COMPARING OFF-POLICY EVALUATION METHODS FOR RECOMMENDATION SYSTEMS

Phachara Laohrenu
Columbia University
New York, USA
pl2839@columbia@edu

1 Introduction

Propensity-Score-based estimators (e.g. IPS, SNIPS) have been popular methods for off-policy evaluation. In 2022, Marginalize Inverse Propensity Score (MIPS) [1] was proposed to handle large action space by making use of action embedding. In IPS, the importance weight for sample i is a function of context x_i and action a_i.

$$w^{IPS}(x_i, a_i) = \pi(a_i|x_i)/\pi_0(a_i|x_i)$$

However, for MIPS, the importance weight for sample i is a function of **context** x_i and **action embedding** e_i.

$$w^{MIPS}(x_i, e_i) = \mathbb{E}_{\pi_0(a|x_i,e_i)}[w(x_i, a)] = \sum_a \pi_0(a|x_i, e_i) * w(x_i, a)$$

Where $w(x_i, a) = w^{IPS}(x_i, a)$ and $\pi_0(a|x_i, e_i)$ is computed by regressing a on (x_i, e_i).

From here, we see the fundamental limitations of Propensity-Score-based estimators.

1. **The need for** $\pi_0(\mathbf{a_i}|\mathbf{x_i})$. In order to obtain the propensity score of the logged policy, the system must log the probability of the logging policy choosing the logged action, given the context. This is even worse for MIPS where we marginalize over all possible actions because now we need to log the probability of all actions, given the context. Moreover, the propensity score is not so easily available because the recommendation system is sometimes deterministic, or involves a deterministic final component such as the re-ranking component. In this case, propensity score can be obtained by building a classification model to predict the probability of selecting the actions, given context. However, the accuracy will degrade significantly as the size of the action space grows because there are more classes to predict.
2. **The need for** $\pi_0(\mathbf{a}|\mathbf{x_i}, \mathbf{e_i})$. Regressing a on (x_i, e_i) is also harder as the size of the action space grows for the same reason as above.
3. **High variance from the division**. Since the importance weight involves the division of propensity score, for actions with low probability, a small error in probability estimation can cause a large error. However, self-normalizing techniques can be used to mitigate this problem (SNIPS and SNMIPS).

1.1 Balanced Off-Policy Estimator

Balance Off-Policy Estimator (B-OPE) [2] was proposed in 2020 to solve the first two limitations of Propensity-Score-based estimators. B-OPE directly estimates the importance weight via density ratio estimation. Specifically, off-policy evaluation using B-OPE consists of four steps:

1. Create a supervised learning problem using the concatenated target policy instances (x, e') and logging policy instances (x, e), as covariates and giving a label (C) of 0 to the logging policy and 1 to the target policy.
2. Learn a classifier to distinguish between the logging and target policy.

3. Take the importance weight as

$$w^{BOPE}(x_i, e_i) = \frac{\hat{p}(C=1|e_i, s_i)}{\hat{p}(C=0|e_i, s_i)}$$

Note that we can now replace importance weight in any Propensity-Score-based estimators with this one. If action embedding is not available, e can be one-hot encoding representation of each action. We can see that B-OPE can naturally handle large action space in the same way as MIPS through the use of action embedding.

2 Research Goal

With less required information, B-OPE seems to be a more practical OPE method to implement in real-world systems, compared to MIPS. However, it is still unclear how the performance of these two compared. Thus, the research question is: **Given the same real-world recommendation system settings, how is the performance of B-OPE and MIPS compared with each other?** If the performance of B-OPE is better or worse, but at an acceptable level, then B-OPE should be a go-to method in the industry.

Performance Comparison So far, MIPS and B-OPE have never been compared on the same data and B-OPE has never been tested on real-world recommendation system data before. It is also hard to gauge the performance of MIPS and B-OPE, even individually, for real-world recommendation system setting, since it is very common to have more than 1000 items to recommend in the real-world system, but neither MIPS nor B-OPE has been evaluated on real-world recommendation system data with more than 1000 items. In this project, we will use the self-normalized version of MIPS, called Self-Normalized Marginalize Inverse Propensity Score (SMIPS), which leads to a much lower variance, compared to MIPS.

