Off-policy evaluation for ranking in information retrieval

Gaomin Wu
New York University
gw1107@nyu.edu

Haoxue Li
New York University
hl3664@nyu.edu

Yafu Ruan
New York University
yr942@nyu.edu

Hongxu Hao
New York University
hh2302@nyu.edu

Abstract

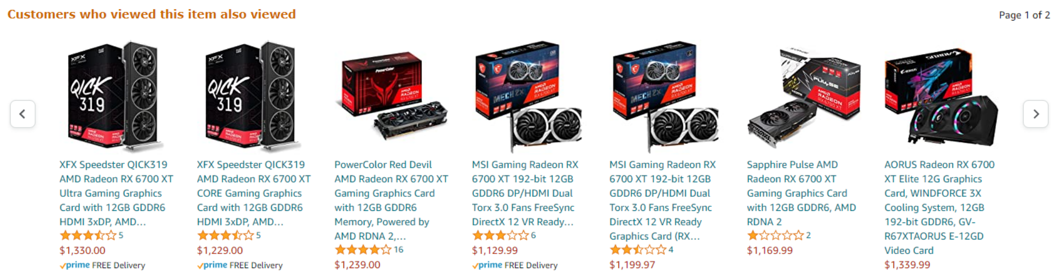
Off-policy evaluation (OPE) aims to estimate the online performance of a policy using pre-collected data. Evaluation for ranking in information retrieval is different from other OPE, since we need evaluate an ordered list of actions instead of a single one. We transform the problem into combinatorial contextual bandits, and test various OPE estimators empirically under different situations with a semi-synthetic evaluation mechanism using a public dataset. This report makes extension on paper "Off-policy evaluation for slate recommendation"[1] and provides insights of why the proposed psudeo-inverse estimator performs very well in the experiment and why the very standard importance-weighted estimator fails drastically. Besides, we extend the deterministic target policy as proposed in the original paper to a stochastic one, and conclude that the extreme failure in OPE of importance-weighted estimator from the original paper might due to the choice of making target policy deterministic.

1 Introduction

Online services like Youtube, News, E-commerce etc have relied on data-driven marketing, which use users' portfolio and data to make better service. These recommended items are selected from an often large collection of items and ranked basis user behaviour and interests. The objective is to maximise user engagement by offering personalised content that best aligns with a user's interests and likes. Recommendation system has been widely used by most of the tech companies, especially Amazon. According to report published by McKinsey in 2013, 35% of revenue come from recommendation engine at that time[2]. If people click one item at Amazon, it will deliver a list of item that "Customers who viewed this item also viewed", as show in Figure 1. This is an on-site recommendation and it increases average possibility of selling through similar items. Similar ideas are applied in other parts of Amazon. Recommendation has become an essential method to make content serving to customers. After developing new content serving algorithms (called policies), usually we test their performance using previous collected data, which is off-policy evaluation.

We consider the above problem as recommending an ordered set of items (i.e. ranking), and formalize the problem as a slate bandit problem (also referred to as the combinatorial bandit problem) [3]. It is similar to the traditional multi-armed bandit problem, except that the agent selects a slate that consists of multiple slots, each one of which is occupied by an action.

Preprint. Under review.



Customers who viewed this item also viewed

Page 1 of 2

XFX Speedster QICK319 AMD Radeon RX 6700 XT Ultra Gaming Graphics Card with 12GB GDDR6 HDMI 3xDP, AMD...
★★★½☆ 5
$1,330.00
prime FREE Delivery

XFX Speedster QICK319 AMD Radeon RX 6700 XT CORE Gaming Graphics Card with 12GB GDDR6 HDMI 3xDP, AMD...
★★★½☆ 5
$1,229.00
prime FREE Delivery

PowerColor Red Devil AMD Radeon RX 6700 XT Gaming Graphics Card with 12GB GDDR6 Memory, Powered by AMD RDNA 2,...
★★★★☆ 16
$1,239.00

MSI Gaming Radeon RX 6700 XT 192-bit 12GB GDDR6 DP/HDMI Dual Torx 3.0 Fans FreeSync DirectX 12 VR Ready...
★★★☆☆ 6
$1,129.99
prime FREE Delivery

MSI Gaming Radeon RX 6700 XT 192-bit 12GB GDDR6 DP/HDMI Dual Torx 3.0 Fans FreeSync DirectX 12 VR Ready Graphics Card (RX...
★★½☆☆ 4
$1,199.97

