A Scalable Probabilistic Model for Reward Optimizing Slate Recommendation

Imad Aouali *† i.aouali@criteo.com

Achraf Ait Sidi Hammou *† aitsidihammou.achraf@gmail.com

Sergey Ivanov † s.ivanov@criteo.com

Otmane Sakhi ‡ o.sakhi@criteo.com

David Rohde † d.rohde@criteo.com

Flavian Vasile † f.vasile@criteo.com

Abstract

We introduce **P**robabilistic **R**ank and **R**eward model (`PRR`), a scalable probabilistic model for personalized slate recommendation. Our model allows state-of-the-art estimation of user interests in the ubiquitous scenario where the user interacts with at most one item from a slate of K items. Our contribution is to show that we can learn more effectively the probability of the recommended slate being successful by combining the reward - whether the slate was clicked - and the rank - the item on the slate that was clicked. `PRR` learns more efficiently than bandit methods that use only the reward, and user preference methods that use only the rank. It also provides better estimation performance to independent inverse propensity score (IPS) methods and is far more scalable. Our method is state-of-the-art in terms of both speed and accuracy on massive datasets with up to 1 million items. Finally, our method allows fast delivery of recommendations powered by maximum inner product search (MIPS), making it suitable in extremely low latency domains such as computational advertising.

1 INTRODUCTION

Recommender systems are becoming ubiquitous in society helping users navigate enormous catalogs of items allowing them to identify items relevant to their interests. Real world recommender systems typically do not recommend a single item, but rather, provide a slate of K recommendations simultaneously. The recommendation usually takes the form of a menu and the user can choose to interact with at most one of the K items. A successful slate of recommendations is one in which the user interacted with one of the items.

This paper is concerned with developing a machine learning model that extracts information from historical recommendations, both successful and unsuccessful. Our model coined **P**robabilistic **R**ank and **R**eward model (`PRR`) uses both the reward signal, i.e. the slate received a click, and the rank signal, i.e. the item that was selected within the slate.

An example of the output of our model is shown in Section 1. In this situation, we imagine that a user is interested in *technology*. We show three candidate slates of size 2. In the top left panel, the slate consists of

*Equal contribution.
†Criteo AI Lab.
‡Criteo AI Lab / ENSAE CREST.

Figure 1: Example of 3 slates of size 2 on a technology website. From left to right (top to bottom) are good, mixed and bad recommendations. $\bar{c}, c_1, c_2$ denote the actual probabilities of no-click, click on the first item, and click on the second item, respectively.

two 'good' recommendations: *phone* and *microphone*. The model's labels [0.91 0.06 0.03] are the probabilities for no click, click on *phone* and click on *microphone* respectively. Clearly, the probability of a click on slate *phone, microphone* is higher than the other slates, and is equal to 0.09. For comparison, the top right panel contains a good recommendation with the *phone* in the prime first position but the *shoe* in second position, which is a poor match with the user's interest in technology. As a consequence, the probabilities become: [0.94 0.04 0.01] for no click, click on *phone* and click on *shoe*. Finally, in the bottom, we show two poor recommendations of *shoe* and *pillow* resulting in the lowest click probabilities [0.97 0.02 0.01].

The goal of the recommender-system learning algorithm is to establish how strong the association of each item (in this case *phone*, *microphone*, *shoe* and *pillow*) is with a particular context (in this case *technology*). Analyzing logs of successful or unsuccessful recommendations (the bandit signal) is the best possible way to learn this association. However, in a real world recommendation system, there is a great deal more that influences the probability of a click than just the quality of recommendations. In this example, we see that when the recommendation is very good, the non-click probability is 0.91 (click probability of 0.09). On the other hand, when the recommendation is very poor, the non-click probability is 0.97 (click probability of 0.03). The change in the click probability from poor to good recommendations is relatively modest at only 0.06.

To account for this, our model incorporates a real world observation: the most informative features for click prediction are *user engagement features*: e.g. how active is the user or how big is the recommendation slate. In contrast, the *recommendation features*, i.e. the items shown in the slate and the users' interest, provide a relatively modest signal. Similarly, our model incorporates the information that different positions in the slate may have different properties. Some positions may boost a recommendation by making it very visible, and other positions may lessen the impact of the recommendation.

It may be useful to draw an analogy. Optical astronomers who take images of far away galaxies need to develop a sophisticated understanding of many local phenomena: the atmosphere, the ambient light, the milky way and by understanding all these large effects they are able to take precise images of extremely faint far away objects. Similarly, `PRR` is able to build up a model of a relatively weak recommendation signal by carefully modeling all of the other factors that often have relatively large contributions to a user clicking.

It is in this light that we may compare our method with existing alternatives. Bandit approaches focus exclusively on the reward or click signal. This has a very profound advantage in that what is being optimized offline is aligned with the click-through rate observed at A/B test time. On the other hand, the selected item (rank) is ignored (in the basic approach). These methods also frequently employ the Horvitz-Thompson or Inverse Propensity Score (IPS) based estimation to deal with the fact that engagement features, rather than recommendations, are key predictors of clicks, yet we are most interested in estimating the recommendation contribution to clicks. Propensity scores are minimal balancing scores Rosenbaum & Rubin (1983) and as such can replace the high dimensional $\boldsymbol{X}$, but in doing this they violate the conditionality principle Robins & Ritov (1997).

User preference modeling using only greedy ranking Covington et al. (2016) is an alternative way to avoid dealing with the fact that the user engagement features are a key predictor of clicks. This perspective considers that if a *phone* is selected above a *shoe*, then *phone* is assumed to be of better quality (*phone* is preferred over *shoe*).

In this work, we focus on using likelihood-based methods for slate recommendation where a slate consists of K simultaneous recommended items i.e. $a_1, ..., a_K$. We introduce a model that assumes that the user interacts at most with one of the K items and a positive reward is obtained if an interaction occurs. This model corresponds to the common case where an interaction (e.g. a click) causes the slate to disappear in an interactive system. The information obtained after a slate is shown consists of two parts: *did the user interacted with one of the item?* and *if an item was interacted with, which item was it?* We refer to these two types of feedback as reward and rank, respectively.

An important advantage of our approach is that the task of recommending a slate of K recommendations is reduced to solving a MIPS for K items Abbasifard et al. (2014); Ding et al. (2019). While this constrains the parametric form of the proposed model, it makes the methodology applicable to massive scale slate recommendation tasks with very low latency requirements, e.g., as the computational advertising industry face. We note that this is different from the common approach where a small subset of potential candidates for recommendation is generated, and then a ranking model is used to select top items from that subset.

We show via extensive simulation that this model is able to perform very well with simulated A/B tests, outperforming commonly used baselines such as slate IPS Dudík et al. (2012); Bottou et al. (2013) and independent IPS Li et al. (2018). We argue that the first of these methods performs poorly because of a failure to borrow strength resulting in extremely high variance. While slate IPS methods make very few assumptions, they are not able to exploit similarities between slate $a_1, \ldots, a_K$ and slate $a'_1, \ldots, a'_K$, even very basic similarities such as having common recommendations are ignored. In contrast independent IPS performs poorly by making too strong and incorrect assumptions. Although these assumptions are slightly relaxed by the reward interaction IPS McInerney et al. (2020), they are unrealistic for many use cases.

Our modeling approach has some superficial similarities with Richardson et al. (2007) which builds a latent variable model that incorporates the position of the item into a click model. Our model differs from this approach in making the (usually realistic) assumption of at most one item being interacted with. However, similar to Richardson et al. (2007), we explicitly incorporate in our model position biases (both multiplicative and additive). The Bayesian probit approach also has similarities with Graepel et al. (2010), although it only applies to single item recommendations.

