Practical Counterfactual Policy Learning for Top-K Recommendations

Yaxu Liu[*†]
National Taiwan University
d08944012@ntu.edu.tw

Jui-Nan Yen[†]
National Taiwan University
juinanyen@gmail.com

Bowen Yuan[†‡]
Amazon
bwyuan@amazon.com

Rundong Shi
Meituan
shirundong@meituan.com

Peng Yan
Meituan
yanpeng04@meituan.com

Chih-Jen Lin
National Taiwan University
cjlin@csie.ntu.edu.tw

ABSTRACT

For building recommender systems, a critical task is to learn a policy with collected feedback (e.g., ratings, clicks) to decide which items to be recommended to users. However, it has been shown that the selection bias in the collected feedback leads to biased learning and thus a sub-optimal policy. To deal with this issue, counterfactual learning has received much attention, where existing approaches can be categorized as either value learning or policy learning approaches. This work studies policy learning approaches for top-K recommendations with a large item space and points out several difficulties related to importance weight explosion, observation insufficiency, and training efficiency. A practical framework for policy learning is then proposed to overcome these difficulties. Our experiments confirm the effectiveness and efficiency of the proposed framework.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

Policy learning, Selection bias, Counterfactual learning, Recommender systems

ACM Reference Format:
Yaxu Liu, Jui-Nan Yen, Bowen Yuan, Rundong Shi, Peng Yan, and Chih-Jen Lin. 2022. Practical Counterfactual Policy Learning for Top-K Recommendations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539295

[*]Work done at Meituan as an intern.
[†]Contribute equally for this work.
[‡]Work done at National Taiwan University.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539295

1 INTRODUCTION

In many applications, users are overwhelmed by a large number of items. To satisfy their needs more efficiently, recommender systems have been constructed to pre-select items according to their preference and some contextual information. Learning a policy to decide which items to be recommended is the core problem of building a recommender system. A widely used approach is to consider rewards (like ratings, clicks, and dwell time) for historical recommendations as labels and solve classification/regression problems to learn a model (e.g., matrix factorization [10], factorization machines [17]) as an estimator of rewards. Then the system decides future recommendations according to the estimated rewards from the learned model.

However, recent works argue that such a learning scheme may lead to biased recommendations. The reason is that the historical recommendations are decided by a previously deployed policy, which is generally referred to as the behavior policy. The mismatch between data distributions of the behavior policy and the new policy introduces the selection bias. To remove the bias, the key idea is to consider rewards for both recommended and non-recommended items. However, the realization of this idea is non-trivial. Unlike rewards for recommended items, rewards for non-recommended items, often known as counterfactual data, are generally not available. Learning an unbiased policy without counterfactual data, which is also referred to as counterfactual learning, is an emerging topic in recommender systems.

Generally speaking, existing approaches of counterfactual learning fall into two groups, value learning approaches [25, 27] and policy learning approaches [9, 19, 20]. A review of them will be given in Section 2.2. Between the two approaches, besides few works (e.g., [8]) that propose combining these approaches for recommender systems, a systematic comparison between these two paradigms is lacking. Our original purpose is to fill this gap. However, when we attempt to implement policy learning approaches for a large-scale top-K recommender system, where K items are recommended for each context, we meet the following challenges.

- The propensity of a top-K recommendation can be very small, resulting in the explosion of importance weights and poor performances of the learned policy.
- For recommender systems with a large item space, it is almost impossible to observe every item for every context. Thus, for some rarer contexts, the empirical reward of each item can be far away from the underlying true reward distribution. In particular, the empirical reward can be wrongly concentrated on one single

Table 1: Main notation.

m, n, K	numbers of contexts, items, and positions
β, π	behavior (deployed) policy and new policy
$\mathbb{A}$	set of n items
$\mathbf{u}, \mathbf{c}$	random variable for the feature vector of a context and a reward vector of n items
$\boldsymbol{V}$	feature matrix of n items
$\mathbf{A}, \mathrm{A}_k$	random variable for a permutation of K items and the item placed at position k
$\mathbf{C}$	random variable for a reward vector of items in $\mathbf{A}$

item if only a few items are observed and one of them is largely amplified by its low propensity weight. This can make the learned policy to be very sharp, a situation which has been pointed out to be harmful to robustness and generalization.
- Training efficiency is a serious issue as optimization problems of policy learning require computations over the entire item space, the size of which can reach over millions in a real-world system.

This work aims to overcome these difficulties with the following contributions.

- We propose a regularized per-item estimator to quell the first issue mentioned above.
- We propose a formulation to handle the probability mass and thus avoid the sharpness of the learned policy.
- We introduce a novel and efficient training method for policy learning. It can be effectively deployed for large-scale top-K recommender systems.

The paper is organized as follows. The preliminary is in Section 2, which includes the problem setting and a review of existing approaches of counterfactual learning. We position the above-mentioned difficulties and propose the corresponding solutions in Section 3. Related works are given in Section 4. In Section 5, a series of experiments confirm our contributions. Finally, Section 6 concludes this work. A list of notations used in this work is in Table 1. Supplementary materials and programs/data used for our experiments are available at https://www.csie.ntu.edu.tw/~cjlin/papers/counterfactual_topk/.

2 PRELIMINARY

In this section, after describing the problem settings, we briefly review existing approaches for counterfactual learning.

2.1 Problem Setting

In this work, we focus on top-K recommender systems with n items (e.g., movies, products). Let $\mathbb{A} = \{1, \ldots, n\}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times D_v}$ be the associated side-feature matrix. The interactions between users and a top-K recommender system can be described as the following procedures.

- When a user visits a webpage with K recommendation positions, a feature vector $\mathbf{u} \in \mathbb{R}^{D_u}$ including the contextual information (e.g., information of a user or a webpage) is sent to the system, where $\mathbf{u}$ follows an underlying distribution $\Pr(\mathbf{u})$.
- Rewards (e.g., clicks, views) $\mathbf{c}$ for all items follow another underlying distribution $\Pr(\mathbf{c} \mid \mathbf{u}; \boldsymbol{V})$, where for each item j, the corresponding c_j depends only on its feature vector $\boldsymbol{v}_j$ in the matrix $\boldsymbol{V}$ and the feature vector $\mathbf{u}$ above. Note that $\boldsymbol{v}_j$ is the jth row of $\boldsymbol{V}$.
- Given the received $\mathbf{u}$, a recommendation policy β deployed in the system recommends K distinct items according to the probability $\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V})$. These items form a K-element permutation of $\mathbb{A}$ as following,

$$\mathbf{A} \in G(\mathbb{A}, K) \equiv \{(\mathrm{A}_1, \ldots, \mathrm{A}_K) \mid \mathrm{A}_1, \ldots, \mathrm{A}_K \in \mathbb{A}; \mathrm{A}_i \neq \mathrm{A}_j, \forall i \neq j\}. \tag{1}$$

- Once items in $\mathbf{A}$ are recommended to users, their corresponding rewards $\mathbf{C} = (\mathrm{c}_{\mathrm{A}_1}, \ldots, \mathrm{c}_{\mathrm{A}_K})$ are revealed to the system, while rewards of other non-recommended items are still unknown.

With the repetition of the above procedures, the system can collect a set $\mathbb{S}$, which includes vast logged events $(\mathbf{u}, \mathbf{A}, \mathbf{C})$. One of the most crucial tasks of building a recommender system is to learn a new policy π so that the probability distribution $\pi(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}, \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ maximizes the expected cumulated reward

$$V^{\pi} = \mathbb{E}_{P_\pi}\Big[r(\mathbf{c}, \mathbf{A})\Big], \tag{2}$$

where

$$\mathbb{E}_{P_\pi}[\cdot] = \mathbb{E}_{\Pr(\mathbf{u})}\mathbb{E}_{\pi(\mathbf{A}|\mathbf{u};\boldsymbol{V},\boldsymbol{\theta})}\mathbb{E}_{\Pr(\mathbf{c}|\mathbf{u};\boldsymbol{V})}[\cdot] \tag{3}$$

and

$$r(\mathbf{c}, \mathbf{A}) = \sum\nolimits_{k=1}^{K} \mathrm{c}_{\mathrm{A}_k} \tag{4}$$

is the cumulated reward from $\mathbf{A}$. To distinguish π from the previous policy β, the latter is also called the behavior policy. A natural setting to obtain a useful π is by using the historical events in $\mathbb{S}$. However, the challenge comes from the fact that directly maximizing the expected reward in (2) requires rewards $\mathbf{c}$ for all items. Unfortunately, among all $\frac{n!}{(n-K)!}$ permutations as the candidate of $\mathbf{A}$, we only observe the one recommended by β for each context $\mathbf{u}$ in $\mathbb{S}$. Opposed to the logged events in $\mathbb{S}$, these events of non-recommended permutations are called counterfactual data. Conducting policy learning with partial rewards in logged events is also referred to as counterfactual learning or batch learning from logged bandit feedback [19].

2.2 A Review on Existing Approaches of Counterfactual learning

Existing approaches of counterfactual learning can be roughly categorized into two groups, the value learning approach and the policy learning approach. They differ on explicitly or implicitly learning the new policy π. Subsequently, we give a brief review of them.

