3 (log) are **unimodal**.[4]
3. On the other hand, the histogram of Reward 2 (log) conditioned on being non-zero is **bimodal**.

5. Reward-Vector-Generator Model

We will assume a simple reward-vector generation model that incorporates the three properties we mentioned above. Let us denote the context and action random variables at time t by C_t and A_t respectively. For the random variables that represent the metrics Reward 1, Reward 2, Reward 3, and Reward 4 at time t, we will use $R_t^{(1)}$, $R_t^{(2)}$, $R_t^{(3)}$, and $R_t^{(4)}$ respectively.

- **Conditional Distribution of $R_t^{(1)}$ and $R_t^{(3)}$**: $R_t^{(1)}$ and $R_t^{(3)}$ respectively correspond to Reward 1 and Reward 3. We assume that the conditional distribution of $R_t^{(j)}$ $(j = 1, 3)$ is given by

$$d\mathbb{P}^{(j)}(r|c,a) = \begin{cases} p_0^j(c,a), & r = 0, \\ p_1^j(c,a) f_{\widetilde{N}_1}^{(j)}(r|c,a), & r > 0, \end{cases} \tag{7}$$

where $p_0^j(c,a) + p_1^j(c,a) = 1$ and $f_{\widetilde{N}_1^{(j)}}(\cdot|c,a)$ represents a pdf obtained from trimming some gaussian-pdf $f_{N_1^{(j)}}(\cdot|c,a)$ to the positive real line $(0,\infty)$. If we respectively denote

[4]Importantly, the histogram of log(Reward 1) is significantly skewed to the right.

the mean and standard-deviation of $N_1^{(j)}$ by $m_1^{(j)}$ and $\sigma_1^{(j)}$, then

$$f_{N_1^{(j)}}(r|c,a) = \frac{1}{\sqrt{2\pi}\sigma_1^{(j)}(c,a)} \exp(-\frac{(r - m_1^{(j)}(c,a))^2}{2(\sigma_1^{(j)}(c,a))^2}). \tag{8}$$

Intuitively speaking, we are assuming that in response to a (context, action) pair, (c,a), the nature conducts a Bernoulli trial with (failure, success) probabilities, $(p_0^{(j)}(c,a), p_1^{(j)}(c,a))$. If the result of this trial is a success, the nature draws a sample using a pdf that is obtained from trimming some gaussian pdf to the positive real line. In the case of failure, the reward assumes the zero value.

- **Conditional Distribution of $R_t^{(2)}$**: $R_t^{(2)}$ corresponds to Reward 2. We assume that the conditional distribution of $R_t^{(2)}$ is given by

$$d\mathbb{P}^{(2)}(r|c,a) = \begin{cases} p_0^{(2)}(c,a), & r = 0, \\ p_1^{(2)}(c,a) f_{\tilde{N}_1}^{(2)}(r|c,a), & r > 0, \\ p_2^{(2)}(c,a) f_{\tilde{N}_2}^{(2)}(r|c,a), & r < 0. \end{cases} \tag{9}$$

Here, $p_0^{(2)}(c,a) + p_1^{(2)}(c,a) + p_2^{(2)}(c,a) = 1$. The symbols $f_{\tilde{N}_1^{(2)}}(\cdot|c,a)$, $f_{\tilde{N}_2^{(2)}}(\cdot|c,a)$ represent pdfs that are obtained from trimming some gaussian pdfs, $f_{N_1^{(2)}}, f_{N_2^{(2)}}$, to the positive and negative real lines respectively. The intuition behind this conditional distribution is similar to above except that we have three categories, category-0 for zero-inflation, and categories 1 and 2 for the right and left modes.

