Efficient Offline Learning of Ranking Policies via Top-k Policy Decomposition

Ren Kishimoto[*]
Tokyo Institute of Technology
kishimoto.r.ab@m.titech.ac.jp

Koichi Tanaka[*]
Keio University
kouichi_1207@keio.jp

Haruka Kiyohara
Cornell University
hk844@cornell.edu

Yusuke Narita
Yale University
yusuke.narita@yale.edu

Nobuyuki Shimizu
LY Corporation
nobushim@lycorp.co.jp

Yasuo Yamamoto
LY Corporation
yasyamam@lycorp.co.jp

Yuta Saito
Cornell University
ys552@cornell.edu

Abstract

We study *Off-Policy Learning* (OPL) of ranking policies, which enables us to learn new ranking policies using only historical logged data. Ranking settings make OPL remarkably challenging because their action spaces consist of permutations of unique items, being extremely large. Existing methods, which primarily use either policy- or regression-based approaches, suffer from high variance and bias respectively. To circumvent these issues of existing methods, we propose a new OPL method for ranking, named ***Ranking Policy Optimization via Top-k Policy Decomposition (R-POD)***, which combines the policy- and regression-based approaches in an effective fashion. Specifically, R-POD decomposes a ranking policy into a first-stage policy for selecting *top-k* actions and a second-stage policy for choosing the bottom actions given the top-k actions. It then learns the first-stage policy via the policy-based approach and the second-stage policy via the regression-based approach. In particular, we propose a new policy gradient estimator to learn the first-stage policy via the policy-based approach. This method can substantially reduce variance, since it applies importance weighting only to the top-k actions. We also demonstrate that our policy-gradient estimator for the first-stage policy is unbiased under a *conditional pairwise correctness* condition, which only requires that the expected reward differences of pairs of rankings sharing the same top-k actions can be estimated correctly. Comprehensive experiments illustrate that R-POD provides substantial improvements in OPL for ranking where existing methods fail due to large action spaces.

1 Introduction

Intelligent systems in the real-world, such as recommender systems, search engines, and news applications, often present items (e.g., products, hotels, news) in the form of rankings. In these systems, our goal is to learn new ranking policies to improve outcomes, using only historical logged data. This learning task is known as *Off-Policy Learning* (OPL). OPL is highly relevant in many practical applications involving automated decision-making regarding ranking interface [25, 14].

[*]Equal contribution

38th Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2024).

The main approaches to OPL include policy-based and regression-based methods [25]. The policy-based approach learns new policies by estimating the policy gradient, often through importance-weighting [21, 4]. Although this approach can be based on unbiased policy gradients and learns effective policies with sufficient logged data, it can be sample-inefficient particularly under large action spaces [19, 24]. In ranking settings, in particular, where the action space corresponds to all possible permutations of items, most policy gradient estimators collapse due to extremely high variance [12, 27, 26, 19]. To mitigate the variance issue caused by importance weighting in ranking setups, several methods introduce assumptions about user behavior such as independence [14] and cascade [18]. While these methods succeed in reducing variance, restrictive assumptions can lead to large bias in policy gradient estimations [13, 12, 18]. On the other hand, the regression-based approach, which learns the reward function and selects a ranking with the highest predicted reward, can avoid variance issues but is known to suffer from high bias due to the difficulty of accurately modeling the rewards of every unique ranking in the action space.

To address the bias and variance issues caused by ranking action spaces, we develop a novel OPL algorithm for ranking called ***Ranking Policy Optimization via Top-k Policy Decomposition (R-POD)***. The crux of R-POD is to decompose a ranking policy into a first-stage policy that chooses the best top-k actions for each context and a second-stage policy that selects the best bottom actions given the top-k actions. Leveraging this decomposition of a ranking policy, we learn the first-stage policy through the policy-based approach with a novel policy gradient estimator, called the R-POD gradient estimator. The R-POD gradient estimator includes importance weighting in the top-k action space to account for the effect of top-k actions and a reward regression model to consider the effect of the bottom actions. We demonstrate that the R-POD gradient estimator is unbiased under a conditional pairwise correctness (CPC) condition, which only requires that the regression model accurately preserves the relative expected reward differences of rankings sharing the same top-k actions. The second-stage policy of R-POD is then learned by the regression-based approach. We show that the second-stage policy can be based on a reward model that is used as a part of the R-POD gradient estimator for the first stage-policy, since the CPC condition ensures that the second-stage policy performs optimally in terms of choosing the bottom actions.

Compared to existing policy-based methods for ranking, our R-POD gradient estimator applies importance weighting only to the top-k actions, substantially reducing variance, as we theoretically demonstrate. In addition, R-POD avoids introducing large bias to achieve substantial variance reduction, as it does not introduce any assumptions about user behavior. Moreover, compared to existing regression-based approaches, R-POD relaxes the modeling requirement regarding the reward function. Specifically, it only needs to accurately learn the relative value differences between pairs of unique rankings that have the same top-k actions, a condition that is generally milder than learning the global expected rewards for every unique ranking. Comprehensive experiments on both synthetic and real-world ranking data illustrate that our R-POD algorithm performs more effectively than existing policy- and regression-based methods for a variety of experiment settings.

2 Preliminaries

In our formulation of OPL for ranking, $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ denotes a d_x-dimensional context vector (e.g., user demographics) drawn i.i.d. from an unknown distribution $p(\boldsymbol{x})$. The finite set of discrete (unique) actions is denoted as $\mathcal{A}$, with $a \in \mathcal{A}$ corresponding to an action like a single movie, song, news article, or product. Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_l, \ldots, a_L)$ be a ranking action vector, where L denotes the length of the ranking. The function $\pi : \mathcal{X} \to \Delta(\prod_L(\mathcal{A}))$ is referred to as a *ranking policy*, with $\prod_L(\mathcal{A})$ indicating the set of L-permutations of $\mathcal{A}$, i.e., the ranking action space. Furthermore, $\boldsymbol{r} = (r_1, r_2, \ldots, r_l, \ldots, r_L)$ represents a reward vector, sampled from an unknown conditional distribution $p(\boldsymbol{r}|\boldsymbol{x}, \boldsymbol{a})$, where r_l is the reward observed at the l-th position. The effectiveness of a policy π is measured through its *value*, which is defined as follows.

$$V(\pi) \coloneqq \mathbb{E}_{p(\boldsymbol{x})\pi(\boldsymbol{a}|\boldsymbol{x})}\left[\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a})\right], \tag{1}$$

where $q_l(\boldsymbol{x}, \boldsymbol{a}) \coloneqq \mathbb{E}\left[r_l|\boldsymbol{x}, \boldsymbol{a}\right]$ represents the *position-wise* expected reward function. Here, α_l is a non-negative weight given to each position. This definition of policy value in Eq. (1) can represent various types of ranking metrics. For instance, when $\alpha_l \coloneqq 1/\log_2(l+1)$, it represents the discounted cumulative gain (DCG).

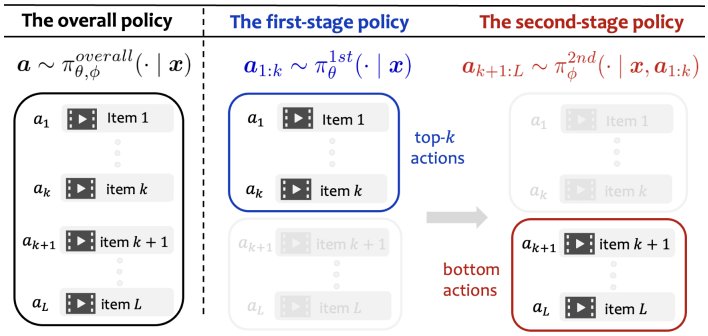


Figure 1: The R-POD algorithm leverages the concept of top-k policy decomposition in Eq. (6).

Let $\mathcal{D} \coloneqq \{(\boldsymbol{x}^{(i)}, \boldsymbol{a}^{(i)}, \boldsymbol{r}^{(i)})\}_{i=1}^{n}$ be the logged data we can use for performing OPL, which contains n independent observations drawn from the logging policy π_0 as $(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{r}) \sim p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})p(\boldsymbol{r}|\boldsymbol{x}, \boldsymbol{a})$. In OPL of ranking policies, using only $\mathcal{D}$, we aim to optimize a ranking policy π_θ, which is parameterized by θ, to maximize the policy value as

$$\theta^* = \arg\max_{\theta \in \Theta} V(\pi_\theta). \tag{2}$$

3 Limitations of Existing Methods

This section reviews existing approaches and their limitations in the ranking setup.

Firstly, **the policy-based approach** aims to learn the policy parameter θ via iterative gradient ascent, $\theta_{t+1} \leftarrow \theta_t + \eta \nabla_\theta V(\pi_\theta)$, where

$$\nabla_\theta V(\pi_\theta) \coloneqq \mathbb{E}_{p(\boldsymbol{x})\pi_\theta(\boldsymbol{a}|\boldsymbol{x})}\left[\left(\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a})\right) \nabla_\theta \log \pi_\theta(\boldsymbol{a}|\boldsymbol{x})\right] \tag{3}$$

is called the policy gradient (we can derive it via the log-derivative trick, i.e., $\nabla_\theta \pi_\theta = \pi_\theta \nabla_\theta \log \pi_\theta$). The problem here is that we do not know the true policy gradient $\nabla_\theta V(\pi_\theta)$ since we do not know the true reward functions $\{q_l(\boldsymbol{x}, \boldsymbol{a})\}_{l=1}^{L}$. Therefore, it needs to estimate the policy gradient with only available logged data $\mathcal{D}$. A standard approach to do it is to apply *inverse propensity scoring (IPS)* as

$$\nabla_\theta \widehat{V}_{\mathrm{IPS}}(\pi_\theta; \mathcal{D}) \coloneqq \frac{1}{n} \sum_{i=1}^{n} w(\boldsymbol{x}^{(i)}, \boldsymbol{a}^{(i)}) \left(\sum_{l=1}^{L} \alpha_l r_l^{(i)}\right) s_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{a}^{(i)}), \tag{4}$$

where $w(\boldsymbol{x}, \boldsymbol{a}) \coloneqq \pi_\theta(\boldsymbol{a}|\boldsymbol{x})/\pi_0(\boldsymbol{a}|\boldsymbol{x})$ is called the ranking-level importance weight, which is defined as the ratio of probabilities that a unique ranking $\boldsymbol{a}$ is chosen under two different policies. We also use $s_\theta(\boldsymbol{x}, \boldsymbol{a}) \coloneqq \nabla_\theta \log \pi_\theta(\boldsymbol{a}|\boldsymbol{x})$ to denote the policy score function in Eq. (4). It is widely known that the IPS gradient estimator given above is unbiased under the full support condition.

Condition 3.1. *(Full Support) The logging policy π_0 is said to have full support if $\pi_0(\boldsymbol{a}|\boldsymbol{x}) > 0$ for all $\boldsymbol{a} \in \Delta(\prod_L(\mathcal{A}))$ and $\boldsymbol{x} \in \mathcal{X}$.*

Unfortunately, in ranking situations, the full support condition is often hard to guarantee due to a large number of unique rankings [27], potentially resulting in substantial bias for the IPS gradient estimator [23, 7, 26].

To deal with the variance issue of IPS, we can possibly apply the *Doubly Robust (DR)* estimator [5] which uses a reward estimator $\hat{f}(\boldsymbol{x}, \boldsymbol{a}) \approx \sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a})$ as a control variate. However, DR still suffers from extremely high variance due to the use of ranking-level importance weighting.

