Learning Action Embeddings for Off-Policy Evaluation

Matej Cief[1,2][0000−0001−9225−5155], Jacek Golebiowski[3][0000−0001−8053−8318], Philipp Schmidt[3], Ziawasch Abedjan[4][0000−0002−2846−1373], and Artur Bekasov[5]

[1] Brno University of Technology, Brno, Czech Republic**
[2] Kempelen Institute of Intelligent Technologies, Bratislava, Slovakia
matej.cief@kinit.sk
[3] Amazon, Berlin, Germany
[4] Leibniz University Hannover, Hanover, Germany
[5] Amazon, London, United Kingdom

Abstract. Off-policy evaluation (OPE) methods allow us to compute the expected reward of a policy by using the logged data collected by a different policy. However, when the number of actions is large, or certain actions are under-explored by the logging policy, existing estimators based on inverse-propensity scoring (IPS) can have a high or even infinite variance. Saito and Joachims [13] propose marginalized IPS (MIPS) that uses action *embeddings* instead, which reduces the variance of IPS in large action spaces. MIPS assumes that good action embeddings can be defined by the practitioner, which is difficult to do in many real-world applications. In this work, we explore *learning* action embeddings from logged data. In particular, we use intermediate outputs of a trained reward model to define action embeddings for MIPS. This approach extends MIPS to more applications, and in our experiments improves upon MIPS with pre-defined embeddings, as well as standard baselines, both on synthetic and real-world data. Our method does not make assumptions about the reward model class, and supports using additional action information to further improve the estimates. The proposed approach presents an appealing alternative to DR for combining the low variance of DM with the low bias of IPS.

Keywords: off-policy evaluation · multi-armed bandits · large action space · representation learning · recommender systems

1 Introduction

The *multi-armed bandit* (MAB) framework is commonly used to model the interaction between recommender systems or search engines with their users [21]. In MAB, an *agent* performs *actions* and receives *rewards*, where each reward is sampled from an unknown reward distribution conditioned on the action. The goal is to learn a *policy* for the agent that maximizes the cumulative reward

** Work done during an internship at Amazon.

over multiple iterations. In a *contextual* MAB, the reward distribution is also conditioned on a *context* that is observed by the agent, and is used as an additional input to the policy function.

Bandit policies are commonly learned *online*, where we run a policy in production and update it iteratively using the observed rewards [21]. However, deploying untested policies to production is risky. An alternative is to leverage the logged data from historical customer interactions to learn a policy *offline*. The fundamental problem of offline learning is *off-policy evaluation* (OPE), where we try to estimate the expected reward of a new policy without deploying it. The disadvantage of OPE is that the computed value is only an *estimate* of the true value of the policy, and the success of offline learning depends on how accurate this estimate is. When the policy used to gather logged data has a low probability of choosing a subset of actions, existing estimators based on *inverse propensity scoring* [IPS, 10] can be inaccurate [11]. This problem is especially potent when the number of actions is large, as is often the case in recommender systems [19].

Embedding contexts into a lower-dimensional space can improve the off-policy estimates when dealing with a large number of contexts [21]. Similarly, Saito and Joachims [13] propose to embed the actions, and to use the embeddings in *marginalized* IPS (MIPS). Saito and Joachims use additional action information to define the embeddings. For example, in fashion recommendation, where an action is a particular fashion item to recommend, they use the price, the brand and the category of an item as embedding dimensions. For many problems of interest defining action embeddings by hand will be difficult, and/or will require expert knowledge.

Motivated by the two considerations above, we propose methods for *learning* or *fine-tuning* the action embeddings, aiming to improve the performance of MIPS. Our key hypothesis is that an action embedding that is useful for reward prediction will also be useful for MIPS. In particular, we propose to use the intermediate outputs of a trained reward model as action embeddings for MIPS, instead of using them for direct reward prediction as in DM.

We study this approach on both synthetic and real-world datasets, demonstrating that it consistently outperforms MIPS with pre-defined action embeddings, as well as common baselines like standard IPS, *direct method* (DM), and *doubly robust* estimation [DR, 3]. We demonstrate that the method is not sensitive to reward model misspecification: a linear regression reward model produces useful action embeddings even when the true reward function itself is non-linear. We propose methods for utilizing additional action information when it is available, and show that learning/fine-tuning action embeddings is especially useful when the additional action information is high-dimensional, or when the information affecting the reward is incomplete. Finally, we provide a theoretical analysis of our method, showing that it can be interpreted as kernelized DM.

2 Background and related work

The interaction of a recommender system with the user starts with the system receiving the context information (e.g. user's purchase history) $x \in \mathcal{X} \subseteq \mathbb{R}^{d_{\mathcal{X}}}$ drawn i.i.d. from an unknown distribution $p(x)$. The system then chooses an action (e.g. a ranked set of products) $a \sim \pi(a \mid x)$ from the set of available actions $\mathcal{A}$ according to the system's policy π. Finally, the user interacts with the displayed products via clicks or purchases, and the system receives a reward $r \in [r_{min}, r_{max}]$ sampled from a reward distribution $p(r \mid x, a)$. This process repeats for n rounds, where in round t the system observes a context x_t, draws an action a_t according to $\pi(a \mid x_t)$, and observes a reward $r_t \sim p(r \mid x_t, a_t)$. All interactions are recorded in a *logged dataset* $\mathcal{D} = \{(x_t, a_t, r_t)\}_{t=1}^{n}$. The policy π_0 that collects $\mathcal{D}$ is called a *logging policy*. The goal of *off-policy evaluation* [OPE, 3, 20, 16, 15, 8, 13] is to develop $\hat{V}$, an estimator of the policy *value* $V(\pi) = \mathbb{E}_{p(x)\pi(a|x)p(r|x,a)}[r]$ such that $\hat{V}(\pi) \approx V(\pi)$.