3 Experiments

3.1 Synthetic Data

The data is simulated using Open Bandit Pipeline [3]. An overview of how the data is generated is as follows:

1. n samples of d_x-dimensional context $x \in \mathcal{X}$ are generated where each dimension are sampled from $\mathcal{N}(0, 1)$ distribution. In a recommendation system, a context can be thought of as user embedding.
2. m samples of d_e-dimensional action embedding $e \in \mathcal{E}$ are generated where each dimension are sampled from $\mathcal{N}(0, 1)$ distribution.
3. The expected reward of each context and action embedding $q(x, e)$ is calculated as:

$$q(x, e) := x^T M_{\mathcal{X},\mathcal{E}} e + \theta_x^T x + \theta_e^T e$$

where $M_{\mathcal{X},\mathcal{E}} \in \mathbb{R}^{d_x \times d_e}, \theta_x \in \mathbb{R}^{d_x}, \theta_e \in \mathbb{R}^{d_e}$ are parameter matrix and vectors, all sampled from the uniform distribution. Note that $q(x, e)$ is unobservable by any models or off-policy evaluators.

4. The logging policy π_0 (which is the policy to generate logged feedback) is created by applying the softmax function to the expected reward, which is amplified by β. More specifically,

$$\pi_0(x, e) = \frac{e^{\beta \cdot q(x,e)}}{\sum_{e' \in \mathcal{E}} e^{\beta \cdot q(x,e')}}$$

Thus, the optimality of the behavior policy can be controlled by adjusting the β parameter. A large value leads to a near-deterministic behavior policy, while a small value leads to a near-uniform behavior policy. A positive value leads to a near-optimal behavior policy, while a negative value leads to a sub-optimal behavior policy.

5. The logged feedback, for each context x, is then generated by sampling action a according to the probability $\pi_0(x, e)$.

3.2 Marginal Importance Weight Estimation $w^{MIPS}(x, e)$

As previously mentioned, an important key for MIPS is to estimate $\pi_{\mathbf{0}}(\mathbf{a}|\mathbf{x_i}, \mathbf{e_i})$, which is done by regressing a on (x_i, e_i). In the original MIPS paper [1], the author assumes the availability of $p(e|a)$ which is used to directly compute $w^{MIPS}(x, e)$, bypassing the need for regression. Since this is impossible in practice, we will perform regression to estimate $\pi_{\mathbf{0}}(\mathbf{a}|\mathbf{x_i}, \mathbf{e_i})$ by using multi-class logistic regress to predict action a, given context-embedding pair (x, e).

3.3 Balanced Importance Weight Estimation $w^{BOPE}(x, e)$

As mentioned in 1.1, the first step to compute $w^{BOPE}(x, e)$ is to build a binary classifier to distinguish between the context-embedding pairs generated from logging and target policy. In this experiment, we use a Neural Network with 3 hidden layers, each with 32 nodes as our binary classifier.

3.4 Target Policies

We test a total of 5 target policies. 2 of them are created by applying softmax to the expected reward, just like the logging policy, but with $\beta = -1, \beta = 5$. 2 of them are greedy strategy, generated by

$$\pi(x, e) = \begin{cases} 1 - \varepsilon + \varepsilon/m & \text{if e is the best action} \\ \varepsilon/m & \text{otherwise} \end{cases}$$

where we use $\varepsilon = 0.2, \varepsilon = 0.6$

The last policy is content-based filtering which we train a binary classification to predict the feedback based on given context and action embedding.

3.5 Evaluation Criteria

In other research of off-policy evaluation methods, RMSE and Bias are used to measure the deviation of the estimated policy value against the true policy value. However, in this project, we will look at the **online-offline plots** to gain deeper understanding of the estimators' performances. Online-Offline plots (see Figure 1) are created by plotting the online vs offline metric of multiple policies. Referring to the example plot (Figure 1) below, the black dotted line represents the ideal line, which is where the offline metric exactly matches the online metric. The blue points are the performance of different policies. Ideally, we would like the blue points to be as close to the ideal line as possible. The deviation from the ideal line represents the Bias. Another important criteria is the **Monotonicity**. Monotonicity is very important in practice because high monotonicity leads to an accurate ranking of policies. In practice, we usually test multiple models offline and would select top k best model, based on the offline performance, to be deployed in production to test them live. If the ranking of policies from offline metric is not accurate, then we will end up deploying models that are not in top k, and thus waste the deployment resources.

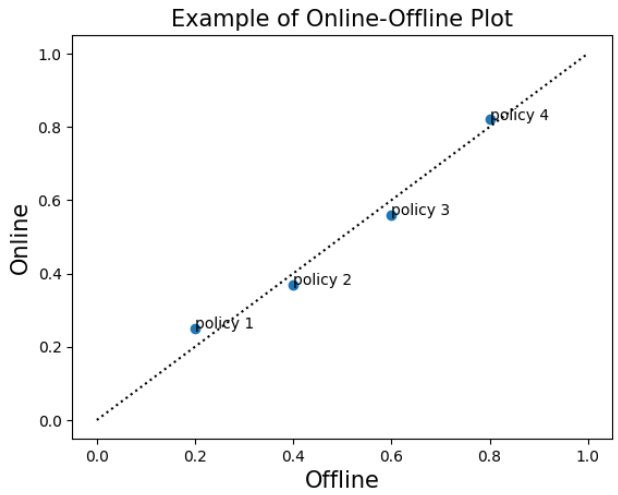

Figure 1: Example of Online-Offline Plot

3.6 Experiment Procedure

1. The synthetic data is generated with 20000 numbers of context c.
2. For more reliable results, we generated 10 bootstrapped dataset from the whole data by sampling 2000 contexts from the whole dataset for 10 times (with replacement).