Sapphire Pulse AMD Radeon RX 6700 XT Gaming Graphics Card with 12GB GDDR6, AMD RDNA 2
★☆☆☆☆ 2
$1,169.99

AORUS Radeon RX 6700 XT Elite 12G Graphics Card, WINDFORCE 3X Cooling System, 12GB 192-bit GDDR6, GV-R67XTAORUS E-12GD Video Card
$1,339.99

Figure 1: Recommendation example in Amazon

2 Related Work

Off policy evaluation is an important part in reinforcement learning area. And recently, there are lots of scientist doing research on this area, because they would like to compare the models in real-world background for which the purposes of early stopping and hyper-parameter tuning methods are costly and often practically infeasible, and they found off-policy policy evaluation (OPE) works well in such situations [4]. In addition, in order to solve multi-class cost-sensitive classification data-set problem, Nikos Vlassis and his colleagues proposed a method by giving a new design for contextual bandits that can give rise to an estimator[5].

People did a lot for improving OPE for different situations, and when things turned to specific application of off-policy evaluation, about how it works in recommendation system area, researchers are more interested, because there are a lot of industrial applications for off-policy evaluation in this area. For example, some researchers focus on how to address biases in logged data and the representative article was published in 2020, the authors demonstrated how to address biases in logged data through incorporating a learned logging policy and a novel top-k off-policy correction[6]. Besides, more researchers chose to face the key challenge when using off-policy evaluation, that is the large number of slates. This problem was proposed by Yu Xiang Wang and his colleagues in 2017, the evaluating policy has high probability to choose different slates from those recorded in the logs, unless it is very similar to the data-collection policy. This issue is fundamental[7]. Hence, there are lot of scientist doing research on using OPE for slate recommendation. And most of them are based on the paper we mainly used in our project, and most of the researchers made some improvement by changing risk minimization function, adding control variate approach and adding shrinkage etc [8][9]. For example, the recent paper published in 2021 by Nikos Vlassis and his colleagues took Bayes risk as the criterion to evaluate estimators and found when analyzing family of estimators using Bayes risk, it gave different ranking results with MSE [8].

3 Combinatorial Contextual Bandit

We consider this as a combinatorial contextual/ slate bandits. It proceeds in rounds of the following steps:

1. observe context x from a distribution $D(x)$.
2. take action $A_1, A_2, ...A_l$ separately from action space $\{1, 2, ..., m\}$, then we can have one slate $s = (A_1, A_2, ...A_l)$. That is one slate contains l slot and for each slot, there are m actions to choose from.
3. receive corresponding reward $r \in [-1, 1]$ from a distribution $D(r \mid x, s)$.

Here we take action A_i according to $A_i \sim \pi(\cdot \mid x)$, where $\pi(\cdot \mid x)$ is a policy that we want to evaluate. The value of the policy $V(\pi)$ is defined as following:

$$V(\pi) = E_{x\sim D} E_{A\sim\pi(\cdot|x)} E_{r\sim D(\cdot|x,A)}[r]$$

For example, we consider web search and ranking scenario. Context x is the query along with user profile. Action A_i is the webpage the policy chooses. Due to the problem of webpage search, there is no repetition between actions $A_1, A_2, ...A_l$, that is, the policy chooses l out of m webpages. In this way, our slate $s = (A_1, A_2, ..., A_l)$. The number of possible slates is $\frac{m!}{(m-l)!}$, which is exponential in l and our estimator is used for this setting. We can set reward as negative of the time taken by the user to find a relevant webpage. The goal of the policy is to maximize the reward.

4 Off-policy Evaluation (OPE) Estimators

In this section, we will talk about all the OPE estimators we used for the report. First we revisit three very standard approaches for off-policy evaluation - Mean, Direct method, and Importance weighted.

To avoid exponential variance in Importance weighted value estimators, we use a new estimator - psudeo-inverse (PI) estimator proposed by Swaminathan et al [1]. By positing a linear assumption on the structure of rewards, the PI estimator is unbiased and variance bounded which can be proved mathematically.