- **Conditional Distribution of $R_t^{(4)}$**: $R_t^{(4)}$ corresponds to Reward 4 which is a Bernoulli reward. Therefore, the model of a Bernoulli trial with (context, action)-dependent failure and success probabilities suffices, i.e.,

$$d\mathbb{P}^{(4)}(r|c,a) = \begin{cases} p_0^{(4)}(c,a), & r = 0, \\ p_1^{(4)}(c,a), & r = 1. \end{cases} \tag{10}$$

Joint Conditional Distribution of Reward Vector R_t: We assume that the scalar rewards are conditionally independent given a (context, action) pair. Therefore, the joint conditional distribution is given by

$$d\mathbb{P}(r|c,a) = \Pi_{j=1}^4 d\mathbb{P}^{(j)}(r^{(j)}|c,a). \tag{11}$$

This gives us the below negative log-likelihood,

$$-\log d\mathbb{P}(r|c,a) = -\sum_{j=1}^{4} \log d\mathbb{P}^{(j)}(r^{(j)}|c,a). \tag{12}$$

5.1. Advantages of Reward-Vector Generator Model

1. **Inherently Random**: Can play the role of a simulator that can generate instantaneous-reward-vectors.

2. **Retrievable Conditional Means and Variances**: Both the conditional means and variances of individual metrics are retrievable from the model outputs.
3. **Better Performance as Mean-Reward-Vector Predictor**: Our Bayesian model tries to better capture the distributions of Reward 1, Reward 2, and Reward 3. Compared to a typical regression, we should expect better performance in mean-reward-vector prediction. Our results show this is true.

6. Reward-Vector Generator Training

6.1. Framing Reward-Vector Generator Learning as a Supervised Multi-Task Learning Problem

Given our reward-vector generator model, we have to learn 17 functions of the (context, action) pair. These are

- $\{m_1^{(j)}(c,a)\}_{j=1}^3$, $m_2^{(2)}(c,a)$ (4 tasks for predicting conditional means);
- $\{\sigma_1^{(j)}(c,a)\}_{j=1}^3$, $\sigma_2^{(2)}(c,a)$ (4 tasks for predicting conditional variances);
- $\{p_0^{(j)}(c,a), p_1^{(j)}(c,a)\}_{j=1}^4$, $p_2^{(j)}(c,a)$ (together make 4 classification tasks from 8 functions).

The learning of these 17 functions can be framed as a supervised multi-task learning (sMTL) problem with 12 tasks because while $\{p_0^{(j)}(c,a), p_1^{(j)}(c,a)\}_{j=1}^4$ are 8 functions, they are two classification tasks (as indicated in the parentheses above). Since conditional variances are part of our reward-vector generator model, there is no need to set any weights for the first 8 tasks. For the remaining tasks, (the 4 classification tasks), let us use temperatures $\sigma = \{\sigma_c^{(j)}, j = 1,2,3,4\}$. Then, using the model-uncertainty method [11], we can derive the below neural-network (θ-based) approximation of the log-likelihood function (12).

$$
\begin{aligned}
&\text{-log}\, d(r|c,a;\theta,\sigma) \\
&\approx -[\textstyle\sum_{j=1,3} 1[r^{(j)}=0]\frac{\log p_0^{(j)}}{(\sigma_c^{(j)})^2} + 1[r^{(j)}>0](\frac{\log p_1^{(j)}}{(\sigma_c^{(j)})^2} + \frac{1}{2(\sigma_1^{(j)})^2}(r^{(j)}-m_1^{(j)})^2 + \log\sqrt{2\pi}\sigma_1^{(j)})] \\
&- [1[r^{(2)}=0]\frac{\log p_0^{(2)}}{(\sigma_c^{(2)})^2} + 1[r^{(2)}>0](\frac{\log p_1^{(2)}}{(\sigma_c^{(2)})^2} + \frac{1}{2(\sigma_1^{(2)})^2}(r^{(2)}-m_1^{(2)})^2 + \log\sqrt{2\pi}\sigma_1^{(2)}). \\
&. + 1[r^{(2)}<0](\frac{\log p_2^{(2)}}{(\sigma_c^{(2)})^2} + \frac{1}{2(\sigma_2^{(2)})^2}(r^{(2)}-m_2^{(2)})^2 + \log\sqrt{2\pi}\sigma_2^{(2)})] \\
&- \frac{1}{(\sigma_c^{(4)})^2}[\log p_0^{(4)} 1[r^{(4)}=0] + \log p_1^{(4)} 1[r^{(4)}=1]] + \textstyle\sum_{j=1}^4 \log\sigma_c^{(j)} = -\log d\tilde{\mathbb{P}}(r|c,a;\theta,\sigma). \quad (13)
\end{aligned}
$$