To address the high variance caused by ranking-level importance weighting, some methods introduce assumptions about user behavior [14, 18, 12]. For instance, under a cascade assumption, which posits that users interact with items one by one from the top position, a policy gradient estimator can be defined as follows [18].

$$\nabla_\theta \hat{V}_{\mathrm{RIPS}}(\pi_\theta; \mathcal{D}) \coloneqq \frac{1}{n} \sum_{i=1}^{n} \sum_{l=1}^{L} \frac{\pi_\theta(\boldsymbol{a}_{1:l}^{(i)}|\boldsymbol{x}^{(i)})}{\pi_0(\boldsymbol{a}_{1:l}^{(i)}|\boldsymbol{x}^{(i)})} \alpha_l r_l^{(i)} \nabla_\theta \log \pi_\theta(\boldsymbol{a}_{1:l}^{(i)}|\boldsymbol{x}^{(i)}), \tag{5}$$

where $\pi(\boldsymbol{a}_{1:l}|\boldsymbol{x}) = \sum_{\boldsymbol{a}' \in \prod_L(\mathcal{A})} \pi(\boldsymbol{a}'|x)\mathbb{I}\{\boldsymbol{a}'_{1:l} = \boldsymbol{a}_{1:l}\}$. This RIPS gradient estimator is unbiased under the cascade assumption and significantly reduces variance compared to IPS and DR. However, the RIPS gradient estimator can exhibit high bias due to mismatches between actual user behavior and its underlying assumption [13].

Secondly, **the regression-based approach** estimates the reward function as $\hat{q}_l(\boldsymbol{x}, \boldsymbol{a}) \approx q_l(\boldsymbol{x}, \boldsymbol{a})$, $\forall l$ using conventional supervised machine learning methods. It then converts the estimated reward functions $\{\hat{q}_l(\boldsymbol{x}, \boldsymbol{a})\}_{l=1}^L$ into a ranking policy, for example, by applying the argmax operator as below.

$$\pi(\boldsymbol{a} \,|\, \boldsymbol{x}) := \begin{cases} 1 & (\boldsymbol{a} = \arg\max_{\boldsymbol{a}' \in \Pi_L(\mathcal{A})} \sum_{l=1}^L \alpha_l \hat{q}_l(\boldsymbol{x}, \boldsymbol{a}')) \\ 0 & (\text{otherwise}) \end{cases}$$

Regarding the variance, this approach is superior to the policy-based approach as it does not involve importance weighting. However, it can suffer from high bias due to the difficulty of accurately regressing the expected rewards for every unique ranking, i.e., $\forall \boldsymbol{a} \in \Pi_L(\mathcal{A})$, based only on partial feedback in the logged data $\mathcal{D}$.

As discussed above, classic approaches to OPL are considered ineffective in the ranking setup, which is prevalent in the real-world. To achieve more efficient OPL of ranking policies, the following develops a novel algorithm for ranking that circumvents the variance issue of the policy-based approach and the bias issue of the regression-based approach simultaneously.

4 The "R-POD" Algorithm

This section introduces a novel OPL method called **R-POD** that addresses the issues of bias and variance in ranking situations. The core concept of our proposed method involves decomposing a ranking policy into two components: a first-stage policy and a second-stage policy, as follows.

The Top-k Policy Decomposition:

$$\pi_{\theta,\phi}^{overall}(\boldsymbol{a} \,|\, \boldsymbol{x}) = \pi_\theta^{1st}(\boldsymbol{a}_{1:k} \,|\, \boldsymbol{x}) \cdot \pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L} \,|\, \boldsymbol{x}, \boldsymbol{a}_{1:k}), \tag{6}$$

where $\boldsymbol{a}_{k_1:k_2} \coloneqq (a_{k_1}, a_{k_1+1}, \cdots, a_{k_2-1}, a_{k_2})$. As depicted in Figure 1, the first-stage policy is a segment of the overall policy responsible for selecting the top-k actions ($\boldsymbol{a}_{1:k}$). Conversely, the second-stage policy is a segment to choose the bottom actions ($\boldsymbol{a}_{k+1:L}$), conditional on the top-k actions already sampled by the first-stage policy.

Intuitively, higher-ranked items are considered more crucial for producing effective rankings. Hence, we consider optimizing the first-stage policy using a method with low bias, and then the second-stage policy using a method with low variance to control the overall variance of the algorithm. With this idea in mind, our R-POD algorithm for ranking learns an overall policy in two separate stages. In the first stage, it optimizes a first-stage policy through the policy-based approach within the top-k action space. By applying importance weighting exclusively to the top-k action space, which is considerably smaller than the entire ranking space, we can learn the first-stage policy with significantly reduced variance, despite using the policy-based approach. In the second stage, we optimize the second-stage policy through the regression-based approach. By applying the regression model only to the remaining action space, conditional on a set of top-k actions, we can learn the second-stage policy with smaller bias than the conventional regression-based approach while taking advantage of its low variance. In the following, we describe how to learn first- and second-stage policies in order to improve the value of the overall policy.

4.1 Optimizing the First-stage Policy π_θ^{1st}

We first consider optimizing the first-stage policy π_θ^{1st}, parameterized by θ, via the policy-based approach given a second-stage policy π_ϕ^{2nd}. We consider learning the first-stage policy under a pre-trained second-stage policy because the overall policy is dependent both on the first- and second-stage policies, and the optimal first-stage policy becomes different given a different second-stage policy, as we will describe below.

Given that our ultimate goal is training a better overall policy, we should train the first-stage policy so that the overall policy is improved. Hence, given a second-stage policy π_ϕ^{2nd}, we aim to update the first-stage policy parameter θ as

$$\theta_{t+1} \leftarrow \theta_t + \eta \nabla_\theta V(\pi_{\theta,\phi}^{overall}) \tag{7}$$

The following derives the policy gradient for the overall policy regarding the first-stage policy parameter, i.e., $\nabla_\theta V(\pi_{\theta,\phi}^{overall})$.

Proposition 4.1. *(The overall policy gradient) The policy gradient for the overall policy regarding the first-stage policy parameter, i.e., $\nabla_\theta V(\pi_{\theta,\phi}^{overall})$, is given as follows.*

$$\nabla_\theta V(\pi_{\theta,\phi}^{overall}) = \mathbb{E}_{p(\boldsymbol{x})\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k})\right], \tag{8}$$

where $q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \coloneqq \mathbb{E}_{\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x}, \boldsymbol{a}_{1:k})}\left[\sum_{l=1}^L \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a})\right]$ denotes the value of top-k actions under π_ϕ^{2nd} and $s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \coloneqq \nabla_\theta \log \pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})$ is the policy score function of the first-stage policy. See Appendix C.1 for the proof.

Proposition 4.1 suggests that if the first-stage policy can choose the top-k actions that are evaluated highly according to the function $q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k})$, we can improve the effectiveness of the overall policy. An interesting observation here is that the top-k actions that the first-stage policy should choose are different given different second-stage policies as implied by the fact that the function $q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k})$ is dependent on π_ϕ^{2nd}. This is indeed the reason why we are considering training the first-stage policy given a second-stage policy.

As discussed, the true $\nabla_\theta V(\pi_{\theta,\phi}^{overall})$ would lead to an improved overall policy, however we do not know the ground-truth policy gradient due to the inability to know $q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k})$, so we have to estimate it using logged data to train the first-stage policy. To achieve this, we propose a new policy gradient estimator, called the **R-POD gradient estimator**, defined as follows.

$$\nabla_\theta \widehat{V}_{\mathrm{RPOD}}(\pi_{\theta,\phi}^{overall}; \mathcal{D}) \coloneqq \frac{1}{n}\sum_{i=1}^n \Bigg\{ w(\boldsymbol{x}^{(i)}, \boldsymbol{a}_{1:k}^{(i)}) \left(\sum_{l=1}^L \alpha_l r_l^{(i)} - \hat{f}(\boldsymbol{x}^{(i)}, \boldsymbol{a}^{(i)})\right) s_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{a}_{1:k}^{(i)}) + \mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x}^{(i)})}\left[\hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x}^{(i)}, \boldsymbol{a}_{1:k}) s_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{a}_{1:k})\right] \Bigg\}, \tag{9}$$

where $w(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \coloneqq \frac{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}{\pi_0^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})} = \frac{\sum_{\boldsymbol{a}'} \mathbb{I}\{\boldsymbol{a}'_{1:k}=\boldsymbol{a}_{1:k}\}\pi_\theta^{1st}(\boldsymbol{a}'|\boldsymbol{x})}{\sum_{\boldsymbol{a}'} \mathbb{I}\{\boldsymbol{a}'_{1:k}=\boldsymbol{a}_{1:k}\}\pi_0^{1st}(\boldsymbol{a}'|\boldsymbol{x})}$ is the *top-k importance weight*. Specifically, the first term of Eq. (9) estimates the value of top-k actions $\boldsymbol{a}_{1:k}$ via importance weighting and the second term deals with the value of bottom actions ($\boldsymbol{a}_{k+1:L}$) via the regression model $\hat{f}(\boldsymbol{x}, \boldsymbol{a})$. Since the top-$k$ importance weight considers only the difference in probabilities of choosing the top-k actions between policies, it is expected to have much lower variance than existing gradient estimators such as IPS, DR, and RIPS. Additionally, it does not introduce any assumption on user behavior such as independence [14] or cascade [18], so it does not produce large bias regarding the violation of such assumptions when reducing variance. Note that we will discuss how we should optimize the regression model $\hat{f}(\boldsymbol{x}, \boldsymbol{a})$ based on the analysis of the R-POD estimator provided below.

As a theoretical analysis, we first characterize the bias of the R-POD gradient estimator under the full "top-k" support condition, which is less restrictive than the full support condition needed for the unbiasedness of IPS.

Condition 4.1. *(Full top-k support) The logging policy π_0 is said to have full top-k support if $\pi_0(\boldsymbol{a}_{1:k}|\boldsymbol{x}) > 0$ for all $\boldsymbol{a} \in \prod_L(\mathcal{A})$ and $\boldsymbol{x} \in \mathcal{X}$.*

Theorem 4.1. *(Bias of the R-POD gradient estimator) If Condition 4.1 is true, the R-POD gradient estimator has the following bias for some given regression model $\hat{f}(\boldsymbol{x}, \boldsymbol{a})$.*

$$\begin{aligned}
&Bias(\nabla_\theta \widehat{V}_{\mathrm{RPOD}}(\pi_{\theta,\phi}^{overall}; \mathcal{D})) \\
&= \mathbb{E}_{\pi_0^{1st}(\boldsymbol{x}, \boldsymbol{c}_{1:k})}\Bigg[\sum_{\boldsymbol{a}<\boldsymbol{b}:\boldsymbol{a}_{1:k}=\boldsymbol{b}_{1:k}=\boldsymbol{c}_{1:k}} \pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x}, \boldsymbol{c}_{1:k})\pi_0^{2nd}(\boldsymbol{b}_{k+1:L}|\boldsymbol{x}, \boldsymbol{c}_{1:k}) \\
&\times (\Delta_q(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) - \Delta_{\hat{f}}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}))\left(\frac{\pi_\theta(\boldsymbol{b}|\boldsymbol{x}, \boldsymbol{c}_{1:k})}{\pi_0(\boldsymbol{b}|\boldsymbol{x}, \boldsymbol{c}_{1:k})} - \frac{\pi_\theta(\boldsymbol{a}|\boldsymbol{x}, \boldsymbol{c}_{1:k})}{\pi_0(\boldsymbol{a}|\boldsymbol{x}, \boldsymbol{c}_{1:k})}\right) s_\theta(\boldsymbol{x}, \boldsymbol{c}_{1:k})\Bigg],
\end{aligned} \tag{10}$$

where $\boldsymbol{a}, \boldsymbol{b} \in \prod_L(\mathcal{A})$. $\Delta_q(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) \coloneqq q(\boldsymbol{x}, \boldsymbol{a}) - q(\boldsymbol{x}, \boldsymbol{b})$ *represents the difference of the expected rewards between a pair of rankings* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *given* $\boldsymbol{x}$*, which we call* ***the relative value difference of rankings****.* $\Delta_{\hat{f}}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) \coloneqq \hat{f}(\boldsymbol{x}, \boldsymbol{a}) - \hat{f}(\boldsymbol{x}, \boldsymbol{b})$ *is a relative value of rankings between* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *given* $\boldsymbol{x}$ *estimated by the regression model* $\hat{f}(\boldsymbol{x}, \boldsymbol{a})$.