4.1 Standard Approaches Off-policy evaluation

Suppose we run a static policy $\pi_0(s \mid x)$ on a slate bandit for n rounds, we get logged feedback $D = (x_1, s_1, r_1), (x_2, s_2, r_2), ..., (x_n, s_n, r_n)$ Then we use D from logging policy π_0 to estimate the performance of a new policy π, that is, the target policy. For off-policy evaluation, we have the following approaches:

1. Mean
   The Mean estimator is the average of rewards from logged feeback D:
   $$\hat{V}_{mean}(\pi) = \frac{1}{n}\sum_{i=1}^{n} r_i$$
   The Mean estimator only consider the logging policy and here we use this unweighted mean for reference.
2. Direct Method (DM)
   The direct method (DM) uses the logged data to train a (parametric) model $r(x, s)$ for estimating the expected reward given context x and slate s. then
   $$\hat{V}_{DM}(\pi) = \frac{1}{n}\sum_{i=1}^{n}[\sum_{s \in S(x)} \hat{r}(x_i, s)\pi(s \mid x_i)]$$
   The direct method is often biased due to mismatch between distribution shift of (x, s) between logged data and the target policy.
3. importance-weighted (IW) value estimator
   The IW value estimator for policy π, based on logged feedback D with slates chosen under a static logging policy π_0 is
   $$\hat{V}_{iw}(\pi) = \frac{1}{n}\sum_{i=1}^{n} r_i \frac{\pi(s_i \mid x_i)}{\pi_0(s_i \mid x_i)}$$
   Here IW value estimator is unbiased under weak assumption.
4. Self-normalized IW (snIW) value estimators
   The self-normalized IW value estimator for policy π, based on logged feedback D with slates chosen under a static logging policy π_0 is
   $$\hat{V}_{sn\_iw}(\pi) = \frac{1}{n}\sum_{i=1}^{n} r_i \frac{\pi(s_i \mid x_i)}{\pi_0(s_i \mid x_i)} / \sum_{i=1}^{n} \frac{\pi(s_i \mid x_i)}{\pi_0(s_i \mid x_i)}$$
   Here self-normalized IW value estimator is unbiased and has less variance compared with IW value estimator.

4.2 Linearity Assumption

We assume the reward has a linear structure, that is, we assume that the reward is a sum of unobserved action-level rewards.

Linearity Assumption: For each context $x \in X$, there exists an unknown intrinsic reward vector $\phi_x \in R^{lm}$ such that the slate value satisfies $V(x, s) = \mathbf{1}_s^T \phi_x = \sum_{j=1}^{l} \phi_x I(j, A_j)$.

In the assumption, ϕ_x can be viewed as a context-specific weight vector for the sum. The indicator $I(j, A_j)$ represents whether the slate s has action A_j in jth slot, if it has, $I(j, A_j) = 1$, otherwise, $I(j, A_j) = 0$. And we can use a slate indicator vector 1_s^T to describe all these l indicators.

The linearity assumption is reasonable in the ranking setting. For example, a common metric in web ranking, *normalized discounted cumulative gain* (NDCG) satisfies the linearity assumption.

4.3 The psudeo-inverse estimator (PI)

Applying the linearity assumption, the slate value estimator is $V(x, s) = \mathbf{1}_s^T \phi_x$. To estimate the weight vector ϕ_x, we can minimize the MSE objective function:

$$\begin{aligned}
&\min E_{s\sim\mu(\cdot|x)} E_{r\sim D(\cdot|s,x)}[(\mathbf{1}_s^T \phi_x - r)^2] \\
=&\min E_\mu[(\mathbf{1}_s^T \phi_x - r)^2 \mid x]
\end{aligned}$$

We can get the estimated $\hat{\phi}_x$ in closed form:

$$\hat{\phi}_x = (E_\mu[\mathbf{1}_s \mathbf{1}_s^T \mid x])^\dagger E_\mu[r\mathbf{1}_s \mid x]$$

Here $M^\dagger$ refers to the Moore-Penrose pseudoinverse of a matrix. We denote $\Gamma_{\mu,x} = E_\mu[\mathbf{1}_s \mathbf{1}_s^T \mid x]$ and $\theta_{\mu,x} = E_\mu[r\mathbf{1}_s \mid x]$. Then we will have

$$\hat{\phi}_x = (\Gamma_{\mu,x})^\dagger \theta_{\mu,x}$$

In this way, the slate value is

$$\begin{aligned}
V(x, s) &= \mathbf{1}_s^T \phi_x \\
V(x, s) &= \mathbf{1}_s^T (\Gamma_{\mu,x})^\dagger \theta_{\mu,x}
\end{aligned}$$