Inverse propensity scoring [IPS, 10] is a commonly used estimator that works by re-weighting the rewards in $\mathcal{D}$ based on how likely the corresponding actions are to be selected under the target policy:

$$\hat{V}_{\mathrm{IPS}}(\pi) = \frac{1}{n} \sum_{t=1}^{n} \frac{\pi(a_t \mid x_t)}{\pi_0(a_t \mid x_t)} r_t. \tag{1}$$

Many state-of-the-art OPE methods are based on IPS [3, 20, 16, 15, 8]. A key assumption of IPS, however, is that the logging policy π_0 and the target policy π have *common support*, i.e. $\pi(a \mid x) > 0 \implies \pi_0(a \mid x) > 0$ [13]. When dealing with large action spaces, it can be difficult to select a logging policy that puts non-zero probability on *all* actions, yet does not degrade the customer experience. Even if the two policies have common support, a low probability of selecting an action under the logging policy results in a large weight $\pi(a_t|x_t)/\pi_0(a_t|x_t)$, which increases the variance of the IPS estimate. To reduce the variance of IPS, we can either clip the importance weights according to a tunable clipping parameter [17] or self-normalize them [SNIPS, 18], but the former biases the estimator and the latter can still result in a relatively high variance. To deal with continuous space, Kallus and Zhou [7] use a kernel function to calculate propensities for IPS. We also use a similar technique to estimate the propensities of learned embeddings, and we show that the propensity estimation used in MIPS can be interpreted as kernel regression in Section 3.3.

In their work Saito and Joachims [13] instead assume that useful *action embeddings* are available to the estimator, and that we can pool information across similar actions to improve the estimates. Suppose we have an action embedding $e \in \mathcal{E} \subseteq \mathbb{R}^{d_{\mathcal{E}}}$ for each action a. Saito and Joachims show that if the logging policy π_0 has common *embedding support*, i.e. $p(e \mid \pi, x) > 0 \implies p(e \mid \pi_0, x) > 0$, and the action a has no *direct effect* on the reward r, i.e. $a \perp r \mid x, e$, then the

proposed *marginalized* IPS (MIPS) estimator

$$\hat{V}_{\mathrm{MIPS}}(\pi) = \frac{1}{n}\sum_{t=1}^{n} \frac{p(e_t \mid \pi, x_t)}{p(e_t \mid \pi_0, x_t)} r_t \tag{2}$$

is unbiased. Experiments by Saito and Joachims demonstrate that in practice MIPS has a lower variance than IPS, and produces estimates with a lower MSE. Our goal in this work is to develop a method for *learning* the action embeddings from logged data to further improve MIPS, and to make it applicable to problems where pre-defined embeddings are not available.

The *direct method* (DM) learns the model of the expected reward for the context-action pair, and uses it to estimate the expected reward of a policy. DM has a low variance, but suffers from a high bias when the model is misspecified. *Doubly robust* estimator [DR, 3, 5, 20] combines DM and IPS: it uses the learned model of the reward as a control variate. In other words, it computes a non-parametric estimation on the reward model residuals. If the expected reward is correctly specified, DR can achieve a lower variance than IPS, and a lower bias than DM. Several extensions of DR have been proposed [20, 15], but, to the best of our knowledge, none of them aim to improve the performance of DR in large action spaces.

Two methods for off-policy evaluation with large action spaces have been developed in parallel with our work [9, 14]. Peng et al. [9] propose to cluster similar actions, and replace individual action propensity scores with those of action clusters. Saito et al. [14] train a two-step regression model, hypothesizing that similar actions share a cluster effect on the reward, but also have their own residual effect.

3 Methods

In this work we propose to use the reward signal to learn the embeddings of actions for the MIPS estimator, which would estimate the propensities of these embeddings, and use them instead of the original action propensities to re-weight the observed rewards.

3.1 Motivation

We first formalize the main assumption that motivates the proposed method.

Assumption 1 (Action Embedding Space) *For every set of actions $\mathcal{A}$, there exists a lower-dimensional embedding space $\mathcal{E} \subseteq \mathbb{R}^{d_\mathcal{E}}$ so that every action $a \in \mathcal{A}$ can be mapped to an embedding $e \in \mathcal{E}$ while $p(r \mid a, x) \approx p(r \mid e, x)$ holds for every context x. We denote $d_\mathcal{E} < |\mathcal{A}|$ to be the number of dimensions of embedding space $\mathcal{E}$.*

This assumption holds when $d_\mathcal{E} = |\mathcal{A}|$ as the actions can be mapped as one-hot encoded representations. But we hypothesize that in practice some actions are

Table 1: Mean squared error of the estimators on a synthetic experiment while varying the number of actions. Reporting mean and standard error averaged over 50 runs.

$\lvert\mathcal{A}\rvert$	IPS	DM	Learned MIPS
50	0.63 ± 0.05	0.64 ± 0.05	$\mathbf{0.56 \pm 0.05}$
100	0.68 ± 0.04	2.13 ± 0.11	$\mathbf{0.58 \pm 0.05}$
200	0.61 ± 0.04	7.16 ± 0.19	$\mathbf{0.55 \pm 0.04}$
500	0.71 ± 0.05	27.9 ± 0.32	$\mathbf{0.56 \pm 0.05}$
1000	0.70 ± 0.04	62.3 ± 0.31	$\mathbf{0.55 \pm 0.03}$

"similar", and that the action space can be represented more efficiently. For example, if certain types of customers mostly clicked on shoes in the logged data, we expect to observe a high reward for a shoe product, even if we have never recommended this particular product for this type of customer before.