The rest of this paper is organized as follows. In Section 2, we describe the setting for slate recommendation and our proposed algorithm. In Section 3, we review related work to the slate recommendation problem. In Section 4, we describe popular baselines and present our qualitative and quantitative results on simulated A/B tests. In Section 5, we make concluding remarks and outline potential directions for the future work.

2 PROPOSED ALGORITHM

2.1 Setting

We consider the setting where the recommender system displays a slate of K recommendations and the user chooses at most one of the K recommended items, which causes the slate to disappear. We denote by c and $\bar{c} = 1 - c$ a 'reward' (e.g., click on the slate), and 'regret' (e.g., no click on the slate), respectively. We also denote by $r_1, ..., r_K$ the rank of the items within a slate of size K (e.g., r_k is equal to 1 if the user clicks on the item at position k in the slate and 0 otherwise). We also assume access to some user engagement features $\boldsymbol{X} \in \mathbb{R}^{d_1}$, with $d_1 > 0$ which are useful for predicting if an interaction with a slate will occur, independently of its items. These features typically include the slate size and how active the user is. It is important to note that these features do not provide any information about user interests, but they are good predictors of a click. Finally, we also assume access to features $\boldsymbol{\Omega}$ which represent users interests. Furthermore, items are referenced by integer indices, so that $\{1, 2, \ldots, P\}$ denotes the entire catalog that contains P items. Let $a = [a_1, \ldots, a_K]$ be a slate of size K, where $a_k, k = 1, \ldots, K$ is the index of the item placed at position k in the slate a. Given $\boldsymbol{A} \in \mathbb{R}^P$, a vector of scores of each item in the catalog, and $\boldsymbol{B} \in \mathbb{R}^{P \times d_2}$, a list of embeddings of each item in the catalog. $\boldsymbol{A}_{a_k}$ and $\boldsymbol{B}_{a_k}$ produce the score of item a_k and the embedding of item a_k, respectively. Section 2.1 summarizes the remaining variables as well as the ones we have already mentioned.

Notation	**Definition**
$\boldsymbol{X}$	User engagement features.
$\boldsymbol{\Omega}$	Context (user interests).
c	Binary flag that indicates interaction with the slate (reward).
$r_k, k = 1, \ldots, K$	Binary flag that indicates interaction with k-th item (rank).
$\boldsymbol{\phi}$	Engagement parameters.
$\gamma_k, k = 1, \ldots, K$	Multiplicative position biases.
$\alpha_k, k = 1, \ldots, K$	Additive position biases.
$g_\Gamma(\boldsymbol{\Omega})$	User embedding.
$\boldsymbol{\Psi}$	Items embeddings.
θ_0	Score for no-interaction with the slate.
$\theta_k, k = 1, \ldots, K$	Score for interaction k-th item.
$f_\Xi(\boldsymbol{\Omega})$	Decision rule for MIPS query creation.
$\boldsymbol{\beta}$	Decision rule item embeddings.
$\pi_0(a_1, \ldots, a_K \mid \Omega)$	Propensity score for the full slate $a_1, \ldots, a_K$.
$\pi_0(a \mid \Omega)$	Propensity score for a single item.

Table 1: Notations and Definitions

2.2 Modeling Rank and Reward

The proposed model has the following form:

$$\bar{c}, r_1, \ldots, r_K \mid \boldsymbol{\Omega}, \boldsymbol{X}, a_1, \ldots, a_K \sim \text{cat}\left(\frac{\theta_0}{Z}, \frac{\theta_1}{Z}, \ldots, \frac{\theta_K}{Z}\right), \quad Z = \sum_{k=0}^{K} \theta_k,$$

where $\bar{c}, r_1, \ldots, r_K, \boldsymbol{\Omega}$, and $\boldsymbol{X}$ are defined in Section 2.1, cat() is the categorical distribution, θ_0 is the score of no interaction with the slate and θ_k, for $k = 1, \ldots, K$ is the score that an interaction occurs with the item at position k in the slate. We discuss how to derive such scores in the sequel.

As highlighted previously, our model accounts for an important observation made by practitioners. It is often possible to produce a good model for predicting clicks on slates while discarding user interests and the items recommended to the user. Instead, features such as slate size, attractiveness and level of user activity can be strong predictors of a click. While a model using only these features might have excellent ability to predict clicks and thus high likelihood, it is useless for recommendation tasks. This observation is often used to justify abandoning likelihood based reward models for recommendation in favor of either ranking likelihood or IPS based methods. Instead, `PRR` uses both the user engagement features $\boldsymbol{X}$, user interests $\boldsymbol{\Omega}$ and recommendations to predict clicks accurately, enabling reward optimizing recommendation.

The user engagement features $\boldsymbol{X}$ are used to produce a positive score θ_0 which is high if the chance of no interaction is high:

$$\theta_0 = \exp(\boldsymbol{X}^T \boldsymbol{\phi}), \tag{1}$$

where $\boldsymbol{\phi}$ is a vector of learnable nuisance parameters of size $d_1 > 0$.

Similarly, the positive scores θ_k for $k = 1, \ldots, K$ associated with each of the K items in the slate are calculated in a way that captures user interests, position biases, and interactions that occur by 'accident'. Precisely, given a recommended slate $a_1, \ldots, a_K$, the score θ_k has the form:

$$\theta_k = \exp\{g_\Gamma(\boldsymbol{\Omega})^T \boldsymbol{\Psi}_{a_k}\} \exp(\gamma_k) + \exp(\alpha_k). \tag{2}$$

Again this formulation is motivated by practitioners experience. The quantity $\exp(\alpha_k), k = 1, \ldots, K$ denotes the additive bias for position k in the slate. It is high if there is a high chance of interaction with the $k-$th item irrespective of how appealing it is to the user. This quantity also explains clicks that are not associated at all with the recommendation (e.g., clicks by 'accident'). It follows that the probability of a click on slates is always larger than $\frac{\sum_{k=1}^{K} \exp(\alpha_k)}{Z}$. The quantity $\gamma_k, k = 1, \ldots, K$ is the multiplicative bias, it is high if a good recommendation in position k is 'boosted' by being in position k.

The main quantity of interest for recommendation is the recommendation score which is $g_\Gamma(\boldsymbol{\Omega})^T \boldsymbol{\Psi}_{a_k}$. This score can be understood as follows. The vector $\boldsymbol{\Omega}$ represents the user's interests and the vector $\boldsymbol{\Psi}_{a_k} \in \mathbb{R}^{d_2}$ represents the embedding of the recommended item a_k. The vector $\boldsymbol{\Omega}$ is first mapped into an embeddings space of dimension d_2 using $g_\Gamma(\cdot)$. The resulting inner product $g_\Gamma(\boldsymbol{\Omega})^T \boldsymbol{\Psi} a_k$ produces a positive or negative score that represents how good is a_k (recommended item in position k) to the user.

This model has the following properties:

- It combines the reward and the rank in an elegant and simple formulation. This is important as both signals shall contain useful information, and discarding one of these signals leads to inferior performance, as we show in our experiments.
- It makes use of non-recommendation relevant features $\boldsymbol{X}$ in order to help learn the recommendation signal more accurately. Practical experience with real world recommender systems can lead to the conclusion that the probability of a click on a slate may have relatively little to do with the quality of recommendations in the slate and more to do with its size and how active the user is. This means that it can be possible to improve the likelihood of the model without improving the recommendation performance. By using all the information that we have available we are able to improve performance.