We further introduced $q_{\pi,x} := E_\pi[\mathbf{1}_s \mid x]$ as the expected slate indicator under π conditional on x. And the empirical estimate for $\theta_{\mu,x}$ is $\hat{\theta}_i := r_i \mathbf{1}_{s_i}$. The Monte Carlo estimate for $V(x, s)$ is the pseudoinverse (PI) estimator:

$$\begin{aligned}
\hat{V}_{PI}(\pi) &= \frac{1}{n} \sum_{i=1}^{n} \sum_{s\in S} \pi(s \mid x_i) \mathbf{1}_s^T (\Gamma_{\mu,x_i})^\dagger \hat{\theta}_i \\
\hat{V}_{PI}(\pi) &= \frac{1}{n} \sum_{i=1}^{n} r_i \cdot q_{\pi,x}^T \Gamma_{\mu,x}^\dagger \mathbf{1}_s^T
\end{aligned}$$

Similar to snIW, we can also derive a weighted variant of PI - snPI:

$$\hat{V}_{snPI}(\pi) = \frac{1}{n} \sum_{i=1}^{n} r_i \cdot q_{\pi,x}^T \Gamma_{\mu,x}^\dagger \mathbf{1}_s^T / \sum_{i=1}^{n} q_{\pi,x}^T \Gamma_{\mu,x}^\dagger \mathbf{1}_s^T$$

4.4 How to understand the proposed psudeo-inverse estimator?

In the above section, we show that PI is the slate value $\hat{V}_{PI}(\pi) = \mathbb{E}_{\mathbf{s}\sim\pi(\cdot|x)} \mathbf{1_s}^T \hat{\phi}_x$, where $\hat{\phi}_x$ is estimated from the logging data, i.e. the logging distribution s $\sim \mu(\cdot \mid x)$ and $r \sim D(\cdot \mid \text{s}, x)$. The Monte Carlo estimate of PI is $\hat{V}_{PI}(\pi) = \frac{1}{n} \sum_{i=1}^{n} r_i \cdot q_{\pi,x_i}^T \Gamma_{\mu,x_i}^\dagger \mathbf{1}_{\mathbf{s}_i}$.

We consider a general form of estimator

$$\hat{V}(\pi) = \frac{1}{n} \sum_{i=1}^{n} r_i w_i$$

Table 1: Toy example to compare w^{IW} and w^{IP}

	logging slate = [1, 0]	
target slate	w^{IW}	w^{IP}
[1, 0]	6	4.5
[0, 1]	0	0.5
[1, 2]	0	1.0
[2, 1]	0	0.0
[2, 0]	0	1.5
[0, 2]	0	-1.5

IW, snIW, PI and snPI all fits in this general form. For IW, $w_i^{IW} = \frac{\pi(s_i|x_i)}{\pi_0(s_i|x_i)}$; for PI, $w_i^{PI} = q_{\pi,x_i}^T \Gamma_{\mu,x_i}^{\dagger} \mathbf{1}_{\mathbf{s}_i}$;

In the following part, we compare w_i from IW and PI in a toy example to see their difference in a clear way. We consider $m = 3$ (action number for each slot) and $l = 2$ (slate length, i.e. number of slots in a slate) under a uniform logging policy. To match the experiment setup, we only consider deterministic target policy, then $\mathbf{q}_{\pi,x} := \mathbb{E}_\pi [\mathbf{1}_\mathbf{s} \mid x]$ degenerate to $\mathbf{q}_{\pi,x} = \mathbf{1}_\mathbf{s}(x)$, and $w_i^{PI} = \mathbf{1}_\mathbf{s}(x_i)^T \Gamma_{\mu,x_i}^{\dagger} \mathbf{1}_{\mathbf{s}_i}$. Because under deterministic target policy, given context x, one and only slate would be selected by the target policy. We use notation $[a_1, a_2]$ to represent action selections in the two slots in this toy example and $a_i \in [0, 1, 2]$. Table-1 shows the change of w^{IW} and w^{IP} with different choices of target slate, while logging slate is fixed. We can see that w^{IW} is nonzero only when target policy select the exact same slate as in the logging data. This makes sense as the target policy is deterministic. w^{IP} has both positive and negative value, and when the target slate match logging slate, w^{IP} value increases.

In this toy example, the size of the slate space is 6. In a more realistic setting, the slate space can be enormous. let's say $m = 100$, $l = 10$, the size of the slate space is $6.2815651e + 19$! This sparsity of target slate matching the logging slate is the main problem why IW does not work in this case (You will see this in Section 6). By imposing a linearity assumption on how the individual slot rewards relate to the overall rewards for a slate, PI has no sparsity issue. Hence, compared with IW, PI has advantage in sample efficiency and thus, has better performance in experiment.