The most important factor in Eq. (10) is $\Delta_q(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) - \Delta_{\hat{f}}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b})$, which implies that when a regression model $\hat{f}(\boldsymbol{x}, \boldsymbol{a})$ accurately preserves the relative value differences of rankings containing the same top-k actions, the bias of the R-POD gradient estimator becomes small. Intuitively, the R-POD gradient estimator already unbiasedly estimates the value of top-k actions via its top-k importance weighting, and thus it is sufficient for the regression model to identify only relative value differences of rankings given the same top-k actions to make the gradient estimator unbiased. Moreover, Theorem 4.1 implies that the R-POD gradient estimator becomes unbiased under the following condition.

Condition 4.2. *(Conditional Pairwise Correctness; CPC) A regression model* $\hat{f}(\boldsymbol{x}, \boldsymbol{a})$ *satisfies conditional pairwise correctness if* $\Delta_q(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) = \Delta_{\hat{f}}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b})$ *for all* $x \in \mathcal{X}$ *and* $\boldsymbol{a}, \boldsymbol{b} \in \prod_L(\mathcal{A})$ *s.t.* $\boldsymbol{a}_{1:k} = \boldsymbol{b}_{1:k}$.

Corollary 4.1. *Under Conditions 4.1 and 4.2, the R-POD gradient estimator is unbiased, i.e.,* $\mathbb{E}_{\mathcal{D}}[\nabla_\theta \widehat{V}_{\mathrm{RPOD}}(\pi_{\theta,\phi}^{overall}; \mathcal{D})] = \nabla_\theta V(\pi_{\theta,\phi}^{overall})$.

Thus, the R-POD gradient estimator can be unbiased under a condition where the regression model satisfies conditional pairwise correctness, which is less restrictive than aiming for correctly estimating the global reward functions $\{q_l(\boldsymbol{x}, \boldsymbol{a})\}_{l=1}^L$ like implicitly assumed for the regression-based approach. The above bias analysis also implies that we should ideally optimize the regression model $\hat{f}(\boldsymbol{x}, \boldsymbol{a})$ so that it preserves the relative value differences to minimize the bias of the resulting gradient estimator.

Next, the following calculates the variance of R-POD to show its relation with the accuracy of the regression model $\hat{f}(\boldsymbol{x}, \boldsymbol{a})$.

Proposition 4.2. *(Variance of the R-POD gradient estimator) Under Conditions 4.1 and 4.2, the variance of the R-POD gradient estimator is given by*

$$
\begin{aligned}
n\mathbb{V}_{\mathcal{D}}(\nabla_\theta \widehat{V}_{\mathrm{RPOD}}^{(j)}(\pi_{\theta,\phi}^{overall}; \mathcal{D})) = \; & \mathbb{E}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\left(w(\boldsymbol{x}, \boldsymbol{a}_{1:k}) s_\theta^{(j)}(\boldsymbol{x}, \boldsymbol{a}_{1:k})\right)^2 \sigma^2(\boldsymbol{x}, \boldsymbol{a})\right] \\
& + \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{V}_{\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \Delta_{q,\hat{f}}(\boldsymbol{x}, \boldsymbol{a}) s_\theta^{(j)}(\boldsymbol{x}, \boldsymbol{a}_{1:k})\right]\right] \\
& + \mathbb{V}_{p(\boldsymbol{x})}\left[\mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) s_\theta^{(j)}(\boldsymbol{x}, \boldsymbol{a}_{1:k})\right]\right], \qquad (11)
\end{aligned}
$$

where $\Delta_{q,\hat{f}}(\boldsymbol{x}, \boldsymbol{a}) \coloneqq q(\boldsymbol{x}, \boldsymbol{a}) - \hat{f}(\boldsymbol{x}, \boldsymbol{a})$ *is the estimation error of* $\hat{f}(\boldsymbol{x}, \boldsymbol{a})$ *against the global expected reward function* $q(\boldsymbol{x}, \boldsymbol{a})$*,* $\sigma^2(\boldsymbol{x}, \boldsymbol{a}) \coloneqq \mathbb{V}[(\sum_{l=1}^L \alpha_l r_l) | \boldsymbol{x}, \boldsymbol{a}]$ *is the conditional variance of the rewards, and* $s_\theta^{(j)}(\boldsymbol{x}, \boldsymbol{a}_{1:k})$ *is the j-th dimension of the score function.*

Proposition 4.2 suggests that, in terms of variance minimization, we should optimize the regression model in a way that minimizes $|\Delta_{q,\hat{f}}(\boldsymbol{x}, \boldsymbol{a})|$ compared to minimizing $|\Delta_q(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) - \Delta_{\hat{f}}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b})|$ for the bias. Based on the theoretical observations, an ideal strategy to optimize the regression model $\hat{f}(\boldsymbol{x}, \boldsymbol{a})$ would be a two-step procedure to directly optimize the bias and variance of the R-POD gradient estimator in each step. Specifically, the first step focuses on minimizing the bias by optimizing a pairwise regression function $\hat{h}_\phi(\boldsymbol{x}, \boldsymbol{a})$ towards accurately estimating the relative reward differences $|\Delta_q(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) - \Delta_{\hat{f}}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b})|$, which needs pairwise logged data. The second step then aims for variance minimization via minimizing $|\Delta_{q,\hat{f}}(\boldsymbol{x}, \boldsymbol{a})|$. More details of the ideal procedure of two-step regression can be found in Appendix B.

However, we do not expect this ideal two-step procedure to be always feasible in practice due to its need of pairwise logged data and implementation costs. Even if it is impractical, we can still employ a conventional regression to estimate the expected absolute reward to construct the regression model. This can be done by optimizing a parameterized function $\hat{f}_\phi : \mathcal{X} \times \Pi_L(\mathcal{A}) \to \mathbb{R}$ via:

$$
\min_\phi \sum_{(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{r}) \in \mathcal{D}} \ell_f\big(\boldsymbol{r}, \hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a})\big), \qquad (12)
$$

and $\hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a})$ is used in Eq. (9). ℓ_f is a loss function to measure the accuracy of $\hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a})$, which can be defined, for example, as $\ell_f\big(\boldsymbol{r}, \hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a})\big) = \big(\sum_{l=1}^{L} \alpha_l r_l - \hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a})\big)^2$. Even with this practical and simple procedure, the R-POD gradient estimator retains advantages over existing policy gradient estimators by its significant variance reduction. The following section empirically demonstrates that R-POD performs more effectively than existing approaches with this practical regression procedure.

4.2 Optimizing the Second-stage Policy π_ϕ^{2nd}

The objective of the second-stage policy π_ϕ^{2nd} is to identify the bottom actions to optimize the value given the top-k actions chosen by the first-stage policy π_θ^{1st}. Specifically, the second-stage policy should rank the bottom actions that yield the highest value among rankings containing the same top-k actions. From the regression procedure mentioned earlier, we have already obtained $\hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a})$ to estimate the reward function via the regression-based approach. This allows us to define the second-stage policy based on $\hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a})$, for example, as follows.

$$\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L} | \boldsymbol{x}, \boldsymbol{a}_{1:k}) := \begin{cases} 1 & (\boldsymbol{a} = \arg\max_{\boldsymbol{a}' : \boldsymbol{a}'_{1:k} = \boldsymbol{a}_{1:k}} \hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a}')) \\ 0 & (otherwise) \end{cases} \tag{13}$$

When the regression model satisfies the CPC condition, the above second-stage policy is optimal because CPC ensures a regression model to accurately estimate the relative value of rankings that share the same top-k actions. This is a more relaxed modeling requirement compared to the existing regression-based approach.

4.3 The Overall R-POD Algorithm

The overall process of the R-POD algorithm is conducted as follows. First, we construct the regression model $\hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a})$, for example, via performing Eq. (12). We then formulate the second-stage policy π_ϕ^{2nd} based on the regressor $\hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a})$ as in Eq. (13). We are also based on $\hat{f}_\phi(\boldsymbol{x}, \boldsymbol{a})$ and optimize the first-stage policy π_θ^{1st} via iterative gradient ascent using the R-POD gradient estimator in Eq. (9).

Once we obtain first- and second-stage policies via the R-POD algorithm, for an incoming context $\boldsymbol{x}$ in the inference phase, we first sample top-k actions from the 1st-stage policy as $\boldsymbol{a}_{1:k} \sim \pi_\theta^{1st}(\boldsymbol{a}_{1:k} \,|\, \boldsymbol{x})$. We then apply the 2nd-stage policy to rank the bottom actions given the top-k actions as $\boldsymbol{a}_{k+1:K} \sim \pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L} \,|\, \boldsymbol{x}, \boldsymbol{a}_{1:k})$. This procedure is equivalent to sampling a ranking from the joint distribution induced by π_θ^{1st} and π_ϕ^{2nd}, i.e., $\boldsymbol{a} \sim \pi_{\theta,\phi}^{overall}(\boldsymbol{a} \,|\, \boldsymbol{x})$.

5 Empirical Evaluation

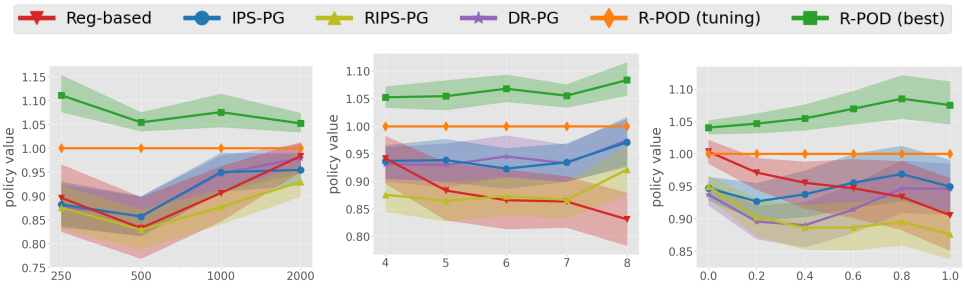
We first evaluate R-POD on synthetic data to identify the situations where R-POD enables more effective OPL than the baseline methods. We then assess the applicability of R-POD in real-world situations using public datasets. Note that our experiment code will be available upon publication.

5.1 Synthetic Data

To generate synthetic datasets, we randomly sample 5-dimensional contexts from the standard normal distribution. Then, for each context-ranking pair, we first synthesize the *position-wise* expected reward function for l-th position in a ranking as

$$q_l(\boldsymbol{x}, \boldsymbol{a}) := \tilde{q}_l(\boldsymbol{x}, a_l) + F(\boldsymbol{x}, \boldsymbol{a}), \tag{14}$$

where $\tilde{q}_l(\boldsymbol{x}, a_l)$ depends only on the corresponding *position-wise* action a_l, while $F(\boldsymbol{x}, \boldsymbol{a})$ depends on the whole ranking action $\boldsymbol{a}$. Specifically, the former term is defined as $\tilde{q}_l(\boldsymbol{x}, a_l) := \theta_{a_l}^\top \boldsymbol{x} + b_{a_l}$, where θ_{a_l} is a parameter vector sampled from the standard normal distribution and b_{a_l} is a bias term defined for action a_l. $F(\boldsymbol{x}, \boldsymbol{a}) = \sum_{m \neq l} \mathbb{W}(a_m, a_l) \mathbb{I}\{X_l \leq \lambda\}$ is called the interaction term, where $\mathbb{W}(a_m, a_l)$ indicates the effect of action a_m on the reward of action a_l. X_l is a random variable sampled from the standard uniform distribution, and $\lambda \in [0, 1]$ is a parameter that controls the probability of a *position-wise* reward affected from other actions within the same ranking. We then sample the reward r_l from a normal distribution with mean $q_l(\boldsymbol{x}, \boldsymbol{a})$ and standard deviation $\sigma = 0.5$.