Making the standard assumption that the training-examples $\{(c_i,a_i,r_i)\}_{i=1}^n$ are generated in an i.i.d[5] manner, one justifies training a neural-network for minimizing the negative log-likelihood, i.e.,

$$
{}_{\theta,\sigma} - \frac{1}{n}\sum_{i=1}^{n} \log d\tilde{\mathbb{P}}(r_i|c_i,a_i;\theta,\sigma). \quad (14)
$$

[5]Independent and identically distributed.

There is one little caveat we need to address. The empirical-risk-function in 14 has variances and temperatures in the denominators which would cause errors in our implementation if they are initialized to zero or get close to zero during the training. To solve this issue, let us use the substitution,

$$\alpha = \log \sigma. \tag{15}$$

Thus, the optimization problem we will solve in our implementation is given by

$$_{\theta,\sigma} - \frac{1}{n} \sum_{i=1}^{n} \log d\tilde{\mathbb{P}}(r_i | c_i, a_i; \theta, e^{\alpha}). \tag{16}$$

6.2. Other Reward-Vector Generator Models

1. We currently use regression for estimation of Reward 1, Reward 2, and Reward 3, i.e., by assuming the conditional-distributions to be some gaussian pdfs. Together with Reward 4, this assumption about conditional distributions results in 4 tasks. Let us call a neural-network model learnt by setting the weights of each task to 1 as **type-1 model**, whereas when the weights are obtained using the model-uncertainty method, we will call it **type-2 model**.
2. In our reward-vector generator model, if we enforce all (context, page-layout)-conditioned variances of Reward 1, pos. Reward 2, neg. Reward 2, Reward 3 to be the same, we retrieve the model-uncertainty method. This reduces the number of tasks to 8. Let us call this **type-3 model**.

We will call the model learned by solving function 16 as **type-4 model**. Without loss of generality, we used the Multi-gate Mixture of Experts (MMoE) architecture [12] for training purposes.

7. Evaluation Results

Table 1
Comparision of model-types as reward-vector generators on test dataset.

Reward	Statistic	Ground-truth	Model-type			
			1	2	3	4
Reward 1	Mean	0.5702	1.1617	1.0102	0.4809	0.4840
	Variance	4.3392	0.0248	0.3040	1.2947	2.8331
Reward 2	Mean	0.8123	1.0809	0.4165	0.7444	0.7723
	Variance	48.2461	0.1171	1.8136	24.9843	56.6592
Reward 3	Mean	14.1212	2.1394	19.5134	12.4750	14.0277
	Variance	9656.9371	1.3290	31689.6621	2004.1919	85994.1797
Reward 4	Mean	0.7438	0.7455	0.7409	0.7402	0.7409

While the primary purpose of the models is not as a mean reward predictor, we performed an experiment to compare the model types for the task of predicting mean rewards. These results are summarized in Table 5 in Appendix A. We consider this as evidence that our choice of conditional distributions better models the rewards.

7.1. Comparision of Model-Types as Reward-Vector Generators

The results of our experiments for the comparision of model-types 1 through 4 for the task of generating reward-vectors are summarized in Table 1. Each row corresponds to some statistic of a reward signal and the highlighted values are the ones closest to the ground-truth. We can observe that overall type-4 model performs better in replicating the statistics.

Table 2
Using model-4 as Random Vector Generator (RVG) and Direct methods on **small dataset**. The policy being evaluated (π) is the logging policy (μ) itself (policy that generated the **small dataset**).

	Reward 1	Reward 2	Reward 3	Reward 4
Ground-Truth	0.57	0.816	14.1578	0.7441
Direct-method	1.1764	1.0835	2.1532	0.7404
RVG	0.4853	0.7697	14.0201	0.7404

7.2. Using Model-4 in Simulation and Direct Methods

Having obtained a trained reward-vector-generator which can also output mean-reward-vector predictors, we ran the simulation and direct methods on the entire **small dataset** using model-type 4. The results of these two methods are listed in Table 2. The values closest to the ground-truth are highlighted. We note that the simulation-method performed better than the direct-method. In Table 3, we see that on the **online experiment dataset**, Model-4 was able to correctly predict winners among two treatments in 3 out of 4 metrics. Here the model was trained using experience data from a separate online logging policy from the same period. These results demonstrate the generalisability of our simulator/reward vector generator.

