Representation Learning in Low-rank Slate-based Recommender Systems

Yijia Dai [1] **Wen Sun** [1]

Abstract

Reinforcement learning (RL) in recommendation systems offers the potential to optimize recommendations for long-term user engagement. However, the environment often involves large state and action spaces, which makes it hard to efficiently learn and explore. In this work, we propose a sample-efficient representation learning algorithm, using the standard slate recommendation setup, to treat this as an online RL problem with low-rank Markov decision processes (MDPs). We also construct the recommender simulation environment with the proposed setup and sampling method.

1. Introduction

Recommender systems surface personalized content based on learned models of user preferences, as in collaborative filtering (Breese et al., 2013; Konstan et al., 1997; Srebro et al., 2004; Mnih & Salakhutdinov, 2007) and content-based filtering (Van Meteren & Van Someren, 2000), or predictive models of user responses to specific recommendations (Ricci et al., 2010). A good recommender increases user engagement, and keeps the user interacting within the system. Popular platforms, like Youtube (Covington et al., 2016), Spotify (Jacobson et al., 2016), and Netflix (Gomez-Uribe & Hunt, 2015), make extensive use of recommender systems. As a user interacts with the system for a longer term, it becomes necessary to keep track of the user dynamics. Traditional methods focus on myopic predictions, where they estimate a user's immediate response. Current research has increased to narrate the problem as a Markov decision process (Rendle et al., 2010; He & McAuley, 2016), and do long-term planning using reinforcement learning algorithms (Shani et al., 2005; Gauci et al., 2018). However, using RL methods in recommender systems faces an issue regarding the large observation and action space, and doing efficient exploration becomes a harder question. Prior RL methods in recommender systems often overlook exploration, or use ϵ-greedy and Boltzmann exploration (Afsar et al., 2022).

In this work, we do representation learning under the low-rank MDP assumption. More specifically, we focus on the case where a user can be learned using a history of observations into a low-dimension representation. And the user dynamics can be modeled as transitions of such representation. Concretely, a low-rank MDP assumes that the MDP transition matrix admits a low-rank factorization, i.e., there exists two unknown mappings $\mu(s')$, $\phi(s,a)$, such that $P(s'|s,a) = \mu(s')^\top \phi(s,a)$ for all s, a, s', where $P(s'|s,a)$ is the probability of transiting to the next state s' under the current state and action (s,a). The representation $\phi(s,a)$ in a low-rank MDP not only linearizes the optimal state-action value function of the MDP (Jin et al., 2020), but also linearizes the transition operator. Such low-rankness assumption has been shown to be realistic in movie recommender systems (Koren et al., 2009).

Our main contribution is using upper confidence bound (UCB) driven representation learning to efficiently explore in user representation space. It is a practical extension from REP-UCB, which provides theorical guarantee on efficient exploration under low-rank MDP with sample complexity $O(d^4|\mathcal{A}|^2/(\epsilon^2(1-\gamma)^5))$ (Uehara et al., 2021). We focus on the case where action is slate recommendation. And under the mild assumption of user choice behavior which we will elaborate in Section 2, the combinatorial action space $O(|\mathcal{A}|)$ can be reduced to $O(k|\mathcal{I}|)$ where k is the slate size and $\mathcal{I}$ is the item space.

To evaluate our method systematically, we introduce a recommender simulation environment, *RecSim NG*, that allows the straightforward configuration of an item collection (or vocabulary), a user (latent) state model and a user choice model (Mladenov et al., 2021). We describe specific instantiations of this environment suitable for user representation learning, and the construction of our REP-UCB-REC learning and optimization methods.

1.1. Related Works

Recommender Systems Recommender systems have relied on collaborative filtering (CF) techniques (Konstan et al., 1997; Srebro et al., 2004; Mnih & Salakhutdinov, 2007)

[1] Department of Computer Science, Cornell University, Ithaca, USA. Correspondence to: Yijia Dai <yd73@cornell.edu>, Wen Sun <ws455@cornell.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

to exploit user responses on items. CF includes clustering users and items, and embedding users and items in a low-dimensional representation (e.g., probabilistic matrix factorization) (Krestel et al., 2009; Moshfeghi et al., 2011).

To capture more nuanced user behaviors, deep neural networks (DNNs) are commonly used in real world applications (Van den Oord et al., 2013; Covington et al., 2016). It naturally follows to be studied as a RL problem, as the user dynamics can be modeled as MDPs. Embeddings are commonly used to learn latent representations from immediate user interactions (Liu et al., 2020). And predictive models are commonly used to improve sample efficiency (Chen et al., 2021) and do self-supervised RL (Xin et al., 2020; Zhou et al., 2020).

Low-rank MDPs Oracle-efficient algorithms for low-rank MDPs (Agarwal et al., 2020; Uehara et al., 2021) provide sample complexity bounds easing the difficulty of exploration in large state space. Under more restricted setting, such as block MDPs (Misra et al., 2020) and m-step decodable MDPs (Efroni et al., 2022), methods are also studied to deal with the curse of dimensionality.

Slate recommendation and choice modeling Slate recommondation is common in recommender systems (Deshpande & Karypis, 2004; Viappiani & Boutilier, 2010). Within the context, off-policy evaluation and optimization with inverse propensity scores is studied (Swaminathan et al., 2017). Hierarchical models are also used for studying user behavior interacting with slates (Mehrotra et al., 2019).

The user choice model is linked with the slate recommendation. A common choice model is multinomial logit model (Louviere et al., 2000). A good representation of user choice has been studied in areas of econometrics, psychology, and operations research (Luce, 2012). Within ML community, including recommendation systems and learning to rank, cascade models are also popular as they capture the fading attention introduced by browsing behavior (Joachims, 2002).