2.2.1 Value Learning Approaches. The main idea of value learning methods is to learn a reward estimation model $\hat{s}(\cdot)$ parametrized by $\boldsymbol{\theta}$ to model the reward probability $\Pr(\mathbf{c} \mid \mathbf{u}; \boldsymbol{V})$. To have an unbiased estimate, it has been pointed out [25] that we should minimize the following full-labeled risk

$$L(\boldsymbol{\theta}) = \mathbb{E}_{\Pr(\mathbf{u})}\mathbb{E}_{\Pr(\mathbf{c}|\mathbf{u};\boldsymbol{V})}[\sum\nolimits_{j=1}^{n} \ell(\mathrm{c}_j, \hat{s}(\boldsymbol{\theta}; \mathbf{u}, \boldsymbol{v}_j))], \tag{5}$$

where $\boldsymbol{v}_j$ is the feature vector of the jth item extracted from row j of $\boldsymbol{V}$, and $\ell(a, b)$ is a loss function (e.g., logistic loss or squared loss). However, for each event $(\mathbf{u}, \mathbf{A}, \mathbf{C})$, we only have rewards contained in $\mathbf{C}$, which correspond to those selected items included in $\mathbf{A}$. It has been shown [25, 27] that because the behavior policy $\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V})$

non-uniformly selects items with the highest reward, considering a modified form of (5) with only items included in $\mathbf{A}$ results in a heavily biased $\hat{s}(\cdot)$.

For addressing the bias, a commonly used way is to apply an inverse-propensity-score (IPS) method [5]. The main idea is to re-weigh the loss term of each recommended item by its inverse propensity score such that the following unbiased risk is minimized

$$L_{\mathrm{IPS}}(\boldsymbol{\theta}) = \mathbb{E}_{\Pr(\mathbf{u})}\mathbb{E}_{\beta(\mathbf{A}|\mathbf{u};\boldsymbol{V})}\mathbb{E}_{\Pr(\mathbf{c}|\mathbf{u};\boldsymbol{V})}\Big[\sum_{k=1}^{K}\frac{\ell(\mathrm{c}_{\mathrm{A}_k}, \hat{s}(\boldsymbol{\theta};\mathbf{u},\boldsymbol{v}_{\mathrm{A}_k}))}{z(\mathrm{A}_k \mid \mathbf{u};\boldsymbol{V})}\Big], \tag{6}$$

where $z(\mathrm{A}_k \mid \mathbf{u};\boldsymbol{V})$ is the propensity score, which is a value proportional to the probability of the item being recommended at the position. For $z(\mathrm{A}_k \mid \mathbf{u};\boldsymbol{V})$, we can infer it with the behavior policy β as follows

$$z(\mathrm{A}_k \mid \mathbf{u};\boldsymbol{V}) = \sum\nolimits_{\hat{\mathbf{A}}\in G(\mathbb{A},K)} \beta(\hat{\mathbf{A}} \mid \mathbf{u};\boldsymbol{V})\mathbb{1}[\mathrm{A}_k = \hat{\mathrm{A}}_k], \tag{7}$$

where $\mathbb{1}[\cdot]$ is the indicator function. In some case, when the behavior policy β is not reserved, we need to estimate $z(\mathrm{A}_k \mid \mathbf{u};\boldsymbol{V})$ from $\mathbb{S}$. Unfortunately, it is known that accurately estimating $z(\mathrm{A}_k \mid \mathbf{u};\boldsymbol{V})$ is not an easy task. Therefore, some recent works (e.g., [25, 27]) extend (6) to minimize the following doubly robust risk [4] for mitigating the issue caused by poor estimates of propensity scores.

$$\begin{aligned} L_{\mathrm{DR}}(\boldsymbol{\theta}) = \mathbb{E}_{\Pr(\mathbf{u})}\mathbb{E}_{\beta(\mathbf{A}|\mathbf{u};\boldsymbol{V})}\mathbb{E}_{\Pr(\mathbf{c}|\mathbf{u};\boldsymbol{V})}\Big[\gamma\sum\nolimits_{j=1}^{n}\bar{\ell}(\hat{\mathrm{c}}_j, \hat{s}(\boldsymbol{\theta};\mathbf{u},\boldsymbol{v}_j)) \\ + \sum\nolimits_{k=1}^{K}\frac{\ell(\mathrm{c}_{\mathrm{A}_k}, \hat{s}(\boldsymbol{\theta};\mathbf{u},\boldsymbol{v}_{\mathrm{A}_k})) - \gamma\ell(\hat{\mathrm{c}}_{\mathrm{A}_k}, \hat{s}(\boldsymbol{\theta};\mathbf{u},\boldsymbol{v}_{\mathrm{A}_k}))}{z(\mathrm{A}_k \mid \mathbf{u};\boldsymbol{V})}\Big], \end{aligned} \tag{8}$$

where $\hat{\mathrm{c}}_j$ is an imputed reward of item j and $\bar{\ell}(\cdot)$ is the loss function for the imputation part. Because the imputed rewards of those non-recommended items are less reliable than recommended items, a small user-specified hyper-parameter γ is applied to balance the two parts.

The reward estimation model $\hat{s}(\cdot)$ is learned from the following minimization problem

$$\min_{\boldsymbol{\theta}} \quad L_{\mathrm{IPS}}(\boldsymbol{\theta}) (\text{or } L_{\mathrm{DR}}(\boldsymbol{\theta})) + \lambda\Phi(\boldsymbol{\theta}), \tag{9}$$

where $\Phi(\boldsymbol{\theta})$ is a regularizer to avoid overfitting (e.g., l2 regularizer), and λ is a pre-specified regularization coefficient. Then, the optimal $\pi(\mathbf{A} \mid \mathbf{u};\boldsymbol{V},\boldsymbol{\theta})$ can be obtained by selecting the top-K items with the highest reward estimates from $\hat{s}(\cdot)$. However, it has been pointed out [25] that such a deterministic policy causes difficulties for optimizing future policies. Thus the ϵ-greedy method is always applied to impose stochasticity. That is, we greedily select the item with the highest reward estimates with probability $1-\epsilon$, and uniformly draw a random item with probability ϵ.

2.2.2 Policy Learning Approaches. Instead of estimating rewards, policy learning approaches [4, 19] directly find an optimal $\pi(\mathbf{A} \mid \mathbf{u};\boldsymbol{V},\boldsymbol{\theta})$ to maximize V^{π} in (2). Similar to value learning approaches discussed in the previous section, since $\mathbb{S}$ is generally collected by a non-uniform β, we should apply the inverse propensity weighting method to correct the distribution mismatching between β and π. This leads to the following estimate of V^{π},

$$V^{\pi}_{\mathrm{IPS}} = \mathbb{E}_{P_\beta}\Big[\mathrm{w}_{\mathbf{A}} r(\mathbf{c},\mathbf{A})\Big], \tag{10}$$

where

$$\mathbb{E}_{P_\beta}[\cdot] = \mathbb{E}_{\Pr(\mathbf{u})}\mathbb{E}_{\beta(\mathbf{A}|\mathbf{u};\boldsymbol{V})}\mathbb{E}_{\Pr(\mathbf{c}|\mathbf{u};\boldsymbol{V})}[\cdot] \tag{11}$$

and

$$\mathrm{w}_{\mathbf{A}} = \frac{\pi(\mathbf{A} \mid \mathbf{u};\boldsymbol{V},\boldsymbol{\theta})}{\beta(\mathbf{A} \mid \mathbf{u};\boldsymbol{V})}. \tag{12}$$

It is known that V^{π}_{IPS} is an unbiased estimate of V^{π} under the following assumption. The proof can be found in our supplementary materials.

Assumption 1. *For any* $\mathbf{A}$ *and* $\mathbf{u}$, $\pi(\mathbf{A} \mid \mathbf{u};\boldsymbol{V},\boldsymbol{\theta}) \neq 0$ *only if* $\beta(\mathbf{A} \mid \mathbf{u};\boldsymbol{V}) \neq 0$.

It indicates that our new policy π only considers items with non-zero probability in our behavior policy β.

Thus we can solve the following problem to learn π,

$$\max_{\boldsymbol{\theta}} \quad R(\boldsymbol{\theta}) = V^{\pi}_{\mathrm{IPS}}(\boldsymbol{\theta}) - \lambda\Phi(\boldsymbol{\theta}), \tag{13}$$

where similar to (5), a regularizer $\Phi(\boldsymbol{\theta})$ is considered to avoid overfitting. To solve (13), the following policy gradient method is applied,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta\nabla R(\boldsymbol{\theta}), \tag{14}$$

where η is a specified learning rate. In (14), $\nabla R(\boldsymbol{\theta})$ is computed as follows,

$$\begin{aligned} \nabla R(\boldsymbol{\theta}) =& \mathbb{E}_{P_\beta}\left[r(\mathbf{c},\mathbf{A})\frac{\nabla\pi(\mathbf{A} \mid \mathbf{u};\boldsymbol{V},\boldsymbol{\theta})}{\beta(\mathbf{A} \mid \mathbf{u};\boldsymbol{V})}\right] - \lambda\nabla\Phi(\boldsymbol{\theta}) \\ =& \mathbb{E}_{P_\beta}\left[r(\mathbf{c},\mathbf{A})\frac{\pi(\mathbf{A} \mid \mathbf{u};\boldsymbol{V},\boldsymbol{\theta})}{\beta(\mathbf{A} \mid \mathbf{u};\boldsymbol{V})}\nabla\log\pi(\mathbf{A} \mid \mathbf{u};\boldsymbol{V},\boldsymbol{\theta})\right] - \lambda\nabla\Phi(\boldsymbol{\theta}), \end{aligned} \tag{15}$$

where the second equality is from the following chain rule,

$$\nabla\log\pi = \frac{1}{\pi}\nabla\pi. \tag{16}$$

3 A PRACTICAL FRAMEWORK FOR POLICY LEARNING

Except a recent work [8] that proposes a new model that combines approaches of both value learning and policy learning for recommender systems, few studied these two paradigms together in this field. Thus the original purpose of this work is to detailedly compare them for recommendation problems. However, to implement them for a large-scale top-K recommender system, where both the number of logged events and the number of items are tremendous, we find some extra difficulties in policy learning approaches. In this section, after positioning a difficulty in each of the following subsections, we propose the corresponding solution. Finally, integrating these solutions leads to a practical framework of policy learning for top-K recommendations.