Based on Assumption 1, the environment can be represented as a graphical model $a \to e \to r \leftarrow x$. This can be learned the same way as learning a reward model for DM, but instead of using the model for reward prediction, we use the intermediate model outputs as action embeddings. As model misspecification is one of the biggest issues with DM [5], it is often difficult to predict the reward end-to-end. But even when the true reward function is complex, and the model class is not rich enough to recover it, we hypothesize that the model will learn a useful $a \to e$ mapping that can be used in MIPS as defined in Equation (2).

To test our hypothesis, we design the following toy example[6]. Let $\mathcal{D} = \{(x_t, a_t, r_t)\}_{t=1}^{n}$ for $n = 1000$ be the logged dataset, where a context $x \in [-1, 1]^{d_\mathcal{X}}$, $d_\mathcal{X} = 5$, is drawn from the standard normal distribution, action $a \sim \pi_0$ is chosen from set $\mathcal{A}$ by the unconditional logging policy $\pi_0(a) = {}^{\nu_a}/_{\sum_{a' \in \mathcal{A}} \nu_{a'}}$, where $\nu_a \sim \text{Exp}(1)$ is drawn from an exponential distribution, and the reward is generated by $r = f_a(x) + \eta$, where $f_a(x) = 1/(1 + e^{-(x^\top \theta_a + \mu_a)})$ is a logistic model for action a with randomly initialized parameters $\theta_a \sim \mathcal{N}(0, 1)^{d_\mathcal{X}}, \mu_a \sim \mathcal{N}(0, 1)$, and $\eta \sim \mathcal{N}(0, 0.1)$ is random noise. We use the uniform random target policy. For DM, we fit a linear regression $\hat{f}_a(x)$ to estimate r. Of course, linear regression is not expressive enough to fit the non-linear f_a, but it learns the correct *order* of the contexts, meaning $\forall x_1, x_2 \in \mathcal{X}, \hat{f}_a(x_1) < \hat{f}_a(x_2) \Leftrightarrow f_a(x_1) < f_a(x_2)$. We take these learned embeddings and use them in the MIPS estimator [12] and refer to this method as *Learned MIPS*. We vary the number of actions $|\mathcal{A}| \in \{50, 100, 200, 500, 1000\}$, while keeping the size of the logged data fixed. For each $|\mathcal{A}|$ we synthesize 50 reward functions, and for each reward function we generate 15 logged datasets to evaluate the methods. The results are summarized in Table 1.

As the number of samples per action decreases, the learned embeddings become increasingly inaccurate for *direct* reward prediction (see the increasing

[6] Code to reproduce this and further experiments is available at https://github.com/amazon-science/ope-learn-action-embeddings.

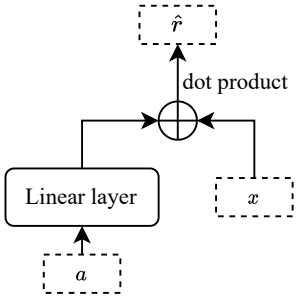

Fig. 1: The model we use to learn the action embeddings. We use a simple linear model in this work, but any model class can be used. Input a can be a one-hot encoded representation of the action identity, a pre-defined action embedding, or a concatenated vector of both.

error of DM in Table 1), but they still reflect the structure of the problem, and we can build an accurate model-free estimator on top of them (Learned MIPS outperforms both DM and IPS). This simple example demonstrates the potential of the proposed method as a novel way to combine the strengths of model-based and model-free off-policy estimation methods.

3.2 Algorithm

As shown in the previous section, we can learn action embeddings by fitting a separate linear regression model $\hat{f}_a(x)$ for each action a. In general, as in DM, we can use any model class. For example, we can fit a deep neural network, and use the intermediate outputs as action embeddings. In this work, we use a linear model visualized in Figure 1. In our preliminary experiments, more complex neural network models demonstrated comparable or worse performance. We hypothesize that this is due to the high expressivity (and hence high variance) of neural networks, which is counter to our goal of reducing estimator variance. We leave further empirical study and theoretical analysis of this phenomenon for future work.

The model takes the action identity a, and computes a reward estimate $\hat{r}$ as a dot product between the action embedding and the context vector x. We train the model to minimize the MSE between $\hat{r}$ and the reward r observed in the logged data. The learned embedding for action a is the output of the linear layer. As we show in Section 4, we can also include additional action information (e.g. content-based features of the corresponding product), which can further improve the performance of the estimator.

Having learned the action embeddings, we follow MIPS [13] and fit a logistic regression model to estimate $p(a \mid e)$. This estimate is used to compute the embedding propensities ($p(e \mid \pi, x)$ and $p(e \mid \pi_0, x)$) in Equation (2), noting that $p(e \mid \pi, x) = p(a \mid e)\pi(a \mid x)$. We discuss this step in more detail in the next section, where we show a connection between our method and kernelized DM.

3.3 Connection to DM

Learned MIPS resembles DM in that both methods solve the OPE task by first training a reward predictor. The two methods differ, however, in how they apply said predictor. In this section, we show a direct connection between learned MIPS and DM. In particular, we show that we can interpret the proposed method as DM that uses *kernel regression* with a learned feature embedding as the reward predictor.