We note that this methodology is more complicated in some respects to IPS based recommender systems because not only does it learns $\boldsymbol{\Gamma}, \boldsymbol{\Psi}$ - required for recommendation - but also the nuisance parameters $\boldsymbol{\phi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}$. However, this also gives performance increases and means we have a more faithful model available to us. Additionally in some situations these nuisance parameters may be interpretable.

2.3 Decision Making

It follows from Eqs. (1) and (2) that the probability of a click on the slate which we take to be reward as:

$$P(c|\boldsymbol{\Omega}, \boldsymbol{X}, a_1, ..., a_K) = \frac{\sum_{k=1}^{K} \theta_k}{Z} = 1 - \frac{\theta_0}{Z}.$$

Without loss of generality, we assume that $\gamma_1 > \gamma_2 > \ldots > \gamma_K$. We now select a slate of recommendations that maximize the probability of a click by placing the most relevant item into position 1 and the second most relevant item into position 2 and so on. In other words, recommendation is performed by doing a sort of the recommendation scores.

Sorting algorithms unfortunately have a computational complexity that is $\mathcal{O}(P \log P)$ which is unsuitable for large catalog sizes common to modern recommendation systems. Fortunately, fast approximate MIPS algorithms Abbasifard et al. (2014); Ding et al. (2019) are capable of quickly performing this task in $\mathcal{O}(\log P)$ as long as the recommendation score satisfies the following structure:

$$a_1, ..., a_K = \text{argsort}(f_\Xi(\boldsymbol{\Omega})^T \boldsymbol{\beta})_{1:K}.$$

Here $\boldsymbol{\Omega}$ is the representation of the user's interests, $f_\Xi(\boldsymbol{\Omega})$ is a deep network that maps $\boldsymbol{\Omega}$ to a vectorial query and $\boldsymbol{\beta}$ is an embedding matrix, containing an embedding for every item. The product $f_\Xi(\boldsymbol{\Omega})^T \boldsymbol{\beta}$ produces a vector of P scores. A suitably optimized $f_\Xi(\boldsymbol{\Omega})^T \boldsymbol{\beta}$ will return scores that sort the items by recommendation quantity for every user interest Ω.

One way to achieve a correct ordering of recommendations given by $\text{argsort}(f_{\Xi}(\mathbf{\Omega})^T\boldsymbol{\beta})_{1:K}$ is to accurately predict the click probability for every recommendation in a given banner. This is in fact what `PRR` does by learning click calibrated probabilities. We can then say $f_{\Xi}(\cdot) \leftarrow g_{\Gamma}(\cdot)$ and $\boldsymbol{\beta} \leftarrow \mathbf{\Psi}$ as per Algorithm 3 (See the Appendix).

The `PRR` model learns the recommendation scores $g_{\Gamma}(\mathbf{\Omega})^T\Psi$ along with the nuisance parameters $\boldsymbol{\phi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}$. The structure of the model is such that the score $g_{\Gamma}(\mathbf{\Omega})^T\Psi$ produces a vector of P scores, that can be calibrated to click through rates (CTRs). Producing CTRs by applying a monotonic transform on $g_{\Gamma}(\mathbf{\Omega})^T\Psi$ is sufficient, but not necessary to get a valid ordering of recommendations. On this basis, we can set the decision rule $f_{\Xi}(\mathbf{\Omega})^T\boldsymbol{\beta}$ to be equal to the model parameters $g_{\Gamma}(\mathbf{\Omega})^T\Psi$.

There are strengths and weaknesses to this approach. The requirement for the model parameters $g_{\Gamma}(\mathbf{\Omega})^T\Psi$ to accurately predict CTRs is more than we require for the decision rule $f_{\Xi}(\mathbf{\Omega})^T\boldsymbol{\beta}$ as we only need to correctly order the first K items. For this reason, more parameters may be required to successfully use a modeling approach because it carries the burden of correctly predicting clicks of all items.

Policy learning algorithms, to be discussed shortly, do not train a model and directly optimize the decision rule $f_{\Xi}(\mathbf{\Omega})^T\boldsymbol{\beta}$ allowing direct optimization of the task at hand. The price policy learning algorithms pay for this is that as they do not have access to a model of the reward and they suffer high variance.

3 RELATED WORK

Reward optimizing recommendation is a relatively new area of recommender systems research which aims to directly optimize the reward measured online by A/B test using offline logs. The earliest work in this field Dudík et al. (2012); Bottou et al. (2013) used small action spaces and the IPS estimator Horvitz & Thompson (1952). The attraction of this estimator is due to the fact that it is simple, unbiased and makes no parametric assumptions on the reward model. However, it can have very high variance, and this is particularly the case in recommender systems. In typical recommender systems, this problem is driven by three factors. First, the number of items available for recommendation is large. Second, the recommendation task often entails selecting not only one but multiple items, which leads to a combinatorially large action space. Finally, the training data usually contains training logs that primarily *exploited* certain recommendations with minimal *exploration*.

The high variance aspect of the IPS estimator in recommender systems is acknowledged and several relatively simple fixes have been proposed including capping IPS weights, self normalizing importance sampling and using a model via doubly robust estimation (see e.g. (Gilotte et al., 2018) for more details). Altering the estimator in this way has the practical impact of avoiding recommendations about which little is known (i.e. only in exceptional circumstances will the training log change the preferred recommendations) and the learned policy will be very close to the original logging policy. In the context of slate recommendation, recent works made simplifying structural assumptions that help reduce the variance. For instance, certain studies restricted the search space by, for example, assuming that items within the slate contribute to the reward individually (Li et al., 2018) or that slate-level reward is additive w.r.t some unobserved and independent item-level rewards (i.e., ranks) (Swaminathan et al., 2017). These assumptions are restrictive and can be violated in many production settings. It is unclear how we can avoid recommending slates with repeated items under policies using independence assumption. Also, the rank information is observed in many settings and it shall contain some useful information about users preference. A relaxed assumption was proposed in (McInerney et al., 2020) where the interaction between the user and the item at position k in the slate depends only on the items at that position and the item at position $k-1$ and its rank. This sequential dependence scheme is not sufficient to encode the at most one interaction which is the problem of interest to us.

Another popular family of methods for reward estimation that do not fully address the ubiquitous case of slate recommendation is the so-called *click models* Chuklin et al. (2015). Click models define dependencies between the user browsing events and estimate full or conditional probabilities of a click, based on a log of user interactions with a slate. The simplest models of this kind, called *click-through-rate models*, define a single parameter for a probability of an item being clicked, possibly depending on a position or a user

Joachims et al. (2017); Craswell et al. (2008). Another type of models is called *cascade models* Dupret & Piwowarski (2008); Guo et al. (2009); Chapelle & Zhang (2009), where a probability of an item being clicked depends not only on the position and the user but also on the interactions with other previous items. These models assume that items are presented in sequential order, which is not always available in real settings (e.g., ads can have arbitrary positions of items in a slate). Later, these models were extended to accommodate multiple user sessions Chuklin et al. (2013); Zhang et al. (2011), granularity levels Hu et al. (2011); Ashkan & Clarke (2012), and additional user signals Huang et al. (2012); Liu et al. (2014).

Traditional click models are represented as graphical models and as such define dependencies manually and are not always scalable to large catalogs. Also, these models were primarily designed for search engine retrieval and often do not accommodate for extra features available at the moment of slate selection. Neural network approaches have been recently proposed to obtain better expressive power and arbitrary dependencies. These models often represent user behaviors as hidden representations, i.e. embeddings, instead of binary events as in graphical models. The models differ in the types of the neural architecture Borisov et al. (2018), loss objectives Chen et al. (2020), and data representation Lin et al. (2021). However, these methods do not consider optimization for the whole slate and instead formulate single item recommendations or learning to rank problems. While our modeling approach makes use of neural networks, it is different from previous models as it accounts for both the reward and the ranks of the items within slates.