3.1 Regularized Per-item IPS Estimator

3.1.1 Weight Explosion in IPS Estimators. In practice, given finite historical events logged by β, we generally derive an unbiased estimate of V^{π}_{IPS} via Monte Carlo approximation. Expressly, assuming that the collected set includes m logged events, we maximize the following estimate of V^{π}_{IPS}.

$$\hat{V}^{\pi}_{\mathrm{IPS}}(\boldsymbol{\theta}) = \frac{1}{m}\sum\nolimits_{i=1}^{m}\mathrm{w}^{i}_{\mathbf{A}} r(\mathbf{c}_i,\mathbf{A}_i). \tag{17}$$

The quality of $\hat{V}^{\pi}_{\mathrm{IPS}}$ estimating V^{π} deeply relies on the range of $\mathrm{w}_{\mathbf{A}}$. To explain this, we make the following assumptions.

Assumption 2. *For any* $\mathbf{A}$ *and* $\mathbf{u}$*, the reward* $\mathrm{c_A}$ *for every* $\mathrm{A} \in \mathbf{A}$ *is bounded in* $[0, 1]$.

Assumption 3. *For any* $\mathbf{A}$ *and* $\mathbf{u}$*, if* $\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}) \neq 0$*, then*

$$\mathrm{w}_{\mathbf{A}} = \frac{\pi(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}, \boldsymbol{\theta})}{\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V})} \leq w_{max}.$$

Then, we give the following theorem.[1]

Theorem 1. *Given a finite policy class* $\mathcal{H}$ *with* $|\mathcal{H}| = N$*, we assume that Assumptions 1-3 are all satisfied. Then, with probability at least* $1 - \delta$*, we have*

$$\sup_{\pi \in \mathcal{H}} |\hat{V}^{\pi}_{IPS} - V^{\pi}| \leq K w_{max} \sqrt{\frac{2}{m} \log \frac{2N}{\delta}}.$$

From Theorem 1, we conclude that the quality of $\hat{V}^{\pi}_{\mathrm{IPS}}$ can suffer when the range of $\mathrm{w}_{\mathbf{A}}$ is large. Unfortunately, this situation commonly occurs for large-scale top-K recommender systems. The leading cause of this issue is that the number of possible item permutations $\mathbf{A}$ is an extremely large number $n!/(n-K)!$. As a consequence, for most item permutations, both $\pi(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}, \boldsymbol{\theta})$ and $\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V})$ can be very small. During the training procedure, any slight disagreement between $\pi(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}, \boldsymbol{\theta})$ and $\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V})$ may cause an explosion of $\mathrm{w}_{\mathbf{A}}$. This conjecture has been confirmed in a real-world top-K advertising system [11]. When K grows, by increasing the disagreement between $\pi(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}, \boldsymbol{\theta})$ and $\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V})$, they demonstrate a vast gap between $\hat{V}^{\pi}_{\mathrm{IPS}}$ and V^{π}.

3.1.2 Weight Factorization and Pruning. The explosion of $\mathrm{w}_{\mathbf{A}}$ is caused by the massive number of possible permutations of K items. Thus, for avoiding this issue, an intuitive idea is to assume that $\pi(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}, \boldsymbol{\theta})$ and $\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V})$ can be respectively factorized as

$$\pi(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}, \boldsymbol{\theta}) = \prod_{k=1}^{K} \pi(\mathrm{A}_k \mid \mathbf{u}, \mathbf{A}_{1:k-1}; \boldsymbol{V}, \boldsymbol{\theta}), \tag{18}$$

and

$$\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}) = \prod_{k=1}^{K} \beta(\mathrm{A}_k \mid \mathbf{u}, \mathbf{A}_{1:k-1}; \boldsymbol{V}). \tag{19}$$

This decomposes the decision of $\mathbf{A}$ into sequential decisions of A_k in K sub-contexts. Now each sub-context consists of not only $\mathbf{u}$ but also items $\mathbf{A}_{1:k-1}$ that have been recommended beforehand. From (4), we can rewrite (10) as

$$V^{\pi}_{\mathrm{IPS}} = \mathbb{E}_{P_\beta}\left[\mathrm{w}_{\mathbf{A}}\left(\sum_{k=1}^{K} \mathrm{c}_{\mathrm{A}_k}\right)\right] = \mathbb{E}_{P_\beta}\left[\sum_{k=1}^{K} \mathrm{w}_{\mathbf{A}} \mathrm{c}_{\mathrm{A}_k}\right]. \tag{20}$$

From (18) and (19), we can rewrite (12) as

$$\mathrm{w}_{\mathbf{A}} = \prod_{k=1}^{K} \mathrm{w}_k, \tag{21}$$

where

$$\mathrm{w}_k = \frac{\pi(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V}, \boldsymbol{\theta})}{\beta(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V})} \text{ and } \hat{\mathbf{u}}_k = [\,{}^{\mathbf{u}}_{\mathbf{A}_{1:k-1}}\,] \tag{22}$$

are respectively the importance weight and the feature vector of the kth sub-context.

However, an explosive $\mathrm{w}_{\mathbf{A}}$ may still occur as it involves the product of K unbounded w_k. To address the issue, our idea is to prune w_k included in $\mathrm{w}_{\mathbf{A}}$. The following theorem shows that under certain conditions, pruning $\prod_{j=k+1}^{K} \mathrm{w}_j$ will not change the unbiasedness of the estimate.[2]

Theorem 2. *Assume that*

$$P_\beta(\mathrm{c}_{\mathrm{A}_k} | \mathbf{u}, \mathbf{A}_{1:k}, \mathbf{A}_{k+1:K}) = P_\beta(\mathrm{c}_{\mathrm{A}_k} | \mathbf{u}, \mathbf{A}_{1:k}).$$

The resulting estimate

$$\mathbb{E}_{P_\beta}\left[\sum_{k=1}^{K} (\prod_{j=1}^{k} \mathrm{w}_j) \mathrm{c}_{\mathrm{A}_k}\right]$$

after pruning $\prod_{j=k+1}^{K} \mathrm{w}_j$ *is still unbiased to* V^{π}.

According to the problem setting described in Section 2.1, we assume c_{A_k} is only dependent on $\mathbf{u}$ and $\boldsymbol{v}_{\mathbf{A}_k}$, so the conditions required in Theorem 2 can hold for our considered scenarios. Thus we can prune $\prod_{j=k+1}^{K} \mathrm{w}_j$ in (20) without introducing any bias. Next we discuss how to prune $\prod_{j=1}^{k-1} \mathrm{w}_{\mathrm{A}_j}$. Let

$$V^{\pi}_{\mathrm{IIPS}} = \mathbb{E}_{P_\beta}[\sum_{k=1}^{K} \mathrm{w}_k \mathrm{c}_{\mathrm{A}_k}] \tag{23}$$

be the resulting estimator after pruning $\prod_{j=1}^{k-1} \mathrm{w}_j$. From (18), (19), and (21),

$$\begin{aligned} |V^{\pi}_{\mathrm{IPS}} - V^{\pi}_{\mathrm{IIPS}}| &= \mathbb{E}_{P_\beta}[\sum_{k=1}^{K} \left| \prod_{j=1}^{k-1} \mathrm{w}_j - 1 \right| \mathrm{w}_k \mathrm{c}_{\mathrm{A}_k}] \\ &= \mathbb{E}_{P_\beta}[\sum_{k=1}^{K} \frac{\left|\pi(\mathbf{A}_{1:k-1} \mid \mathbf{u}; \boldsymbol{V}, \boldsymbol{\theta}) - \beta(\mathbf{A}_{1:k-1} \mid \mathbf{u}; \boldsymbol{V})\right|}{\beta(\mathbf{A}_{1:k-1} \mid \mathbf{u}; \boldsymbol{V})} \mathrm{w}_k \mathrm{c}_{\mathrm{A}_k}]. \end{aligned} \tag{24}$$

This then implies that the loss of pruning $\prod_{j=1}^{k-1} \mathrm{w}_j$ can be bounded by the difference between π and β. Intuitively, as long as π imitates β, the product can be pruned without any loss. However, any improvement of π over β can not be achieved, which is contradictory to the purpose of policy optimization. Instead, we alleviate the loss caused by pruning $\prod_{j=1}^{k-1} \mathrm{w}_j$ through constraining the difference between π and β. Consider Pinsker's inequality defined as follows

$$\sup |\pi - \beta| \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(\beta \,||\, \pi)}, \tag{25}$$

where $D_{\mathrm{KL}}(\beta \,||\, \pi)$ is the Kullback-Leibler divergence between π and β.