(a) Training data size (n)

(b) Number of unique actions ($|\mathcal{A}|$)

(c) Probability affected from different positions (λ)

Figure 2: Comparing the test policy values (normalized by R-POD (tuning)) of the OPL methods, with varying (a) training data sizes, (b) numbers of actions, and (c) probabilities affected from each action.

Table 1: Comparison of the OPL methods on MSLR-WEB30K regarding DCG@5. $f_0(x, a)$ in the logging policy uses either constant value (uniform), random forest regression, or ridge regression. Values outside and inside the parenthesis are the mean and the standard deviation based on 5 random seeds, respectively.

MSLR-WEB30K	Logging Policy		
OPL methods	uniform	random forest	ridge
Reg-based	0.506 (0.083)	0.516 (0.064)	0.501 (0.064)
IPS-PG	0.456 (0.015)	0.500 (0.011)	0.506 (0.011)
RIPS-PG	0.495 (0.014)	0.514 (0.014)	0.517 (0.015)
DR-PG	0.490 (0.015)	0.514 (0.011)	0.517 (0.033)
R-POD (tuning, 10% estimation error)	0.522 (0.033)	0.528 (0.021)	0.527 (0.014)
R-POD (tuning, 5% estimation error)	**0.528 (0.039)**	**0.535 (0.039)**	**0.530 (0.019)**
R-POD (best)	**0.531 (0.042)**	**0.539 (0.040)**	**0.532 (0.020)**

We define the logging policy that produces the logged data based on the Plackett-Luce model [20] as follows.

$$\pi_0(\boldsymbol{a}|\boldsymbol{x}) = \prod_{l=1}^{L} \pi_0(a_l|\boldsymbol{x}) = \prod_{l=1}^{L} \frac{\exp(f_0(\boldsymbol{x}, a_l))\mathbb{I}[a_l \notin \boldsymbol{a}_{1:l-1}]}{\sum_{a' \in \mathcal{A} \setminus \boldsymbol{a}_{1:l-1}} \exp(f_0(\boldsymbol{x}, a'))} \tag{15}$$

where $f_0(\boldsymbol{x}, a_l) = \tilde{\theta}_{a_l}^\top \boldsymbol{x} + \tilde{b}_{a_l}$. We sample both $\tilde{\theta}_{a_l}$ and $\tilde{b}_{a_l}$ from the standard uniform distribution.

Compared Methods. We compare R-POD with IPS-PG, DR-PG, RIPS-PG, and Regression-based approach (Reg-based). To determine the hyperparameter k for R-POD, we perform grid-search in range $k \in [0, L]$ based on the policy value estimated by IPS. R-POD with data-driven tuning of k is denoted as "R-POD (tuning)" in our experiment results. We also report the results of "R-POD (best)", which uses the hyperparameter k with the best ground-truth policy value among the possible values of k and provides the best achievable value as a reference.

Results Figure 2 compares the value of policies learned by each OPL method over 100 simulations with different random seeds. Each figure in Figure 2 compares the policy learning effectiveness with varying data sizes, numbers of unique actions, and the probabilities of the *position-wise* reward function ($q_l(\boldsymbol{x}, a_l)$) affected by the whole ranking ($F(\boldsymbol{x}, \boldsymbol{a})$), where the default parameters are n = 1000, $|\mathcal{A}| = 5$, $L = 3$, and $\lambda = 1.0$, respectively.

First, Figure 2a, which varies the training data size n from 250 to 2000, shows that R-POD (tuning) performs consistently better than the baseline methods across various data sizes. The advantage of R-POD over the baselines becomes particularly large when the data size is small, suggesting

that R-POD effectively achieves a substantial reduction in variance regarding the policy gradient estimation to enable a more data-efficient OPL for ranking policies. It would also be interesting to see that R-POD (tuning) performs competitively compared to R-POD (best), particularly when the data size is large, even though R-POD (best) always performs even better than R-POD (tuning) leveraging its unfair access to the ground-truth policy value to identify the optimal value of k.

Next, when varying the numbers of unique actions $|\mathcal{A}|$ from 4 to 8 (this varies the number of unique rankings from 24 to 336) in Figure 2b, we observe that R-POD outperforms the baseline methods in all situations. This suggests that R-POD can perform satisfactory even when the number of candidate rankings grows.

Finally, Figure 2c demonstrates that RIPS-PG degrades in performance with larger violations of the cascade assumption (larger λ), because RIPS-PG completely ignores the interactions from actions placed in lower positions in the ranking by assuming the cascade user behavior. In contrast, it is much more robust to the violations of the cascade model than RIPS-PG because R-POD considers the effect of lower positions using its regression model. R-POD also performs consistently better than IPS-PG and DR-PG, which are unbiased irrespective of λ, but have extremely high variance.

5.2 Real-World Data

We then conduct real-world experiments on the learning-to-rank datasets, namely the Microsoft Learning to Rank Challenge dataset (MSLR-WEB30K) [22]. MSLR-WEB30K has 124 documents per query on average, and we randomly sample 100 documents per query.

This dataset contains 5-level relevance scores $\mathit{rel}(\boldsymbol{x}, a) \in \{0, ..., 4\}$ for all of their query($\boldsymbol{x}$)-document(a) pairs and we define the position-wise expected reward function of the form: $q_l(\boldsymbol{x}, \boldsymbol{a}) = \mathit{rel}(\boldsymbol{x}, a_l)/4 + \eta_{a_l}$ where η_{a_l} is a noise parameter sampled separately for each a_l from a normal distribution whose mean is 0 and standard deviation is 0.05. Then, we sample the *position-wise* reward from a normal distribution with mean $q_l(\boldsymbol{x}, \boldsymbol{a})$ and standard deviation $\sigma = 0.05$. We use the Plackett-Luce logging policy defined in Eq. (15), the same logging policy as used in the synthetic experiment. However, in the real-world experiment, $f_0(\boldsymbol{x}, a)$ in the logging policy definition is either a constant value (uniform) or a regression model (either random forest regression or ridge regression), which is trained with 10 % of the training data. Note that we set $L = 5$. R-POD (tuning) tunes its hyperparameter k by a noise-added ground-truth policy value where the noise (or estimation error) is sampled from a uniform distribution of range $(-\Delta_{\text{max}}, \Delta_{\text{max}})$, and $|\Delta_{\text{max}}|$ is either 5% or 10% of the ground-truth policy value $V(\pi_\theta)$, varying the accuracy of the tuning of its hyperparameter k.

Results Table 1 reports the results with varying logging policies, including uniform, random forest, svr, and ridge on MSLR-WEB30K. The results demonstrate that R-POD works better than the baseline methods across varying logging policies. Specifically, we observe that R-POD outperforms the policy-based methods (RIPS-PG, IPS-PG, and DR-PG) by a large margin, even when tuning the hyperparameter k with 10% estimation error, resulting from effectively reducing variance without inducing a significant bias. Moreover, comparing R-POD with Reg-based, we observe that Reg-based underperforms R-POD by a large margin when the regression is inaccurate. This result suggests that R-POD is more robust to the regression error than Reg-based, which is consistent with our theoretical analysis. We can also see that R-POD (best) is always the best, which suggests the even higher potential of R-POD (best) on the real-world dataset with a better procedure to tune its hyperparameter.

6 Conclusion

This work studies off-policy learning (OPL) for ranking policies. Existing methods often fail due to substantial bias and variance. To facilitate more effective OPL for ranking, we develop a novel algorithm called R-POD. R-POD optimizes the first-stage policy, responsible for selecting the top-k actions in a ranking, through a new policy gradient estimator. This estimator is unbiased under conditional pairwise correctness and exhibits lower variance compared to existing gradient estimators. The second-stage policy, responsible for selecting the bottom actions, is learned via a reward regression. This component of our algorithm is considered more robust to reward modeling errors than traditional regression-based methods. Empirical evaluations demonstrate the effectiveness of R-POD in optimizing ranking policies, particularly in challenging situations such as with small sample sizes and large action spaces.

References

[1] Alina Beygelzimer and John Langford. The offset tree for learning with partial labels. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 129–138, 2009.

[2] Alexey Borisov, Ilya Markov, Maarten De Rijke, and Pavel Serdyukov. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*, pages 531–541, 2016.

[3] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. Click models for web search. *Synthesis lectures on information concepts, retrieval, and services*, 7(3):1–115, 2015.

[4] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1097–1104, 2011.

[5] Miroslav Dudík, Dumitru Erhan, John Langford, and Lihong Li. Doubly robust policy evaluation and optimization. *Statistical Science*, 29(4), November 2014.

[6] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. More robust doubly robust off-policy evaluation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1447–1456. PMLR, 2018.

[7] Nicolo Felicioni, Maurizio Ferrari Dacrema, Marcello Restelli, and Paolo Cremonesi. Off-policy evaluation with deficient support using side information. *Advances in Neural Information Processing Systems*, 35, 2022.

[8] Fan Guo, Chao Liu, and Yi Min Wang. Efficient multiple-click models in web search. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining*, pages 124–131, 2009.

[9] Olivier Jeunen and Bart Goethals. Pessimistic reward models for off-policy learning in recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pages 63–74, 2021.

[10] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pages 652–661. PMLR, 2016.

[11] Nathan Kallus, Yuta Saito, and Masatoshi Uehara. Optimal off-policy evaluation from multiple logging policies. In *International Conference on Machine Learning*, pages 5247–5256. PMLR, 2021.

[12] Haruka Kiyohara, Yuta Saito, Tatsuya Matsuhiro, Yusuke Narita, Nobuyuki Shimizu, and Yasuo Yamamoto. Doubly robust off-policy evaluation for ranking policies under the cascade behavior model. In *Proceedings of the 15th International Conference on Web Search and Data Mining*, 2022.

[13] Haruka Kiyohara, Masatoshi Uehara, Yusuke Narita, Nobuyuki Shimizu, Yasuo Yamamoto, and Yuta Saito. Off-policy evaluation of ranking policies under diverse user behavior. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1154–1163, 2023.

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

[15] Dawen Liang and Nikos Vlassis. Local policy improvement for recommender systems, 2023.

[16] Yao Liu, Pierre-Luc Bacon, and Emma Brunskill. Understanding the curse of horizon in off-policy evaluation via conditional importance sampling. In *International Conference on Machine Learning*, pages 6184–6193. PMLR, 2020.

[17] Yifei Ma, Yu-Xiang Wang, and Balakrishnan Narayanaswamy. Imitation-regularized offline learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2956–2965. PMLR, 2019.

[18] James McInerney, Brian Brost, Praveen Chandar, Rishabh Mehrotra, and Benjamin Carterette. Counterfactual evaluation of slate recommendations with sequential reward interactions. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1779–1788, 2020.

[19] Jie Peng, Hao Zou, Jiashuo Liu, Shaoming Li, Yibao Jiang, Jian Pei, and Peng Cui. Offline policy evaluation in large action spaces via outcome-oriented action grouping. In *Proceedings of the ACM Web Conference 2023*, pages 1220–1230, 2023.

[20] Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

[21] Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning*, page 759–766, 2000.

[22] Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013.

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

[24] Noveen Sachdeva, Lequn Wang, Dawen Liang, Nathan Kallus, and Julian McAuley. Off-policy evaluation for large action spaces via policy convolution. *arXiv preprint arXiv:2310.15433*, 2023.

[25] Yuta Saito and Thorsten Joachims. Counterfactual learning and evaluation for recommender systems: Foundations, implementations, and recent advances. In *Proceedings of the 15th ACM Conference on Recommender Systems*, page 828–830, 2021.

[26] Yuta Saito and Thorsten Joachims. Off-policy evaluation for large action spaces via embeddings. In *International Conference on Machine Learning*, pages 19089–19122. PMLR, 2022.