Table 3
Comparision of Random Vector Generator (RVG) Metrics for **Online Experiment Dataset** with Attributed/Ground Truth Data using model-type 4 trained on randomized traffic. Here logging policy and evaluation policies are different. GT means Ground Truth.

Means	Reward 1		Reward 2		Reward 3		Reward 4	
	T2	T1	T2	T1	T2	T1	T2	T1
GT	0.389	0.387	0.621	0.619	9.984	9.983	0.822	0.822
RVG	0.372	0.372	0.549	0.548	11.46	11.45	0.811	0.811

Since the reward vector generator/simulation method samples from a distribution, which is atypical of model based methods, we expect some variance in the estimates. We empirically estimated the variance of this method and compared it to the variance of the the Self Normalized IPS (SNIPS) method in Table 4. We see that our random vector generator method has 3 to 6 orders of magnitude less variance than the SNIPS estimator.

Table 4
Comparing the Simulator/Reward-Vector Generator (RVG) Variance with SNIPS

Method	Reward 1		Reward 2		Reward 3	
	T1	T2	T1	T2	T1	T2
RVG	$2.84{-}7$	$2.26{-}7$	$3.57{-}6$	$4.15{-}6$	$5.24{-}4$	$6{-}4$
SNIPS	0.16	0.13	4892.07	2052.97	0.23	0.24

8. Conclusion

Many eCommerce applications use contextual bandit models for decision making. These models often optimize for an objective that is a linear combination of multiple objectives. Finding a pareto-optimal linear scalarization of the objectives is an OPE problem in this setting.

Amongst the OPE methods, theoretically, under mild assumptions, the Dobuly Robust (DR) estimator has low bias, and also enjoys good variance properties as long as one of the estimators (model or IPS) is accurate. However, in practice, this is rarely the case. On the other hand, the simulation and direct methods do not have good theoretical guarantees of the DR-estimator, but they do not require any attributions apart from tuples of the form (context, considered-actions, action, reward), which are already widely available in our current deployment of PLR. Keeping this in mind, we developed a random vector generator methods. In doing so, we formulated and implemented a Bayesian model (model-4) about the ground-truth conditional distributions of rewards. In our results, we noted that model-4 can improve upon the performance of a baseline MMOE model as the mean reward-vector predictor. Interestingly, we also found that the random vector generator method performed better than the direct-method on our **small dataset**. Further, we trained our random vector generator on an online policy and validated that it predicted the correct winner among two treatments of an A/B test (**online experiment dataset**). The random vector generator method does this at 3 to 6 orders of magnitue lower variance than Self Normalized IPS (SNIPS) method. This is an interesting observation, one that is worth exploring both theoretically and empirically.

References

[1] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, J. Tang, Deep reinforcement learning for page-wise recommendations, in: Proceedings of the 12th ACM Conference on Recommender Systems, RecSys ’18, Association for Computing Machinery, New York, NY, USA, 2018, p. 95–103. URL: https://doi.org/10.1145/3240323.3240374. doi:10.1145/3240323.3240374.

[2] A. Slivkins, Introduction to multi-armed bandits, CoRR abs/1904.07272 (2019). URL: http://arxiv.org/abs/1904.07272. arXiv:1904.07272.

[3] N. Wang, H. Wang, M. Karimzadehgan, B. Kveton, C. Boutilier, Imo^3: Interactive multi-objective off-policy optimization, CoRR abs/2201.09798 (2022). URL: https://arxiv.org/abs/2201.09798. arXiv:2201.09798.

[4] Z. Gao, Y. Han, Z. Ren, Z. Zhou, Batched multi-armed bandits problem, 2019. arXiv:1904.01763.

[5] L. Li, W. Chu, J. Langford, X. 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 - WSDM '11, ACM Press, 2011. URL: https://doi.org/10.1145%2F1935826.1935878. doi:10.1145/1935826.1935878.