Some work on slate recommendation focuses on the idea that certain combinations of recommendation may be particularly virtuous (or not) (Ie et al., 2019; Jiang et al., 2019; Wilhelm et al., 2018; Zhao et al., 2018) in this case the value of a slate is not simply the sum of its constituent recommendations, interacting recommendation is beyond the scope of this work.

4 EXPERIMENTS

4.1 Simulation Based Evaluation

Currently, recommender systems are often evaluated using some Information Retrieval-inspired offline metrics like recall, precision or nDCG on real-world datasets. We argue that this evaluation method is sub-optimal in slate recommendation. First, there is a huge lack of publicly available slate recommendation datasets. In addition, minor modeling or engineering considerations can make a dataset unusable, e.g., when the model requires additional features that are not present in the available dataset. Even when the dataset is well-suited to our problem formulation, it has been shown that there can be a striking gap between offline evaluation metrics and A/B test results (see e.g. Section 5.1 in (Garcin et al., 2014)). While these offline evaluation techniques have clearly shown their value and have fostered research progress, they remain just proxies that don't necessarily reflect the actual performance of the models.

Another evaluation method that has been successful in academia is IPS based offline evaluation. As discussed before, IPS suffers high variance in this setting. In practice one can imagine two scenarios that are likely to happen. First, the target policy is very similar to the logging policies, so that recommendations almost entirely match logged data. In this case, the variance of the estimate of the target policy expected reward can be small, but recommendation improvements would be minor. Second, the target policy is far from the logging policies. In this case, it is likely that recommendations would be different from the ones recorded in data due to the large size of the search space. The resulting estimate would ignore almost all data points and would suffer high variance.

Simulation helps avoid the aforementioned problems since it provides a flexible framework that can easily match modeling assumptions and engineering considerations. In addition, simulators can mimic the actual sequential decision making process of recommender systems where an agent recommends an item/slate and receives a feedback that depends on the quality of the recommendation. Simulators also help avoid, in early candidates selection experiments, the costly A/B testing. That said, we have seen an increase interest in using simulation and semi-synthetic data in the context of slate recommendation (e.g. (McInerney et al., 2020; Swaminathan et al., 2017; Jiang et al., 2019)). For these reasons we opted for simulation based experiments.

4.2 Experimental Design

To compare the different baselines against our approach, we design a simulated A/B test protocol that takes a list of different recommender systems as inputs, and outputs for each one of them the click-through rate on a list of simulated impressions. The protocol is composed of the following steps. Firstly, we define the problem consisting of the true parameters, the logging policy and the dataset size. Precisely, a problem is defined by the oracle parameters $\{\boldsymbol{\phi}, \gamma, \alpha, g_\Gamma(\cdot), \boldsymbol{\Psi}, P(\boldsymbol{X}), P(\boldsymbol{\Omega}), P(K), N_{\text{train}}\}$, where $P(\boldsymbol{X}), P(\boldsymbol{\Omega}), P(K)$ are the distributions of user engagement features, context and slate sizes respectively. Given this oracle, we produce offline training logs $\{\mathcal{D}, \mathcal{P}\}$ using Algorithm 1. These logs will then be used to train recommender system algorithms. We consider two families of algorithms, one based upon explaining the data with maximum likelihood, and another based upon direct optimization of the decision rule. Finally we compare how the different algorithms perform in a simulated A/B test (Algorithm 2).

Algorithm 1: Simulated Logs

Input: Oracle parameters $\{\boldsymbol{\phi}, \gamma, \alpha, g_\Gamma(\cdot), \boldsymbol{\Psi}, P(\boldsymbol{X}), P(\boldsymbol{\Omega}), P(K), N_{\text{train}}\}$ and, Logging policy $\pi_0(a_1, \ldots, a_K|\Omega)$, Marginal logging policy $\pi(a_1|\Omega), \ldots, \pi(a_K|\Omega)$

Output: logs $\mathcal{D}$, propensities $\mathcal{P}$

$\mathcal{D} \leftarrow [\,]$

$\mathcal{P} \leftarrow [\,]$

for $n = 1, \ldots, N_{\text{test}}$ **do**

  $X_n \sim P(X), \quad \boldsymbol{\Omega}_n \sim P(\boldsymbol{\Omega}) \quad K_n \sim P(K)$

  $a_{n,1}, \ldots, a_{n,K} \sim \pi_0(a_1, \ldots, a_K|\Omega_n)$

  $\theta_0 \leftarrow \exp(\boldsymbol{X}_n^T \boldsymbol{\phi})$

  $\theta_k \leftarrow \exp(g_\Gamma(\boldsymbol{\Omega}_n)^T \boldsymbol{\Psi}_{a_k}) \exp(\gamma_k) + \exp(\alpha_K), \forall k, 1 \leq k \leq K$

  $\bar{c}_n, c_{n,1}, \ldots, c_{n,K} \sim cat(\frac{\theta_0}{Z}, \frac{\theta_1}{Z}, \ldots, \frac{\theta_K}{Z}) \quad Z = \sum_{k=0}^{K} \theta_k$

  $\mathcal{D}$.append($\{\bar{c}_n, c_{n,1}, \ldots, c_{n,K}, \boldsymbol{X}_n, \boldsymbol{\Omega}_n, a_{n,1}, \ldots, a_{n,K}\}$)

  $\mathcal{P}$.append($\{\pi_0(a_{n,1}, \ldots, a_{n,K}|\boldsymbol{\Omega}_n), \pi_0(a_{n,1}|\boldsymbol{\Omega}_b), \ldots, \pi_0(a_{n,K}|\boldsymbol{\Omega}_n)\}$)

end

Algorithm 2: Simulated A/B Test

Input: Oracle parameters $\{\boldsymbol{\phi}, \gamma, \alpha, g_\Gamma(\cdot), \boldsymbol{\Psi}, P(\boldsymbol{X}), P(\boldsymbol{\Omega}), P(K)\}$ and list of decision rule A $\{f_\Xi^A(\cdot), \boldsymbol{\beta}^A\}$ and decision rule B $\{f_\Xi^B(\cdot), \boldsymbol{\beta}^B\}$ and N_{test}

Output: Reward_A, Reward_B

$\text{Reward}_A \leftarrow [\,]$;

$\text{Reward}_B \leftarrow [\,]$;

for $n = 1, \ldots, N_{\text{test}}$ **do**

  for $d \in \{A, B\}$ **do**

    $X_n \sim P(X)$

    $\boldsymbol{\Omega}_n \sim P(\boldsymbol{\Omega})$

    $K_n \sim P(K)$

    $a_{n,1}, \ldots, a_{n,K} \leftarrow \text{argsort}\{f_\Xi^d(\boldsymbol{\Omega}_n)^T \boldsymbol{\beta}^d\}_{1:K}$

    $\theta_0 \leftarrow \exp(\boldsymbol{X}_n^T \boldsymbol{\phi})$

    $\theta_k \leftarrow \exp(g_\Gamma(\boldsymbol{\Omega}_n)^T \boldsymbol{\Psi}_{a_k}) \exp(\gamma_k) + \exp(\alpha_K), \forall k, 1 \leq k \leq K$

