# Offline Contextual Bandits in the Presence of New Actions

Ren Kishimoto
Institute of Science Tokyo
kishimoto.r.ab@m.titech.ac.jp

Tatsuhiro Shimizu
Yale University
tatsuhiro.shimizu@yale.edu

Kazuki Kawamura
Sony Group Corporation
Kazuki.Kawamura@sony.com

Takanori Muroi
Sony Group Corporation
Takanori.Muroi@sony.com

Yusuke Narita
Yale University
yusuke.narita@yale.edu

Yuki Sasamoto
Sony Group Corporation
Yuki.Sasamoto@sony.com

Kei Tateno
Sony Group Corporation
Kei.Tateno@sony.com

Takuma Udagawa
Sony Group Corporation
Takuma.Udagawa@sony.com

Yuta Saito
Cornell University
ys552@cornell.edu

## Abstract

Automated decision-making algorithms drive applications in domains such as recommendation systems and search engines. These algorithms often rely on off-policy contextual bandits or *off-policy learning* (OPL). Conventionally, OPL selects actions that maximize the expected reward within an existing action set. However, in many real-world scenarios, actions—such as news articles or video content—change continuously, and the action space evolves over time compared to when the logged data was collected. We define actions introduced after deploying the logging policy as *new actions* and focus on the problem of OPL with new actions. Existing OPL methods cannot learn and select new actions because no relevant data are logged. To address this limitation, we propose a new OPL method that leverages action features. In particular, we first introduce the Local Combination PseudoInverse (LCPI) estimator for the policy gradient, generalizing the PseudoInverse estimator initially proposed for off-policy evaluation of slate bandits. LCPI controls the trade-off between reward-modeling condition and the condition for data collection regarding the action features, capturing the interaction effects among different dimensions of action features. Furthermore, we propose a generalized algorithm called ***Policy Optimization for Effective New Actions (PONA)***, which integrates LCPI, a component specialized for new action selection, with Doubly Robust (DR), which excels at learning within existing actions. We define PONA as a weighted sum of the LCPI and DR estimators, optimizing both the selection of existing and new actions, and allowing the proportion of new action selections to be adjusted by controlling the weight parameter.

**ACM Reference Format:**
Ren Kishimoto, Tatsuhiro Shimizu, Kazuki Kawamura, Takanori Muroi, Yusuke Narita, Yuki Sasamoto, Kei Tateno, Takuma Udagawa, and Yuta Saito. 2025. Offline Contextual Bandits in the Presence of New Actions. In . ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn


*Conference'17, July 2017, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 Introduction

We increasingly rely on data-driven algorithms to optimize decision-makings in various domains, including recommendation systems [6, 8, 22, 23, 27], search engines [15], advertising [2], and medical treatments [11]. These problems take the form of contextual bandits, where a policy selects actions based on observed contexts, and the corresponding rewards become available. The goal in such problems is to learn a policy that maximizes the expected reward. *Off-Policy Learning* (OPL) [17, 21, 32, 35] enables learning such policies by leveraging previously collected data, eliminating the need for online deployment and avoiding the risk of negatively impacting user experiences through active exploration [14, 23]. Typically, OPL focuses on identifying the best actions within a predefined set of existing actions. However, in many real-world scenarios, the action space evolves over time as new actions emerge or existing ones are updated. For example, search systems must process new articles daily, and recommendation systems for products or videos continually welcome new items. We define these actions, introduced after the data collection phase, as ***new actions***. In this work, we rigorously address the challenges of OPL regarding the presence of such new actions for the first time in the relevant literature.

Existing OPL methods such as policy-based and regression-based approaches are effective at identifying optimal actions within the set of existing ones [23, 28]. However, the naive applications of existing methods cannot choose new actions at all, failing to give fair opportunities to them. To address this critical limitations of the typical OPL methods, we consider and formulate a scenario where actions are represented as multi-dimensional action features. There exist many real-life problems where the actions are represented with features [6, 24]. For example, in the problem of thumbnail optimization (Figure 1) like done at Netflix [1], actions correspond to thumbnails described by features such as character type (e.g., male, female, child), title position (e.g., top, middle, bottom), and title size (e.g., large, small). In particular, the left side of the figure depicts existing thumbnails in the predefined action set, while the right side illustrates new thumbnails that are not at all observed in the logged data. As in this illustration, we consider leveraging the action features to develop a new OPL method that effectively selects new actions (i.e., unobserved combinations of action features) while still achieving a competitive overall policy performance.

Merely applying existing regression-based or policy-based approaches to observable action features does not resolve the problem

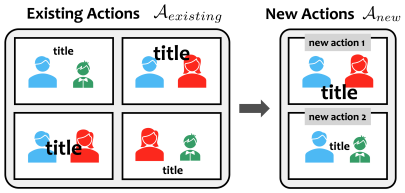


**Figure 1: Examples of Existing and New Actions.** *Note*: There are three types of action features: **character type** (chosen as two from male, female, or child), **title position** (top, center, or bottom), and **title size** (large or small). The left side of the figure illustrates the thumbnails for existing actions, while the right side shows examples of new actions: "Male and female, title position at the bottom, large title" (**new action 1**) and "Male and child, title position at the center, small title" (**new action 2**).

of new actions. This is because the combinations of action features that represent new actions are never observed in the logged data. A possible solution might be to draw inspiration from the PseudoInverse (PI) method introduced in the slate bandit setting [12, 35, 37]. Slate bandits refer to the setting where a policy aims to optimize the selection of slates, which consist of multiple action features (which are also called sub-actions) [35]. To deal with the combinatorial slate spaces and resulting variance issues, the PI estimator relies on two conditions [35, 37]: (1) *the logging policy provides full support for each action feature independently*, and (2) *the expected reward function (q-function) is a linear combination of the intrinsic value of each feature dimension*. Under these conditions, the PI estimator can estimate the policy value or policy gradient without bias even if there are new actions. However, the linearity condition is particularly restrictive because it treats the effects of action features as completely independent, ignoring potentially significant interaction effects and thus introducing bias into the estimation [12]. Therefore, we relax the linearity condition of PI to deal with local interactions among action features and propose **Local Combinatorial Feature Interaction (LCPI)**—a generalized version of the PI estimator. LCPI enables new policies to account for interactions among different dimensions of action features, allowing a more effective selection of new actions. Although LCPI is better at identifying effective new actions, existing OPL methods, such as the policy-based approach using DR to estimate the policy gradient [4], can be more effective for learning within existing actions. This is because they do not rely on any conditions about the form of the q-function. To leverage the strengths of both approaches, we finally introduce a hybrid algorithm named ***Policy Optimization for Effective New Actions (PONA)***, which integrates LCPI and DR by weighting them through a weight parameter. This design allows PONA to optimize the overall policy performance via the effective selection of existing actions while identifying promising new actions. Furthermore, by adjusting the weight parameter, PONA can control how aggressively it selects new actions, supporting both conservative (greater focus on existing actions) and aggressive (greater focus on new actions) strategies. Results on synthetic and real-world experiments demonstrate that PONA efficiently selects new actions and achieves satisfactory overall performance, whereas existing methods fail to choose new actions severely.

## 2 Preliminaries: Conventional OPL

This section formulates the typical problem of OPL without new actions and introduces existing OPL methods.

First, let $x \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ represent a context vector such as user features in recommender systems. Given a context $x$, a possibly stochastic policy $\pi(a|x)$ selects an action $a \in \mathcal{A}$ such as products or movies. Furthermore, let $r \in \mathbb{R}_{\geq 0}$ denote a reward, such as conversion label and play duration, drawn from an unknown conditional distribution $p(r|x,a)$. In the setting of OPL, we are provided with a logged dataset $\mathcal{D} := \{(x_i, a_i, r_i)\}_{i=1}^n$ collected under a logging policy $\pi_0$ such that $(x,a,r) \sim p(x)\pi_0(a|x)p(r|x,a)$.

We define the expected reward under the deployment of a policy $\pi$ (policy value) as a measure of its overall performance:

$$V(\pi) := \mathbb{E}_{p(x)\pi(a|x)p(r|x,a)}[r] = \mathbb{E}_{p(x)\pi(a|x)}[q(x,a)],$$

where $q(x,a) := \mathbb{E}[r|x,a]$ is the expected reward function or *q-function* for a given context $x$ and action $a$.

The goal in OPL is to learn a new policy $\pi_\theta$, parameterized by $\theta$, using only the logged dataset $\mathcal{D}$. When learning a policy, we aim to maximize the policy value as $\theta^* = \mathrm{argmax}_{\theta\in\Theta} V(\pi_\theta)$. The following describes two typical approaches to OPL.

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

$$\nabla_\theta V(\pi_\theta) := \mathbb{E}_{p(x)}\left[\sum_{a\in\mathcal{A}} \pi_\theta(a|x) q(x,a) \nabla_\theta \log \pi_\theta(a|x)\right], \quad (1)$$

is the policy gradient (PG). Since the true PG is unknown, we need to estimate it using the logged data $\mathcal{D}$. A common approach for this estimation is to use the IPS method [19]:

$$\nabla_\theta \widehat{V}_{\mathrm{IPS}}(\pi_\theta; \mathcal{D}) := \frac{1}{n}\sum_{i=1}^n \frac{\pi_\theta(a_i|x_i)}{\pi_0(a_i|x_i)} r_i \nabla_\theta \log \pi_\theta(a_i|x_i). \quad (2)$$

where $\pi_\theta(a|x)/\pi_0(a|x)$ is called the *importance weight*. It plays a key role in ensuring the unbiasedness of IPS but often causes the issue of high variance and inefficient policy learning [28].