[6] D. G. Horvitz, D. J. Thompson, A generalization of sampling without replacement from a finite universe, Journal of the American Statistical Association 47 (1952) 663–685. URL: http://www.jstor.org/stable/2280784.

[7] M. Dudik, J. Langford, L. Li, Doubly robust policy evaluation and learning, in: Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 1097–1104. URL: https://arxiv.org/pdf/1503.02834.

[8] E. L. Ionides, Truncated importance sampling, Journal of Computational and Graphical Statistics 17 (2008) 295–311. URL: http://www.jstor.org/stable/27594308.

[9] A. Swaminathan, T. Joachims, The self-normalized estimator for counterfactual learning, in: C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 28, Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/39027dfad5138c9ca0c474d71db915c3-Paper.pdf.

[10] M. Farajtabar, Y. Chow, M. Ghavamzadeh, More robust doublt robust off-policy evaluation, in: Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. URL: https://arxiv.org/pdf/1802.03493.

[11] A. Kendall, Y. Gal, R. Cipolla, Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, 2017. URL: https://arxiv.org/abs/1705.07115. doi:10.48550/ARXIV.1705.07115.

[12] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, E. H. Chi, Modeling task relationships in multi-task learning with multi-gate mixture-of-experts, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, KDD ’18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1930–1939. URL: https://doi.org/10.1145/3219819.3220007. doi:10.1145/3219819.3220007.

A. Supplementary Evaluation Results

More results are shown in Table 5.

B. Training Data Statistics

The training data statistics are in Table 6.

Table 5
Comparision of model-types as mean reward-vector predictors. Each row corresponds to a model-type and lists its prediction metrics on train and test datasets for Reward 1, Reward 2, Reward 3, and Reward 4 rewards. These metrics are root-mean-square-error (RMSE) for Reward 1, Reward 2, Reward 3, and area-under-receiver-operating-curve (AUROC) for Reward 4. The best metric values in each column are highlighted.

Model	Reward 1		Reward 2		Reward 3		Reward 4	
Type	Train	Test	Train	Test	Train	Test	Train	Test
1	2.2209	2.161	6.8975	6.9468	111.4532	98.9076	.6425	.6409
2	2.2211	2.1613	6.8974	6.9468	111.4529	98.9067	.6432	.6412
3	2.1334	2.0711	6.8834	6.9329	110.6313	97.9745	.6430	.6412
4	2.1343	2.0717	6.8824	6.9319	110.6608	98.0126	.6434	.6417

Table 6
Important Statistics of Train, Validation and Test Data-sets.

Metric	Statistic	Data-set		
		Train	Validation	Test
Reward 1	Mean	0.5713	0.5706	0.5702
	Variance	4.6034	4.33	4.3392
	Zeroes	14511963	3110180	3108955
	Positives	3230605	691800	693024
	Cond'l Mean Pos	3.1377	3.1358	3.1283
	Cond'l Mean Pos	17.2293	15.7528	15.8030
Reward 2	Mean	0.8154	0.8236	0.8123
	Variance	47.5675	305.2215	48.2461
	Zeroes	16571130	3551870	3551820
	Positives	1028124	219576	219373
	Negatives	143314	30534	30786
	Cond'l Mean Non-zero	12.3501	12.5191	12.3449
	Cond'l Var Non-zero	578.003	4493.3414	590.8888
	Cond'l Mean Pos	14.9443	15.1495	14.9548
	Cond'l Var Pos	578.2699	5029.9224	598.3940
	Cond'l Mean Neg	-6.2609	-6.3966	-6.2523
	Cond'l Var Neg	181.4392	227.1482	143.0220
Reward 3	Mean	14.1044	14.4436	14.1212
	Variance	12296.421	301423.6155	9656.9371
	Zeroes	14619342	3132516	3132948
	Positives	3123226	669464	669031
	Cond'l Mean Pos	80.1251	82.0273	80.2480
	Cond'l Mean Pos	64564.1866	1706285.3960	49572.0939
Reward 4	Mean	0.7441	0.7441	0.7438