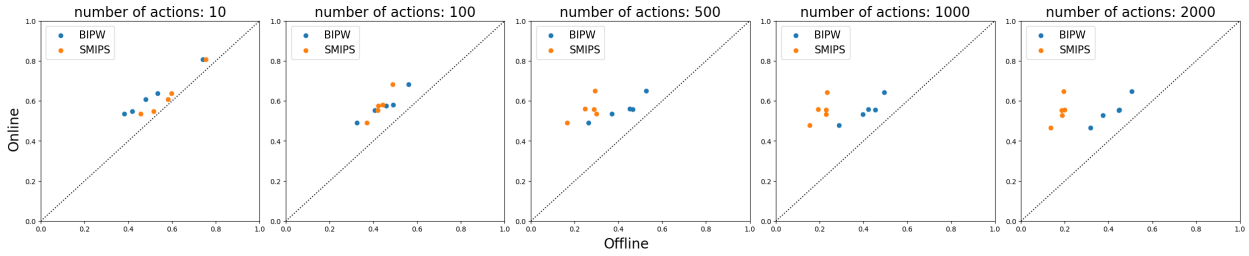

Figure 2: Offline-Online plots of SMIPS vs BOPE at various number of actions

3. for each bootstrapped dataset and for each target policy, we compute the offline metric using BOPE and SMIPS methods, and also online metric using the expected reward which is unobservable in practice.
4. The result is then averaged across different bootstrapped round.

4 Results

4.1 Varying numbers of actions

According to Figure 2, we can see that at low m ($m = 10$), SMIPS performs a little bit better than BOPE because its points are closer to the ideal line. However, as the number of actions m, increases, the performance of SMIPS drops significantly, while BOPE's performance stays relatively the same. For SMIPS, the bias increase significantly with m, and the monotonicity also decrease. This is because SMIPS relies on the estimation of $\pi_{\mathbf{0}}(\mathbf{a}|\mathbf{x_i}, \mathbf{e_i})$, which becomes harder as m gets larger as there are more classes to predict. This shows the very first and important advantage of BOPE, which is BOPE is not sensitive the number of actions, while SMIPS performance drops significantly with higher numbers of actions.

4.2 Varying action embedding dimension

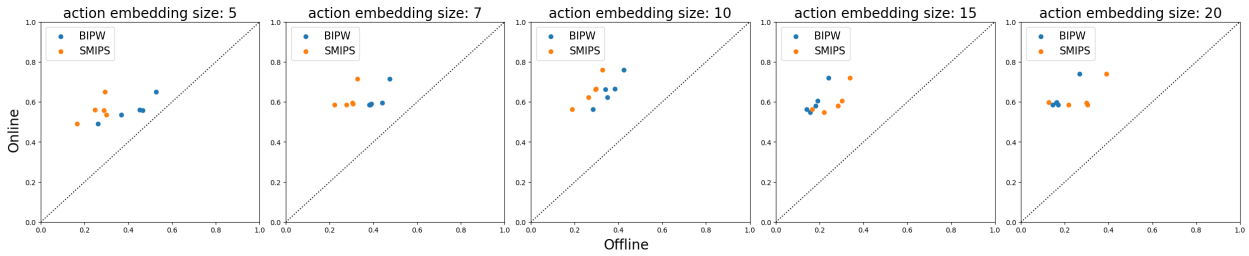

Figure 3: Offline-Online plots of SMIPS vs BOPE at various sizes of action embedding

According to Figure 3, we can see that the bias of both SMIPS and BOPE reduces from having larger dimension of action embedding. This is not surprising since larger size of e, means more features for the classification models to fit. However, it is worth noting that the monotonicity of BOPE does not suffer that much.

References

[1] Yuta Saito and Thorsten Joachims. Off-policy evaluation for large action spaces via embeddings, 2022.

[2] Arjun Sondhi, David Arbour, and Drew Dimmery. Balanced off-policy evaluation in general action spaces, 2020.

[3] Yuta Saito, Shunsuke Aihara, Megumi Matsutani, and Yusuke Narita. Open bandit dataset and pipeline: Towards realistic and reproducible off-policy evaluation, 2021.