An improved approach to estimate the PG is using the DR estimator [4]. It leverages a q-function estimator $\hat{q}(x,a)$ to reduce the variance in the PG estimation compared to IPS as

$$\nabla_\theta \widehat{V}_{\mathrm{DR}}(\pi_\theta; \mathcal{D}) := \frac{1}{n}\sum_{i=1}^n \frac{\pi_\theta(a_i|x_i)}{\pi_0(a_i|x_i)} \left(r_i - \hat{q}(x_i,a_i)\right) \nabla_\theta \log \pi_\theta(a_i|x_i) + \frac{1}{n}\sum_{i=1}^n \sum_{a\in\mathcal{A}} \pi_\theta(a|x_i)\hat{q}(x_i,a)\nabla_\theta \log \pi_\theta(a|x_i). \quad (3)$$

IPS and DR are both unbiased against the true PG under the *full support* condition (**a condition to ensure sufficient data collection by the logging policy** $\pi_0$).

**Condition 2.1.** *(Full Support) The logging policy $\pi_0$ is said to have full support if $\pi_0(a|x) > 0, \forall x \in \mathcal{X}, \forall a \in \mathcal{A}$.*

Secondly, the **regression-based** approach estimates the q-function $q(x,a)$ using off-the-shelf supervised machine learning methods. It then transforms the estimated q-function $\hat{q}_\theta(x,a)$ into a decision-making policy, for example, by applying the softmax function. This approach may fail due to bias issues resulting from the difficulty in

accurately estimating the q-function $q(x, a)$. However, it can avoid the issue of high variance compared to the policy-based approach based on IPS and DR [9].

Existing policy-based or regression-based methods perform effectively in identifying existing actions with high expected rewards [17, 28, 32]. However, these methods cannot learn to select new actions, even if some of the new actions have high expected rewards. This is because we do not observe any data about new actions in $\mathcal{D}$, which severely violates *full support* (Condition 2.1). Therefore, we need to first relax the typical condition of full support to effectively learn policies that can choose promising new actions.

# 3 The Proposed Approach

The previous sections discussed the challenges associated with selecting new actions using existing OPL methods. To overcome, we propose a solution that leverages action features, which are often available in real-world decision-making problems [6, 18, 24].

## 3.1 Formulation of OPL with Action Features

We now represent action $a$ by the $d$-dimensional action features $f(a) = (f_1(a), \ldots, f_l(a), \ldots, f_d(a))$, where $f_l(a) \in \mathcal{F}_l$ denotes the $l$-th dimension. In particular, we consider the setting where each action feature $f_l(a)$ is discrete and can be represented as a one-hot vector. For example, the genre feature such as 'documentary', 'romance', and 'horror' are encoded as $[1, 0, 0]$, $[0, 1, 0]$, and $[0, 0, 1]$.

In OPL with action features, we can possibly address the presence of new actions by being able to represent them as combinations of already observed factors. Figure 1 illustrates an example in the context of thumbnail selection with the 3-dimensional action features. The left side of the figure shows the existing action set, which includes thumbnails already considered by the logging policy. The right side introduces new thumbnails as new actions. For example, **new action 1** consists of "male and female characters," "title positioned at the bottom," and "large title size." We can see that we have already observed every individual feature of **new action 1** separately in the existing action set. **Despite this, existing OPL methods struggle to handle these new actions because the logged data does not include specific combinations of action features representing the new actions**. Therefore, we propose a novel method and algorithm that leverage these action features through a new estimator for the PG to identify effective new actions.

Before introducing our proposed approach, we rigorously introduce three key types of action spaces.

(1) **The space of *all* actions**: The set of all possible combinations of action features:
$$\mathcal{A} := \{a \mid a = (f_1, \ldots, f_d),\ \forall f_1 \in \mathcal{F}_1, \ldots, \forall f_d \in \mathcal{F}_d\}.$$
(2) **The space of *existing* actions**: The set of actions that have some positive probability under the logging policy:
$$\mathcal{A}_{existing} := \{a \in \mathcal{A} \mid \exists x \in \mathcal{X},\ \pi_0(a|x) > 0\}.$$
(3) **The space of *new* actions**: The set of actions that are not at all chosen under the logging policy:
$$\mathcal{A}_{new} := \{a \in \mathcal{A} \mid \forall x \in \mathcal{X},\ \pi_0(a|x) = 0\}.$$

These action spaces satisfy the following relations:
$$\mathcal{A}_{existing} \cap \mathcal{A}_{new} = \emptyset, \quad \mathcal{A}_{existing} \cup \mathcal{A}_{new} = \mathcal{A}.$$

In the typical formulation for OPL, the objective is to identify actions that maximize the expected reward from a set of *existing* actions $\mathcal{A}_{existing}$. In contrast, we consider a more general and realistic scenario, where the goal is to select actions that maximize the expected reward from $\mathcal{A}$, which includes new actions $\mathcal{A}_{new}$.

## 3.2 The Proposed Algorithm: PONA

To design a new algorithm for OPL with new actions, we first draw inspiration from the PI estimator initially proposed for OPE in slate bandits [35], and provide a generalization.

To introduce PI, we begin with formally defining an one-hot vector for each action feature as $\mathbb{I}_{f_l(a)} \in \mathbb{R}^{dm}$. This is a flattened vector that represents the one-hot encoding of each action feature. Using this notation, we can first extend the PI estimator [35] as an estimator for the PG as follows.

$$\nabla_\theta \widehat{V}_{\mathrm{PI}}(\pi_\theta; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{a \in \mathcal{A}} \pi_\theta(a|x_i) \nabla_\theta \log \pi_\theta(a|x_i) \mathbb{I}_{f_l(a)}{}^T \right) \Gamma^\dagger_{\pi_0, x_i} \mathbb{I}_{f_l(a_i)} r_i, \quad (4)$$

where $\Gamma_{\pi_0,x} := \mathbb{E}_{\pi_0(a|x)}[\mathbb{I}_{f_l(a)} \mathbb{I}^T_{f_l(a)} | x]$, and $M^\dagger$ denotes the Moore-Penrose pseudoinverse of the matrix $M$.

The PI estimator introduced above relies on the following conditions (3.1 and 3.2) to ensure unbiased estimation of the PG:

**Condition 3.1.** *(Independent Support) The logging policy $\pi_0$ ensures independent support if $\pi_0(f_l|x) > 0, \forall x \in \mathcal{X}, \forall l \in [1, ..., d], \forall f_l \in \mathcal{F}_l$. Note that $\pi_0(f_l|x) = \sum_{a \in \mathcal{A}: f_l(a) = f_l} \pi_0(a \mid x)$ is the marginal probability of observing $f_l$ under $\pi_0$.*

This condition requires the logging policy to independently support every dimension of the action features. The condition of Independent Support is weaker than the Full Support condition (Condition 2.1), which requires the logging policy to choose every combination of action features. When new actions exist, the Full Support condition cannot hold, but the Independent Support condition may still hold even in the presence of new actions. **Due to this relaxation of the support condition, the PI estimator can learn a new policy that chooses new actions**.

The next condition of PI is with regard to the q-function.

**Condition 3.2.** *(Linearity) For each context $x \in \mathcal{X}$, there exists an (unknown) intrinsic reward vector $\phi_{x,l} \in \mathbb{R}^{dm}$ such that:*

$$q(x, a) = \sum_{l=1}^{d} q_l(x, f_l(a)) = \mathbb{I}^T_{f_l(a)} \phi_{x,l}.$$

The linearity condition requires that the q-function be expressed as a linear combination of latent value functions for each dimension of the feature. **This condition essentially assumes no interaction effects between different dimensions of the action feature**. While extending the PI estimator to estimate the PG is straightforward as done in Eq. (4), the linearity condition rarely holds and introduces significant bias in PG estimation [12, 37].

To overcome this critical limitation of the naive extension of PI, we first generalize Independent Support to account for local interactions of action features as follows.

**Condition 3.3.** *(Local Combination Support) The logging policy $\pi_0$ is said to ensure local combination support if, for the the first $s$ dimensions of the action features, $\pi_0(f_{1:s}|x) > 0, \forall x \in \mathcal{X}, \forall f_{1:s} \in \prod_{j=1}^{s} \mathcal{F}_j$ as well as $\pi_0(f_l|x) > 0$ for the rest $s+1 \leq l \leq d$.*

Note that $f_{1:s} = (f_1, ..., f_s)$ represents the first $s$ dimensions of the action features, where $1 \leq s \leq d$. The condition of Local Combination Support requires $f_{1:s}$ to be fully supported under the logging policy.[1] Condition 3.3 remains weaker than the original Full Support condition (Condition 2.1), because Full Support requires every dimension of the action features to be simultaneously supported (equivalent to Local Combination Support with $s = d$). When $s = 1$, Local Combination Support reduces to Condition 3.1. Along with this extension, we generalize the linearity condition for the q-function (Condition 3.2) as described below.