$$\begin{aligned} D_{\mathrm{KL}}(\beta \,||\, \pi) &= -\mathbb{E}_{\Pr(\mathbf{u})} \sum_{\mathbf{A}} \beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}) \log(\frac{\pi(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}, \boldsymbol{\theta})}{\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V})}) \\ &= -\mathbb{E}_{\Pr(\mathbf{u})} \mathbb{E}_{\beta(\mathbf{A}|\mathbf{u}; \boldsymbol{V})} \log(\frac{\pi(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V}, \boldsymbol{\theta})}{\beta(\mathbf{A} \mid \mathbf{u}; \boldsymbol{V})}) \\ &= -\mathbb{E}_{\Pr(\mathbf{u})} \mathbb{E}_{\beta(\mathbf{A}|\mathbf{u}; \boldsymbol{V})} \sum_{k=1}^{K} \log(\frac{\pi(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V}, \boldsymbol{\theta})}{\beta(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V})}). \end{aligned} \tag{26}$$

The last equality in (26) follows from (18) and (19). Then, the task of alleviating the loss caused by pruning $\prod_{j=1}^{k-1} \mathrm{w}_j$ can be accomplished by bounding $D_{\mathrm{KL}}(\beta \,||\, \pi)$. Therefore, we consider to bound $D_{\mathrm{KL}}(\beta \,||\, \pi)$ as an additional constraint by the method of adding a penalty function and propose the following regularized per-item IPS (RIIPS) estimator.

$$V^{\pi}_{\mathrm{RIIPS}} = V^{\pi}_{\mathrm{IIPS}} - \alpha D_{\mathrm{KL}}(\beta \,||\, \pi), \tag{27}$$

[1] See the proof in our supplementary materials.

[2] See the proof in our supplementary materials.

where α is the penalty hyper-parameter decided by a validation procedure. Finally, for policy learning, by equipping the regularization term, we achieve the following problem.

$$\max_{\boldsymbol{\theta}} \quad \mathbb{E}_{P_\beta}\left[\sum_{k=1}^{K}\left(\frac{\pi(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V}, \boldsymbol{\theta})}{\beta(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V})}\mathrm{c}_{\mathrm{A}_k} + \alpha \log(\pi(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V}, \boldsymbol{\theta}))\right)\right] - \lambda\Phi(\boldsymbol{\theta}), \tag{28}$$

where β in $D_{\mathrm{KL}}(\beta \,||\, \pi)$ has been omitted due to its independence to $\boldsymbol{\theta}$. It is straightforward to solve the problem by the policy gradient method in (14). Specifically, with the trick defined in (16), the gradient of the objective function in (28) can be computed as

$$\mathbb{E}_{P_\beta}\left[\sum_{k=1}^{K}\left((\frac{\pi(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V}, \boldsymbol{\theta})}{\beta(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V})}\mathrm{c}_{\mathrm{A}_k} + \alpha)\nabla \log(\pi(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V}, \boldsymbol{\theta}))\right)\right] + \lambda\nabla\Phi(\boldsymbol{\theta}). \tag{29}$$

Now, it comes to how to model π. Let $\hat{y}(\cdot)$ be a score function parametrized by $\boldsymbol{\theta}$. To guarantee that the $\mathbf{A}$ resulting from π is a K-element permutation of $\mathbb{A}$, we apply the Plackett-Luce (PL) model [15], which requires to exclude items recommended beforehand for each sub-context. Recalling (22), we can divide each sub-context $\hat{\mathbf{u}}_k$ into two parts: $\mathbf{u}$ for the inference of $\hat{y}(\cdot)$ and $\mathbf{A}_{1:k-1}$ for item exclusion. Then, with the softmax function, π is defined as,

$$\pi(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V}, \boldsymbol{\theta}) = \frac{e^{\hat{y}(\boldsymbol{\theta}; \mathbf{u}, \boldsymbol{v}_{\mathrm{A}_k})}}{\sum_{j=1, j\notin\mathbf{A}_{1:k-1}}^{n} e^{\hat{y}(\boldsymbol{\theta}; \mathbf{u}, \boldsymbol{v}_j)}}. \tag{30}$$

According to $\mathbf{A}_{1:k-1}$, items recommended previously are explicitly excluded from the item space $\mathbb{A}$. For practical convenience, instead of (30), during training we can use the following formulation,

$$\pi(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V}, \boldsymbol{\theta}) = \frac{e^{\hat{y}(\boldsymbol{\theta}; \hat{\mathbf{u}}_k, \boldsymbol{v}_{\mathrm{A}_k})}}{\sum_{j=1}^{n} e^{\hat{y}(\boldsymbol{\theta}; \hat{\mathbf{u}}_k, \boldsymbol{v}_j)}}, \tag{31}$$

which does not explicitly exclude $\mathbf{A}_{1:k-1}$ but instead includes $\mathbf{A}_{1:k-1}$ as additional contextual features for $\hat{y}(\cdot)$. The PL model is then only used for generating a K-element permutation of $\mathbb{A}$ during the online deployment. Empirically, the models trained with (30) and (31) have similar performance in our experiments.

3.2 Adaptive Policy Learning

The main idea of IPS and its derivatives is to divide the observed reward $\mathrm{c}_{\mathrm{A}_k}$ by the propensity β, so that in expectation this adjusted empirical reward will match the underlying reward distribution. As described in [18], IPS generally works when the behavior policy is stochastic and a sufficient amount of events are observed.

However, for recommender systems, the context feature $\mathbf{u}$ is often very sparse and high-dimensional. This makes some contexts very rare while the number of items is usually very large. Consequently, it is almost impossible to go through every item for every context. In fact, it is more likely that we only observed a small fraction of items for each context.

To see why this can be bad for IPS, we give a simple example in Figure 1, which only involves two items and two independent contexts. Specifically, we assume the first context appears more frequently than the second one. Under these two contexts, the empirical rewards of the two items are observed. We show the normalized empirical reward in Figure 1(a).

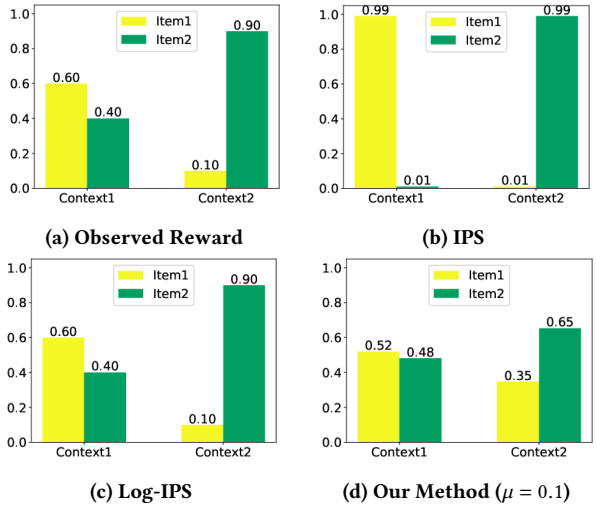

Figure 1: A toy example with two contexts and two items.

From Figure 1(b), we can see that for both contexts, the learned policy from IPS almost puts the probability mass entirely on the item with the highest empirical reward, and the other item is simply ignored. This is known as the winner-takes-all effect, and it has been pointed out [8] that the effect can be harmful to the robustness and generalization of the learned policy from IPS. The issue is particularly acute when we do not have enough observations, and as in this case, the item with the highest empirical reward might not actually be the best one.

To handle the winner-takes-all effect, [8] proposes the so-called log-IPS as below. By applying the log transformation to our RIIPS and switching to a minimization setting, (28) becomes

$$\min_{\boldsymbol{\theta}} \quad \mathbb{E}_{P_\beta}[\sum_{k=1}^{K}\left(-\mathrm{Z}_{\mathrm{A}_k}\log(\pi(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V}, \boldsymbol{\theta}))\right)] + \lambda\Phi(\boldsymbol{\theta}), \tag{32}$$

where

$$\mathrm{Z}_{\mathrm{A}_k} = \frac{\mathrm{c}_{\mathrm{A}_k}}{\beta(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V})} + \alpha. \tag{33}$$

According to [8], the log transformation in (32) moves the global optimum and thus makes the learned policy from log-IPS allocate probability mass proportional to the normalized empirical reward in Figure 1(a). This results in what is shown in Figure 1(c). Thus, if the empirical reward spreads across multiple items with enough observations, then log-IPS can alleviate the winner-takes-all effect successfully. For example, as the first context occurs frequently, both items have been observed for several times and the ratio of their empirical reward becomes even and stable. A comparison between Figure 1(c) and Figure 1(b) shows that log-IPS makes good use of this even ratio while IPS does not.

However, we argue that for contexts with few occurrences, it is likely that one item will wrongly occupy a large portion of the empirical reward. The reason behind this is that when one of the many lower propensity items is selected, its empirical reward will be largely amplified by the inverse propensity scaling. If we do not have enough observations, this will result in a high concentration of empirical rewards as the second context in Figure 1(a). In this

case, log-IPS does not help much, as the probability mass of the learned policy is still highly concentrated on one single item.