[27] Yuta Saito, Qingyang Ren, and Thorsten Joachims. Off-policy evaluation for large action spaces via conjunct effect modeling. *arXiv preprint arXiv:2305.08062*, 2023.

[28] Yuta Saito, Jihan Yao, and Thorsten Joachims. Potec: Off-policy learning for large action spaces via two-stage policy decomposition. *arXiv preprint arXiv:2402.06151*, 2024.

[29] Alex Strehl, John Langford, Lihong Li, and Sham M Kakade. Learning from logged implicit exploration data. In *Advances in Neural Information Processing Systems*, volume 23, pages 2217–2225, 2010.

[30] Yi Su, Maria Dimakopoulou, Akshay Krishnamurthy, and Miroslav Dudík. Doubly robust off-policy evaluation with shrinkage. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 9167–9176. PMLR, 2020.

[31] Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *International Conference on Machine Learning*, pages 814–823. PMLR, 2015.

[32] Adith Swaminathan and Thorsten Joachims. The self-normalized estimator for counterfactual learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[33] Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pages 2139–2148. PMLR, 2016.

[34] Cameron Voloshin, Hoang M Le, Nan Jiang, and Yisong Yue. Empirical study of off-policy policy evaluation for reinforcement learning. *arXiv preprint arXiv:1911.06854*, 2019.

[35] Yu-Xiang Wang, Alekh Agarwal, and Miroslav Dudık. Optimal and adaptive off-policy evaluation in contextual bandits. In *International Conference on Machine Learning*, pages 3589–3597. PMLR, 2017.

[36] Danqing Xu, Yiqun Liu, Min Zhang, Shaoping Ma, and Liyun Ru. Incorporating revisiting behaviors into click models. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 303–312, 2012.

[37] Zeyu Zhang, Yi Su, Hui Yuan, Yiran Wu, Rishab Balasubramanian, Qingyun Wu, Huazheng Wang, and Mengdi Wang. Unified off-policy learning to rank: a reinforcement learning perspective, 2023.

A Related Work

This section summarizes notable related works.

A.1 General Off-Policy Evaluation

Off-policy evaluation (OPE), which aims to estimate the performance of a counterfactual policy using historical data, has widely been acknowledged as a safe, ethical, and cost-effective alternative for online A/B tests [25], and has attracted a lot of interest in both contextual bandits [5, 6, 11, 35] and reinforcement learning [10, 16, 33]. There are three standard OPE methods to estimate the policy value with context-action-reward tuples and two advanced estimators that leverage the action embeddings for efficient OPE. First, *Direct Method* (DM) [1] learns the expected reward function via conventional machine learning methods and then estimates the value of new policies with the estimated rewards. DM has low variance, and enables accurate estimation when the regression model learns the expected reward function $q(x, a)$ accurately across context-action pairs. However, DM suffers from high bias due to model misspecification and covariate shift [6, 34]. Second, *Inverse Propensity Scoring* (IPS) [29] applies importance weighting to the observed rewards to estimate the value of new policies. IPS is unbiased under the common support condition ($\pi(a|x) > 0 \rightarrow \pi_0(a|x) > 0 \; \forall (x, a) \in \mathcal{X} \times \mathcal{A}$). However, the common support condition is frequently violated in large action spaces, and IPS incurs bias under deficient support [23**?**]. In contrast, satisfying the common support condition causes severe variance because the logging policy $\pi_0(a|x)$ should allocate a small probability for every context-action pair, leading to large importance weights. There are several techniques to reduce the scale of the importance weight, such as clipping [31, 30] and self-normalization [32]. However, these methods can still suffer from high variance and introduce high bias in the estimation. Third, *Doubly Robust* (DR) [5] combines DM and IPS in a way to reduce variance while being unbiased. DR uses the regression-based estimations as its baseline and applies importance weighting only to the difference between the reward r and the estimated reward $\hat{q}(x, a)$. By doing so, DR can reduce the variance of IPS under a reasonable assumption about the regression ($|q(x, a) - \hat{q}(x, a)| \leq |q(x, a)|$), while being unbiased under the same common support condition as IPS. However, it has been demonstrated that the variance reduction is limited and DR suffers from high variance when the action space is very large [26]. This is because DR uses the same importance weights as IPS.

To mitigate the above variance issues, *Marginalized IPS* (MIPS) [26] leverages auxiliary information called action embeddings. MIPS applies importance weighting only to embeddings. MIPS significantly reduces variance, as the embedding space becomes smaller than the original action space. Moreover, MIPS remains unbiased under the *no direct effect* assumption, which assumes that embeddings fully mediate every possible effect of the actions on the rewards. However, this estimator may have a nonnegligible bias when two actions that have the same embedding have dissimilar rewards [27].

Off-policy evaluation estimator based on the Conjunct Effect Model (OffCEM) [27] aims to remove the no direct assumption of MIPS by introducing the conjunct effect model (CEM), which decomposes the expected reward function into cluster effects and residual effects. Based on this decomposition, OffCEM estimates the cluster and residual effect in different ways as follows.

$$\hat{V}_{\mathrm{OffCEM}}(\pi; \mathcal{D}, \hat{f}) = \frac{1}{n} \sum_{i=1}^{n} \left\{ w(x_i, \phi(x_i, a_i))(r_i - \hat{f}(x_i, a_i)) + \hat{f}(x_i, \pi) \right\},$$

where $w(x, c) \coloneqq \frac{\sum_{\mathcal{A} \in a} \mathbb{I}\{\phi(x,a)=c\}\pi(a|x)}{\sum_{\mathcal{A} \in a} \mathbb{I}\{\phi(x,a)=c\}\pi_0(a|x)}$ is the cluster importance weight. OffCEM deals with the cluster effects through the cluster importance weight and the residual effects by the regression model. OffCEM is unbiased under the local correctness assumption, which only requires that the regression model preserves the relative expected reward differences between actions in the same cluster. OffCEM can achieve lower variance, as the cluster importance weight becomes small like MIPS. Moreover, OffCEM outperforms MIPS since OffCEM deals with the residual effects ignored by MIPS. Thus, OffCEM provides substantial improvements for OPE in large action spaces.

Based on the OffCEM's concept, our proposed gradient estimator deals with the effects of top-k actions via importance weighting and the effects of the bottom actions via a reward regression model. By doing so, our proposed gradient estimator can achieve low MSE in ranking settings.

A.2 Off-Policy Evaluation for Ranking Policies

In ranking settings, the action spaces consist of permutations of unique items, thus we encounter severe variance problems caused by large action spaces. To deal with the bias and variance issues of DM, IPS, and DR, existing estimators have tried to reduce variance by leveraging sub-rewards and some assumptions on user behavior in examining rankings based on click models [8, 3]. For instance, *Independent Inverse Propensity Scoring* (IIPS) [14] assumes that users interact with items independently across positions. IIPS applies the position-wise importance weights to the observed position-wise rewards. Therefore, IIPS reduces variance substantially. However, when the independent assumption does not hold, this estimator introduces high bias, as empirically demonstrated in [18, 12]. Similarly, *Reward interaction Inverse Propensity Scoring* (RIPS) [18] is based on another assumption called the *cascade* assumption, which assumes that users interact with items in a ranking sequentially from the top position. RIPS is unbiased under the cascade assumption, which is weaker and more reasonable than IIPS, while also reducing the variance compared to IPS. However, this estimator still incurs considerable bias by ignoring the interactions from lower positions. Cascade-DR [12] is also based on the cascade assumption, which introduces a baseline estimator as a control variate similar to DR. Cascade-DR can further reduce variance compared to RIPS. The estimators above put a single assumption about user behavior for every user to deal with the variance issue in ranking settings. However, in practice, user behavior is diverse among users [2, 36]. To conduct effective estimations under diverse user behavior, *Adaptive IPS* (AIPS) [13] generalizes the idea of importance weighting based on the user behavior, by considering *adaptive* importance weight depending on each user (x). AIPS puts different user behavior assumptions for different users adaptively to deal with diverse user behavior. Under diverse user behavior, AIPS provides substantial improvements compared to other estimators, such as IIPS and RIPS. Unlike these estimators that rely on some user behavior assumption, [37] views ranking settings as episodic RL settings by considering a click model as a Markov Decision Process (MDP). This method enables us to leverage some estimators from offline RL.

Our proposed OPL method depends on the intuitive idea that the effects of higher-ranked items largely dominate the expected reward of a ranking. Although this idea is similar to the cascade assumption, our proposed gradient estimator includes a reward regression model to consider the effect of the bottom actions, ignored by the cascade assumption. By doing so, our proposed gradient estimator provides substantial improvements in OPL without incurring extra bias under any user behavior.

A.3 Off-Policy Learning

Off-policy learning (OPL) for contextual bandits enables us to learn a new policy only using the logged data. In OPL, two main approaches are the regression- and the policy-based approaches. The regression-based approach learns the expected reward function via a familiar supervised learning method and then chooses the highest predicted reward. A drawback of this approach is bias due to the difficulty of accurately learning the expected rewards. In contrast, the policy-based approach optimizes a new policy π_θ, parameterized by θ, via iterative gradient ascent as $\theta_{t+1} \leftarrow \theta_t + \eta \nabla_\theta V(\pi_\theta)$. However, we do not know the true policy gradient $\nabla_\theta V(\pi_\theta) (= \mathbb{E}_{p(x)\pi_\theta(a|x)} [q(x,a) \nabla_\theta \log \pi_\theta(a|x)])$, so we need to estimate it from the historical logged data. A typical way to do so is to apply importance weighting as follows.

$$\nabla_\theta \widehat{V}_{\mathrm{IPS}}(\pi_\theta; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} w(x_i, a_i) r_i s_\theta(x_i, a_i),$$

where $w(x,a) = \frac{\pi_\theta(a|x)}{\pi_0(a|x)}$ is the (vanilla) importance weight and $s_\theta(x,a) = \nabla_\theta \log \pi_\theta(a|x)$ is the policy score function. This policy gradient estimator is unbiased under the full support condition ($\pi_0(a|x) > 0$ for all $a \in \mathcal{A}, x \in \mathcal{X}$). In large action spaces, it struggles with two critical issues. First, the requirement is often violated, resulting in introducing substantial bias in gradient estimations[23, 7]. Second, the vanilla importance weight takes a large value when the requirement is satisfied. This leads to serious variance. To tackle the variance issues in large action spaces, some OPL methods utilize some regularization techniques [9, 17, 15]. These methods could avoid the variance issues by imposing penalties on the difference between an optimized policy and the logging policy. However, it might not improve the policy value, as a policy is likely to be close to the logging policy.

To deal with the limitations of existing approaches in large action spaces, a novel two-stage OPL algorithm called ***P**olicy **O**ptimization via **T**wo-Stage Policy D**ec**omposition (**POTEC**)* [28] leverages

the Conjunct Effect Model (CEM) from [27]. POTEC decomposes a policy into a first-stage policy and a second-stage policy as follows.

$$\pi_{\theta,\phi}^{overall}(a|x) = \sum_{c \in \mathcal{C}} \pi_{\theta}^{1st}(c|x)\pi_{\phi}^{2nd}(a|x,c),$$

where $c \in \mathcal{C}$ denotes an action cluster. The first-stage policy π_{θ}^{1st} chooses a promising cluster and then the second-stage policy π_{ϕ}^{2nd} chooses a best action within the cluster selected by the first-stage policy. The POTEC algorithm optimizes the first-stage policy via the policy-based approach and the second-stage policy via the regression-based approach. The POTEC gradient estimator for the first-stage policy is defined as:

$$\nabla_{\theta}\hat{V}_{\mathrm{POTEC}}\left(\pi_{\theta,\phi}^{overall};\mathcal{D}\right) = \frac{1}{n}\sum_{i=1}^{n}\left\{w(x_i,c_{a_i})(r_i - \hat{f}(x_i,a_i))s_{\theta}(x_i,c_{a_i}) + \mathbb{E}_{\pi_{\theta}^{1st}}\left[\hat{f}^{\pi_{\phi}^{2nd}}(x_i,c)s_{\theta}(x_i,c)\right]\right\},$$

where $w(x,c) = \frac{\pi_{\theta}^{1st}(c|x)}{\pi_0(c|x)}$ is the cluster importance weight and $\hat{f}^{\pi_{\phi}^{2nd}}(x,c) = \mathbb{E}_{\pi_{\phi}^{2nd}(a|x,c)}\left[\hat{f}(x,a)\right]$. The POTEC gradient estimator applies importance weight over the action cluster space, which is much smaller than the original action space, so it provides a substantial reduction of variance rather than conventional gradient estimators. Moreover, it is unbiased under the same local correctness condition as OffCEM. The second-stage policy, on the other hand, should select the optimal actions within a cluster selected by the first-stage policy. The second-stage policy is optimized based on the regression model. To summarize, the POTEC algorithm applies a policy-based approach to the first-stage policy to select promising clusters with low bias and variance using the cluster importance weight. It then optimizes the second-stage policy via a regression-based approach with low variance to identify the best actions within a cluster chosen by the first-stage policy.