**Condition 3.4.** *(Local Linearity) Define a vector with binary values for representing the first $s$ dimensions of the action features as $\mathbb{I}_{f_{1:s}} \in \mathbb{R}^{m^s}$. The overall action indicator $\mathbb{I}_a$ is then defined as $\mathbb{I}_a := concat[\mathbb{I}_{f_l}, \mathbb{I}_{f_{1:s}}]$. We say that the q-function meets Local Linearity, if for each context $x \in \mathcal{X}$, there exists an (unknown) intrinsic reward vector $\phi_x = concat[\phi_{x,l} \in \mathbb{R}^{dm}, \phi_{x,1:s} \in \mathbb{R}^{m^s}]$ such that:*

$$q(x,a) = \underbrace{\sum_{l=1}^{d} q_l(x, f_l(a)) + q(x, f_{1:s}(a))}_{\mathbb{I}_{f_l(a)}^T \phi_{x,l} \quad + \quad \mathbb{I}_{f_{1:s}(a)}^T \phi_{x,1:s}} = \mathbb{I}_a^T \phi_x.$$

**The Local Linearity condition allows the first $s$ dimensions of the action features to have interaction effects with each other**. Therefore, this condition is milder than linearity of PI (Condition 3.2), because linearity does not allow any interaction effects across different action features. When $s = 1$, the condition of local linearity reduces to the linearity condition (Condition 3.2).

Building on these generalized conditions, we propose a new policy gradient estimator, called the Local Combination PseudoInverse (LCPI) estimator, generalizing the PI estimator [35] as below.

$$\nabla_\theta \widehat{V}_{\mathrm{LCPI}}(\pi_\theta; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{a \in \mathcal{A}} \pi_\theta(a|x_i) \nabla_\theta \log \pi_\theta(a|x_i) \mathbb{I}_a{}^T \right) \Gamma_{\pi_0, x_i}^{\dagger} \mathbb{I}_{a_i} r_i, \quad (5)$$

where $\Gamma_{\pi_0,x} := \mathbb{E}_{\pi_0(a|x)}[\mathbb{I}_a \mathbb{I}_a^T | x]$.

LCPI satisfies unbiasedness against the PG $\nabla_\theta V(\pi_\theta)$ under Conditions 3.3 and 3.4, i.e., $\mathbb{E}_{\mathcal{D}}[\nabla_\theta \widehat{V}_{\mathrm{LCPI}}(\pi_\theta; \mathcal{D})] = \nabla_\theta V(\pi_\theta)$. This indicates that LCPI can estimate the reward of new actions while dealing with the interaction effects among action features (the original PI estimator cannot deal with such interaction effects).

Although LCPI is effective for selecting new actions than PI by dealing with the interaction effects, traditional policy- or regression-based methods may achieve better performance within existing actions. This is because they do not rely on any condition by focusing only on existing actions. Therefore, there is an interesting tradeoff between our LCPI and traditional methods. While traditional methods focus on identifying the optimal one within existing actions, LCPI is superior in choosing new actions leverafing action features and local linearity. This interesting tradeoff leads us to finally develop the **Policy Optimization for New Actions (PONA)** algorithm, which integrates LCPI with the DR estimator as follows:

$$\nabla_\theta \widehat{V}_{\mathrm{PONA}}(\pi_\theta; \kappa, \mathcal{D}) = \kappa \cdot \nabla_\theta \hat{V}_{\mathrm{LCPI}}(\pi_\theta; \mathcal{D}) + (1-\kappa) \cdot \nabla_\theta \hat{V}_{\mathrm{DR}}(\pi_\theta; \mathcal{D}), \quad (6)$$

where $\kappa \in [0,1]$ is a hyperparameter that controls the trade-off between focusing on new actions and maximizing the overall policy value by prioritizing existing actions. A larger value of $\kappa$ increases the weight of the LCPI component, which leads to greater reliance on the local linearity condition and a stronger focus on optimizing new actions. However, this approach sacrifices the effectiveness of identifying the optimal actions within the set of existing actions. In contrast, a smaller value of $\kappa$ increases the weight of DR, reducing reliance on local linearity and shifting the focus toward optimizing existing actions. This results in a more conservative policy that rarely selects new actions. We can tune the value of $\kappa$ in a data-driven manner using a cross-validation procedure. Multiple formulations of cross-validation can be considered for tuning $\kappa$, depending on the specific objectives. For instance, one can simply maximize the policy value $V(\pi)$ on the validation set when tuning $\kappa$. Alternatively, it is possible to impose constraints on the proportions of new actions under the learned policy, allowing control over how aggressive or conservative the new policy should be, as follows.

$$\max_{\kappa} \hat{V}(\pi_{\theta,\kappa}; \mathcal{D}) \quad \text{s.t. } \rho_L \leq \mathbb{E}_{p(x)}[\sum_{a \in \mathcal{A}_{new}} \pi_{\theta,\kappa}(a|x)] \leq \rho_U, \quad (7)$$

where $\rho_L$ and $\rho_U$ define the desired range for the proportion of new actions. In the next section, we will empirically demonstrate that, by performing this cross-validation procedure and changing the values of $\rho_L$ and $\rho_U$, we can flexibly control the proportion between existing and new actions under the learned policy. Note that, when performing cross-validation, we can estimate the policy value $\hat{V}(\pi_{\theta,\kappa})$ of the policy induced by $\kappa$ by applying off-the-shelf OPE estimators such as IPS and DR in the validation set [25].

## 4 Empirical Evaluation

This section empirically evaluates our proposed method, PONA, and its special case (LCPI) on synthetic and real-world datasets (**empirical results on real data can be found in Appendix D.2**).

We begin by sampling context vectors $x$ from a normal distribution. The action features span $d = 5$ dimensions, with each feature having $m = 3$ possible values. The total number of possible actions is thus $|\mathcal{A}| = 3^5 = 243$. We then define the synthetic q-function as

$$q(x,a) = \sum_{l=1}^{d} \underbrace{q_l(x, f_l(a))}_{:= x^T M_{f_l(a)}} + \underbrace{q_{1:s}(x, f_{1:s}(a))}_{:= x^T M_{f_{1:s}(a)}} + \gamma \cdot \underbrace{q_{1:d}(x, f_{1:d}(a))}_{:= x^T M_{f_{1:d}(a)}},$$

where $\gamma$ is a parameter that controls the contribution of the last term $q_{1:d}(x, f_{1:d}(a))$. $M_{f_l}$, $M_{f_{1:s}}$, and $M_{f_{1:d}}$ are parameter matrices sampled from a uniform distribution. The first term of the q-function, $q_l(x, f_l(a))$, represents the independent effect of each dimension of the action feature. The second term, $q_{1:s}(x, f_{1:s}(a))$, captures the interaction effects within the first $s$ dimensions of the action feature. Most methods, including IPS, DR, and ours, can handle the first two terms because they do not involve interaction effects beyond

[1] Note that we focus on the first $s$ dimensions just for ease of exposition, as the dimensions of the action feature can be arbitrarily reordered.

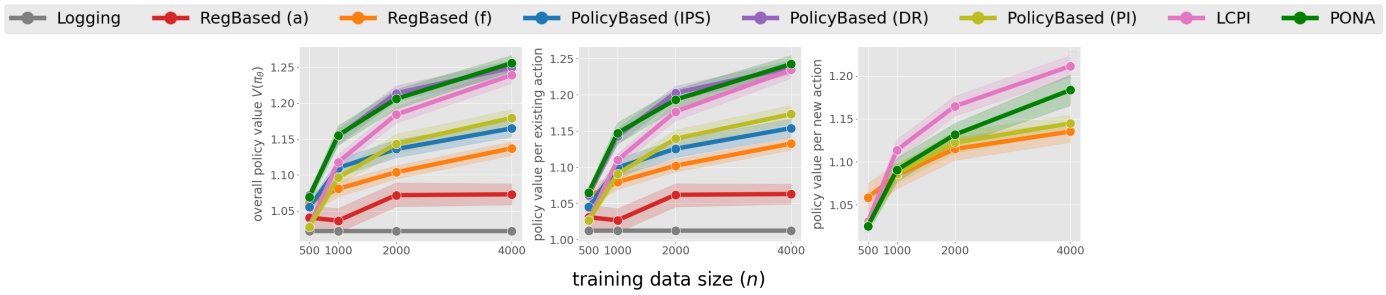


**Figure 2: Comparisons of the overall policy value, policy value per existing action, and policy value per new action with varying training data sizes ($n$). Note that the metrics are normalized by those of the uniform random policy.**

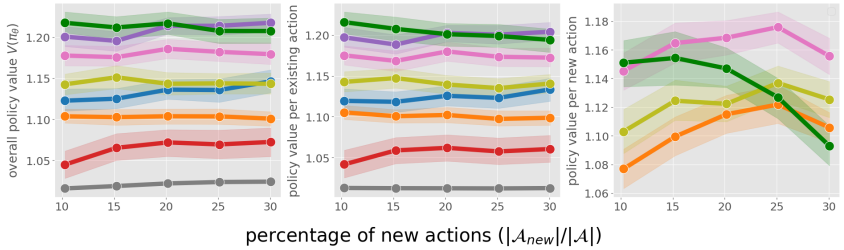


**Figure 3: Comparisons of the overall policy value, policy value per existing action, and policy value per new action with varying percentages of new actions ($|\mathcal{A}_{new}|/|\mathcal{A}|$). Note that the metrics are normalized by those of the uniform random policy.**

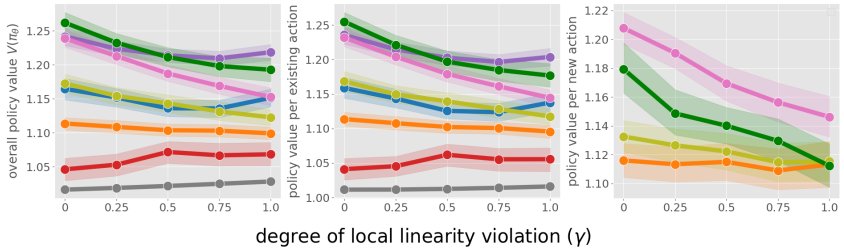


**Figure 4: Comparisons of the overall policy value, policy value per existing action, and policy value per new action with varying degrees of local linearity violation ($\gamma$). Note that the metrics are normalized by those of the uniform random policy.**