5 Experiment Setup

In order to empirically evaluate the performance of the pseudoinverse estimator for ranking problems, we apply the same semi-synthetic evaluation set up proposed by Swaminathan et al [1] to run the simulation. All of our code is available online[1].

5.1 Dataset

The data-sets we used are from Microsoft Learning to Rank Challenge data-sets[10].These data-sets include two parts: MSLR-WEB10K and MSLR-WEB30K, and here our experiments are mainly based on the first one, MSLR-WEB10K, which has 1000 queries.

The MSLR-WEB10K are queries data, which all queries and urls are represented by IDs. The attributes of this data-set are feature vectors and relevance labels. The feature vectors are extracted from query-url pairs and the relevance labels are values from 0 to 4, where 0 means irrelevant and 4 means perfectly relevant. In the data file, each row represents a query-url pair, the first column is relevance label, the second column is query id and the left columns corresponds to the different features.

5.2 Feedback Simulation

In order to evaluate the OPE estimators, we need the full feedback data $D_{full} = (x_1, s_1, r_1), (x_2, s_2, r_2), ..., (x_n, s_n, r_n)$. We generate semi-synthetic feedback data based on the dataset mentioned above. We consider queries as contexts x and the available documents to retrieve

[1]`https://github.com/YafuR/slates_semisynth_expts`

are actions a, and each pair of x and a has a feature vector $f(x, a)$ and a label $rel(x, a)$ which represent the relevance of action a on context x (our queries).

In our experiment, we use ERR to generate slate rewards. The expected reciprocal rank, also called ERR is calculated as follow:

$$\mathrm{ERR}(x, \mathbf{s}) := \sum_{r=1}^{\ell} \frac{1}{r} \prod_{i=1}^{r-1} \left(1 - R\left(s_i\right)\right) R\left(s_r\right), \quad \text{where } R(a) = \frac{2^{rel(x,a)} - 1}{2^{\text{maxel}}} \text{ with maxrel } = 4$$

Here, the $rel(x, a)$ is the label from the data.

Additionally, in order to use this metric, we need to make a standard assumption here: The logging policy should give positive probability(non-zero) for all slates s which can be chosen by the target policy, otherwise, the off-policy evaluation cannot be unbiased. A note here is that ERR does not satisfy the linearity assumption we mentioned above. The original paper actually also implement another metrics called normalized discounted cumulative gain or NDCG, which satisfies the linearity assumption. And the paper also shows similar performance that under both rewards.

5.3 Policy Simulation

In this experiment, the policy is the selection of slates that consisting components actions/documents. In order to generate a policy, we first train a machine learning model, specifically a decision tree regressor, to predict the relevance $\hat{rel}(x, a)$ between the query-document pairs (x, a) . Then we rank all the available documents by $\hat{rel}(x, a)$. To generate a deterministic policy, we can simply choose the top l documents from the rank according to the model.

To make the policy stochastic, we construct a multinomial distribution for the policy:

$$p_\alpha(a \mid x) \propto 2^{-\alpha\lfloor\log_2 \mathrm{rank}(x,a)\rfloor}$$

We denote $\mathrm{rank}(x, a)$ as the rank of document a using context x from the machine learning model. Note that the policy is parametrized by the model through $\mathrm{rank}(x, a)$. The parameter α represents the decaying weights along the $\mathrm{rank}(x, a)$. Varying α interpolates between uniformly random and deterministic policy decided by the ML model. We could construct slates by sampling actions through p_α without replacement.

We derive several different variants of policies: Uniform, Optimal, Sub-optimal and Anti-optimal.

1. **Uniform policy** is a stochastic policy that select documents for each slot uniformly from the available documents.

2. **Optimal policy** is based on ML models that trained with the full set of features f. Since the whole feature set we have is highly relevant to the relevance score, and the reward depends on the the relevance score. We consider the logging policy trained with all features as the optimal policy. It can be deterministic or stochastic.

3. **Sub-Optimal policy** is based on ML models that trained with the a subset of features f_{url}. To generate a biased model, we extract features related to the url as f_{url}. It can be deterministic or stochastic.

4. **Anti-Optimal policy** is based on ML models that trained with the full set of features. However, after the trained models generate $\hat{rel}(x, a)$, we sort the rank with the least relevant document on the top, which make the policy select slates with very bad rewards. It can also be deterministic or stochastic.