    $Z \leftarrow \sum_{k=0}^{K} \theta_k$

    Reward_d.append($1 - \frac{\theta_0}{Z}$)

  end

end

4.3 Baselines

4.3.1 Slate IPS

IPS-based methods are often proposed as a way to estimate the reward in recommender systems. Unlike the modeling approach that we propose, they do not have a parametric form. In order to allow fast recommendation at massive scale, we adopt the following parametric form that applies to a single (non-slate) recommendation:

$$\pi_{\Xi,\beta}(a|\mathbf{\Omega}) = \frac{\exp\{f_{\Xi}(\mathbf{\Omega})^T \beta_a\}}{\sum_{a'} \exp\{f_{\Xi}(\mathbf{\Omega})^T \beta_{a'}\}}. \tag{3}$$

This is simply the most flexible model that conforms to the *Policy Optimization for Exponential Models* Swaminathan & Joachims (2015) and enables MIPS. In order to extend Eq. (3) to slate recommendation, a simple approach is to make an independence assumption:

$$\pi_{\Xi,\beta,K}(a_1, \dots, a_K|\mathbf{\Omega}) = \prod_{k=1}^{K} \pi_{\Xi,\beta}(a_k|\mathbf{\Omega}).$$

While this is convenient, there is a significant limitation in this assumption as it allows, and in some circumstances will encourage, selecting slates with repeated items. The expected number of clicks can be estimated using:

$$E[c|\mathbf{\Xi}, \boldsymbol{\beta}] \approx \sum_{n=0}^{N} \frac{c_n \pi_{\Xi,\beta,K}(a_1, \dots, a_K|\mathbf{\Omega})}{\pi_0(a_1, \dots, a_K|\mathbf{\Omega})}$$

where N is the number of data points and $\pi_0(a_1, \dots, a_K|\mathbf{\Omega})$ is the propensity score of the slate. We then optimize $\pi_{\Xi,\beta,K}(\cdot)$ with respect to Ξ, β. We refer to this as the 'Slate IPS baseline'. It optimizes against a low bias but very high variance reward estimator.

4.3.2 Independent IPS (IIPS)

The 'independent IPS' baseline substitutes variance for bias and uses the rank r as an item-level reward:

$$E[c|f_{\Xi}(\cdot), \beta] \approx \sum_{n=0}^{N} \sum_{k=1}^{K} \frac{r_{k,n} \pi_{\Xi,\beta}(a_k|\mathbf{\Omega})}{\pi_0(a_k|\mathbf{\Omega})}$$

This formulation is based on three dubious assumptions, these are $E[c|a_1, \dots, a_K|X] = \sum_{k=1}^{K} E[r_k|a_k|X]$, which is incorrect when the slate is constrained to have at most one item clicked (this causes a dependency and covariance in $r_1, \dots, r_K$). Also as previously stated an independence assumption is usually wrong for both $\pi_k(a_1, \dots, a_K|\mathbf{\Omega})$ and $\pi_0(a_1, \dots, a_K|\mathbf{\Omega})$ because slates with repeated items are forbidden. In our experiments, we ignore the former and in order to deal with the latter we approximate by letting $\pi_0(a_k|\mathbf{\Omega})$ be the appropriate marginal.

4.3.3 Top-K IPS

Any slate recommendation task that is based upon a policy applied to single item recommendations i.e. $\pi_k(a_k|\mathbf{\Omega})$ is at risk of converging to a policy that creates slates with repeated items. In order to mitigate this effect, we also consider using the Top-K heuristic developed in Chen et al. (2019) which causes the probability mass in $\pi_k(a_k|\mathbf{\Omega})$ to be spread out over the top K recommended items. The Top-K heuristic contains a parameter K^* which controls how many items the policy should spread over. We set it to 3 on all of our experiments, which we found to be a good setting for the relatively small slate sizes we were considering.

As well as IPS baselines we also investigate simplified likelihood models of the following forms:

4.3.4 Reward Model

$$\bar{c}, c|\boldsymbol{\Omega}, \boldsymbol{X}, a_1, \ldots, a_K \sim \text{cat}\left(\frac{\theta_0}{Z}, \sum_{k=1}^{K} \frac{\theta_k}{Z}\right), \quad Z = \sum_{k=0}^{K} \theta_k.$$

4.3.5 Ranking Model

$$r_1, \ldots, r_K|\boldsymbol{\Omega}, a_1, \ldots, a_K \sim \text{cat}\left(\frac{\theta_1}{Z}, \ldots, \frac{\theta_K}{Z}\right), \quad Z = \sum_{k=1}^{K} \theta_k.$$

4.3.6 `PRR` Bias Only

$$\bar{c}, r_1, \ldots, r_K|\boldsymbol{\Omega}, a_1, \ldots, a_K \sim \text{cat}\left(\frac{\theta_0}{Z}, \frac{\theta_1}{Z}, \ldots, \frac{\theta_K}{Z}\right), \quad Z = \sum_{k=0}^{K} \theta_k.$$

but now $\theta_0 = \phi$ is a scalar that does not depend on $\boldsymbol{X}$.

4.4 Toy Example

To help visualizing the results, we start with a toy example using a small set of context categories and products:

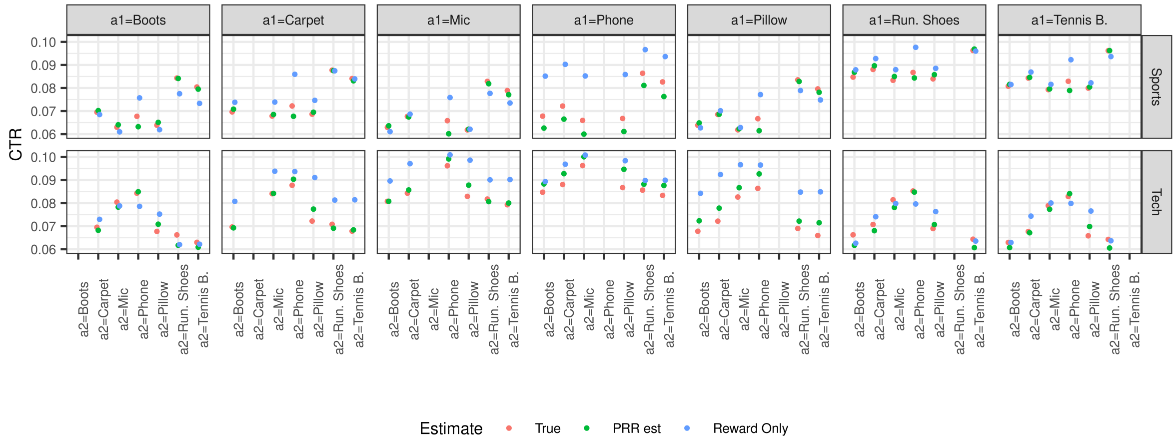
- products: *running shoes, phone, boots, microphone, carpet, pillow, tennis balls*
- context categories: *sports, tech, house, fashion*

For each context category, we set a strict ordering of the best recommendations, e.g., for *sports*, the best recommendations are (in descending order): *running shoes, tennis balls, carpet, phone, boots, pillow, microphone*.

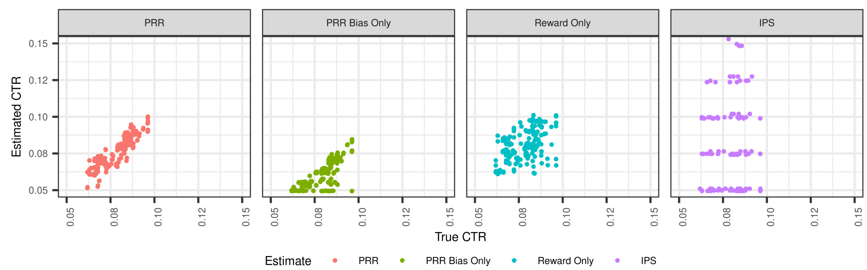
Using Algorithm 1, we produced training logs. We generated 20,000 records with slates uniformly of size 2, 3 and 4 and user engagement features of dimension 2 and $\boldsymbol{X} \sim \mathcal{U}[0, 1]$. All algorithms were trained for 300 epochs using Adam with a learning rate of 0.005 and a batch size of 1024. The user had precisely one of the four possible context/user interest ($\boldsymbol{\Omega}$) drawn uniformly.