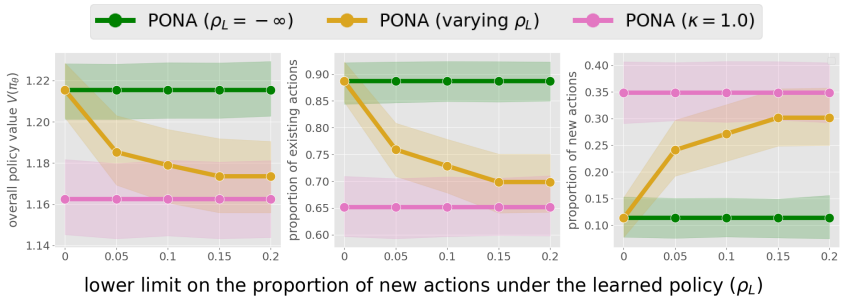


**Figure 5: Comparisons of the overall policy value, proportion of existing actions, and proportion of new actions under the learned policy with varying lower limits on the proportion of new actions ($\rho_L$ in Eq. (7)).**

the first $s$ dimensions. Only the original PI method [35] cannot estimate the second term due to its reliance on the linearity condition. The last term of the q-function, $q_{1:d}(x, f_{1:d}(a))$, accounts for the interaction effects among all dimensions of the action feature. The proposed methods cannot estimate the last term without bias due to their reliance on local linearity, while existing methods can achieve unbiased estimation of the q-function within existing actions regardless of the interaction effects. By adjusting the $\gamma$ parameter, we control the degree of the condition's violation and evaluate the robustness of our methods under these violations.

***Compared Methods.*** We compare PONA and LCPI with the logging policy $\pi_0$, the regression-based method using action index (RegBased ($a$)), the regression-based method using action feature (RegBased ($f$)), policy-based method w/ IPS (PolicyBased (IPS)), policy-based method w/ DR (PolicyBased (DR)), and policy-based

method w/ PI (PolicyBased (PI)). We report RegBased ($f$) as a baseline method that estimates the q-function using action features as input. It then picks the best action according to the estimated reward function $\hat{q}_\theta(x, f_{1:d}(a))$. RegBased ($f$) differs from RegBased ($a$) in that it may select new actions. To tune the hyperparameter $\kappa$ for PONA, we perform a grid search over the range $[0, 0.25, 0.5, 0.75, 1.0]$, selecting the best value in terms of optimizing the policy value $V(\pi)$ on validation data.

***Result.*** We report the following metrics of policies learned by each OPL method, averaged across 200 simulations.

- **overall policy value**: $\mathbb{E}[\sum_{a\in\mathcal{A}} \pi_\theta(a|x)q(x,a)]$
- **policy value per existing action**: $\frac{\mathbb{E}[\sum_{a\in\mathcal{A}_{existing}} \pi_\theta(a|x)q(x,a)]}{\mathbb{E}[\sum_{a\in\mathcal{A}_{existing}} \pi_\theta(a|x)]}$
- **policy value per new action**: $\frac{\mathbb{E}_{p(x)}[\sum_{a\in\mathcal{A}_{new}} \pi_\theta(a|x)q(x,a)]}{\mathbb{E}[\sum_{a\in\mathcal{A}_{new}} \pi_\theta(a|x)]}$

**Overall policy value** measures the overall effectiveness of the learned policy. **Policy value per existing action** evaluates the quality of selecting existing actions, while **policy value per new action** assesses that of selection within new actions. **Note that the logging policy $\pi_0$, RegBased ($a$), PolicyBased (IPS), and PolicyBased (DR) do not choose new actions at all, and thus their results do not appear in the figures of policy value per new action.**

***How does PONA perform with varying sizes of training data?*** First, we compare the OPL methods using varying sizes of training logged data ($n \in \{500, 1000, 2000, 4000\}$), as shown in Figure 2. The figure shows that most methods, including PONA and LCPI, improve in overall policy value, policy value per existing action, and policy value per new action as the training data size increases, which aligns with expectations. Notably, LCPI performs particularly well in terms of the selection of new actions when the training data size is relatively large ($n = 2000, 4000$). However, it underperforms PolicyBased (DR) in terms of overall policy value. In contrast, PONA combines the strengths of LCPI and PolicyBased (DR), maintaining a policy value per new action comparable to RegBased ($f$) while matching PolicyBased (DR) in terms of overall policy value. Considering that existing methods, including PolicyBased (DR), cannot select new actions at all, it is noteworthy that **PONA not only selects new actions and gives fair opportunities to them as effectively as RegBased ($f$) but also achieves overall performance competitive with PolicyBased (DR).**

***How does PONA perform with varying numbers of new actions?*** Next, Figure 3 illustrates how the OPL algorithms perform as the percentage of new actions, $100 \cdot \frac{|\mathcal{A}_{new}|}{|\mathcal{A}|}$, increases. As the percentage of new actions grows, the number of existing actions decreases. The figure shows that methods focusing exclusively on existing actions (RegBased ($a$), PolicyBased (IPS), PolicyBased (DR)) exhibit a slight improvement in policy value for existing actions as the percentage of new actions increases. This is because, with fewer existing actions, it becomes easier to identify the optimal action within the reduced set. Nevertheless, PONA consistently achieves the same or slightly better overall policy value compared to PolicyBased (DR), while also selecting new actions for any percentage of new actions. In comparison, PolicyBased (DR) and other traditional methods never select new actions, even as the percentage of new actions increases. PolicyBased (PI) is mostly worse than PONA and LCPI in every metric due to its reliance on linearity, which is severely violated in our experimental environment.

***How does PONA perform when Condition 3.4 is violated?*** We next evaluate the robustness of the proposed methods against violations of their key condition, namely, local linearity. To test this, we increase the value of $\gamma$ in the definition of the synthetic reward function in the x-axis of Figure 4. The figure demonstrates that as $\gamma$ increases, the violations of the local linearity condition become more pronounced, leading to a gradual decrease in the overall policy value of LCPI. In contrast, existing methods that do not rely on any assumptions about the q-function exhibit little change in policy value with varying values of $\gamma$, which is reasonable. Most interestingly, PONA, which also relies on local linearity, is more robust to the condition's violation than LCPI. This occurs because its data-driven tuning process in Eq. (7) adjusts its hyperparameter $\kappa$ to assign greater weight to the DR component as the violation becomes more severe. This adjustment adaptively maintains the policy value for existing actions, remaining a high overall policy value. Even in scenarios where the ability to learn new actions is at its weakest, PONA performs comparably to RegBased ($f$) for new actions, significantly outperforming PolicyBased (DR) and other baseline methods in selecting new actions. These results highlight PONA's ability to maintain a high policy value and effectively learn new actions, even when its key condition is severely violated

***How does PONA perform when varying the constraints about the proportions of new actions during parameter tuning?*** In the previous experiments, we did not impose any constraints on the behavior of the learned policy when tuning the weight parameter $\kappa$ of PONA (i.e., always setting $\rho_L = -\infty$ and $\rho_U = \infty$ in Eq. (7)). As a result, the tuning procedure selected the $\kappa$ value that simply maximized the overall policy value in the validation data.

In Figure 5, we report the overall policy value and the proportions of existing and new actions under the learned policies for varying lower limit $\rho_L$ in Eq. (7), while keeping $\rho_U = \infty$. Note that we include the metrics for PONA ($\rho_L = -\infty$) and PONA ($\kappa = 1.0$) as references in the figure. This experiment allows us to investigate how well PONA's behavior can be controlled through the tuning of $\kappa$. The right plot in Figure 5 demonstrates that, as the lower limit on the proportion of new actions becomes larger, the proportion of new actions under **PONA (with varying $\rho_L$)** increases. This result shows that we can control the behavior and aggressiveness of the learned policy by appropriately formulating the tuning process of the weight parameter $\kappa$. It is also reasonable that the overall policy value of PONA decreases with larger $\rho_L$, as this increases the reliance of PONA on the local linearity condition.

## 5 Conclusion

This work tackled the challenge of OPL with new actions. We first proposed the LCPI estimator for the policy gradient, which relaxes the restrictive conditions of PI by accounting for interactions among action features, enabling the more effective selection of new actions. To balance the strengths of LCPI for new actions and DR for existing actions, we also introduced the PONA algorithm. By integrating these estimators through a weight parameter, PONA efficiently selects new actions while maintaining strong overall performance.

# References


[1] Fernando Amat, Ashok Chandrashekar, Tony Jebara, and Justin Basilico. 2018. Artwork personalization at Netflix. In *Proceedings of the 12th ACM conference on recommender systems*. 487–488.

[2] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising. *Journal of Machine Learning Research* 14, 11 (2013).

[3] Shreyas Chaudhari, David Arbour, Georgios Theocharous, and Nikos Vlassis. 2024. Distributional Off-Policy Evaluation for Slate Recommendations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8265–8273.

[4] Miroslav Dudík, Dumitru Erhan, John Langford, and Lihong Li. 2014. Doubly Robust Policy Evaluation and Optimization. *Statist. Sci.* 29, 4 (2014), 485–511.

[5] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. 2018. More Robust Doubly Robust Off-Policy Evaluation. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. PMLR, 1447–1456.