To simplify the notation, we consider a non-contextual estimation task: given logged data $\mathcal{D} = \{a_t, r_t\}_{t=1}^n$ gathered with a policy $\pi_0(a)$, we want to estimate the value of a policy $\pi(a)$. In this setting, the DM estimate is simply

$$\hat{V}_{\mathrm{DM}}(\pi) = \sum_{a \in \mathcal{A}} \pi(a) \hat{r}(a), \tag{3}$$

where $\hat{r}(a)$ is the model of expected reward for action a that we learn using $\mathcal{D}$ as the training data. We now define $\tilde{\mathcal{D}} = \{a_t, e_t, r_t\}_{t=1}^n$, where $e_t \sim p\left(e \mid a_t\right)$ is the corresponding action embedding. The MIPS estimate is then

$$\hat{V}_{\mathrm{MIPS}}(\pi) = \frac{1}{n} \sum_{t=1}^{n} \mathbb{E}_{p(a|e_t)} \left[\frac{\pi(a)}{\pi_0(a)} \right] r_t = \frac{1}{n} \sum_{t=1}^{n} \sum_{a \in \mathcal{A}} \frac{\pi(a)}{\pi_0(a)} p(a \mid e_t) r_t \tag{4}$$

$$= \sum_{a \in \mathcal{A}} \pi(a) \frac{1}{n} \sum_{t=1}^{n} \frac{p(a \mid e_t)}{\pi_0(a)} r_t = \sum_{a \in \mathcal{A}} \pi(a) \tilde{r}(a), \tag{5}$$

where $p(a \mid e_t)$ is estimated as in Equation (7). In other words, the MIPS estimate matches the DM estimate with a different expected reward model:

$$\tilde{r}(a) = \frac{1}{n} \sum_{t=1}^{n} \frac{p(a \mid e_t)}{\pi_0(a)} r_t. \tag{6}$$

We now try to understand the nature of $\tilde{r}$. Intuitively, $p(a \mid e)$ will be higher when the embedding of a is "closer" to e according to some distance measure. In particular, let $e \sim \mathcal{N}(f(a), \sigma^2 I)$ be the embedding distribution, where $f(a)$ is the learned deterministic embedding. As in MIPS, we estimate $p(a \mid e)$ by fitting a probabilistic classifier to the dataset $\{a_t, e_t\}_{t=1}^n$. Saito and Joachims use logistic regression, but here we consider *linear discriminant analysis* [LDA, 6, Section 4.3], a comparable model. LDA fits a normal distribution to each class, hence we expect it to recover $\mathcal{N}(f(a), \sigma^2 I)$. We also expect it to recover $\pi_0(a)$ as the class prior, because we sampled $a_t \sim \pi_0(a)$ to produce the training data $\{a_t, e_t\}_{t=1}^n$. The predictive probability of LDA is then

$$p(a \mid e) \propto \mathcal{N}(e; f(a), \sigma^2 I) \pi_0(a). \tag{7}$$

Combining Equations (6) and (7) we get

$$\tilde{r}(a) \propto \frac{1}{n} \sum_{t=1}^{n} \mathcal{N}(e_t; f(a), \sigma^2 I) r_t, \tag{8}$$

where

$$\mathcal{N}(e_t; f(a), \sigma^2 I) \propto \exp\left(-\frac{1}{2\sigma^2}(e_t - f(a))^\top (e_t - f(a))\right). \tag{9}$$

This confirms our intuition: if we use LDA as the classifier, the expected reward $\tilde{r}(a)$ in Equation (5) is a weighted combination of all rewards in the logged data, where samples with e_t "closer" to the action embedding $f(a)$ have a higher weight. For the normal embedding distribution, the weight is determined by a squared Euclidean distance between vectors e_t and $f(a)$, but other distributions will induce different distance functions. In fact, Equation (8) is equivalent to *kernel regression* [2, Chapter 10] in a learned embedding space determined by f, with a kernel determined by the embedding distribution.

While the above holds when using LDA as the classifier, in our experiments we follow Saito and Joachims and use the logistic regression classifier. Both of these methods fit linear decision boundaries and give similar results in practice (4; 6, Chapter 4), which means that our interpretation likely holds for logistic regression-based MIPS as well. As DM, our method does not provide theoretical guarantees on the bias/variance for a general reward model class, but, as we show in the next section, demonstrates good performance in practice.

4 Empirical evaluation

In this section we extend our proof-of-concept experiments in Section 3 to a more complex synthetic dataset, and conduct experiments on a real-world fashion recommendation dataset. In the synthetic experiments, we evaluate two classes of methods based on whether the method uses additional action information or not. We find that our methods outperform all the standard baselines and, in some cases, improve upon the true embeddings used to generate the reward.

4.1 Synthetic experiments

For our empirical evaluation, we follow the setup by Saito and Joachims [13]. We generate synthetic data by first sampling $d_{\mathcal{X}}$-dimensional context vectors x from the standard normal distribution. We also sample $d_{\mathcal{E}}$-dimensional categorical action embeddings $e \in \mathcal{E}$ from the following distribution

$$p(e \mid a) = \prod_{k=1}^{d_{\mathcal{E}}} \frac{\exp \alpha_{a,e_k}}{\sum_{e' \in \mathcal{E}_k} \exp \alpha_{a,e'}}, \tag{10}$$

where $\{\alpha_{a,e_k}\}$ is a set of parameters sampled independently from the standard normal distribution. Each dimension of $\mathcal{E}$ has a cardinality of 10. We then synthesize the expected reward as

$$q(x, e) = \sum_{k=1}^{d_{\mathcal{E}}} \eta \cdot (x^\top M x_{e_k} + \theta_x^\top x + \theta_e^\top x_{e_k}), \tag{11}$$