In this toy example, we focus on analysing slates of size 2 (even though the training log contained varying slate sizes). A key construct of interest is the probability of a click in a particular context irrespective of the user engagement features that is $\int P(c|a_1, a_2, \boldsymbol{\Omega}, \boldsymbol{X})P(\boldsymbol{X})d\boldsymbol{X}$. We approximate this integral by Monte Carlo over 600 realizations over $\boldsymbol{X}$. The results are shown in Fig. 2a. To avoid cluttering the diagram we only show two contexts and the reward only estimator. For the model based algorithms, the recommender system will simply select for a given context the pair of a_1, a_2 with the highest click probability. Estimation error can lead to the highest estimated click probability not having the highest actual click probability which is also shown and marked “True”. Indeed, this occurs in the *Sports* category where the *Phone* item is significantly over-estimated by the reward only estimator. This would result in the reward only method recommending: *Running Shoes, Phone* on the *Sports* category, resulting in a reduced click-through rate. In contrast, `PRR` produces a superior estimate.

The high accuracy of the (`PRR`) model compared with the alternatives is further demonstrated in Fig. 2b. The IPS based estimator is notably worse than other alternatives because it cannot benefit from borrowing



(a) Estimated CTRs for slates of size 2, approximation of $\int P(c|a_1, a_2, \mathbf{\Omega}, \boldsymbol{X})P(\boldsymbol{X})d\boldsymbol{X}$..



(b) Estimated CTRs compared with ground truth.

strength between slates of different size, it also ignores the rank signal. The independent IPS algorithm performs much better, but doesn't estimate full slate click probabilities but rather maps full slate recommendation back to a single item recommendation.

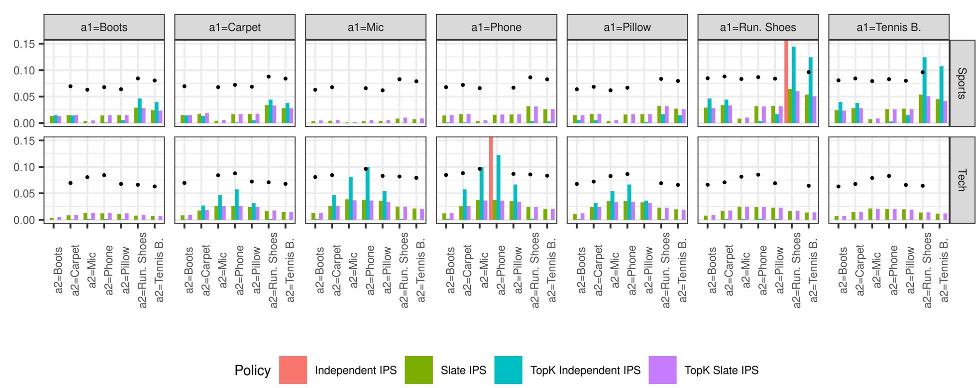


Figure 3: Policies for slates of size 2. Black dots represent true CTRs.

Finally we analyze the policy based methods in Fig. 3. Here the bars represent the probability mass for each of the four policy algorithms place on the actions for each context (y-axis is truncated by 0.15). The black dot represents the oracle CTR and the orange dot represents the Slate IPS CTR estimate. Independent IPS does not estimate slate level CTR and is not shown.

Policy learning involves optimizing to find the best decision, and the selected recommendation for each policy is the highest non-duplicate recommendation (i.e. when $a_1 \neq a_2$).

Because of the independence assumption in the policies, all policies place significant probability mass on duplicate actions. This is especially so for the independent IPS algorithm which places almost probability 1 on duplicate actions for every given context. For example when the context is *House* it places almost all the mass on a_1 =pillow, a_2 =pillow. Indeed *Pillow* would be a good recommendation for *Home* if the slate was of size 1. The independent IPS algorithm in fact is not a slate recommendation algorithm and only discovers a single good recommendation for each context. The Top-K correction here offers significant value as it distributes the probability mass among all the other actions.

The Slate IPS policy does not benefit so much from Top-K. Due to independence, it is not possible to place all the mass on the best action. For example in the case of the *Sports* context the best recommendations are *Tennis balls* and *Running shoes*. If a single action policy places a probability of 0.5 on each of *Tennis balls* and *Running shoes*, then there would be a 0.25 probability of (illegally) recommending *Tennis balls* twice and 0.25 probability of recommending *Running shoes* twice, both of which receive zero reward. The fact that duplicate recommendations are penalized naturally encourages diversification of an independent policy.

Algorithm	**CTR**
Oracle	$8.58 \pm 0.2\%$
Random	$7.22 \pm 0.18\%$
`PRR`	$8.56 \pm 0.2\%$
`PRR` (Bias only)	$8.54 \pm 0.2\%$
Ranking Model	$8.53 \pm 0.2\%$
Reward Model	$7.68 \pm 0.19\%$
Slate IPS	$7.93 \pm 0.19\%$
Top-K SlateIPS	$7.93 \pm 0.19\%$
IIPS	$8.52 \pm 0.2\%$
Top-K IIPS	$8.56 \pm 0.2\%$

Table 2: Simulated A/B test for Toy Problem

Finally we used Algorithm 2. to run a simulated A/B test with the results shown in Table 2. This shows that the `PRR` model performs very well, although Top-K Independent IPS is also very competitive. This is likely due to the fact that the bias variance trade off made by making the (false) independence assumption is reasonably advantageous in this situation.

4.5 Algorithm Performance on Varying Slate Size and Catalog Size

Algorithm	**K=2**	**K=6**	**K=18**
Oracle	$72.62 \pm 0.45\%$	$64.43 \pm 0.48\%$	$27.08 \pm 0.44\%$
Random	$3.39 \pm 0.18\%$	$7.72 \pm 0.27\%$	$5.24 \pm 0.22\%$
`PRR`	$\mathbf{50.47 \pm 0.5\%}$	$\mathbf{61.70 \pm 0.49\%}$	$\mathbf{26.7 \pm 0.44\%}$
`PRR` (Bias Only)	$36.0 \pm 0.48\%$	$60.12 \pm 0.49\%$	$25.44 \pm 0.44\%$
Ranking Model	35.79 ± 0.48 %	$48.88 \pm 0.5\%$	$\mathbf{26.31 \pm 0.44\%}$
Reward Model	$42.04 \pm 0.49\%$	$22.28 \pm 0.42\%$	$5.43 \pm 0.23\%$
Slate IPS	$20.02 \pm 0.4\%$	$26.67 \pm 0.44\%$	$3.31 \pm 0.18\%$
Top-K Slate IPS	$20.02 \pm 0.4\%$	$17.84 \pm 0.38\%$	$3.31 \pm 0.18\%$
IIPS	$40.42 \pm 0.49\%$	$48.91 \pm 0.5\%$	$25.36 \pm 0.44\%$
Top-K IIPS	$40.86 \pm 0.49\%$	$52.77 \pm 0.5\%$	$25.6 \pm 0.44\%$

Table 3: Simulated A/B test with varying slate size. $P = 1k$, $N_{\text{Train}} = 10k$

We then move on a more realistic setting with much bigger catalogs and contexts. In all the following experiments, 4 user engagement features $\boldsymbol{X} \sim \mathcal{U}[0, 1]$. Finally 20 different discrete contexts ($\boldsymbol{\Omega}$) were used.