[6] Nicolo Felicioni, Maurizio Ferrari Dacrema, Marcello Restelli, and Paolo Cremonesi. 2022. Off-Policy Evaluation with Deficient Support Using Side Information. *Advances in Neural Information Processing Systems* 35 (2022).

[7] Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. 2022. KuaiRec: A fully-observed dataset and insights for evaluating recommender systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 540–550.

[8] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. 2018. Offline A/B Testing for Recommender Systems. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 198–206.

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

[10] Nathan Kallus, Yuta Saito, and Masatoshi Uehara. 2021. Optimal Off-Policy Evaluation from Multiple Logging Policies. In *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139. PMLR, 5247–5256.

[11] Nathan Kallus and Angela Zhou. 2018. Policy evaluation and optimization with continuous treatments. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1243–1251.

[12] Haruka Kiyohara, Masahiro Nomura, and Yuta Saito. 2024. Off-policy evaluation of slate bandit policies via optimizing abstraction. In *Proceedings of the ACM on Web Conference 2024*. 3150–3161.

[13] Haruka Kiyohara, Yuta Saito, Tatsuya Matsuhiro, Yusuke Narita, Nobuyuki Shimizu, and Yasuo Yamamoto. 2022. 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*.

[14] Haruka Kiyohara, Masatoshi Uehara, Yusuke Narita, Nobuyuki Shimizu, Yasuo Yamamoto, and Yuta Saito. 2023. Off-Policy Evaluation of Ranking Policies under Diverse User Behavior. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1154–1163.

[15] Lihong Li, Shunbao Chen, Jim Kleban, and Ankur Gupta. 2015. Counterfactual estimation and optimization of click metrics in search engines: A case study. In *Proceedings of the 24th International Conference on World Wide Web*. 929–934.

[16] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. 2018. Breaking the curse of horizon: infinite-horizon off-policy estimation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 5361–5371.

[17] Alberto Maria Metelli, Alessio Russo, and Marcello Restelli. 2021. Subgaussian and Differentiable Importance Sampling for Off-Policy Evaluation and Learning. *Advances in Neural Information Processing Systems* 34 (2021).

[18] Jie Peng, Hao Zou, Jiashuo Liu, Shaoming Li, Yibao Jiang, Jian Pei, and Peng Cui. 2023. Offline Policy Evaluation in Large Action Spaces via Outcome-Oriented Action Grouping. In *Proceedings of the ACM Web Conference 2023*. 1220–1230.

[19] Doina Precup, Richard S. Sutton, and Satinder P. Singh. 2000. Eligibility Traces for Off-Policy Policy Evaluation. In *Proceedings of the 17th International Conference on Machine Learning*. 759–766.

[20] Noveen Sachdeva, Yi Su, and Thorsten Joachims. 2020. Off-Policy Bandits with Deficient Support. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 965–975.

[21] Yuta Saito, Himan Abdollahpouri, Jesse Anderton, Ben Carterette, and Mounia Lalmas. 2024. Long-term Off-Policy Evaluation and Learning. In *Proceedings of the ACM on Web Conference 2024*. 3432–3443.

[22] Yuta Saito, Shunsuke Aihara, Megumi Matsutani, and Yusuke Narita. 2021. Open Bandit Dataset and Pipeline: Towards Realistic and Reproducible Off-Policy Evaluation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[23] Yuta Saito and Thorsten Joachims. 2021. Counterfactual Learning and Evaluation for Recommender Systems: Foundations, Implementations, and Recent Advances. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 828–830.

[24] Yuta Saito and Thorsten Joachims. 2022. Off-Policy Evaluation for Large Action Spaces via Embeddings. In *International Conference on Machine Learning*. PMLR, 19089–19122.

[25] Yuta Saito and Masahiro Nomura. 2024. Hyperparameter Optimization Can Even be Harmful in Off-Policy Learning and How to Deal with It. *arXiv preprint arXiv:2404.15084* (2024).

[26] Yuta Saito, Qingyang Ren, and Thorsten Joachims. 2023. Off-Policy Evaluation for Large Action Spaces via Conjunct Effect Modeling. *arXiv preprint arXiv:2305.08062* (2023).

[27] Yuta Saito, Takuma Udagawa, Haruka Kiyohara, Kazuki Mogi, Yusuke Narita, and Kei Tateno. 2021. Evaluating the Robustness of Off-Policy Evaluation. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 114–123.

[28] Yuta Saito, Jihan Yao, and Thorsten Joachims. 2024. POTEC: Off-Policy Learning for Large Action Spaces via Two-Stage Policy Decomposition. *arXiv preprint arXiv:2402.06151* (2024).

[29] Tatsuhiro Shimizu, Koichi Tanaka, Ren Kishimoto, Haruka Kiyohara, Masahiro Nomura, and Yuta Saito. 2024. Effective Off-Policy Evaluation and Learning in Contextual Combinatorial Bandits. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 733–741.

[30] Yi Su, Maria Dimakopoulou, Akshay Krishnamurthy, and Miroslav Dudík. 2020. Doubly Robust Off-Policy Evaluation with Shrinkage. In *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119. PMLR, 9167–9176.

[31] Yi Su, Lequn Wang, Michele Santacatterina, and Thorsten Joachims. 2019. Cab: Continuous adaptive blending for policy evaluation and learning. In *International Conference on Machine Learning*, Vol. 84. 6005–6014.

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

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

[34] Adith Swaminathan and Thorsten Joachims. 2015. The Self-Normalized Estimator for Counterfactual Learning. *Advances in Neural Information Processing Systems* 28 (2015).

[35] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miro Dudik, John Langford, Damien Jose, and Imed Zitouni. 2017. Off-Policy Evaluation for Slate Recommendation. In *Advances in Neural Information Processing Systems*, Vol. 30. 3632–3642.

[36] Masatoshi Uehara, Chengchun Shi, and Nathan Kallus. 2022. A review of off-policy evaluation in reinforcement learning. *arXiv preprint arXiv:2212.06355* (2022).

[37] Nikos Vlassis, Ashok Chandrashekar, Fernando Amat Gil, and Nathan Kallus. 2021. Control Variates for Slate Off-Policy Evaluation. *Advances in Neural Information Processing Systems* 34 (2021).

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

# A Related Work

This section discuss the related work.

***Off-Policy Evaluation and Learning.*** Off-Policy Evaluation (OPE) and Learning (OPL) [4, 5, 10, 16, 17, 24, 26, 30, 31, 38] aim to evaluate and optimize decision-making policies using only historical logged data, without the need for direct testing in an online environment. Various methods and estimators have been developed in OPE to control the bias and variance of the estimation of the performance of new policies [4, 5, 10, 12–14, 17, 23, 30, 31, 33, 34, 38]. These advancements in OPE methodology have led to highly accurate estimation of the policy value in the standard setting without new actions [22, 27, 36]. In OPL, these estimation techniques enable accurate estimation of the policy gradient and enhance the resulting effectiveness of policy learning from only logged data [28]. However, most existing methods in OPL rely crucially on the full support condition (Condition 2.1) and focus solely on selecting the most effective actions within those already present in the logged data [20]. Therefore, they cannot deal with new actions that are potentially available in many decision-making problems and severely violate full support. In this paper, we propose a novel formulation and algorithm that leverage action features to address OPL with new actions. In particular, our algorithm can identify and choose effective new actions without sacrificing the overall policy performance, while existing methods cannot handle new actions even with the availability of the action features.

***OPE for Slate Bandits.*** In this study, we address the challenge of incorporating new actions by considering OPL with action features. Specifically, we tackle realistic scenarios where each action is characterized by multi-dimensional action features, and a single reward is observed for each action. By mapping actions to slates and features of an action to slots of a slate, we can establish a connection between our problem with action features and OPE for slate bandits, initially explored by Swaminathan et al. [35].

Slate bandits consider selecting a combinatorial action called *slate* composed of multiple sub-actions. For example, in medical treatment scenarios, one may aim to select the optimal combination of drug doses to improve patient outcomes. While these systems often have access to extensive logged data, an accurate OPE remains challenging due to the exponential increase in variance associated with slate-wise importance weighting [12, 30, 35]. To address this challenge, several versions of *PseudoInverse* (PI) estimators have been proposed [30, 35, 37]. They use slot-wise importance weights to relax the support condition and reduce the variance of typical estimators such as IPS. They have also been proven to enable unbiased OPE under the *linearity* assumption on the q-function. Linearity requires that the q-function be linearly decomposable, ignoring interaction effects among different slots. While PI often outperforms IPS under linear reward structures, it introduces significant bias when the linearity assumption does not hold [12, 29].

To effectively handle new actions in OPL by leveraging action features, we extend the idea of PI to estimate the policy gradient by relaxing the linearity assumption on the reward. This relaxation is crucial for dealing with the real-world non-linearities in reward functions during policy learning. It should be noted that the key contributions of our work are substantially different from those of existing studies for slate bandits [3, 12, 35, 37]. We focus on the problem of policy *learning with new actions*. In contrast, the existing literature around slate OPE addresses only the problem of policy *evaluation without considering new actions*.