where M is the parameter matrix and θ_x, θ_e are parameter vectors sampled uniformly in $[-1, 1]$. x_{e_k} is a parameter vector corresponding to the k-th dimension of the action embedding and is sampled from the standard normal distribution. Parameter η_k specifies the importance of the k-th dimension and is drawn from a Dirichlet distribution $\eta \sim \mathrm{Dir}(\alpha)$ with its parameters $\alpha = (\alpha_1, \ldots, \alpha_{d_\mathcal{E}})$ sampled from a uniform distribution $[0, 1)$. With this setup, the action embeddings are fully informative, and the action identity has no *direct effect* on the reward. On the other hand, if action embeddings have some missing dimensions, they do not fully capture the action's effect on the reward. In this case, we hypothesize that we can learn better embeddings, and we test this in one of the experiments.

We also follow Saito and Joachims [13] in how we synthesize the logging policy π_0. We apply the softmax function to $q(x, a) = \mathbb{E}_{p(e|a)}[q(x, e)]$ as

$$\pi_0(a \mid x) = \frac{\exp \beta \cdot q(x, a)}{\sum_{a' \in \mathcal{A}} \exp \beta \cdot q(x, a')}, \tag{12}$$

where β is the parameter that controls the entropy and the optimality of the logging policy. A large positive value leads to an optimal, near-deterministic logging policy, while 0 leads to a uniform-random policy. We define the target policy π as an ε-greedy policy

$$\pi(a \mid x) = (1 - \varepsilon) \cdot \mathbb{1}\left\{a = \arg\max_{a' \in \mathcal{A}} a(x, a')\right\} + \frac{\varepsilon}{|\mathcal{A}|}, \tag{13}$$

where $\varepsilon \in [0, 1]$ controls the exploration rate of the target policy. Similar to Saito and Joachims [13], we set $\beta = -1$ and $\varepsilon = 0.05$ in our experiments.

To generate a batch of logged data, we follow an iterative process to generate n samples $\mathcal{D} = \{x_t, a_t, e_t, r_t\}_{t=1}^n$. In each iteration, we sample a context x from the standard normal distribution ($d_\mathcal{X} = 10$), and a discrete action a from π_0 according to Equation (12). Given an action a, we then sample a categorical action embedding e using Equation (10). Finally, we sample the reward from a normal distribution with mean $q(x, e)$ defined in Equation (11) and standard deviation $\sigma = 2.5$ [13].

We assume three variants of our method, based on which action representation is passed to the model:

- *Learned MIPS OneHot* only sees one-hot encoded action identities.
- *Learned MIPS FineTune* only sees pre-defined action embeddings.
- *Learned MIPS Combined* sees both of the above concatenated into a single vector.

We compare our methods to all common estimators, including DM, IPS, and DR. To gauge the quality of the learned embeddings, we specifically compare to MIPS with pre-defined embeddings, and *MIPS (true)*, which uses the true propensity scores of the pre-defined embeddings, which are not available in practice. For all experiments we report the MSE and standard errors averaged over 100 different subsampled datasets.

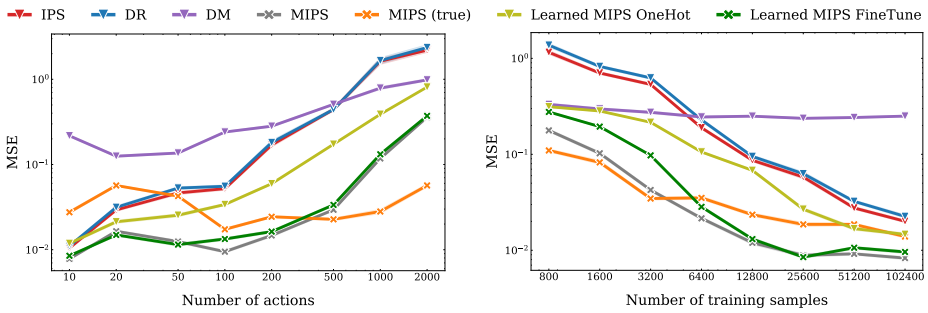

Fig. 2: Synthetic experiments varying the number of actions and training samples. To better distinguish comparable methods, those marked with $\triangledown$ only use action identities, and those marked with $\times$ also use pre-defined action embeddings. *Learned MIPS OneHot* outperforms all standard baselines. When we have enough data for every action, it performs just as well as IPS. As the variance grows with fewer samples per action, its error approaches the one of DM using the same model. The pre-defined embeddings ($d_{\mathcal{E}} = 3$) have low bias and variance; hence our methods can not improve upon them. Shaded areas around the lines are standard errors (almost invisible).

Can we improve upon standard baselines without using pre-defined embeddings? We evaluate our performance against all standard estimators in two sets of experiments with data generated according to the procedure described in Section 4.1. All pre-defined embedding dimensions are preserved, hence these embeddings are *fully-informative*, i.e. $q(x, e) = q(x, e, a)$. We fix the number of pre-defined embedding dimensions to $d_{\mathcal{E}} = 3$. In the first experiment, we vary the number of actions from 10 to 2,000 with a fixed sample size of $n = 20{,}000$. In the second experiment, we vary the number of samples in the training data $n = [800, 1600, \ldots, 102400]$ and leave the number of actions fixed at $|\mathcal{A}| = 100$. The results of these two experiments are presented in Figure 2.