To address this problem, our idea is to introduce a hyper-parameter which can further smoothen the probability mass of the learned policy. However, we find it difficult to introduce such a hyper-parameter in (32), so an alternative formulation is needed. To this end, we connect (32) to the multi-class classification. A closer look at (32) shows that if the softmax function is used for π as in (31), then $-\log \pi$ in (32) is the well-known cross-entropy loss function commonly applied for multi-class classification problems. Specifically, (31) is the probability for an instance to be in class A_k and (32) can be considered as to minimize a weighted negative log-likelihood.

A common alternative used for multi-class classification problems is the one-versus-all scheme. In this setting, a binary probability model is considered for each class. That is, the recommended item A_k is seen as positive and the non-recommended items are seen as negative. Then, for class A_k, the following probability model is considered

$$\Pr(\text{positive} \mid \hat{y}(\boldsymbol{\theta}; \hat{\mathbf{u}}_k, \boldsymbol{v}_{\mathrm{A}_k})) = \sigma(\hat{y}(\boldsymbol{\theta}; \hat{\mathbf{u}}_k, \boldsymbol{v}_{\mathrm{A}_k})), \tag{34}$$

where $\sigma(\cdot)$ is the sigmoid function. In the one-versus-all setting, each binary problem corresponds to its own parameter $\boldsymbol{\theta}_{\mathrm{A}_k}$, so an independent optimization problem to minimize the negative log-likelihood of (34) is solved.

Keeping this process in mind, here we still consider one $\boldsymbol{\theta}$ for all binary models and propose the following formulation,

$$\begin{aligned}\min_{\boldsymbol{\theta}} \quad & \mathbb{E}_{P_\beta} \sum_{k=1}^{K} \mathrm{Z}_{\mathrm{A}_k} \left[\ell^+_{\log}(\hat{y}(\boldsymbol{\theta}; \hat{\mathbf{u}}_k, \boldsymbol{v}_{\mathrm{A}_k})) + \mu \sum_{j=1, j\neq \mathrm{A}_k}^{n} \ell^-_{\log}(\hat{y}(\boldsymbol{\theta}; \hat{\mathbf{u}}_k, \boldsymbol{v}_j)) \right] \\ & + \lambda \Phi(\boldsymbol{\theta}),\end{aligned} \tag{35}$$

where

$$\ell^+_{\log}(x) = \log \sigma(x) \text{ and } \ell^-_{\log}(x) = \log(1 - \sigma(x)), \tag{36}$$

and μ is a hyper-parameter explained below. This formulation sums over all the binary classification problems in the one-versus-all scheme. To construct π, we must combine all n probability models in (34) into one. This issue has been well studied in the literature of the one-versus-all multi-class classification; see, for example, Section 4.1 in [7]. Here, instead of using some sophisticated settings, we use the following heuristic to construct π.

$$\pi(\mathrm{A}_k \mid \hat{\mathbf{u}}; \boldsymbol{V}, \boldsymbol{\theta}) = \frac{\sigma(\hat{y}(\boldsymbol{\theta}; \mathbf{u}, \boldsymbol{v}_{\mathrm{A}_k}))}{\sum_{j=1, j\notin \mathbf{A}_{1:k-1}}^{n} \sigma(\hat{y}(\boldsymbol{\theta}; \mathbf{u}, \boldsymbol{v}_j))}, \tag{37}$$

which also explicitly excludes $\mathbf{A}_{1:k-1}$ as in (30).

One major benefit of (35) is that the loss function is now decoupled for the positive and negative labels, and the hyper-parameter μ can be naturally added in front of $\ell^-_{\log}(\cdot)$ to control the smoothness of the final probability distribution. As we can see in Figure 1(d), if we set $\mu = 0.1$, this new setting can alleviate the winner-takes-all effect for both the first and the second contexts. In practice, we can use a grid search to find a suitable value for μ, and the probability mass of the learned policy will then adapt accordingly. We thus name our approach Adaptive-RIIPS, as it can adapt the smoothness of the learned policy to address the winner-takes-all effect.

3.3 Efficient Training

Another critical challenge of policy learning approaches for recommender systems is the training efficiency, which is rarely discussed in past works. As we mentioned in Section 2.2.2, the policy gradient method is applied to solve a policy learning problem. For each context, this involves the calculation of all possible decisions of π. For (13), the number of possibilities is equal to the number of all K-element permutations of $\mathbb{A}$, which has the order of $O(\frac{n!}{(n-K)!})$. Clearly, for any top-K recommender system with a tremendous number of contexts and items, solving (13) through the policy gradient method is infeasible.

As for the RIIPS estimator proposed in Section 3.1, we decompose the K-element permutation decision for each context into K sequential decisions for K sub-contexts. Thus π is only responsible for deciding a single item A_k from n items for each sub-context, and the decision space of π is remarkably reduced to n. Consequently, given a training set including m logged events, the cost of computing (29) by going through all events becomes $\mathbb{O}(Mn)$, where $M = mK$ is the number of sub-contexts after decomposition.

However, since both M and n can reach over millions in recommender systems, the training time of (29) is still prohibitive with the $\mathbb{O}(Mn)$ cost. This difficulty has been reported and addressed in recent works [3, 13] of policy learning with a large action (item) space. Their main idea is to reduce the cost by subsampling items from $\mathbb{A}$. But just as reported in these works, this kind of subsampling mechanism often degenerates the performance. To have a better trade-off between the training efficiency and the performance, we propose a novel solution in this work based on (35), which considers a non-subsampled training setting. As shown in [23], such a non-subsampled setting empirically leads to less performance loss than the subsampled one.

Define

$$\hat{\boldsymbol{u}}_i = [\, {}^{\boldsymbol{u}_l}_{\mathbf{A}_{1:k-1}} \,],$$

where $i = (l-1) \times K + k$, to be the kth sub-context of the lth context realization $\boldsymbol{u}_l$. Thus, for $M = mK$ sub-contexts after the decomposition from m contexts, we derive the following empirical formulation of (35).

$$\min_{\boldsymbol{\theta}} \quad \frac{1}{m} \sum\nolimits_{i=1}^{M} z_i \left[\ell^+_{\log}(\hat{Y}_{ia_i}) + \mu \sum\nolimits_{j=1, j\neq a_i}^{n} \ell^-_{\log}(\hat{Y}_{ij}) \right] + \lambda \Phi(\boldsymbol{\theta}), \tag{38}$$

where $\hat{Y}_{ij} = \hat{y}(\boldsymbol{\theta}; \hat{\boldsymbol{u}}_i, \boldsymbol{v}_j)$, $\boldsymbol{a} \in \mathbb{A}^M$ and $\boldsymbol{z} \in \mathbb{R}^M$. Here $\boldsymbol{a}$ and $\boldsymbol{z}$ contain realizations of A_k and $\mathrm{Z}_{\mathrm{A}_k}$ for M sub-contexts, respectively. With this formulation, we transform the policy learning problem to a binary classification problem of Mn training instances. Similar to the situation of the RIIPS estimator, solving (38) by conventional optimization methods is not easy. For example, the stochastic gradient (SG) method may face difficulties in the process of sampling Mn samples [23].

To overcoming this difficulty, our idea is to connect (38) with the extreme similarity learning problem, which learns the relation between a huge number of (context, item) pairs. Efficient training has been well studied for such problems, like [23, 24, 26]. This connection relies on the following facts.

- In our experiments, we consider a two-tower structure for the model $\hat{y}(\cdot)$.[3] This setting is very prevalent in modern recommender systems.
- By applying the second-order Taylor expansion, the second term in (38) can be approximated with a weighted squared loss as $\ell_{\text{sq}}(\tilde{Y}_{ij}, \hat{Y}_{ij}) = \omega_{ij}(\tilde{Y}_{ij} - \hat{Y}_{ij})^2$, where ω_{ij} is a cost weight associated with the loss and $\tilde{Y}_{ij}$ is an imputed value.[4]

As these two facts satisfy the requirements in works of extreme similarity learning, we can apply efficient optimization methods for this problem. In our implementation, we consider the Gauss-Newton method proposed in [26], which can factorize all operations involving the $\mathbb{O}(Mn)$ cost into a series of operations with a much smaller $\mathbb{O}(M) + \mathbb{O}(n)$ cost.

4 RELATED WORKS

In previous sections, we discuss the counterfactual learning in a storyline of recommender systems, where actions decided by a policy are specified as recommended items. For general scenarios, counterfactual learning has been immensely discussed in recent studies. The mainline of these studies [9, 19, 20] focus on stabilizing the learning process and consider learning a policy that decides only one single action. To the best of our knowledge, [2] is the first work extending policy learning to top-K recommender systems. Similar to our discussion in Section 3.1, they meet the difficulty caused by the vast number of item permutations. To make the problem tractable, they impose the following two constraints.

- For each context, at most one item in $\mathbf{A}$ has a non-zero reward, which is denoted by A^+.
- For each $\mathbf{A}$, the K recommended items are sampled with replacement during training but without replacement during deployment.