Our OPL algorithm is inspired by the POTEC algorithm in terms of using both policy-based approaches and regression-based approaches. However, there are clear differences in that the proposed R-POD algorithm decomposes a ranking policy by leveraging the unique structure inherent to rankings, while the POTEC algorithm considers cluster spaces. Moreover, POTEC should optimize clustering methods to conduct effective OPL. In contrast, what R-POD has to do is just choose hyperparameter k. This suggests that we can control the bias and variance of R-POD easily. Thus, our algorithm makes the most of ranking situations to deal with challenging OPL for ranking policies.

B Optimizing the regression model via a two-step procedure

We can optimize the regression model via a two-step procedure not the one-step procedure in Eq.12. We show how to optimize and use the two-step procedure in the R-POD algorithm.

B.1 Two-step Procedure

Proposition 4.2 suggests that, in terms of variance minimization, we should optimize the regression model in a way that minimizes $|\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})|$ compared to minimizing $|\Delta_q(\boldsymbol{x},\boldsymbol{a},\boldsymbol{b}) - \Delta_{\hat{f}}(\boldsymbol{x},\boldsymbol{a},\boldsymbol{b})|$ for the bias. Therefore, based on the theoretical observations, we should ideally optimize the regression model via the following two-step procedure in order to optimize the bias and variance of the R-POD gradient estimator.

1. Bias Minimization Step: Optimize a pairwise regression function $\hat{h}_{\phi}$, parameterized by ϕ, to estimate the relative value differences of ranking sharing the same top-k actions.

$$\min_{\phi} \sum_{(\boldsymbol{x},\boldsymbol{a},\boldsymbol{b},\boldsymbol{r_a},\boldsymbol{r_b}) \in \mathcal{D}_{pair}} \ell_h\left(\boldsymbol{r_a} - \boldsymbol{r_b}, h_{\phi}(\boldsymbol{x},\boldsymbol{a}) - h_{\phi}(\boldsymbol{x},\boldsymbol{b})\right) \tag{16}$$

2. Variance Minimization Step: Optimize $\hat{g}_{\omega}$, parameterized by ω, to minimize $\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})$ given $\hat{f} = \hat{g}_{\omega} + \hat{h}_{\phi}$.

$$\min_{\omega} \sum_{(\boldsymbol{x},\boldsymbol{a},\boldsymbol{r}) \in \mathcal{D}} \ell_g\left(\boldsymbol{r}, \hat{g}_{\omega}(\boldsymbol{x},\boldsymbol{a}) + \hat{h}_{\phi}(\boldsymbol{x},\boldsymbol{a})\right) \tag{17}$$

Algorithm 1 The R-POD Algorithm (w/ two-step procedure to optimize the regression model)

Require: logged data $\mathcal{D}$, logging policy π_0, hyperparameter k.
Ensure: first-stage policy $\pi_\theta^{1st}(\boldsymbol{a}_{1:k} \mid \boldsymbol{x})$ and second-stage policy $\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L} \mid \boldsymbol{x}, \boldsymbol{a}_{1:k})$

1: Perform pairwise regression and obtain $\hat{h}_\phi(\boldsymbol{x}, \boldsymbol{a})$ as in Eq. (16), which works as the second-stage policy as in Eq. (17) and also as a part of the regression model to help train the first-stage policy
2: Regress the reward residual from pairwise regression and obtain $\hat{g}_\omega(\boldsymbol{x}, \boldsymbol{a}_{1:k+1})$ as in Eq. (17)
3: Perform policy-based learning of the first-stage policy based on the R-POD gradient estimator in Eq. (9) and the regression model $\hat{f}_{\omega,\phi}(\boldsymbol{x}, \boldsymbol{a}) = \hat{g}_\omega(\boldsymbol{x}, \boldsymbol{a}_{1:k+1}) + \hat{h}_\phi(\boldsymbol{x}, \boldsymbol{a})$

$\ell_h, \ell_g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ are some appropriate loss functions such as squared loss. Note that $\mathcal{D}_{pair}$ is a dataset augmented for performing pairwise regression to minimize the bias of the R-POD gradient estimator, which is defined as

$$\mathcal{D}_{pair} := \left\{ (\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{r_a}, \boldsymbol{r_b}) \mid \begin{array}{l} (\boldsymbol{x_a}, \boldsymbol{a}, \boldsymbol{r_a}), (\boldsymbol{x_b}, \boldsymbol{b}, \boldsymbol{r_b}) \in \mathcal{D} \\ \boldsymbol{x} = \boldsymbol{x_a} = \boldsymbol{x_b}, \boldsymbol{a}_{1:k} = \boldsymbol{b}_{1:k} \end{array} \right\}. \tag{18}$$

As suggested in our analysis, $\hat{h}_\phi(\boldsymbol{x}, \boldsymbol{a})$ characterizes the bias of the R-POD gradient estimator, so the first step focuses on minimizing its bias by optimizing it towards accurately estimating the relative reward differences. The second step then aims for variance minimization by optimizing $\hat{g}_\omega$. Since the bias of the R-POD gradient estimator does not depend on the top-k actions (as in Theorem 4.1), the second step minimizes its variance without affecting its bias. After performing the two-step regression procedure, we can construct a regression model as $\hat{f}_{\omega,\phi}(\boldsymbol{x}, \boldsymbol{a}) = \hat{g}_\omega(\boldsymbol{x}, \boldsymbol{a}_{1:k}) + \hat{h}_\phi(\boldsymbol{x}, \boldsymbol{a})$ that is to be used as a part of the R-POD gradient estimator.

Note that we do not expect that the two-step procedure described above is always feasible and practical due to its need for pairwise data and implementation costs. If the two-step procedure is impractical due to a lack of sufficient pairwise data, we can employ a regression for the expected absolute reward in Eq. (12).

B.2 Optimizing the Second-stage Policy π_ϕ^{2nd}

The objective of the second-stage policy π_ϕ^{2nd} is to identify the bottom actions to optimize the value given the top-k actions chosen by the first-stage policy π_θ^{1st}. Specifically, the second-stage policy should rank the bottom actions that yield the highest value among rankings containing the same top-k actions. Fortunately, from the two-step regression procedure mentioned earlier, we have already obtained $\hat{h}_\phi$ aiming to preserve the relative value difference of rankings. This allows us to readily define the second-stage policy based on the pairwise regressor $\hat{h}_\phi$ as follows.

$$\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L} | \boldsymbol{x}, \boldsymbol{a}_{1:k}) := \begin{cases} 1 & (\boldsymbol{a} = \arg\max_{\boldsymbol{a}' : \boldsymbol{a}'_{1:k} = \boldsymbol{a}_{1:k}} \hat{h}_\phi(\boldsymbol{x}, \boldsymbol{a}')) \\ 0 & (otherwise) \end{cases} \tag{19}$$

The second-stage policy is constructed solely based on pairwise regressor $\hat{h}_\phi$. Although optimized by the regression-based approach, it only needs to learn the relative value difference of rankings, which is easier than precisely learning the global reward function. Thus, the second-stage policy of our method is expected to produce lower bias compared to regression-based approaches.

B.3 The Overall R-POD Algorithm

The overall process of the R-POD algorithm is given in Algorithm 1. First, we construct the regression model $\hat{f}_{\omega,\phi}$ through two-step regression in Eq. (16) and Eq. (17). We then formulate the second-stage policy π_ϕ^{2nd} based on the pairwise regressor $\hat{h}_\phi$ (as in Eq. (19)). We are then be based on $\hat{f}_{\omega,\phi}(\boldsymbol{x}, \boldsymbol{a}) = \hat{g}_\omega(\boldsymbol{x}, \boldsymbol{a}_{1:k+1}) + \hat{h}_\phi(\boldsymbol{x}, \boldsymbol{a})$ to optimize the first-stage policy π_θ^{1st} via iterative gradient ascent using the R-POD gradient estimator in Eq. (9).

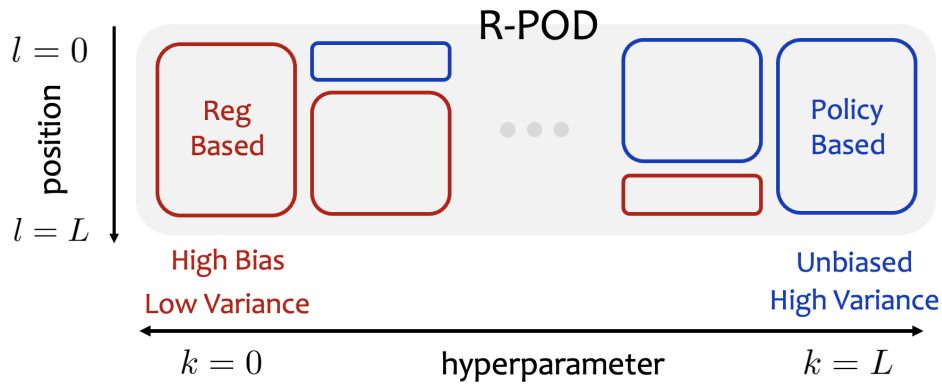


Figure 3: The R-POD algorithm mixes policy- and regression-based approaches via its hyperparameter k. When $k = L$, R-POD is reduced to the policy-based approach, while it is reduced to the regression-based approach with $k = 0$.

B.4 The Role of Hyperparameter k in R-POD

The hyperparameter k plays a crucial role in deciding the effectiveness of the R-POD algorithm. When k is large, the bias of the gradient estimator for the first-stage policy is expected to be small. This is because decreasing the number of bottom actions given the top-k actions makes CPC milder. In the extreme case where $k = L$, the R-POD gradient estimator becomes unbiased irrespective of the accuracy of the regression model because CPC requires nothing. In contrast, the variance of the R-POD gradient estimator may increase because a larger number of top-k actions leads to higher variance in the top-k importance weight. Conversely, when k is small, the variance of the R-POD gradient estimator decreases while its bias increases. It should be noted that when $k = L$, the first-stage policy becomes identical to the overall policy and thus the R-POD algorithm is reduced to the policy-based approach. In contrast, when $k = 0$, the second-stage policy becomes identical to the overall policy and thus R-POD is reduced to the regression-based approach. Therefore, an intriguing interpretation of the hyperparameter k is that it determines the mixture ratio of the policy- and regression-based approaches in R-POD as described in Figure 3.

C Omitted Proofs

Here, we provide the derivations and proofs that are omitted in the main text.