Learned MIPS OneHot outperforms all other methods using action identities. We can observe the difference between the two methods that use the same model, *Learned MIPS OneHot* and *DM*. Using learned embedding propensities in MIPS results in a smaller bias compared to using them in direct reward prediction. The error of *Learned MIPS* keeps shrinking with more data. While IPS and DR do not pool information across actions, MIPS is using the embedding propensities influenced by every action, resulting in a better bias-variance trade-off. When the number of actions is small enough, and we have enough data, IPS estimates already have a low variance, and we do not observe any significant improvements.

We also report the performance of the methods that use pre-defined embeddings. Perhaps unsurprisingly, the learned embeddings perform worse than pre-defined embeddings, and even *Learned MIPS FineTune* does not improve over them. This is because the pre-defined embeddings compress the information about the action well. In further experiments, we study more realistic benchmarks, where embeddings are higher-dimensional or do not fully capture the reward.

Both of these cases are common in practice, for example, when using pre-defined embeddings extracted from LLMs or when extending the setup to lists of items in learning to rank.

Can we improve upon the pre-defined embeddings? In the previous experiment, fine-tuned embeddings did not improve upon the pre-defined embeddings. The reason is that the pre-defined embeddings have low bias and low variance. This is rarely the case in practice. In this experiment, we show that we can learn better embeddings when the pre-defined embeddings have a high number of embedding dimensions or when they do not capture the full effect on the reward. We evaluate how the bias of the pre-defined embeddings influences our methods and whether it is better to learn the embeddings from action identities or fine-tune them from pre-defined embeddings in this setting. We progressively add more bias similarly to Saito and Joachims [13]. We fix the number of dimensions in the action embedding to $d_{\mathcal{E}} = 20$, and after the reward is generated, we hide a certain number of dimensions so the estimators do not have access to fully-informative embeddings. The results are shown in Figure 3. As the embeddings worsen, our methods that learn or fine-tune them outperform MIPS. We can see the bias-variance trade-off between *Learned MIPS OneHot* and *Learned MIPS FineTune*. When the number of unobserved dimensions is small, the variance reduction gained from pre-defined embeddings is greater than induced bias. As we increase the number of unobserved dimensions, the bias of pre-defined embeddings can get even higher than the variance of IPS. In practice, we often do not know how biased the pre-defined embeddings are. Therefore, we introduce *Learned MIPS Combined* and observe performance improvements when using this method for the majority of the values of the unobserved dimensions. When the information in the embedding is complete (we do not hide any dimensions), providing additional dimensions as an action identity only makes the problem more difficult (*Learned MIPS FineTune* outperforms *Learned MIPS Combined* with more embedding dimensions). But as soon as some of the information in the embedding is omitted and the common *embedding support* assumption [13] is violated, the action identity information is very useful, and *Learned MIPS Combined* uses it to model the direct effect of the action missing from the pre-defined embedding. As we show in the next section, it may be difficult to gauge in practice how much information is missing in the real-world embedding and which method would yield the best performance. We leave the optimal method selection for future work.

4.2 Real-world data

We follow the experimental setup of Saito and Joachims [13] to evaluate how the estimators perform on a real-world bandit dataset. In particular, we use a fashion recommendation dataset that was collected on a large-scale fashion e-commerce platform, and comes in the form of $\mathcal{D} = \{(x_t, a_t, e_t, r_t)\}_{t=1}^{n}$ tuples. The dataset was collected during an A/B test under two policies: a uniform policy π_0 and a Thompson sampling policy π. The size of the action space is $|\mathcal{A}| = 240$, and every action (a fashion item) comes with a pre-defined 4-dimensional action embedding

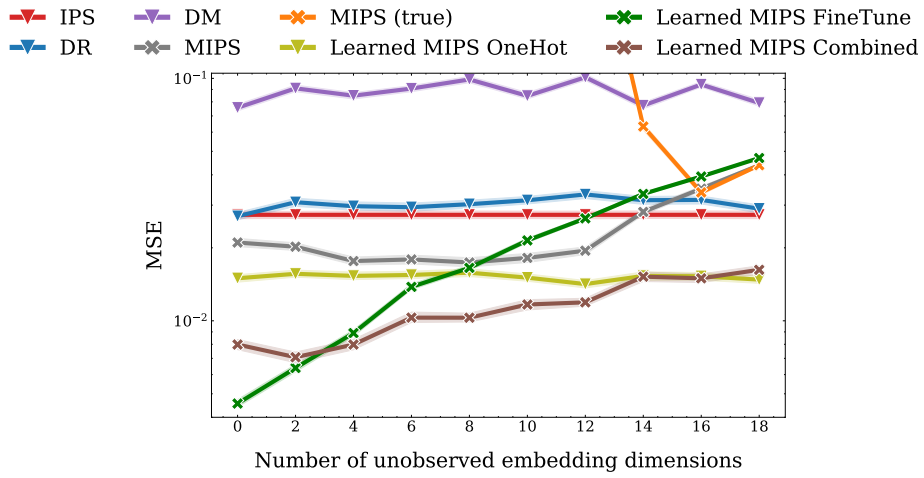

Fig. 3: Synthetic experiments varying the number of unobserved dimensions. As we progressively hide more dimensions, pre-defined embeddings get more biased and methods using them get less accurate. Combining *Learned MIPS OneHot* and *FineTune* yields the most robust results when the bias of pre-defined embeddings is unknown. Shaded areas around the lines are standard errors (almost invisible).

e, which includes the item's price, brand, and hierarchical category information. We use $n = 10000$ sub-sampled observations from the "ALL" campaign in the dataset to be directly comparable to the prior work [13, 9].