Under the above constraints, they propose the following top-K REINFORCE estimator,

$$\hat{V}^{\pi}_{\text{top-K}}(\boldsymbol{\theta}) = \mathbb{E}_{P_\beta}\Big[\frac{1-(1-\pi(\mathrm{A}^+ \mid \mathbf{u};\boldsymbol{V};\boldsymbol{\theta}))^K}{z(\mathrm{A}^+ \mid \mathbf{u};\boldsymbol{V})}\mathrm{c}_{\mathrm{A}^+}\Big], \tag{39}$$

where $1-(1-\pi(\mathrm{A}^+ \mid \mathbf{u};\boldsymbol{V};\boldsymbol{\theta}))^K$ is the probability of A^+ being included in K items sampled by π with replacement, and $z(\mathrm{A}^+ \mid \mathbf{u};\boldsymbol{V})$ is the propensity score the same as that in (6). For the gradient of (39), there will be an additional multiplier derived from $\frac{1-(1-\pi(\mathrm{A}^+|\mathbf{u};\boldsymbol{V};\boldsymbol{\theta}))^K}{z(\mathrm{A}^+|\mathbf{u};\boldsymbol{V})}$, which allocates some probability mass of the learned policy to other items of interest and thus helps to alleviate the winner-takes-all effect.

Our idea of pruning importance weights in the RIIPS estimator is inspired by reinforcement learning [16], where the pruning method is used for decreasing weights from long horizons. A similar idea appeared in the policy evaluation [12, 21] for top-K recommendations, where both β and π are pre-known and fixed. For policy learning, we are not the only work to apply this idea. A parallel work [13] also considers pruning the weight around the current action. However, different from their empirical study, we offer a theoretical analysis on the loss of the pruning in Section 3.1. This analysis leads to the addition of a regularization term to alleviate the loss, whose effectiveness is confirmed in our empirical experiments.

[3] See the details in Appendix A.1.
[4] See the detailed derivations in our supplementary materials.

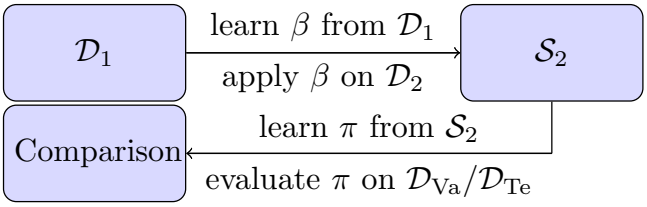

Figure 2: An illustration of the supervised-to-bandit conversion, where β is the behavior policy, and π is the policy to be learnt and evaluated.

On the other hand, while most works of policy learning only consider a relatively small action space (e.g., less than hundreds), [2, 13] are the pioneers to extend policy learning for tremendous actions. To tackle the training efficiency issue described in Section 3.3, which is resulted from going over all the possible actions, they pre-select actions from $\mathbb{A}$ to reduce the cost. Specifically, in [2, 3], they perform sampled softmax [1] for training π. In [13], they consider a heuristic $\rho(\cdot)$ to pre-select actions, which results in their proposed POXM estimator as follows.

$$\hat{V}^{\pi}_{\text{POXM}}(\boldsymbol{\theta}) = \mathbb{E}_{P_\beta}\Big[\sum_{k=1}^{K}\frac{\pi(\mathrm{A}_k \mid \hat{\boldsymbol{u}}_k, \rho(\mathbf{u});\boldsymbol{V},\boldsymbol{\theta})}{\beta(\mathrm{A}_k \mid \hat{\boldsymbol{u}}_k;\boldsymbol{V})}(\mathrm{c}_{\mathrm{A}_k} - \alpha)\Big], \tag{40}$$

where they only consider actions in the pre-selected set $\rho(\mathbf{u})$ of each context such that

$$\pi(\mathrm{A}_k \mid \hat{\boldsymbol{u}}_k, \rho(\mathbf{u});\boldsymbol{V},\boldsymbol{\theta}) = \begin{cases} \frac{e^{\hat{y}(\boldsymbol{\theta};\mathbf{u},\boldsymbol{v}_{\mathrm{A}_k})}}{\sum_{j\in\rho(\mathbf{u})\setminus\mathrm{A}_{1:k-1}} e^{\hat{y}(\boldsymbol{\theta};\mathbf{u},\boldsymbol{v}_j)}} & \mathrm{A}_k \in \rho(\mathbf{u}), \\ 0 & \mathrm{A}_k \notin \rho(\mathbf{u}). \end{cases}$$

Besides, they follow [9] to adjust α in (40). In [13], they empirically construct $\rho(\mathbf{u})$ by referring to the behavior policy β such that

$$\begin{aligned}\rho(\mathbf{u}) =& \{\mathrm{A}_k \mid \mathrm{A}_k \text{ has the largest } \beta(A_k \mid \hat{\boldsymbol{u}}_k;\boldsymbol{V}), \\ & \text{and } 1 \le k \le p\},\end{aligned} \tag{41}$$

where p is a specified number of pre-selected actions. The performance of this kind of methods highly relies on the quality of the pre-selection procedure. For example, in [3], they find the number of sampled actions significantly impacts the performance of the learned policy. For POXM, if ρ can successfully cover actions with the highest rewards with only a small action space, POXM should perform better than approaches learned from the whole action space. However, the task to construct such an effective ρ is just as challenging as learning π. That is to say, if for all contexts, we are already able to obtain the wanted ρ, which includes the actions with the highest rewards, then there may be no need for us to learn a new π. In contrast, our proposed training method in Section 3.3 does not require any pre-selection on items.

5 EXPERIMENTS

In this section, after presenting our experimental setup for online simulation, we conduct a series of experiments to compare our proposed framework in Section 3 with other existing state-of-the-art approaches.

Table 2: Data statistics, where n is the number of items. $|\cdot|$ indicates the number of elements in a set.

Data Set	n	$\lvert\mathcal{D}_1\rvert$	$\lvert\mathcal{D}_2\rvert$	$\lvert\mathcal{D}_{\text{Va}}\rvert$	$\lvert\mathcal{D}_{\text{Te}}\rvert$
ml1m	3,513	12,066	27,153	9,051	12,077
ml10m	10,210	139,022	312,836	104,279	139,081

5.1 Experimental Setup for Online Simulation

To simulate the problem setting described in Section 2.1, we following the supervised-to-bandit conversion in [14, 19] to generate our data from MovieLens 1M (ml1m) and 10M (ml10m) data sets.[5] We first randomly split a data set into four independent subsets: $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_{\text{va}}$ and $\mathcal{D}_{\text{te}}$. Then, as shown in Figure 2, we learn a behavior policy β with $\mathcal{D}_1$ and deploy it on $\mathcal{D}_2$ by logging a $(\mathbf{u}, \mathbf{A}, \mathbf{C})$ event for each context to form a set $\mathcal{S}_2$. Next we conduct different approaches to learn a new policy π, where $\mathcal{D}_{\text{Va}}$ is used as the validation set for tuning hyper-parameters. Finally, we evaluate π on the test set $\mathcal{D}_{\text{Te}}$ and compare the resulting performance. The whole procedure is illustrated in Figure 2, while the statistics of the generated data sets are in Table 2. More details of data preprocessing and constructing β are in Appendices A.2 and A.3.

For learning π, we compare seven approaches, which can be grouped into four categories:

- Value-IPS, Value-DR: They are two value learning approaches solving (9) respectively with (6) and (8).
- BanditNet: This approach considers the estimate in (20) without pruning $\text{w}_{\mathbf{A}}$. To alleviate the possible issue caused by any extremely large $\text{w}_{\mathbf{A}}$, it additionally applies the technique proposed in [9].
- Top-K REINFORCE [2] and POXM [13]: They are two state-of-the-art competitors in the policy learning for top-K decision making; see the discussion in Section 4. For POXM, we follow [13] to construct $\rho(\cdot)$ by (41) and conduct grid searches on p.
- RIIPS, Log-RIIPS and Adaptive-RIIPS: Three approaches derived from our proposed RIIPS estimator, which respectively solve problems in (28), (32) and the approximation of (38). For Adaptive-RIIPS, we empirically set all $\tilde{Y}_{ij} = -1$ and consider $\omega_{ij} = \omega$ as a hyper-parameter to tune.

More details of the hyper-parameter setting and selection are in Appendix A.4.

To evaluate π, we simulate the online environment. Specifically, we compute the following average cumulated reward (ACR) on a data set with fully observed rewards for all items.

$$\text{ACR} = \frac{1}{\hat{m}} \sum_{i=1}^{\hat{m}} r(\boldsymbol{c}_i, \mathcal{A}_i), \tag{42}$$

where $\hat{m}$ is the number of contexts included in this set, $\boldsymbol{c}_i$ is the fully-observed reward vector of context i, and $\mathcal{A}_i$ is the set of K items recommended according to $\pi(\mathcal{A}_i \mid \boldsymbol{u}_i; \boldsymbol{V})$. Here we focus on the exploitation ability of each learned policy π, so the decision of $\mathcal{A}_i$ is made by selecting top K items with the highest scores predicted by $\hat{y}(\cdot)$. For consistency, we use the same K for constructing $\mathcal{S}_2$ and evaluating models on $\mathcal{D}_{\text{Va}}$ and $\mathcal{D}_{\text{Te}}$.

[5] https://grouplens.org/datasets/movielens/

Table 3: For two data sets, we report the ACR of all approaches on $\mathcal{D}_{\text{Te}}$. All scores below are scaled up by 100.

Approach	ml1m		ml10m	
	$K = 1$	$K = 10$	$K = 1$	$K = 10$
Value-IPS	24.04	25.91	25.23	17.07
Value-DR	38.80	28.75	27.46	26.25
POXM	28.66	16.48	22.74	14.54
top-K REINFORCE	34.47	29.13	30.41	25.39
BanditNet	34.51	18.21	30.38	19.14
RIIPS	35.99	28.44	30.41	24.19
Log-RIIPS	35.11	26.40	28.73	20.57
Adaptive-RIIPS	**39.02**	**32.27**	**34.66**	**28.27**
RIIPS ($\alpha = 0$)	34.47	23.01	30.41	23.01

5.2 Comparison on Various Approaches

By comparing results in Table 3, we have the following observations.