C.1 Derivation of Eq. (8)

Proof. We derive the overall policy gradient in Eq. (8).

$$
\begin{aligned}
\nabla_\theta V(\pi_{\theta,\phi}^{overall}) &= \nabla_\theta \mathbb{E}_{p(\boldsymbol{x})\pi_{\theta,\phi}^{overall}(\boldsymbol{a}\,|\,\boldsymbol{x})}\left[\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x},\boldsymbol{a})\right] \\
&= \nabla_\theta \mathbb{E}_{p(\boldsymbol{x})\pi_\theta^{1st}(\boldsymbol{a}_{1:k}\,|\,\boldsymbol{x})\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}\,|\,\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x},\boldsymbol{a})\right] \\
&= \nabla_\theta \mathbb{E}_{p(\boldsymbol{x})}\left[\sum_{\boldsymbol{a}_{1:k}}\sum_{\boldsymbol{a}_{k+1:L}} \pi_\theta^{1st}(\boldsymbol{a}_{1:k}\,|\,\boldsymbol{x})\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}\,|\,\boldsymbol{x},\boldsymbol{a}_{1:k})\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x},\boldsymbol{a})\right] \\
&= \mathbb{E}_{p(\boldsymbol{x})}\left[\sum_{\boldsymbol{a}_{1:k}}\sum_{\boldsymbol{a}_{k+1:L}} \nabla_\theta \pi_\theta^{1st}(\boldsymbol{a}_{1:k}\,|\,\boldsymbol{x})\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}\,|\,\boldsymbol{x},\boldsymbol{a}_{1:k})\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x},\boldsymbol{a})\right] \\
&= \mathbb{E}_{p(\boldsymbol{x})}\left[\sum_{\boldsymbol{a}_{1:k}}\sum_{\boldsymbol{a}_{k+1:L}} \pi_\theta^{1st}(\boldsymbol{a}_{1:k}\,|\,\boldsymbol{x})\nabla_\theta \log \pi_\theta^{1st}(\boldsymbol{a}_{1:k}\,|\,\boldsymbol{x})\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}\,|\,\boldsymbol{x},\boldsymbol{a}_{1:k})\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x},\boldsymbol{a})\right]
\end{aligned}
$$

$$
\begin{aligned}
&= \mathbb{E}_{p(\boldsymbol{x})} \left[\sum_{\boldsymbol{a}_{1:k}} \sum_{\boldsymbol{a}_{k+1:L}} \pi_\theta^{1st}(\boldsymbol{a}_{1:k} \,|\, \boldsymbol{x}) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L} \,|\, \boldsymbol{x}, \boldsymbol{a}_{1:k}) \sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a}) \right] \\
&= \mathbb{E}_{p(\boldsymbol{x}) \pi_\theta^{1st}(\boldsymbol{a}_{1:k} \,|\, \boldsymbol{x})} \left[s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \sum_{\boldsymbol{a}_{k+1:L}} \pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L} \,|\, \boldsymbol{x}, \boldsymbol{a}_{1:k}) \sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a}) \right] \\
&= \mathbb{E}_{p(\boldsymbol{x}) \pi_\theta^{1st}(\boldsymbol{a}_{1:k} \,|\, \boldsymbol{x})} \left[s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \mathbb{E}_{\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L} \,|\, \boldsymbol{x}, \boldsymbol{a}_{1:k})} \left[\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a}) \right] \right] \\
&= \mathbb{E}_{p(\boldsymbol{x}) \pi_\theta^{1st}(\boldsymbol{a}_{1:k} \,|\, \boldsymbol{x})} \left[q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right] \qquad (20)
\end{aligned}
$$

where we use $\mathbb{E}_{\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L} \,|\, \boldsymbol{x}, \boldsymbol{a}_{1:k})} \left[\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a}) \right] \coloneqq q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k})$ and $\nabla_\theta \log \pi_\theta^{1st}(\boldsymbol{a}_{1:k} \,|\, \boldsymbol{x}) \coloneqq s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k})$. □

C.2 Derivation of Eq. (10)

Proof. We derive the bias of R-POD in Eq. (10).

$$
\begin{aligned}
&Bias(\nabla_\theta \widehat{V}_{RPOD}(\pi_{\theta,\phi}^{overall}; \mathcal{D})) \\
&= \mathbb{E}_{p(\boldsymbol{x}) \pi_0(\boldsymbol{a}|\boldsymbol{x})} \left[w(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \left(\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a}) - \hat{f}(\boldsymbol{x}, \boldsymbol{a}) \right) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right] + \mathbb{E}_{p(\boldsymbol{x}) \pi_\theta^{1st}(\boldsymbol{a}_{1:k} \,|\, \boldsymbol{x})} \left[\hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right] \\
&\quad - \mathbb{E}_{p(\boldsymbol{x}) \pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})} \left[q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right] \\
&= \mathbb{E}_{p(\boldsymbol{x}) \pi_0^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x}) \pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x}, \boldsymbol{a}_{1:k})} \left[w(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \left(\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a}) - \hat{f}(\boldsymbol{x}, \boldsymbol{a}) \right) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right] \\
&\quad - \mathbb{E}_{p(\boldsymbol{x}) \pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})} \left[\left(q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) - \hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right] \\
&= \mathbb{E}_{p(\boldsymbol{x}) \pi_0^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})} \left[\sum_{\boldsymbol{a}_{k+1:L}} \pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x}, \boldsymbol{a}_{1:k}) w(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \left(\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a}) - \hat{f}(\boldsymbol{x}, \boldsymbol{a}) \right) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right] \\
&\quad - \mathbb{E}_{p(\boldsymbol{x})} \left[\sum_{\boldsymbol{a}_{1:k}} \pi_\theta^{1st}(\boldsymbol{a}_{1:k} \,|\, \boldsymbol{x}) \left(q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) - \hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right] \\
&= \mathbb{E}_{p(\boldsymbol{x}) \pi_0^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})} \left[w(\boldsymbol{x}, \boldsymbol{a}_{1:k}) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \sum_{\boldsymbol{a}_{k+1:L}} \pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x}, \boldsymbol{a}_{1:k}) \left(\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a}) - \hat{f}(\boldsymbol{x}, \boldsymbol{a}) \right) \right] \\
&\quad - \mathbb{E}_{p(\boldsymbol{x})} \left[\sum_{\boldsymbol{a}_{1:k}} \pi_0^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x}) \frac{\pi_\theta^{1st}(\boldsymbol{a}_{1:k} \,|\, \boldsymbol{x})}{\pi_0^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})} \left(q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) - \hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right] \\
&= \mathbb{E}_{p(\boldsymbol{x}) \pi_0^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})} \left[w(\boldsymbol{x}, \boldsymbol{a}_{1:k}) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \sum_{\boldsymbol{a}_{k+1:L}} \pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x}, \boldsymbol{a}_{1:k}) \left(\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{a}) - \hat{f}(\boldsymbol{x}, \boldsymbol{a}) \right) \right] \\
&\quad - \mathbb{E}_{p(\boldsymbol{x}) \pi_0^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})} \left[w(\boldsymbol{x}, \boldsymbol{a}_{1:k}) s_\theta(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \left(q^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) - \hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x}, \boldsymbol{a}_{1:k}) \right) \right] \\
&= \mathbb{E}_{p(\boldsymbol{x}) \pi_0^{1st}(\boldsymbol{c}_{1:k}|\boldsymbol{x})} \Bigg[s_\theta(\boldsymbol{x}, \boldsymbol{c}_{1:k}) \sum_{\boldsymbol{a}:\boldsymbol{a}_{1:k}=\boldsymbol{c}_{1:k}} w(\boldsymbol{x}, \boldsymbol{a}) \pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x}, \boldsymbol{c}_{1:k}) \\
&\qquad\qquad \sum_{\boldsymbol{b}:\boldsymbol{b}_{1:k}=\boldsymbol{c}_{1:k}} \pi_0^{2nd}(\boldsymbol{b}_{k+1:L}|\boldsymbol{x}, \boldsymbol{c}_{1:k}) \left(\sum_{l=1}^{L} \alpha_l q_l(\boldsymbol{x}, \boldsymbol{b}) - \hat{f}(\boldsymbol{x}, \boldsymbol{b}) \right) \Bigg]
\end{aligned}
$$

$$- \mathbb{E}_{p(\boldsymbol{x})\pi_0^{1st}(\boldsymbol{c}_{1:k}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{c}_{1:k})s_\theta(\boldsymbol{x},\boldsymbol{c}_{1:k})\sum_{\boldsymbol{a}:\boldsymbol{a}_{1:k}=\boldsymbol{c}_{1:k}}\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\left(\sum_{l=1}^{L}\alpha_l q_l(\boldsymbol{x},\boldsymbol{a})-\hat{f}(\boldsymbol{x},\boldsymbol{a})\right)\right] \tag{21}$$

$$= \mathbb{E}_{p(\boldsymbol{x})\pi_0^{1st}(\boldsymbol{c}_{1:k}|\boldsymbol{x})}\left[s_\theta(\boldsymbol{x},\boldsymbol{c}_{1:k})\sum_{\boldsymbol{a}:\boldsymbol{a}_{1:k}=\boldsymbol{c}_{1:k}}w(\boldsymbol{x},\boldsymbol{a})\pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\right.$$
$$\left.\sum_{\boldsymbol{b}:\boldsymbol{b}_{1:k}=\boldsymbol{c}_{1:k}}\pi_0^{2nd}(\boldsymbol{b}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\left(\sum_{l=1}^{L}\alpha_l q_l(\boldsymbol{x},\boldsymbol{b})-\hat{f}(\boldsymbol{x},\boldsymbol{b})\right)\right]$$
$$- \mathbb{E}_{p(\boldsymbol{x})\pi_0^{1st}(\boldsymbol{c}_{1:k}|\boldsymbol{x})}\left[s_\theta(\boldsymbol{x},\boldsymbol{c}_{1:k})\sum_{\boldsymbol{a}:\boldsymbol{a}_{1:k}=\boldsymbol{c}_{1:k}}w(\boldsymbol{x},\boldsymbol{c}_{1:k})\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\left(\sum_{l=1}^{L}\alpha_l q_l(\boldsymbol{x},\boldsymbol{a})-\hat{f}(\boldsymbol{x},\boldsymbol{a})\right)\right]$$
$$= \mathbb{E}_{p(\boldsymbol{x})\pi_0^{1st}(\boldsymbol{c}_{1:k}|\boldsymbol{x})}\left[s_\theta(\boldsymbol{x},\boldsymbol{c}_{1:k})\sum_{\boldsymbol{a}:\boldsymbol{a}_{1:k}=\boldsymbol{c}_{1:k}}w(\boldsymbol{x},\boldsymbol{a})\pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\sum_{\boldsymbol{b}:\boldsymbol{b}_{1:k}=\boldsymbol{c}_{1:k}}\pi_0^{2nd}(\boldsymbol{b}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{b})\right]$$
$$- \mathbb{E}_{p(\boldsymbol{x})\pi_0^{1st}(\boldsymbol{c}_{1:k}|\boldsymbol{x})}\left[s_\theta(\boldsymbol{x},\boldsymbol{c}_{1:k})\sum_{\boldsymbol{a}:\boldsymbol{a}_{1:k}=\boldsymbol{c}_{1:k}}w(\boldsymbol{x},\boldsymbol{c}_{1:k})\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{b})\right]$$
$$= \mathbb{E}_{p(\boldsymbol{x})\pi_0^{1st}(\boldsymbol{c}_{1:k}|\boldsymbol{x})}\left[s_\theta(\boldsymbol{x},\boldsymbol{c}_{1:k})\sum_{\boldsymbol{a}:\boldsymbol{a}_{1:k}=\boldsymbol{c}_{1:k}}w(\boldsymbol{x},\boldsymbol{a})\pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\right.$$
$$\left.\left(\left(\sum_{\boldsymbol{b}:\boldsymbol{b}_{1:k}=\boldsymbol{c}_{1:k}}\pi_0^{2nd}(\boldsymbol{b}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{b})\right)-\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})\right)\right]$$

where $\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a}) := \sum_{l=1}^{L}\alpha_l q_l(\boldsymbol{x},\boldsymbol{a}) - \hat{f}(\boldsymbol{x},\boldsymbol{a})$. We use $q^{\pi_\phi^{2nd}}(\boldsymbol{x},\boldsymbol{a}_{1:k}) = \mathbb{E}_{\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}\,|\,\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[\sum_{l=1}^{L}\alpha_l q_l(\boldsymbol{x},\boldsymbol{a})\right]$ and $\hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x},\boldsymbol{a}_{1:k}) = \mathbb{E}_{\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}\,|\,\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[\hat{f}(\boldsymbol{x},\boldsymbol{a})\right]$ in Eq. (21). We use Lemma B.1 of [26] and then get the following.