In line with Saito and Joachims [13], we repeat the experiment over 150 bootstrap samples, normalize the estimators' MSE relative to the MSE of IPS (i.e. divide the method's MSEs for a given data sample by the MSE of IPS for the same sample), and compute the cumulative distribution function (CDF) of these MSEs. We report the results in Figure 4. We compare against another competitive baseline *MIPS(w/ SLOPE)* that greedily improves the performance by dropping embedding dimensions [13]. *Learned MIPS* methods outperform IPS in about 75% of runs, more often than any other baseline. The next best-performing method is MIPS (w/ SLOPE), which outperforms IPS in about 65% of runs. For DM, we use the same model as in *Learned MIPS OneHot*. These results support our hypothesis that even though DM fails to model the reward, the embeddings it learns in the process can still be useful for model-free estimation.

5 Conclusion

In this work, we propose methods that assume a structured action space in the contextual bandit setting, and reduce the variance of model-free off-policy estimators by *learning* action embeddings from the reward signal. We extend MIPS to settings where pre-defined embeddings are not available, or are difficult to define. At the same time, we show that the method can improve upon the pre-defined

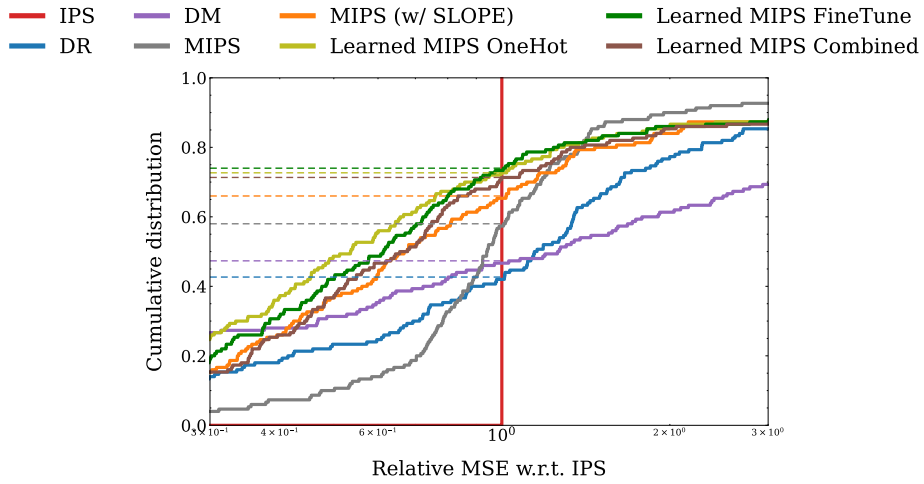

Fig. 4: CDF of relative MSEs w.r.t IPS on the real-world dataset. The intersection of a method's curve with the IPS curve tells us the proportion of experiments in which the method performs better than IPS.

embeddings even when they are available. The proposed method outperforms all standard baselines on synthetic and real-world fashion recommendation data, even if the reward model itself is inaccurate. An interesting future direction is to study the embeddings learned by more complex classes of reward models, such as neural networks. In the current experiments, we estimate the action distribution over the embeddings $p(a \mid e)$ using a discriminative model as proposed by Saito and Joachims [13]. To enable explicit bias-variance control in future work, it would be interesting to experiment with estimating these weights with a generative classifier and a *prescribed* form of $p(a \mid e)$.

We would also like to extend our methods to the learning-to-rank setting, as list-level embeddings can dramatically reduce the combinatorial complexity of the problem. This can be done by concatenating the item-level embeddings of a single list into a longer embedding and applying our method to it. We assume that in the same way our model exploits the similarity relationships between individual items, it would also learn to embed the browsing customer behavior impacting the final reward (e.g., the items at the end of the list are observed less frequently). This would eliminate the need for explicitly modeling these behaviors by click models [1]. Another way to use our method in LTR, as it works with any model class, is to use some of the well-known list-wise LTR methods and use its intermediate output as the learned action embedding.

Acknowledgements We want to thank Mohamed Sadek for his contributions to the codebase. The research conducted by Matej Cief (also with slovak.AI) was partially supported by TAILOR, a project funded by EU Horizon 2020 under GA No. 952215, https://doi.org/10.3030/952215.

Bibliography

[1] Chuklin, A.: Click Models for Web Search **7**(3), 1–115 (2015)

[2] Dhrymes, P.J.: Topics in advanced econometrics: Probability foundations, vol. 1. Springer Science & Business Media (1989)

[3] Dudík, M., Erhan, D., Langford, J., Li, L.: Doubly Robust Policy Evaluation and Optimization. Statistical Science **29**(4), 485–511 (Nov 2014), ISSN 0883-4237, 2168-8745, https://doi.org/10.1214/14-sts500, URL https://projecteuclid.org/journals/statistical-science/volume-29/issue-4/Doubly-Robust-Policy-Evaluation-and-Optimization/10.1214/14-STS500.full, 204 citations (Semantic Scholar/DOI) [2023-05-01] 50 citations (Crossref) [2023-05-01] Publisher: Institute of Mathematical Statistics

[4] Efron, B.: The efficiency of logistic regression compared to normal discriminant analysis. Journal of the American Statistical Association **70**(352), 892–898 (1975)

[5] Farajtabar, M., Chow, Y., Ghavamzadeh, M.: More Robust Doubly Robust Off-policy Evaluation. In: Proceedings of the 35th International Conference on Machine Learning, pp. 1447–1456, PMLR (Jul 2018), URL https://proceedings.mlr.press/v80/farajtabar18a.html, iSSN: 2640-3498