- Firstly, Value-DR is much better than Value-IPS. This result confirms what we described in Section 3.2. That is, the IPS approach may not work properly when the amount of observed events is deficient. Compared to Value-IPS, Value-DR imputes rewards for unobserved items and reweighs the loss term for these items, which can be seen as a way to tune the reward distribution adaptively. This leads to a great improvement on the performance.
- RIIPS makes a significant improvement over BanditNet when $K = 10$. The reason is that RIIPS prunes $\text{w}_{\mathbf{A}}$. As reported in [11], the value of $\text{w}_{\mathbf{A}}$ gets exploded very easily when $K > 1$, which leads to a huge divergence between $\hat{V}^{\pi}_{\text{IPS}}$ and V^{π}. BanditNet's performance suffers without pruning $\text{w}_{\mathbf{A}}$.
- In Table 3, RIIPS is better than RIIPS ($\alpha = 0$) for $K = 10$, which confirms the rationality of adding the regularization term $D_{\text{KL}}(\beta \,||\, \pi)$ with pruning $\prod_{j=1}^{k-1} \text{w}_j$.
- Compared to other approaches, top-K REINFORCE is almost the second optimal approach in policy learning when $K = 10$. This verifies the applicability of (39) proposed in [2] and confirms the necessity for alleviating the winner-takes-all effect.
- Adaptive-RIIPS proposed in this work performs the best. As stated in Section 3.2, by tuning ω, we can adapt the probability mass of the learned policy. This effectively addresses the winner-takes-all effect and overcomes the drawback of Log-RAIPS.
- In our data sets, POXM performs the worst because as described in Section 4, it is tough to get an effective $\rho(\mathbf{u})$. Even though our behavior policy β is trained on $\mathcal{D}_1$ with fully-revealed $\boldsymbol{c}$, the pre-selected ρ still can not cover enough items with the highest rewards. Increasing the size of ρ may help, but it also lowers the training efficiency. This issue might limit the usability of POXM.

To further confirm the efficiency and effectiveness of the framework proposed in this work, we compare the performance and training time of Adaptive-RIIPS trained by two different optimizers: Adagrad and Gauss-Newton. In Table 4, we can observe that Adaptive-RIIPS with Gauss-Newton consistently converges faster and better than the one with Adagrad. Take ml1m with $K = 1$ for example. Gauss-Newton not only speeds up more than 10 times for the model training but also improves the model performance by about 6% respectively when compared to Adagrad.

Table 4: Comparison on Adaptive-RIIPS with two different optimizers: Adagrad and Gauss-Newton. Training time and performance under $K = 1$ and 10 for two datasets are shown.

Dataset	Optimizers	$K = 1$		$K = 10$	
		Time(s)	ACR	Time(s)	ACR
ml1m	Adagrad	739.04	36.74	6028.79	30.70
	Gauss-Newton	58.12	39.02	1422.50	32.70
ml10m	Adagrad	13826.98	34.24	60850.22	27.07
	Gauss-Newton	673.52	34.66	14500.11	28.27

6 CONCLUSION

In this work, we categorize the conventional approaches of counterfactual learning for recommender systems into two classes: value learning and policy learning. To do a comparison between these two kinds of approaches, we point out that some existing difficulties in the policy learning approaches for large top-K recommender systems must be addressed first.

- It is likely that the extremely small propensity of a top-K recommendation would lead to a poor estimation of $\hat{V}^{\pi}_{\text{IPS}}$ and thus a sub-optimal policy.
- The robustness and generalization of policy learning approaches suffer from the limited observations of each item for each context.
- The issue of training efficiency in recommender systems with a large item space limits the usability of policy learning approaches.

To address the above difficulties, we derive a novel policy learning framework. We introduce a regularized per-item approach to balance between the variance and the bias of policy learning. Then, through decoupling the objective function and introducing an extra hyper-parameter for tuning the smoothness, we manage to improve the robustness of policy learning. For efficient training, we integrate our proposed approach with the two-tower structure and the algorithm developed in [26] to be a framework able to train large top-K recommender systems. With experiments conducted on real-world data sets, we confirm the effectiveness and efficiency of our proposed framework.

ACKNOWLEDGMENTS

This work was supported in part by MOST of Taiwan grant 110-2221-E-002-115-MY3.

REFERENCES

[1] Yoshua Bengio and Jean-Sébastien Senécal. 2003. Quick Training of Probabilistic Neural Nets by Importance Sampling. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*. 17–24.

[2] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.

[3] Minmin Chen, Bo Chang, Can Xu, and Ed H. Chi. 2021. User Response Models to Improve a REINFORCE Recommender System. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM)*. 121–129.

[4] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly Robust Policy Evaluation and Learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML)*. 1097–1104.

[5] Daniel G. Horvitz and Donovan J. Thompson. 1952. A Generalization of Sampling Without Replacement From a Finite Universe. *J. Amer. Statist. Assoc.* 47 (1952), 663–685.

[6] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information and Knowledge Management*. 2333–2338.

[7] Tzu-Kuo Huang, Ruby C. Weng, and Chih-Jen Lin. 2006. Generalized Bradley-Terry Models and Multi-class Probability Estimates. *Journal of Machine Learning Research* 7 (2006), 85–115. http://www.csie.ntu.edu.tw/~cjlin/papers/generalBT.pdf

[8] Olivier Jeunen, David Rohde, Flavian Vasile, and Martin Bompaire. 2020. Joint Policy-Value Learning for Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1223–1233.

[9] Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. 2018. Deep learning with logged bandit feedback. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[10] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42 (2009), 30–37.

[11] Damien Lefortier, Adith Swaminathan, Xiaotao Gu, Thorsten Joachims, and Maarten de Rijke. 2016. Large-scale validation of counterfactual learning methods: A test-bed. In *NIPS Workshop on Inference and Learning of Hypothetical and Counterfactual Interventions in Complex Systems*.

[12] Shuai Li, Yasin Abbasi-Yadkori, Branislav Kveton, Shan Muthukrishnan, Vishwa Vinay, and Zheng Wen. 2018. Offline evaluation of ranking policies with click models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1685–1694.

[13] Romain Lopez, Inderjit Dhillon, and Michael I. Jordan. 2021. Learning from eXtreme Bandit Feedback. (2021).

[14] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Ji Yang, Minmin Chen, Jiaxi Tang, Lichan Hong, and Ed H. Chi. 2020. Off-policy Learning in Two-stage Recommender Systems. In *Proceedings of The Web Conference*. 463–473.

[15] John I. Marden. 1995. *Analyzing and Modeling Rank Data*. Chapman & Hall, London.

[16] Doina Precup, Richard S. Sutton, and Satinder P. Singh. 2000. Eligibility Traces for Off-Policy Policy Evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*. 759–766.

[17] Steffen Rendle. 2010. Factorization machines. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*. 995–1000.

[18] Noveen Sachdeva, Yi Su, and Thorsten Joachims. 2020. Off-policy bandits with deficient support. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 965–975.

[19] Adith Swaminathan and Thorsten Joachims. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research* 16, 1 (2015), 1731–1755.

[20] Adith Swaminathan and Thorsten Joachims. 2015. The Self-normalized Estimator for Counterfactual Learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*. 3231–3239.

[21] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miroslav Dudík, John Langford, Damien Jose, and Imed Zitouni. 2017. Off-Policy Evaluation for Slate Recommendation. (2017), 3635–3645.

[22] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 269–277.

[23] Hsiang-Fu Yu, Mikhail Bilenko, and Chih-Jen Lin. 2017. Selection of Negative Samples for One-class Matrix Factorization. In *Proceedings of SIAM International Conference on Data Mining (SDM)*. http://www.csie.ntu.edu.tw/~cjlin/papers/one-class-mf/biased-mf-sdm-with-supp.pdf

[24] Hsiang-Fu Yu, Hsin-Yuan Huang, Inderjit S. Dihillon, and Chih-Jen Lin. 2017. A Unified Algorithm for One-class Structured Matrix Factorization with Side Information. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*. http://www.csie.ntu.edu.tw/~cjlin/papers/ocmf-side/biased-leml-aaai-with-supp.pdf

[25] Bowen Yuan, Jui-Yang Hsia, Meng-Yuan Yang, Hong Zhu, Chihyao Chang, Zhenhua Dong, and Chih-Jen Lin. 2019. Improving Ad Click Prediction by Considering Non-displayed Events. In *Proceedings of the 28th ACM International Conference on Conference on Information and Knowledge Management (CIKM)*. http://www.csie.ntu.edu.tw/~cjlin/papers/occtr/ctr_oc.pdf

[26] Bowen Yuan, Yu-Sheng Li, Pengrui Quan, and Chih-Jen Lin. 2021. Efficient optimization methods for extreme similarity learning with nonlinear embeddings. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. http://www.csie.ntu.edu.tw/~cjlin/papers/similarity_learning/pq.pdf