***OPL with Deficient Actions.*** Another setting similar to new actions is OPL with deficient actions [6, 20], which refer to the actions that have zero probability of being selected under the logging policy for *certain* contexts $x$. Rigorously, the set of deficient actions is written as $\{a \in \mathcal{A} \mid \exists x \in \mathcal{X}, \pi_0(a \mid x) = 0\}$. Deficient action is a broader concept that encompasses the new actions we consider as a special case. Specifically, a deficient action $a$ may still be observed for some contexts $x$ under the logging policy $\pi_0$, whereas new actions are never observed for any context $x$. This distinction becomes significant when estimating the q-function, $\hat{q}(x, a)$. The existence of deficient actions still allows for learning the q-function $q(x, a)$ because we observe every action somewhere in the logged data. However, with the presence of new actions, it becomes extremely challenging to estimate the q-function because we do not observe even a single data point for such actions. Due to this distinction, existing OPL approaches to deal with deficient actions [6, 20] cannot handle new actions. This motivates us to develop new methods specifically dealing with new actions without sacrificing the overall quality of policy learning.

# B Omitted Proofs

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

# C Proof of Unbiasedness of LCPI

We provide a detailed proof that LCPI is unbiased under Conditions 3.4 and 3.3, mirroring the structure of Proposition 1 in [35] while incorporating our own notation. We begin with a key lemma and then use it to establish the final result. To prove this, let us define the following action sets:

$$\mathcal{A}_{\pi_0}(x) := \{a \in \mathcal{A} \mid \pi_0(a|x) > 0\},$$
$$\mathcal{A}_{\mathrm{LCS}}(x) := \{a \in \mathcal{A} \mid \pi_0(f_{1:s}(a)|x) > 0, \pi_0(f_l(a)|x) > 0, \forall l \in [1, \ldots, d]\}.$$

Lemma C.1. *If Condition 3.4 holds and $a \in \mathcal{A}_{LCS}(x)$, then $q(x, a) = \mathbb{I}_a^T \Gamma_{\pi_0,x}^{\dagger} \theta_{\pi_0,x}$, where $\theta_{\pi_0,x} := \mathbb{E}_{\pi_0(a|x)}[\mathbb{I}_a q(x, a)]$.*

Proof. By Condition 3.4, there is a fixed vector $\phi_x$ such that for each action $a$,

$$q(x, a) = \mathbb{I}_a^T \phi_x.$$

Define a binary matrix $M \in \{0,1\}^{|\mathcal{A}_{\mathrm{LCS}}(x)| \times (dm+m^s)}$, whose rows are the row vectors $\mathbb{I}_a^T$ for each $a \in \mathcal{A}_{\mathrm{LCS}}(x)$. The vector $M\phi_x$ enumerates $q(x,a)$ for all $a \in \mathcal{A}_{\mathrm{LCS}}(x)$. Let $\mathrm{Null}(M)$ denote the null space of $M$, and let $\Pi$ denote the orthogonal projection onto $\mathrm{Null}(M)$. Define $\phi_x^\star := (I - \Pi)\phi_x$. Since $M\phi_x = M\phi_x^\star$, it follows that

$$q(x,a) = \mathbb{I}_a^T \phi_x = \mathbb{I}_a^T \phi_x^\star \quad \text{for all } a \in \mathcal{A}_{\mathrm{LCS}}(x).$$

From the definition in Section 3.2, we have

$$\theta_{\pi_0,x} = \mathbb{E}_{\pi_0(a|x)}\left[\mathbb{I}_a q(x,a)\right] = \mathbb{E}_{\pi_0(a|x)}\left[\mathbb{I}_a \mathbb{I}_a^T \phi_x\right] = \Gamma_{\pi_0,x}\phi_x. \tag{8}$$

Observe that

$$\begin{aligned}
\mathrm{Null}(\Gamma_{\pi_0,x}) &= \left\{v \mid v^T \Gamma_{\pi_0,x} v = 0\right\} \\
&= \left\{v \mid \mathbb{E}_{\pi_0(a|x)}\left[v^T \mathbb{I}_a \mathbb{I}_a^T v\right] = 0\right\} \\
&= \left\{v \mid \mathbb{I}_a^T v = 0, \forall a \in \mathcal{A}_{\pi_0}(x)\right\} \\
&= \left\{v \mid \mathbb{I}_a^T v = 0, \forall a \in \mathcal{A}_{\mathrm{LCS}}(x)\right\} \\
&= \mathrm{Null}(M),
\end{aligned}$$

where the first step follows from the positive semi-definiteness of $\Gamma_{\pi_0,x}$. Since $\mathrm{Null}(\Gamma_{\pi_0,x}) = \mathrm{Null}(M)$, it follows from Eq. (8) that

$$\theta_{\pi_0,x} = \Gamma_{\pi_0,x}\phi_x^\star.$$

Moreover, from the definition of $\phi_x^\star$, we know that $\phi_x^\star \perp \mathrm{Null}(\Gamma_{\pi_0,x})$. Thus, by the definition of the pseudoinverse,

$$\Gamma_{\pi_0,x}^\dagger \theta_{\pi_0,x} = \phi_x^\star.$$

Putting this together, for $a \in \mathcal{A}_{\mathrm{LCS}}(x)$:

$$q(x,a) = \mathbb{I}_a^T \phi_x^\star = \mathbb{I}_a^T \Gamma_{\pi_0,x}^\dagger \theta_{\pi_0,x}.$$

This completes the proof of Lemma C.1. □

We now use the above Lemma to prove the unbiasedness of LCPI.

Proof.

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}\left[\nabla_\theta \widehat{V}_{\mathrm{LCPI}}(\pi_\theta;\mathcal{D})\right] &= \mathbb{E}_{\mathcal{D}}\left[\frac{1}{n}\sum_{i=1}^n \left(\sum_{a\in\mathcal{A}} \pi_\theta(a|x_i)\nabla_\theta \log \pi_\theta(a|x_i)\mathbb{I}_a^T\right)\Gamma_{\pi_0,x_i}^\dagger \mathbb{I}_{a_i} r_i\right] \\
&= \mathbb{E}_{p(x)\pi_0(a'|x)p(r|x,a')}\left[\left(\sum_{a\in\mathcal{A}} \pi_\theta(a|x)\nabla_\theta \log \pi_\theta(a|x)\mathbb{I}_a^T\right)\Gamma_{\pi_0,x}^\dagger \mathbb{I}_{a'} r\right] \\
&= \mathbb{E}_{p(x)}\left[\left(\sum_{a\in\mathcal{A}} \pi_\theta(a|x)\nabla_\theta \log \pi_\theta(a|x)\mathbb{I}_a^T\right)\Gamma_{\pi_0,x}^\dagger \mathbb{E}_{\pi_0(a'|x)}\left[\mathbb{I}_{a'} q(x,a')\right]\right] \\
&= \mathbb{E}_{p(x)}\left[\left(\sum_{a\in\mathcal{A}} \pi_\theta(a|x)\nabla_\theta \log \pi_\theta(a|x)\mathbb{I}_a^T\right)\Gamma_{\pi_0,x}^\dagger \theta_{\pi_0,x}\right] \\
&= \mathbb{E}_{p(x)}\left[\sum_{a\in\mathcal{A}} \pi_\theta(a|x)\nabla_\theta \log \pi_\theta(a|x) q(x,a)\right] \quad \because \text{Condition 3.3, Lemma C.1} \\
&= \nabla_\theta V(\pi_\theta).
\end{aligned}$$

□

## D Experiment Details and Additional Result

This section describes the detailed experiment settings and reports additional results.

## D.1 Synthetic Experiments

***Detailed Setup.*** We describe synthetic experiment settings in detail. First, we define the space of existing actions $\mathcal{A}_{existing}$ as follows:

$$\mathcal{A}_{existing} := \mathcal{A}^{\mathrm{IS}}_{\mathrm{base}} \cup \mathcal{A}^{\mathrm{LCS}}_{\mathrm{base}}(s) \cup \mathcal{A}_{\mathrm{random}}$$

where $\mathcal{A}^{\mathrm{IS}}_{\mathrm{base}}$ and $\mathcal{A}^{\mathrm{LCS}}_{\mathrm{base}}(s)$ are artificially constructed sets of actions that satisfy the Independent Support condition (3.1) and Local Combination Support condition (3.3), respectively, as below:

$$\mathcal{A}^{\mathrm{IS}}_{\mathrm{base}} := \{a = (f_{1,i}, f_{2,i}, \cdots, f_{d,i}) | \forall i \in m\},$$

$$\mathcal{A}^{\mathrm{LCS}}_{\mathrm{base}}(s) := \left\{ a = (f_{1:s}, f_{s+1,1} \cdots, f_{d,1}) | \forall f_{1:s} \in \prod_{l=1}^{s} \mathcal{F}_l \right\}.$$

Note that $f_{j,i}$ represent the $i$-th feature value in the $j$-th dimension, where $j \in [d]$ and $i \in [m]$. This construction of the action space ensures that the logging policy selects each feature at least once, satisfying the Independent Support condition. It also ensures that the logging policy selects all combinations of the first $s$ dimensions, satisfying the Local Combination Support condition. Additionally, $\mathcal{A}_{\mathrm{random}}$ contains a set of actions randomly sampled from $\mathcal{A}$.

***Additional Results.*** We report and discuss additional synthetic experiments results.

***How does PONA perform when varying the dimension of Local Combination Support (s)?*** The action features have a dimension of $d = 5$, so the parameter $s$ can take values from 1 to 5. However, when $s = 5$, we have $\mathcal{A} = \mathcal{A}_{existing}$, meaning that no new actions exist. Therefore, we present results only for $s = 1$ to 4. Varying $s$ introduces a trade-off between the support condition and the reward condition: a stronger Local Combination Support (LCS) relaxes the condition on the reward, whereas weaker support conditions impose stricter constraints on the reward function. Note that when $s = 1$, LCPI is identical to PolicyBased (PI). As shown in Figure 6, both LCPI and PONA successfully learn both new and existing actions across different values of $s$. In other words, these methods perform well under both strict reward assumptions and strict support conditions.