We first investigate the impact of slate sizes on the different baselines. In Table 3, we used a fixed catalog of 10,000 products as well as 10,000 training samples. The reward only model and Slate IPS are greatly suffering from growing slate sizes. On the other hand, the rank only model benefits from bigger slates as it leads to displaying more product comparisons. Independent IPS still performs decently across all the different slate sizes as opposed to the Slate IPS. Once again, the addition of Top-K is improving performance by spreading the mass over different products, making IIPS not only focus on retrieving one but several best recommendations. Finally, PRR performs the best on all the different slate sizes.

Algorithm	**P=1k**	**P=5k**	**P=10k**
Oracle	$49.21 \pm 0.50\%$	$76.94 \pm 0.42\%$	$78.01 \pm 0.41\%$
Random	$3.65 \pm 0.19\%$	$7.0 \pm 0.26\%$	$5.68 \pm 0.23\%$
PRR	$\mathbf{42.63 \pm 0.49\%}$	$\mathbf{58.01 \pm 0.49\%}$	$\mathbf{57.18 \pm 0.49\%}$
PRR (Bias only)	$\mathbf{42.88 \pm 0.49\%}$	$49.98 \pm 0.5\%$	$44.31 \pm 0.5\%$
Ranking Model	$26.15 \pm 0.44\%$	$50.57 \pm 0.5\%$	$49.72 \pm 0.5\%$
Reward Model	$2.83 \pm 0.17\%$	$38.35 \pm 0.49\%$	$19.14 \pm 0.39\%$
Slate IPS	$20.93 \pm 0.41\%$	$29.19 \pm 0.45\%$	$18.01 \pm 0.38\%$
Top-K Slate IPS	$27.23 \pm 0.45\%$	$25.38 \pm 0.44\%$	$15.76 \pm 0.36\%$
IIPS	$36.56 \pm 0.48\%$	$51.47 \pm 0.5\%$	$34.85 \pm 0.48\%$
Top-K IIPS	$\mathbf{43.01 \pm 0.5\%}$	$53.96 \pm 0.5\%$	$37.24 \pm 0.48\%$

Table 4: Simulated A/B test with varying catalog size. $K = 5$, $N_{\text{Train}} = 10k$

We then consider different catalog sizes in Table 4 where all slates are of fixed size 5, and the baselines are once again trained with 10,000 training samples. With a small catalog (1,000 products) both PRR and Top-K IIPS are performing extremely well. We notice a major performance boost from the addition of Top-K on the IIPS method, increasing from 36.56% to 43.01% CTR. However, as the catalog size grows, the (relative) performance of Top-K IIPS decreases. With bigger catalogs, the rank only model is better than all IPS methods, and PRR still performs better than all the other baselines. Note that scalability with regards to the catalog size is a crucial feature of a recommender system as they often operate on catalogs of size one million or more in production.

4.6 Speed Benchmarking

The speed of the algorithms we consider are driven by two main considerations. The PRR model and simplified variants all compute a softmax over the slate incurring a per iteration cost of $\mathcal{O}(K)$. In contrast, policy learning algorithms compute a softmax over the usually much larger catalog size, incurring a higher $\mathcal{P}$ cost per iteration. The second consideration is whether the algorithm uses clicked-banners only. If clicks are only 10% of the data, then this produces a 10× speed up. The fastest of all algorithms is the ranking model as it is both $\mathcal{O}(K)$ and trains on clicked-banners only.

When the catalog size becomes relatively large, the $\mathcal{O}(K)$ cost becomes prohibitive. With one million items PRR has far higher CTR than the competitors and is almost 20 times faster to train than IPS methods. In order to highlight the scalability of PRR, we assess its performance and speed on a dataset with 1 million actions. The computational complexity of the algorithms is shown in Table 5. When the data set size is large the catalog size (P) becomes the dominant issue in training the algorithm as demonstrated in Table 6. In the one million item scenario we see that PRR has a fast training time and is a clear winner in terms of performance, significantly beating Top-K IIPS and the Ranking Model.

Method	**Iterate Speed**	**Statistical efficiency**	**Uses non-clicks**
`PRR`	$\mathcal{O}(K)$	High	Yes
Reward only	$\mathcal{O}(K)$	Medium	Yes
Ranking	$\mathcal{O}(K)$	Medium	No
Slate IPS	$\mathcal{O}(P)$	Low	No
IIPS	$\mathcal{O}(P)$	Medium-High	No
Top-K IIPS	$\mathcal{O}(P)$	Medium-High	No
Top-K Slate IPS	$\mathcal{O}(P)$	Low	No

Table 5: Comparison of slate recommendation methods.

Algorithm	**CTR**	**Time/epoch (s)**
Oracle	$94.84 \pm 0.31\%$	-
Random	$6.25 \pm 0.34\%$	-
`PRR`	**67.19 ± 0.66%**	4.36
`PRR` (Bias only)	$46.87 \pm 0.71\%$	4.36
Ranking Model	$52.13 \pm 0.71\%$	**2.81**
Reward Model	$31.37 \pm 0.66\%$	4.37
Slate IPS	$3.11 \pm 0.25\%$	68.38
Top-K SlateIPS	$3.11 \pm 0.25\%$	69.81
IIPS	$42.22 \pm 0.7\%$	68.60
Top-K IIPS	$48.58 \pm 0.71\%$	69.56

Table 6: Simulated A/B test for 1M products. $N_{\text{Train}} = 10k$, $K = 5$, batch size 516.

5 CONCLUSION

In this paper, we presented `PRR`, a scalable probabilistic model for personalized slate recommendation. This approach provides a fully calibrated ordering of items by combining reward and rank signal to produce calibrated CTRs. Experiments using a simulated A/B test protocol, validate the benefit of combining rank and reward, as the `PRR` model outperforms reward-estimation baselines that use the Horvitz-Thompson estimator, including IPS, Independent IPS - with and without the Top-K heuristic. This is even more true in big-catalog, high-data situations, which match real scenarios. Not only does our approach gets a higher click-through rate, but it also can be trained significantly faster than policy-based methods with growing catalogs. There is a cost however, for predicting click-through rates everywhere in a calibrated manner, i.e the `PRR` model requires a richer parameterization than a policy-based method that only estimates the top K items. A possible path for getting the best of both worlds, is to optimize a policy method with respect to a reward model, rather than the Horvitz-Thompson estimator.

References

Mohammad Reza Abbasifard, Bijan Ghahremani, and Hassan Naderi. A survey on nearest neighbor search methods. *International Journal of Computer Applications*, 95(25), 2014.

Azin Ashkan and Charles LA Clarke. Modeling browsing behavior for click analysis in sponsored search. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 2015–2019, 2012.

Alexey Borisov, Martijn Wardenaar, Ilya Markov, and Maarten de Rijke. A click sequence model for web search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 45–54, 2018.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14(11), 2013.

Olivier Chapelle and Ya Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*, pp. 1–10, 2009.

Jia Chen, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. A context-aware click model for web search. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 88–96, 2020.

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

Aleksandr Chuklin, Pavel Serdyukov, and Maarten De Rijke. Modeling clicks beyond the first result page. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pp. 1217–1220, 2013.

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

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pp. 191–198, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340359. doi: 10.1145/2959100.2959190. URL https://doi.org/10.1145/2959100.2959190.

Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*, pp. 87–94, 2008.