[27] Bowen Yuan, Yaxu Liu, Jui-Yang Hsia, Zhenhua Dong, and Chih-Jen Lin. 2020. Unbiased Ad click prediction for position-aware advertising systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*. http://www.csie.ntu.edu.tw/~cjlin/papers/debiases/debiases.pdf

[28] Bowen Yuan, Meng-Yuan Yang, Jui-Yang Hsia, Hong Zhu, Zhirong Liu, Zhenhua Dong, and Chih-Jen Lin. 2019. *One-class Field-aware Factorization Machines for Recommender Systems with Implicit Feedbacks*. Technical Report. National Taiwan University. http://www.csie.ntu.edu.tw/~cjlin/papers/ocffm/imp_ffm.pdf

A APPENDIX

A.1 Details of Model Setting and its Connection with Extreme Similarity Learning Problems

The connection between our proposed Adaptive-RIIPS and extreme similarity learning problems relies on the fact that the two-tower structure models are prevalent in modern recommender systems. Let $\boldsymbol{\theta} = [\begin{smallmatrix}\boldsymbol{\theta}_u \\ \boldsymbol{\theta}_v\end{smallmatrix}]$. The two-tower structure transforms $\hat{\boldsymbol{u}}$ and $\boldsymbol{v}$ into two d-dimensional embeddings respectively by learning two embedding functions $f(\boldsymbol{\theta}_u; \hat{\boldsymbol{u}})$ and $g(\boldsymbol{\theta}_v; \boldsymbol{v})$, and then $\hat{y}(\boldsymbol{\theta}; \hat{\boldsymbol{u}}, \boldsymbol{v}) = f(\boldsymbol{\theta}_u; \hat{\boldsymbol{u}})^\top g(\boldsymbol{\theta}_v; \boldsymbol{v})$ is the score function of this structure. Many commonly-used recommendation models fall into the two-tower structure. For example, when $f(\cdot)$ and $g(\cdot)$ are linear embedding functions, it has been pointed out [28] that $\hat{y}(\cdot)$ is a variant of the factorization machine [17]. Besides, non-linear embedding functions (e.g., neural networks) are also widely applied in two-tower recommendation models [6, 22].

Through applying the two-tower structure models for $\hat{y}(\boldsymbol{\theta}; \hat{\boldsymbol{u}}, \boldsymbol{v})$, the problem (38) can be considered as an extreme similarity learning problem. That is, we learn the similarity between a sub-context and an item from extremely large Mn (sub-context, item) pairs. Here the similarity presents the tendency of an item being recommended by π under a given sub-context. To avoid any $\mathbb{O}(Mn)$ cost in solving an extreme similarity learning problem, existing works in this field [23, 24, 26] impose the following squared loss on a certain part of the loss function. In our case, the second term in (38), which deals with the similarity between sub-contexts and non-recommended items, is replaced with $\ell_{\text{sq}}(\tilde{Y}_{ij}, \hat{Y}_{ij}) = \omega_{ij}(\tilde{Y}_{ij} - \hat{Y}_{ij})^2$, where ω_{ij} is a cost associated with the loss and $\tilde{Y}_{ij}$ is an imputed value. For each ω_{ij} and $\tilde{Y}_{ij}$, it is required that they can be decomposed to multiple parts solely related to i or j, respectively.

Then, the remaining issue is how to choose ω_{ij} and impute $\tilde{Y}_{ij}$. We apply the second-order Taylor expansion of $\ell^-_{\log}(\hat{Y}_{ij})$ at our chosen point $\tilde{y}_j$ for each item to get the following alternative loss function for non-recommended items.[6]

$$\frac{1}{2}\nabla^2\ell^-_{\log}(\tilde{y}_j)\left(\hat{Y}_{ij} + \frac{\nabla\ell^-_{\log}(\tilde{y}_j) - \nabla^2\ell^-_{\log}(\tilde{y}_j)\tilde{y}_j}{\nabla^2\ell^-_{\log}(\tilde{y}_j)}\right)^2. \tag{43}$$

By applying (43) to (38), we attain the following problem to solve.

$$\min_{\boldsymbol{\theta}} \quad \frac{1}{m}\sum_{i=1}^{M} z_i\left[\ell^+_{\log}(\hat{Y}_{ia_i}) + \omega_{ij}\sum_{j=1, j\neq a_i}^{n}\frac{1}{2}(\tilde{Y}_{ij} - \hat{Y}_{ij})^2\right] + \lambda\Phi(\boldsymbol{\theta}), \tag{44}$$

where for all i, $\omega_{ij} = \mu\nabla^2\ell^-_{\log}(\tilde{y}_j)$, and $\tilde{Y}_{ij} = \frac{\nabla^2\ell^-_{\log}(\tilde{y}_j)\tilde{y}_j - \nabla\ell^-_{\log}(\tilde{y}_j)}{\nabla^2\ell^-_{\log}(\tilde{y}_j)}$ are pre-defined constants solely dependent to j, and $\hat{Y}_{ij}$ comes from the two-tower structure. As deciding $\tilde{y}_j$ is like to select a hyper-parameter, we instead treat ω_{ij} and $\tilde{Y}_{ij}$ as two hyper-parameters and tune them by grid search. In our experiments, this simple transformation is found to be effective enough empirically.

A.2 Details of Data Preprocessing

For MovieLens 1M (ml1m) and 10M (ml10m) data sets, we follow [23] to binarize ratings included in original sets. We consider pairs with rating ≥ 4 as positive while the rest including unrated items as negative. As original sets do not include contextual information, we need to generate the contextual features $\boldsymbol{u}$ and context-aware rewards $\boldsymbol{c}$ respectively being realizations of $\mathbf{u}$ and $\mathbf{c}$. Specifically, for each user, we group his/her positive items into a set and then randomly divide it into two subsets equally. Positive items in the first subset, like movies the user has watched, are used as the contextual information to construct $\boldsymbol{u} \in \{0, 1\}^n$ where

$$u_j = \begin{cases} 1 & \text{item } j \text{ is included in the first subset,} \\ 0 & \text{otherwise.} \end{cases}$$

Besides, positive items in the second subset, like movies the user is going to watch, are used for constructing $\boldsymbol{c}$ where

$$c_j = \begin{cases} 1 & \text{item } j \text{ is included in the second subset,} \\ 10^{-3} & \text{otherwise.} \end{cases}$$

We repeat the above procedure to generate ten $(\boldsymbol{u}, \boldsymbol{c})$ pairs for each user. For $\boldsymbol{V}$, we generate a feature vector of each item by one-hot encoding with its identity number.

A.3 Details of Behavior Policy β

For learning β, similar to past works [13, 19], we learn $\hat{s}(\cdot)$ by a value learning approach with the fully-revealed $\boldsymbol{c}$ included in $\mathcal{D}_1$. We apply the approach in [24] for this task. With the learned $\hat{s}(\cdot)$, we transform it into a policy β defined in (19) by the PL ranking model defined in (30) such that for each sub-context, we have

$$\beta(\mathrm{A}_k \mid \hat{\mathbf{u}}_k; \boldsymbol{V}) = \frac{e^{\hat{s}(\boldsymbol{\theta}; \hat{\mathbf{u}}_k, \boldsymbol{v}_{\mathrm{A}_k})}}{\sum_{j=1, j\notin\mathbf{A}_{1:k-1}}^{n} e^{\hat{s}(\boldsymbol{\theta}; \hat{\mathbf{u}}_k, \boldsymbol{v}_j)}}. \tag{45}$$

A.4 Details of Hyper-Parameter Selection

For the two-tower structure we use in all approaches, the output size of the embeddings is set to be 128. Adaptive-RIIPS uses the Gauss-Newton method proposed in [26], while all the other approaches are trained by Adagrad. For Adagrad, the learning rate is initialized to 0.05 and the batch size is set to 10% of the training data. As an exception, for ml10m with $K = 10$, the batch size of Adagrad is set to 1% of the training data due to the size of our RAM. For Value-DR with (8), we set $\hat{\mathrm{c}}_j = \log(\frac{\overline{\mathrm{CTR}}}{1-\overline{\mathrm{CTR}}}), \forall j$, where $\overline{\mathrm{CTR}}$ is the average click-through-rate from an unbiased set collected by a uniform behavior policy. For convenience, we directly compute $\overline{\mathrm{CTR}}$ from $\mathcal{D}_2$, which is unbiased due to fully-observed reward vectors.

We train each approach for T epochs, retain the model at the epoch with the best validation performance, and report the test performance by applying the model to predict the test set. For Adagrad, T = 500 and for Gauss-Newton, T = 30.

The following hyper-parameters are selected by a grid search. Candidates of each hyper-parameter are also listed below.

- $\lambda \in \{4^{-1}, 4^0, ..., 4^4\}$: the $l2$ regularization coefficient.
- $\alpha \in \{0, 0.01, 0.1, 1\}$: the coefficient of $D_{\mathrm{KL}}(\beta \,||\, \pi)$.
- $\gamma, \omega \in \{16^{-5}, 16^{-4}, 16^{-3}, 16^{-2}\}$: the coefficients of the square loss term in (8) and (44).
- $\zeta \in \{0.6, 0.8, 1.0, 1.2\}$: the Lagrange multiplier in [9].
- $\xi \in \{0.1, 0.3, 0.5\}$: the tolerance in the relative stopping condition for Gauss-Newton.
- $p \in \{10, 20, 50\}$: the size of ρ for POXM.

[6] See the detailed derivations in our supplementary materials.