***How does PONA perform when varying the temperature parameter of the logging policy ($\beta$)?*** Next, we compare the OPL methods while varying the temperature parameter of the logging policy ($\beta \in \{-0.2, -0.1, 0, 0.1, 0.2\}$), as shown in Figure 7. As $\beta$ increases, the performance of the logging policy improves, and when $\beta = 0$, the logging policy corresponds to a uniform policy. This result shows that the methods using action index (PolicyBased (IPS), PolicyBased (DR), and RegBased ($a$)) exhibit a decline in the quality of the overall policy value with larger $\beta$. We infer that as $\beta$ increases, the logging policy becomes more likely to select actions with high q-function values, causing the logged data to be skewed and thus increasing both estimation bias and variance. In contrast, RegBased ($f$), PolicyBased (PI) and LCPI remain robust to changes in $\beta$ by incorporating action features. PONA consistently achieves a high overall policy value across all $\beta$. However, because PONA is influenced by PolicyBased (DR) during the $\kappa$ tuning process, its ability to learn new actions diminishes as $\beta$ increases. If the primary goal is to focus on learning new actions, one can tune $\kappa$ by adjusting $\rho_L$ rather than setting $\rho_L = -\infty$. This adjustment can improve performance on new actions even when $\beta$ is large.

***How does PONA perform when varying the constraints on the proportions of new actions during parameter tuning?*** In Figure 8, we present the overall policy value along with the proportions of existing and new actions under the learned policies for different upper limits $\rho_U$ in Eq. (7), while keeping $\rho_L = -\infty$. We also include the metrics for PONA ($\rho_U = \infty$) and PONA ($\kappa = 1$) as references in the figure. This experiment examines how effectively PONA's behavior can be controlled by tuning $\kappa$. The right plot in Figure 8 shows that as the upper limit on the proportion of new actions decreases, the proportion of new actions under PONA (with varying $\rho_U$) also decreases. This result indicates that we can regulate the behavior and conservativeness of the learned policy by properly designing the tuning process of the weight parameter $\kappa$.

***Other metrics.*** We report the following metrics as the additional experiment results,

- **overall policy value**: $\mathbb{E}[\sum_{a \in \mathcal{A}} \pi_\theta(a|x) q(x,a)]$
- **policy value of existing action**: $\mathbb{E}[\sum_{a \in \mathcal{A}_{existing}} \pi_\theta(a|x) q(x,a)]$
- **policy value of new action**: $\mathbb{E}[\sum_{a \in \mathcal{A}_{new}} \pi_\theta(a|x) q(x,a)]$

For all metrics, higher values indicate better performance. Note that the logging policy $\pi_0$, RegBased ($a$), PolicyBased (IPS), and PolicyBased (DR) do not select new actions, so the policy value of new actions is always 0. We present results for varying training data sizes ($n$) in Figure 9, percentages of new actions ($|\mathcal{A}_{new}|/|\mathcal{A}|$) in Figure 10, degrees of local linearity violation ($\gamma$) in Figure 11, the dimension of Local Combination Support ($s$) in Figure 12 and the temperature parameter of the logging policy ($\beta$) in Figure 13.

## D.2 Real-World Data

To evaluate the real-world applicability of PONA, we finally assess its performance on the KuaiRec dataset [7],[2] which are derived from the recommendation logs of a video-sharing mobile app. This dataset includes a matrix with 1,411 users and 3,327 videos, where nearly

[2]https://kuairec.com/

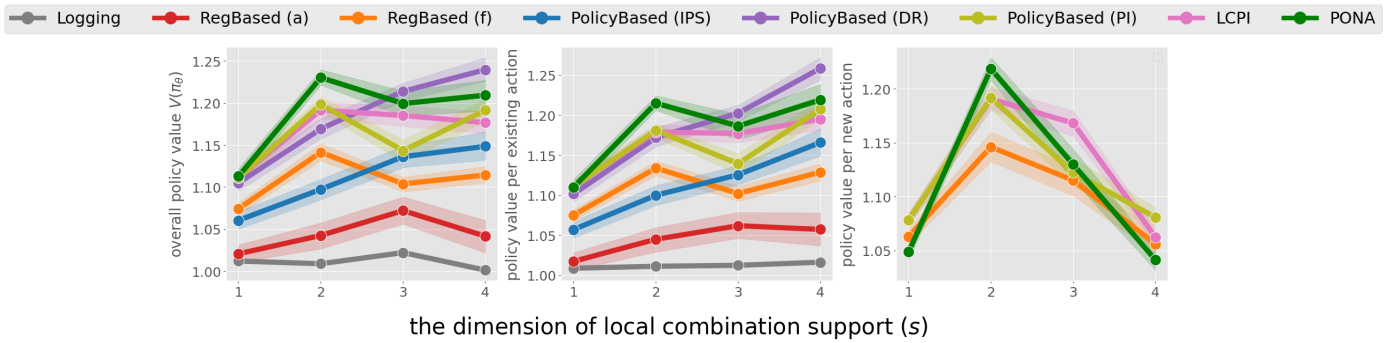


**Figure 6: Comparisons of the overall policy value, policy value per existing action, and policy value per new action with varying the dimension of ocal Combination Support ($s$). Note that the metrics are normalized by those of the uniform random policy.**

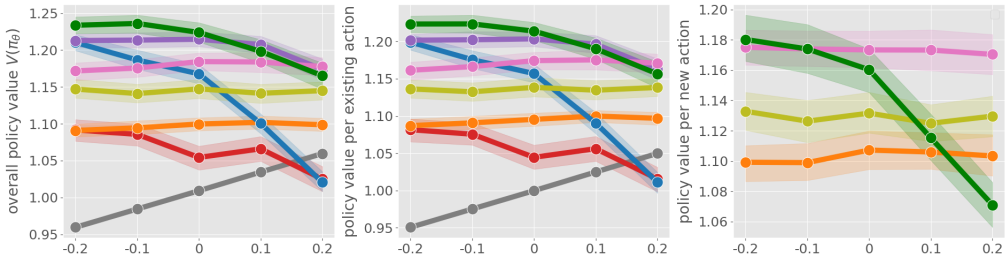


**Figure 7: Comparisons of the overall policy value, policy value per existing action, and policy value per new action with varying temperature parameter of the logging policy ($\beta$). Note that the metrics are normalized by those of the uniform random policy.**

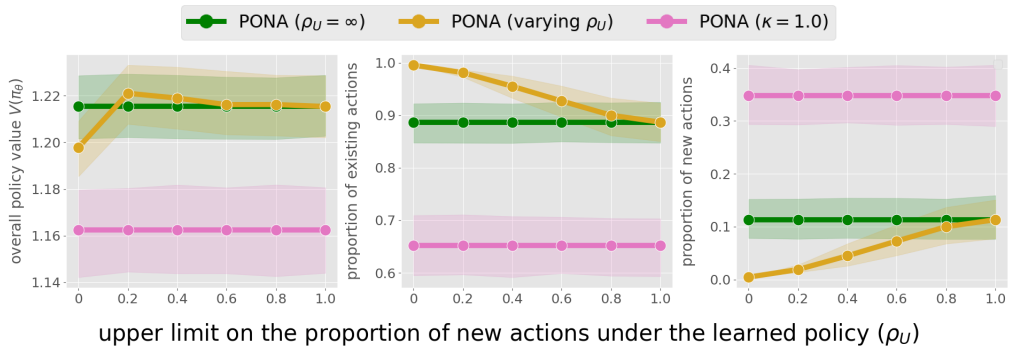


**Figure 8: Comparisons of the overall policy value, proportion of existing actions, and proportion of new actions under the learned policy with varying upper limits on the proportion of new actions ($\rho_U$).**

100% of the rewards are observable. Consequently, we can directly use the observed rewards to conduct experiments on OPL, which makes this dataset the most suitable real-world public data for our experiments. We use user information as the context $x$ and video IDs as the actions $a$. We then rely on video tags and video categories to construct action features $f(a)$. More specifically, the video categories follow a three-level hierarchical structure. Combining the video tags and categories creates a four-dimensional discrete action feature space. To reduce complexity, we extract 15 factors for each action feature, which results in $|\mathcal{A}| = 117$. We use the watch ratio, defined as the play duration divided by the video duration, as the reward $r$. Following the synthetic experiment, we use the softmax function to define the logging policy with $\beta = 0.05$.