[6] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics, Springer New York, New York, NY (2009), ISBN 978-0-387-84857-0 978-0-387-84858-7, https://doi.org/10.1007/978-0-387-84858-7, URL http://link.springer.com/10.1007/978-0-387-84858-7

[7] Kallus, N., Zhou, A.: Policy Evaluation and Optimization with Continuous Treatments. In: Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, pp. 1243–1251, PMLR (Mar 2018), URL https://proceedings.mlr.press/v84/kallus18a.html, iSSN: 2640-3498

[8] Metelli, A.M., Russo, A., Restelli, M.: Subgaussian and Differentiable Importance Sampling for Off-Policy Evaluation and Learning. In: Advances in Neural Information Processing Systems, vol. 34, pp. 8119–8132, Curran Associates, Inc. (2021), URL https://proceedings.neurips.cc/paper/2021/hash/4476b929e30dd0c4e8bdbcc82c6ba23a-Abstract.html

[9] Peng, J., Zou, H., Liu, J., Li, S., Jiang, Y., Pei, J., Cui, P.: Offline Policy Evaluation in Large Action Spaces via Outcome-Oriented Action Grouping. In: Proceedings of the ACM Web Conference 2023, pp. 1220–1230, WWW '23, Association for Computing Machinery, New York, NY, USA (Apr 2023), ISBN 978-1-4503-9416-1, https://doi.org/10.1145/3543507.3583448, URL https://dl.acm.org/doi/10.1145/3543507.3583448

[10] Robins, J.M., Rotnitzky, A., Zhao, L.P.: Estimation of Regression Coefficients When Some Regressors are not Always Observed. Journal of the American Statistical Association **89**(427), 846–866 (Sep 1994), ISSN 0162-1459, https://doi.org/10.1080/01621459.1994.10476818, URL https:

//doi.org/10.1080/01621459.1994.10476818, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01621459.1994.10476818

[11] Sachdeva, N., Su, Y., Joachims, T.: Off-policy Bandits with Deficient Support. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 965–975, KDD '20, Association for Computing Machinery, New York, NY, USA (Aug 2020), ISBN 978-1-4503-7998-4, https://doi.org/10.1145/3394486.3403139, URL https://dl.acm.org/doi/10.1145/3394486.3403139, 42 citations (Semantic Scholar/DOI) [2023-05-01] 9 citations (Crossref) [2023-05-01]

[12] Saito, Y., Aihara, S., Matsutani, M., Narita, Y.: Open Bandit Dataset and Pipeline: Towards Realistic and Reproducible Off-Policy Evaluation (Oct 2021), https://doi.org/10.48550/arXiv.2008.07146, URL http://arxiv.org/abs/2008.07146, arXiv:2008.07146 [cs, stat]

[13] Saito, Y., Joachims, T.: Off-Policy Evaluation for Large Action Spaces via Embeddings. In: Proceedings of the 39th International Conference on Machine Learning, pp. 19089–19122, PMLR (Jun 2022), URL https://proceedings.mlr.press/v162/saito22a.html, iSSN: 2640-3498

[14] Saito, Y., Ren, Q., Joachims, T.: Off-Policy Evaluation for Large Action Spaces via Conjunct Effect Modeling. In: Proceedings of the 40th International Conference on Machine Learning, pp. 29734–29759, PMLR (Jul 2023), URL https://proceedings.mlr.press/v202/saito23b.html, iSSN: 2640-3498

[15] Su, Y., Dimakopoulou, M., Krishnamurthy, A., Dudik, M.: Doubly robust off-policy evaluation with shrinkage. In: Proceedings of the 37th International Conference on Machine Learning, pp. 9167–9176, PMLR (Nov 2020), URL https://proceedings.mlr.press/v119/su20a.html, iSSN: 2640-3498

[16] Su, Y., Wang, L., Santacatterina, M., Joachims, T.: CAB: Continuous Adaptive Blending for Policy Evaluation and Learning. In: Proceedings of the 36th International Conference on Machine Learning, pp. 6005–6014, PMLR (May 2019), URL https://proceedings.mlr.press/v97/su19a.html, iSSN: 2640-3498

[17] Swaminathan, A.: Counterfactual Evaluation And Learning From Logged User Feedback. Ph.D. thesis, Cornell University, Ithaca, NY, United States (May 2017), URL https://ecommons.cornell.edu/handle/1813/51557, accepted: 2017-07-07T12:48:28Z

[18] Swaminathan, A., Joachims, T.: The Self-Normalized Estimator for Counterfactual Learning. In: Advances in Neural Information Processing Systems, vol. 28, Curran Associates, Inc. (2015), URL https://proceedings.neurips.cc/paper/2015/hash/39027dfad5138c9ca0c474d71db915c3-Abstract.html

[19] Swaminathan, A., Krishnamurthy, A., Agarwal, A., Dudik, M., Langford, J., Jose, D., Zitouni, I.: Off-policy evaluation for slate recommendation. In: Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc. (2017), URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/5352696a9ca3397beb79f116f3a33991-Abstract.html

[20] Wang, Y.X., Agarwal, A., Dudik, M.: Optimal and Adaptive Off-policy Evaluation in Contextual Bandits. In: Proceedings of the 34th International

Conference on Machine Learning, pp. 3589–3597, PMLR (Jul 2017), URL https://proceedings.mlr.press/v70/wang17a.html, iSSN: 2640-3498

[21] Zhou, L.: A Survey on Contextual Multi-armed Bandits (Feb 2016), https://doi.org/10.48550/arXiv.1508.03326, URL http://arxiv.org/abs/1508.03326, arXiv:1508.03326 [cs]