Qin Ding, Hsiang-Fu Yu, and Cho-Jui Hsieh. A fast sampling algorithm for maximum inner product search. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3004–3012. PMLR, 2019.

Miroslav Dudík, Dumitru Erhan, John Langford, and Lihong Li. Sample-efficient nonstationary policy evaluation for contextual bandits. *arXiv preprint arXiv:1210.4862*, 2012.

Georges E. Dupret and Benjamin Piwowarski. A user browsing model to predict search engine click data from past observations. SIGIR '08, 2008.

F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber. Offline and Online Evaluation of News Recommender Systems at Swissinfo.Ch. In *Proc. of the 8th ACM Conference on Recommender Systems*, RecSys '14, pp. 169–176, 2014.

Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. Offline a/b testing for recommender systems. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, Feb 2018. doi: 10.1145/3159652.3159687. URL http://dx.doi.org/10.1145/3159652.3159687.

Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. Omnipress, 2010.

Fan Guo, Chao Liu, and Yi Min Wang. Efficient multiple-click models in web search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, 2009.

Daniel G Horvitz and Donovan J Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

Botao Hu, Yuchen Zhang, Weizhu Chen, Gang Wang, and Qiang Yang. Characterizing search intent diversity into click models. In *Proceedings of the 20th international conference on World wide web*, pp. 17–26, 2011.

Jeff Huang, Ryen W White, Georg Buscher, and Kuansan Wang. Improving searcher models using mouse cursor activity. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pp. 195–204, 2012.

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. Slateq: A tractable decomposition for reinforcement learning with recommendation sets. 2019.

Ray Jiang, Sven Gowal, Timothy A. Mann, and Danilo J. Rezende. Beyond greedy ranking: Slate optimization via list-cvae, 2019.

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *ACM SIGIR Forum*, volume 51, pp. 4–11. Acm New York, NY, USA, 2017.

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

Jianghao Lin, Weiwen Liu, Xinyi Dai, Weinan Zhang, Shuai Li, Ruiming Tang, Xiuqiang He, Jianye Hao, and Yong Yu. A graph-enhanced click model for web search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1259–1268, 2021.

Yiqun Liu, Chao Wang, Ke Zhou, Jianyun Nie, Min Zhang, and Shaoping Ma. From skimming to reading: A two-stage examination model for web search. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 849–858, 2014.

James McInerney, Brian Brost, Praveen Chandar, Rishabh Mehrotra, and Benjamin Carterette. Counterfactual evaluation of slate recommendations with sequential reward interactions. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul 2020. doi: 10.1145/3394486.3403229. URL http://dx.doi.org/10.1145/3394486.3403229.

Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pp. 521–530, 2007.

James M Robins and Ya'acov Ritov. Toward a curse of dimensionality appropriate (coda) asymptotic theory for semi-parametric models. *Statistics in medicine*, 16(3):285–319, 1997.

Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

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

Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miroslav Dudík, John Langford, Damien Jose, and Imed Zitouni. Off-policy evaluation for slate recommendation, 2017.

Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H. Chi, and Jennifer Gillenwater. Practical diversified recommendations on youtube with determinantal point processes. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pp. 2165–2173, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360142. doi: 10.1145/3269206.3272018. URL https://doi.org/10.1145/3269206.3272018.

Yuchen Zhang, Weizhu Chen, Dong Wang, and Qiang Yang. User-click modeling for understanding and predicting search-behavior. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1388–1396, 2011.

Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. *Proceedings of the 12th ACM Conference on Recommender Systems*, Sep 2018. doi: 10.1145/3240323.3240374. URL http://dx.doi.org/10.1145/3240323.3240374.

A Likelihood and Utility Functions

The Different optimization problems defined by the model based approaches (Algorithm 3) and the policy based approaches (Algorithm 4) are defined below.

A.1 Model based approaches

Algorithm 3: Model Based RecSys Algorithm

Input: logs $\mathcal{D}$
Output: $f_{\Xi}(\cdot), \boldsymbol{\beta}$
$\hat{\boldsymbol{\Gamma}}, \hat{\boldsymbol{\Psi}} = \text{argmax}_{\Xi,\boldsymbol{\beta}} \mathcal{L}(\mathcal{D}; \boldsymbol{\Xi}, \boldsymbol{\beta})$
$\boldsymbol{\Xi} \leftarrow \hat{\boldsymbol{\Gamma}}$
$\boldsymbol{\beta} \leftarrow \hat{\boldsymbol{\Psi}}$

The model based approaches are trained using the appropriate likelihood function:

$$\mathcal{L}_{\text{Full}}(\mathcal{D}|\boldsymbol{\Gamma}, \boldsymbol{\Psi}) = \sum_n [\bar{c}_n, r_{n,1}, ..., r_{n,K}] \log \boldsymbol{\theta} - \log(\sum_{k=0}^{K} \boldsymbol{\theta}_k),$$

$$\mathcal{L}_{\text{Reward}}(\mathcal{D}|\boldsymbol{\Gamma}, \boldsymbol{\Psi}) = \sum_n [\bar{c}_n, c_n] \log([\boldsymbol{\theta}_0, \sum_k \boldsymbol{\theta}_k]^T) - \log(\sum_{k=0}^{K} \boldsymbol{\theta}_k),$$

and

$$\mathcal{L}_{\text{Rank}}(\mathcal{D}|\boldsymbol{\Gamma}, \boldsymbol{\Psi}) = \sum_n [r_{n,1}, ..., r_{n,K}] \log \boldsymbol{\theta}_{1:K} - \log(\sum_{k=1}^{K} \boldsymbol{\theta}_k).$$

A.2 Policy based approaches

Algorithm 4: Policy Based RecSys Algorithm

Input: logs $\mathcal{D}, \mathcal{P}$
Output: $f_{\Xi}(\cdot), \boldsymbol{\beta}$
$\boldsymbol{\Xi}, \boldsymbol{\beta} = \text{argmax}_{\Xi,\boldsymbol{\beta}} \mathcal{U}(\mathcal{D}, \mathcal{P}; \boldsymbol{\Xi}, \boldsymbol{\beta})$

The policy based approaches are trained with the following utility functions:

$$\mathcal{U}_{\text{IPS}}(\mathcal{D}, \mathcal{P}; \boldsymbol{\Xi}, \boldsymbol{\beta}) = \sum_n \frac{c_n \prod_k \pi(a_k|\Omega)}{\pi_0(a_1, ..., a_K|\Omega)},$$

$$\mathcal{U}_{\text{IPS Top-K}}(\mathcal{D}, \mathcal{P}; \boldsymbol{\Xi}, \boldsymbol{\beta}) = \sum_n \frac{c_n \prod_k^K [1 - \{1 - \pi(a_{n,k}|\Omega)\}^{K^*}]}{\pi_0(a_1, ..., a_K|\Omega)},$$

$$\mathcal{U}_{\mathrm{IIPS}}(\mathcal{D},\mathcal{P};\boldsymbol{\Xi},\boldsymbol{\beta}) = \sum_n \sum_k r_{k,n} \frac{\pi(a_{n,k}|\boldsymbol{\Omega})}{\pi_0(a_{n,k}|\boldsymbol{\Omega}_n)},$$

and

$$\begin{aligned}&\mathcal{U}_{\mathrm{IIPS\ Top-K}}(\mathcal{D},\mathcal{P};\boldsymbol{\Xi},\boldsymbol{\beta})\\&= \sum_n \sum_k r_{k,n} \frac{1-\{1-\pi(a_{n,k}|\Omega)\}^{K^*}}{\pi_0(a_{n,k}|\boldsymbol{\Omega}_n)}.\end{aligned}$$