***Results.*** We compare the overall policy values learned by each OPL method across 50 simulations with different random seeds. All other settings match those of the synthetic experiments. Figures 14 and 15 present the results with varying training data sizes and percentages of new actions. The results demonstrate trends similar to those observed in the synthetic experiments. PONA achieves the overall policy value

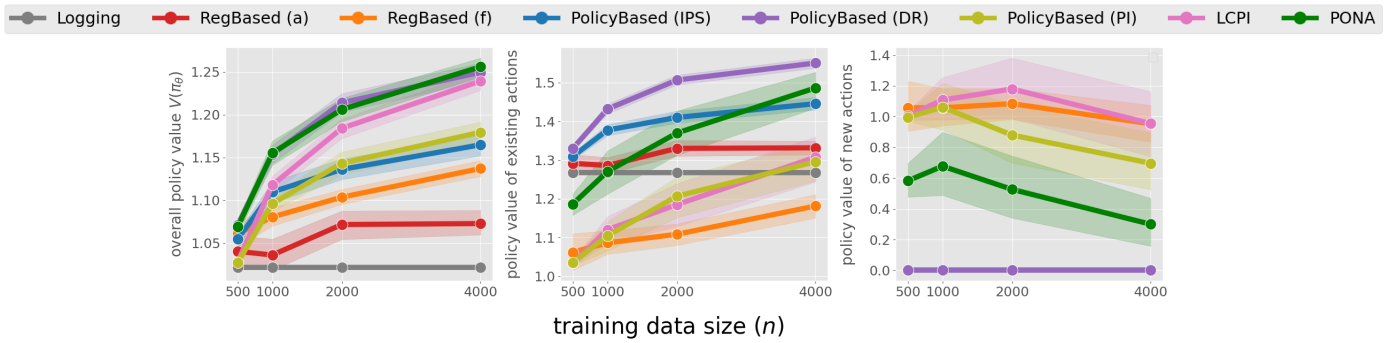


**Figure 9: Comparisons of the overall policy value, policy value of existing action, and policy value of new action with varying training data sizes ($n$). Note that the metrics are normalized by those of the uniform random policy.**

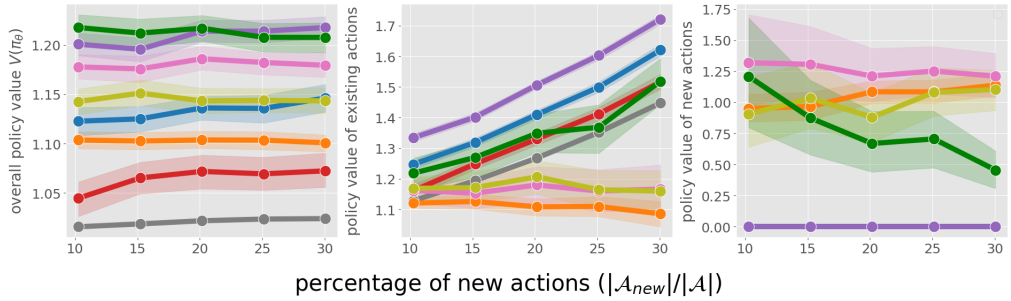


**Figure 10: Comparisons of the overall policy value, policy value of existing action, and policy value of new action with varying percentages of new actions ($|\mathcal{A}_{new}|/|\mathcal{A}|$). Note that the metrics are normalized by those of the uniform random policy.**

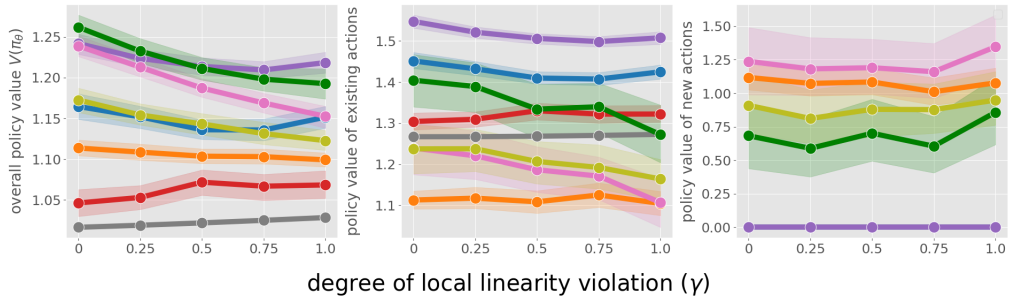


**Figure 11: Comparisons of the overall policy value, policy value of existing action, and policy value of new action with varying degrees of local linearity violation ($\gamma$). Note that the metrics are normalized by those of the uniform random policy.**

comparable to PolicyBased (DR). PONA also successfully identifies promising new actions and performs much better than RegBased ($f$) in terms of **policy value per new action**. Note that typical methods including PolicyBased (DR) did not choose new actions at all on the real data as well. RegBased ($f$) and PolicyBased (PI) perform equally or worse compared to the uniform random policy in selecting new actions. This is because the real-world dataset contains a larger number of values for each action feature, making it more challenging to learn the rewards of new actions via mere regression or based on linearity.

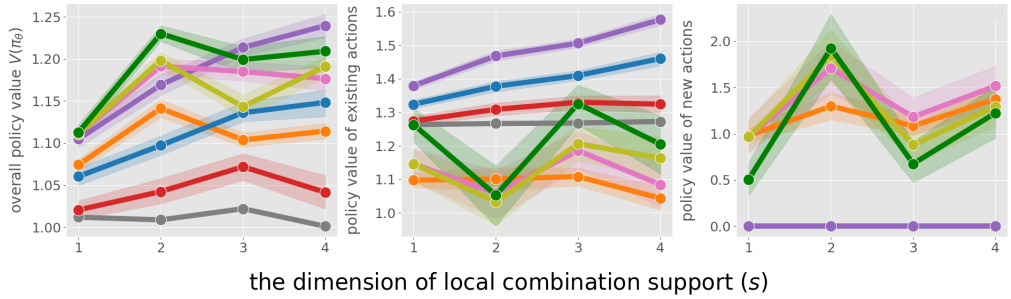


**Figure 12: Comparisons of the overall policy value, policy value of existing action, and policy value of new action with varying the dimension of Local Combination Support ($s$). Note that the metrics are normalized by those of the uniform random policy.**

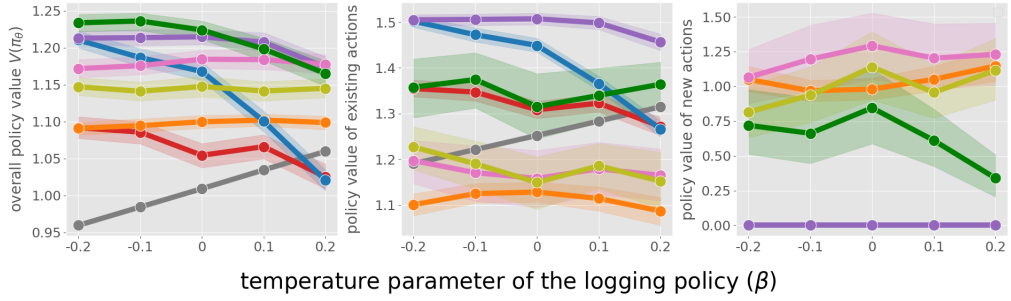


**Figure 13: Comparisons of the overall policy value, policy value of existing action, and policy value of new action with varying temperature parameter of the logging policy ($\beta$). Note that the metrics are normalized by those of the uniform random policy.**

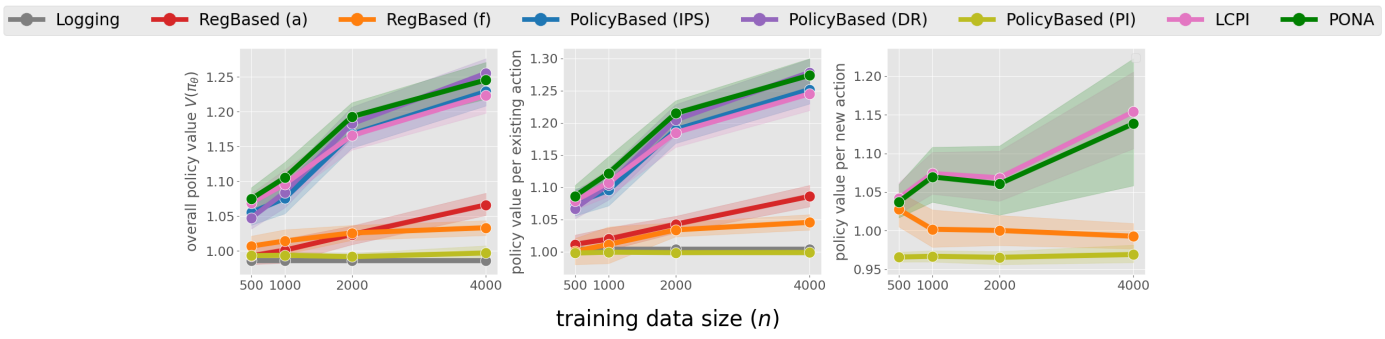


**Figure 14: Comparisons of the overall policy value, policy value per existing action, and policy value per new action with varying training data sizes ($n$) on the KuaiRec dataset. Note that the metrics are normalized by those of the uniform random policy.**

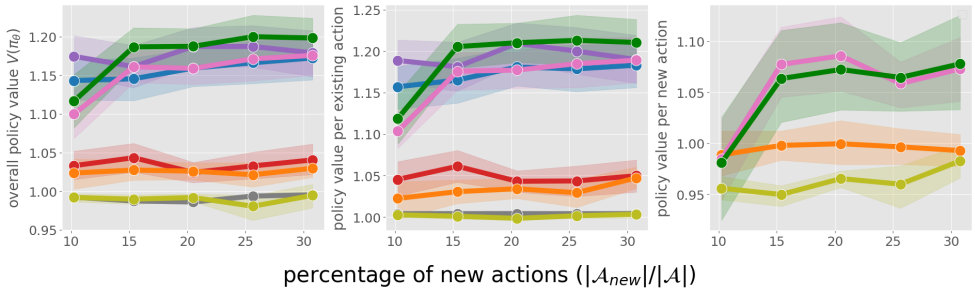


**Figure 15: Comparisons of the overall policy value, policy value per existing action, and policy value per new action with varying percentages of new actions ($|\mathcal{A}_{new}|/|\mathcal{A}|$) on the KuaiRec dataset. Note that the metrics are normalized by those of the uniform random policy.**