5.4 Estimators Implementation

For estimators, there are existing implementations for DM, IPS and PI for the original paper. The DM take half of the dataset to train the model for generating the policy, and the remaining half is used for evaluation. Except for DM, self-normalized version is also provided for scaling. One potential problem is that all target policies are coded as deterministic policy, which could be a special case of stochastic target policy with probability one for one action and 0 elsewhere.

Evaluation of Estimators In the dataset, we have labels $rel(x, a)$ for all available documents/actions for each query. We can generate all possible slates s (combination of documents/actions), and compute its reward by $ERR(x, s)$ with the labels. This means that for any (x, s) the ture reward is known, thus the estimator is evaluated by mean square error(MSE). We run the evaluation for several iterations. The results are recorded so that we could know how far from true values to estimated values. Besides, we are able to get the variance of the result.

6 Results

6.1 How does mismatch between logging and target policy affect off-policy estimation?

We evaluated the effectiveness of different off-policy estimators described in Section 4: OnPolicy, Mean, snIW, snPI, DM_tree, DM_ridge and DMc_ridge. Note that DM_tree, DM_ridge are implementation of DM with decision tree, and ridge regression trained with full set of features respectively. DMc_ridge is just a variant of DM_ridge, which clip the predicted reward into the range of [-1, 1]. On-policy estimator is just for comparison. In the original paper, DM is only trained with url features so here we try DM with full advantage to see if DM can have better performance.

We evaluated these seven estimators under twelve different pairs of logging and target policy. Note that we follow the original paper and set all logging policy as stochastic with α=1.0 and set all target policy deterministic. Figure 2 shows the estimated average reward for the three deterministic target policies while using the logged data from four different stochastic logging policies. The four rows in the figure represent the same experiment repeated with different policies as the logging policy. Note that even the diagonal does not represents the on-policy average reward since the logging is always stochastic and target policy is always deterministic. The description of each policy can be found in Section 5.3. Each experiment was repeated 25 times and the error bars represent the standard deviation of the estimated values. The MSE in log scale for all the twelve situation is also shown in Figure 3.

With these experiment, we further explore the robustness of the estimators under a situation of a large samples of data (sample size = 10 million). We observe that snIW estimator has a large variance, making it less favorable to be used in practice. Further more, snIW always gives estimated value close to zero. DM works pretty well when the target policy and logging policy are similar. However, when they are very different it degrades eventually as it overfits on the logging distribution, which is different from the target distribution. Additionally, we can easily see that snPI outperforms all the other estimators. Even under the case where logging and target policy is very different (when one is optimal and the other is anti-optimal), snPI still provides the most accurate estimate among all.

6.2 Will the SNIPS still perform so bad in the case of stochastic target policy?

When comparing snIW estimator with snPI, we can see that the pseudoinverse estimator snPI easily dominates snIPS across all experimental conditions, as can be seen in Fig 2. This result matches exactly what they show in the original paper [1]. Specifically snIW estimator always give estimated values near zero under all situations and snIW has very large variance compared with all the other estimators.

However, we wonder if this is always the case or this is a result due to their deliberate choice of deterministic target policy. As we have shown in Section 4.4 with a toy example, with deterministic target policy and large slate size (in our experiment $m = 100$ and $l = 10$), snIW and IW both suffer from a large variance because there is very few nonzero weight, and hence very low sample efficiency due to the very sparse action/slate space.

We implement a stochastic target policy in the way as mentioned in Section 5.3. Here we choose to use decision tree as model, and use the full set of features to train the model. The choice of $\alpha = 2.0$. We compare the average reward estimates of snIW for uniform random logging policy with a deterministic target policy and a stochastic target policy, respectively in Figure 4. Note that both target policy share the same trained models. As shown in the plot, clearly after turning the target policy from deterministic to stochastic, the snIW estimated value is closer to the true estimate and the variance reduces a lot. This result supports the analysis in Section 4.4 and also shows that the reason why snIW results in the original paper have extremely poor performance is mainly due to the choice of deterministic target policy.