$$\mathbb{E}_{p(\boldsymbol{x})\pi_0^{1st}(\boldsymbol{c}_{1:k}|\boldsymbol{x})}\left[s_\theta(\boldsymbol{x},\boldsymbol{c}_{1:k})\sum_{\boldsymbol{a}<\boldsymbol{b}:\boldsymbol{a}_{1:k}=\boldsymbol{b}_{1:k}=\boldsymbol{c}_{1:k}}\pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\pi_0^{2nd}\right.$$
$$\left.(\boldsymbol{b}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\left(\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})-\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{b})\right)(w(\boldsymbol{x},\boldsymbol{b})-w(\boldsymbol{x},\boldsymbol{a}))\right]$$
$$= \mathbb{E}_{p(\boldsymbol{x})\pi_0^{1st}(\boldsymbol{c}_{1:k}|\boldsymbol{x})}\left[\sum_{\boldsymbol{a}<\boldsymbol{b}:\boldsymbol{a}_{1:k}=\boldsymbol{b}_{1:k}=\boldsymbol{c}_{1:k}}\pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})\pi_0^{2nd}.\right.$$
$$\left.(\boldsymbol{b}_{k+1:L}|\boldsymbol{x},\boldsymbol{c}_{1:k})(\Delta_q(\boldsymbol{x},\boldsymbol{a},\boldsymbol{b})-\Delta_{\hat{f}}(\boldsymbol{x},\boldsymbol{a},\boldsymbol{b}))s_\theta(\boldsymbol{x},\boldsymbol{c}_{1:k})\left(\frac{\pi_\theta(\boldsymbol{b}|\boldsymbol{x},\boldsymbol{c}_{1:k})}{\pi_0(\boldsymbol{b}|\boldsymbol{x},\boldsymbol{c}_{1:k})}-\frac{\pi_\theta(\boldsymbol{a}|\boldsymbol{x},\boldsymbol{c}_{1:k})}{\pi_0(\boldsymbol{a}|\boldsymbol{x},\boldsymbol{c}_{1:k})}\right)\right],$$

where we use $\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})-\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{b}) \Rightarrow \Delta_q(\boldsymbol{x},\boldsymbol{a},\boldsymbol{b})-\Delta_{\hat{f}}(\boldsymbol{x},\boldsymbol{a},\boldsymbol{b})$. □

C.3 Derivation of Proposition 4.2

Proof. We derive the variance of R-POD under Condition 4.1 and Condition 4.2 by applying the law of total variance several times.

$$n\mathbb{V}_{\mathcal{D}}(\nabla_\theta \widehat{V}_{RPOD}^{(j)}(\pi_{\theta,\phi}^{overall};\mathcal{D}))$$
$$= \mathbb{V}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})p(\boldsymbol{r}|\boldsymbol{x},\boldsymbol{a})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\left(\sum_{l=1}^{L}\alpha_l r_l-\hat{f}(\boldsymbol{x},\boldsymbol{a})\right)s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})+\mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[\hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]$$
$$= \mathbb{E}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\mathbb{V}_{p(\boldsymbol{r}|\boldsymbol{x},\boldsymbol{a})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\left(\sum_{l=1}^{L}\alpha_l r_l-\hat{f}(\boldsymbol{x},\boldsymbol{a})\right)s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})+\mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[\hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\right]$$

$$
\begin{aligned}
&+ \mathbb{V}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\mathbb{E}_{p(\boldsymbol{r}|\boldsymbol{x},\boldsymbol{a})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\left(\sum_{l=1}^{L}\alpha_l r_l - \hat{f}(\boldsymbol{x},\boldsymbol{a})\right)s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k}) + \mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[\hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\\
&= \mathbb{E}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\left(w(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right)^2\sigma^2(\boldsymbol{x},\boldsymbol{a})\right]\\
&\quad + \mathbb{V}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\left(\sum_{l=1}^{L}\alpha_l q_l(\boldsymbol{x},\boldsymbol{a}) - \hat{f}(\boldsymbol{x},\boldsymbol{a})\right)s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k}) + \mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[\hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\\
&= \mathbb{E}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\left(w(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right)^2\sigma^2(\boldsymbol{x},\boldsymbol{a})\right]\\
&\quad + \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{V}_{\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\left(\sum_{l=1}^{L}\alpha_l q_l(\boldsymbol{x},\boldsymbol{a}) - \hat{f}(\boldsymbol{x},\boldsymbol{a})\right)s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k}) + \mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[\hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\right]\\
&\quad + \mathbb{V}_{p(\boldsymbol{x})}\left[\mathbb{E}_{\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\left(\sum_{l=1}^{L}\alpha_l q_l(\boldsymbol{x},\boldsymbol{a}) - \hat{f}(\boldsymbol{x},\boldsymbol{a})\right)s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k}) + \mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[\hat{f}^{\pi_\phi^{2nd}}(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\right]\\
&= \mathbb{E}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\left(w(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right)^2\sigma^2(\boldsymbol{x},\boldsymbol{a})\right] + \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{V}_{\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\\
&\quad + \mathbb{V}_{p(\boldsymbol{x})}\Big[\mathbb{E}_{\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\\
&\quad + \mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[\mathbb{E}_{\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[\hat{f}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\Big]\\
&= \mathbb{E}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\left(w(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right)^2\sigma^2(\boldsymbol{x},\boldsymbol{a})\right] + \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{V}_{\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\\
&\quad + \mathbb{V}_{p(\boldsymbol{x})}\Big[\mathbb{E}_{\pi_0^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})\pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\\
&\quad + \mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[\mathbb{E}_{\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[\hat{f}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\Big]\\
&= \mathbb{E}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\left(w(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right)^2\sigma^2(\boldsymbol{x},\boldsymbol{a})\right] + \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{V}_{\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\\
&\quad + \mathbb{V}_{p(\boldsymbol{x})}\Big[\mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}\,|\,\boldsymbol{x})}\left[\mathbb{E}_{\pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\\
&\quad + \mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[\mathbb{E}_{\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[\hat{f}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\Big]\\
&= \mathbb{E}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\left(w(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right)^2\sigma^2(\boldsymbol{x},\boldsymbol{a})\right] + \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{V}_{\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\\
&\quad + \mathbb{V}_{p(\boldsymbol{x})}\Big[\mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}\,|\,\boldsymbol{x})}\left[\mathbb{E}_{\pi_0^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[g(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\\
&\quad + \mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[\mathbb{E}_{\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[(q(\boldsymbol{x},\boldsymbol{a}) - g(\boldsymbol{x},\boldsymbol{a}_{1:k}))\,s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\Big]
\end{aligned}
\tag{22}
$$

$$
\begin{aligned}
&= \mathbb{E}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\left(w(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right)^2\sigma^2(\boldsymbol{x},\boldsymbol{a})\right] + \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{V}_{\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\\
&\quad + \mathbb{V}_{p(\boldsymbol{x})}\Big[\mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}\,|\,\boldsymbol{x})}\left[g(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right] + \mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\Big[-g(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\\
&\quad + \mathbb{E}_{\pi_\phi^{2nd}(\boldsymbol{a}_{k+1:L}|\boldsymbol{x},\boldsymbol{a}_{1:k})}\left[q(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\Big]\Big]\\
&= \mathbb{E}_{p(\boldsymbol{x})\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[\left(w(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right)^2\sigma^2(\boldsymbol{x},\boldsymbol{a})\right] + \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{V}_{\pi_0(\boldsymbol{a}|\boldsymbol{x})}\left[w(\boldsymbol{x},\boldsymbol{a}_{1:k})\Delta_{q,\hat{f}}(\boldsymbol{x},\boldsymbol{a})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right]\\
&\quad + \mathbb{V}_{p(\boldsymbol{x})}\left[\mathbb{E}_{\pi_\theta^{1st}(\boldsymbol{a}_{1:k}|\boldsymbol{x})}\left[q^{\pi_\phi^{2nd}}(\boldsymbol{x},\boldsymbol{a}_{1:k})s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})\right]\right],
\end{aligned}
$$

where we use the conditional pairwise correctness condition as $g(\boldsymbol{x},\boldsymbol{a}_{1:k}) = q(\boldsymbol{x},\boldsymbol{a}) - \hat{f}(\boldsymbol{x},\boldsymbol{a})$ in Eq. (22). $s_\theta^{(j)}(\boldsymbol{x},\boldsymbol{a}_{1:k})$ is the j-th dimension of the score function. □

D Ablation Study about Hyperparameter Selection of R-POD

We study how the hyperparameter selection strategy affects the performance of R-POD. Specifically, we compare R-POD (tuning, OPE) and R-POD (tuning, estimation error). R-POD (tuning, OPE) selects the hyperparameter k by the policy value estimated by an unbiased OPE estimator, particularly the IPS estimator. In contrast, R-POD (tuning, estimation error) selects the hyperparameter k by a noise-added ground-truth policy value, where the noise (or estimation error) is sampled from a uniform distribution of range $(-\Delta_{\max}, \Delta_{\max})$, where $|\Delta_{\max}|$ is $10\% \sim 90\%$ of the ground-truth policy value. For other configurations, we set n = 1000, $|\mathcal{A}| = 5$, $L = 3$, $\lambda = 1.0$, and $\tau = 1.0$.

Figure 4 shows that R-POD (tuning, estimation error) performs worse as it produces a larger bias with increasing $\Delta_{\max}$. R-POD (tuning, estimation error) outperforms R-POD(tuning, OPE) when the estimation error is small, and becomes competitive when $\Delta_{\max}$ is around 50% of the ground-truth policy value. This result suggests that there is room for improvement in terms of hyperparameter k selection. Still, R-POD (tuning estimation error) outperforms the baseline estimators across varying estimation errors and performs almost identically to the best-performing existing baseline (IPS-PG, DR-PG) only when $\Delta_{\max}$ is more than 0.8. This result indicates R-POD's robustness against the estimation error in the MSE estimation and the choice of hyperparameter k.

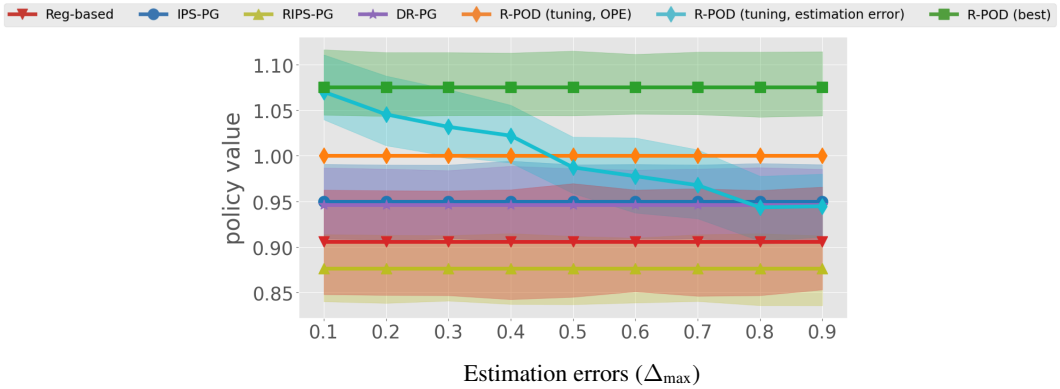


Figure 4: Comparing the policy value (normalized by R-POD (tuning, OPE)) of the OPL methods, with varying estimation errors in the hyperparameter selection process.