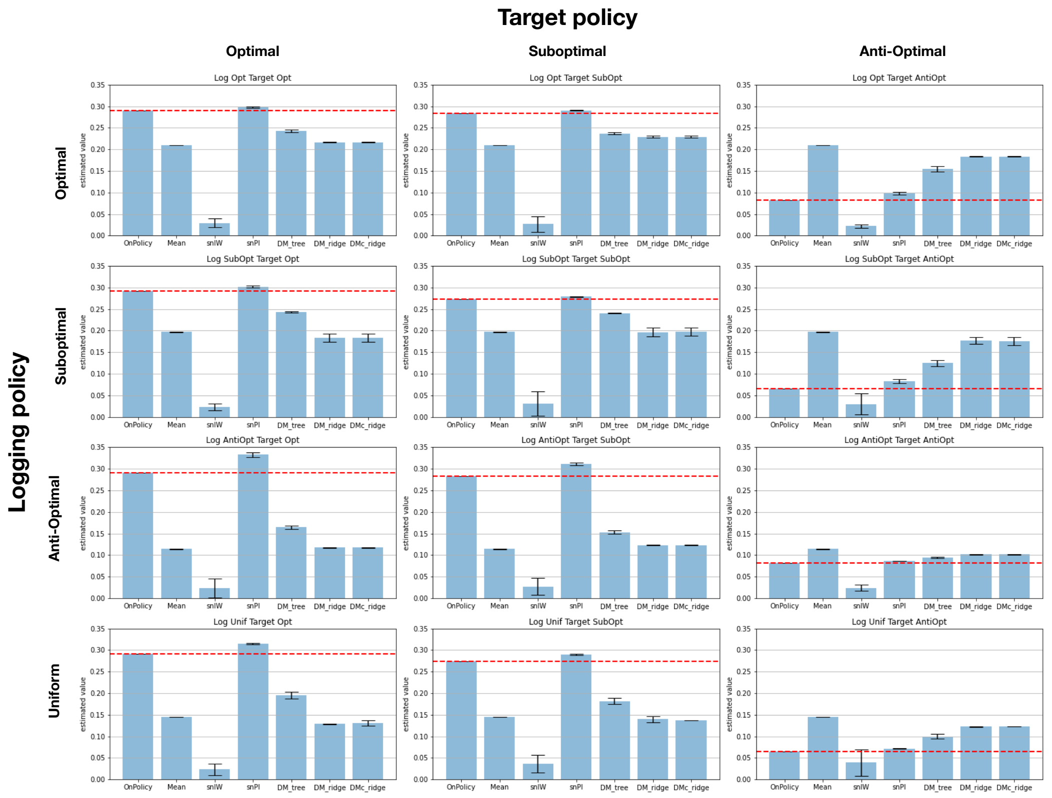


Figure 2: Top: Results of the simulation experiment showing the average reward estimates for three different target policies computed using various off-policy estimators when using optimal stochastic policy as the logging policy. Row 2: same as the top but with sub-optimal stochastic policy as the logging policy. Row 3: same as the top but with anti-optimal stochastic policy as the logging policy. Bottom: same as the top but with uniform random policy as the logging policy. The red horizontal line indicate the true estimate for each experiments. The sample sizes are set to 10 million in all situation.

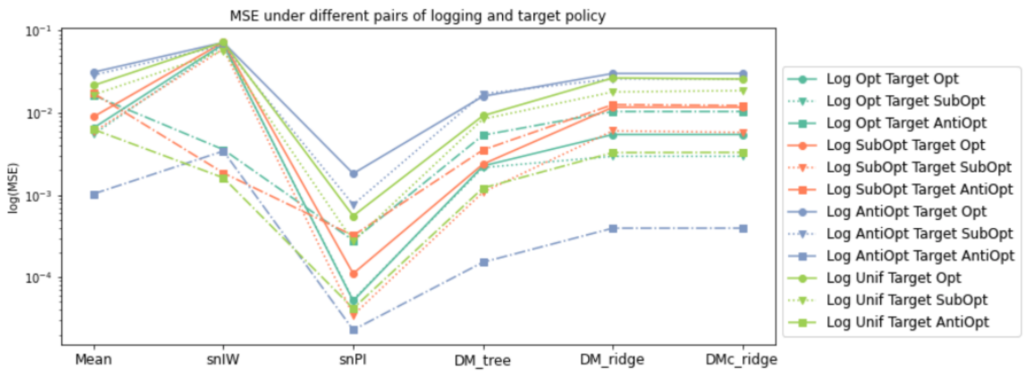


Figure 3: Results of the simulation experiment showing the MSE for twelve different situation computed using various off-policy estimators

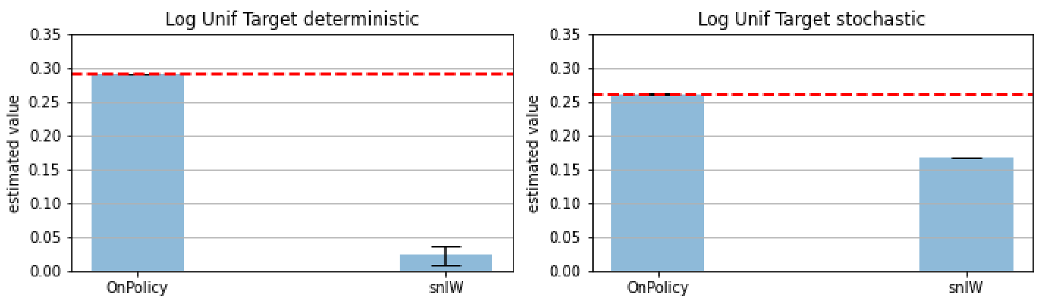


Figure 4: Results of the simulation experiment showing average reward estimates for stochastic and deterministic target policy

7 Conclusion and Future work

In this paper, we replicate the work from Swaminathan et al [1]. We mainly focus on their work in off-policy evaluation for ranking part. They proposed a psudeo-inverse estimator (PI) for slates bandit problem. The attraction of this approach is that it does not try to model the rewards and suffer the model mismatch bias. Instead, they impose a linearity assumption on how the individual rewards at the slot relate to the overall rewards for a slate. After that, they weighted the reward with a single-sample Monte Carlo estimate from the logged data to arrive at the psudeo-inverse estimator (PI). PI has great performance in experiments. They also have proved that PI is unbiased and variance bounded.

Other than replicating the work,

1. we analyze with a toy example in Section 4.4 to understand the difference of weights used in IW and PI. We learn that under a large slate space and a deterministic policy, there would be sparsity of target slate matching the logging slate, which is the main problem why IW does not giving good estimates. By imposing a linearity assumption on how the individual slot rewards relate to the overall rewards for a slate, PI has no sparsity issue. Hence, compared with IW, PI has advantage in sample efficiency and thus, has better performance in experiment. We later support this analysis with empirical experiments results that controlling all other factors, after turning the target policy from deterministic to stochastic, the snIW estimated value is much closer to the true estimate and the variance reduces a lot shown in Section 6.

2. Another major contribution we have is that we test the robustness of various off-policy estimators for ranking problem with simulation experiment under twelve different combination of logging policy and target policy. We show that when the data is abundant, snPI easily outperforms all the other estimators. Even under the case where logging and target policy is very different (when one is optimal and the other is anti-optimal), PI still remains lowest MSE among all.

There is work that we could have been done to make this report more thorough. However, due to lack of time, we are not able to finish it. We think it's interesting to covert all the target policy to stochastic policy in the whole experiment in Figure 2 and further test the robustness of all the estimators.

References

[1] Swaminathan et al. Off-policy evaluation for slate recommendation. *arXiv:1605.04812*, 2017.

[2] Chris Meyer Ian MacKenzie and Steve Noble Open interactive popup. How retailers can keep up with consumers. 2013.

[3] Nicolo Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

[4] Konstantinos Bousmalis Chris Harris Julian Ibarz Sergey Levine Alex Irpan, Kanishka Rao. Off-policy evaluation via off-policy classification. *arXiv:1906.01624*, 2019.

[5] Maria Dimakopoulou Tony Jebara Julian Ibarz Sergey Levine Nikos Vlassis, Aurelien Bibaut. On the design of estimators for bandit off-policy evaluation. *Proceedings of the 36th International Conference on Machine Learning, PMLR 97:6468-6476*, 2019.

[6] Paul Covington Sagar Jain Francois Belletti Ed Chi Minmin Chen, Alex Beutel. Top-k off-policy correction for a reinforce recommender system. *arXiv:1812.02353*, 2018.

[7] Alekh Agarwal Yu-Xiang Wang and Miroslav Dudik. Optimal and adaptive off-policy evaluation in contextual bandits. *In International Conference on Machine Learning*, 2017.

[8] Ashok Chandrashekar Nikos Vlassis, Fernando Amat Gil. Off-policy evaluation of slate policies under bayes risk. *arXiv:2101.02553*, 2021.

[9] Akshay Krishnamurthy Miroslav Dud´ık Yi Su, Maria Dimakopoulou. Doubly robust off-policy evaluation with shrinkage. *arXiv:1907.09623*, 2019.

[10] Tie-Yan Liu Tao Qin. Microsoft learning to rank dataset. *https://www.microsoft.com/en-us/research/project/mslr*.