2. Preliminaries

We consider an episodic MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma, d_0 \rangle$ for content recommendation with slates. We consider a setting in which a recommender system presents a slate to a user, from which the user selects zero or one item for consumption. The user respond to a consumed item with an engagement measure. The above setup is commonly used and easily extensible to real world applications (Ie et al., 2019).

The states $\mathcal{S}$ typically reflect user state, which includes static user features such as demographics, as well as dynamic user features. In particular, summaries of relevant user history with past recommendations made to the user and past user responses play a key role. The summarization of history is often domain specific and can capture certain aspects of user latent state in a partially observable MDP. The state should be predictive of immediate user response (e.g., immediate engagement, hence reward) and self-predictive (i.e., summarizes user history in a way that renders the implied dynamics Markovian).

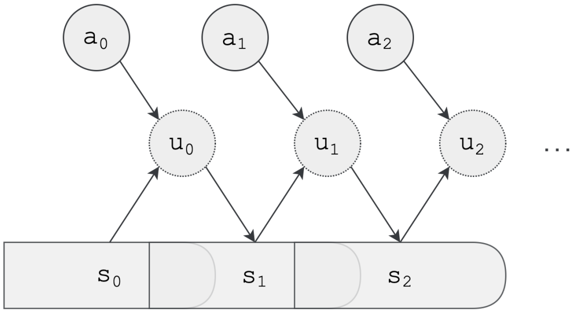

Figure 1. A latent state model captured by low-rank MDP. The states represent the logged user history, including the responses to the recommendations. The $\phi^\star(s, a)$ is a distribution over the latent user representation space $\mathcal{U}$. Note that this is still Markovian model since there is no direct transition between latent states.

The action space $\mathcal{A}$ is simply the set of all possible recommendation slates. We assume a fixed catalog of recommendable items $\mathcal{I}$, so actions are the subsets $a \subseteq \mathcal{I}$ such that $|\mathcal{A}| = \binom{|\mathcal{I}|}{k}$, where k is the slate size. We assume that each item $i \in \mathcal{I}$ and each slate a is recommendable at each state s for ease of exposition. We do not account for positional bias or ordering effects within the slate in this work. Because a user may select no item from a slate, we assume that every slate includes a $(k+1)$-th null item. This is standard in most choice modeling work and is straightforward on specifying all user behavior induced by a choice from the slate.

Transition $P(s'|s, a)$ represents the probability of user transitioning to s' from s when action a is taken. This generally reflects uncertainty in both user response and the future contextual or environmental state. One of the most critical points of uncertainty pertains the probability with which a user will consume a particular recommended item $i \in a$ from the slate. As such, choice models play a critical role in evaluating the quality of a slate. Since the ground truth $P^\star$ is unknown, we need to learn it by interacting with environments in an online manner or utilizing offline data at hand.

The reward $r(s, a)$ is the expected reward of a slate, which measures the expected degree of user engagement with items on the slate. The expectation accounts for the uncertainty in user choice. Without loss of generality, we assume trajectory reward is normalized, i.e., for any trajectory $\{s_h, a_h\}_{h=0}^{\infty}$, we have $\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \in [0, 1]$. We assume that $r(s, a)$ is known. This assumption largely relies on the success of existing myopic, item-level recommender (Covington et al., 2016).

The discounted factor $\gamma \in [0, 1)$ and initial distribution $d_0 \in \Delta(\mathcal{S})$ are also known.

Our goal is to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ which maps from state to distribution over actions (i.e., recommended slates). We use the following notations. Under some probability transition P, the value function $V_P^\pi(s) = \mathbb{E}[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) | s_0 = s, P, \pi]$ to represent the expected total discounted reward of π under P starting at s. Similarly, we define the state-action Q function $Q_P^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_P^\pi(s')$. Then, the expected total discounted reward of a policy π under transition P and reward r is denoted by $V_{P,r}^\pi = \mathbb{E}_{s_0 \sim d_0} V_P^\pi(s_0)$. We define the state-action discounted occupancy distribution $d_P^\pi(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t d_{P,t}^\pi(s, a)$, where $d_{P,t}^\pi(s, a)$ is the probability of visiting (s, a) at time step t under P and π. The state visitation as $d_P^\pi(s) = \sum_{a \in \mathcal{A}} d_P^\pi(s, a)$ Finally, given a vector a, $\|a\|_2 = \sqrt{a^\top a}$, $\|a\|_B = \sqrt{a^\top B a}$, and $\{c_i\}$ where $i \in \mathbb{N}$ are constants.

We focus on low-rank MDP defined as follows, with normalized function classes.

Definition 2.1. (Low-rank MDP) A transition model $P = \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ admits a low-rank decomposition with rank $d \in \mathbb{N}$ if there exist two embedding functions ϕ and μ such that

$$\forall s, s' \in \mathcal{S}, a \in \mathcal{A} : P(s'|s, a) = \mu(s')^\top \phi(s, a)$$

where $\|\phi(s, a)\|_2 \leq 1$ for all (s, a) and for any function $f : \mathcal{S} \to [0, 1]$, $\| \int \mu(s) f(s) d(s) \|_2 \leq \sqrt{d}$. An MDP is low-rank if P admits low-rank decomposition.

Within the context of recommender systems, low-rank MDPs capture the latent user representation dynamics as shown in Figure 2. As states are observed and follow Markovian transition, they mainly contain information of the history of user interactions with the recommender. Note that the states are likely to have overlapping information under this definition, and this simplifies the model class of μ as we will later discuss in Section 2. The actions are the slates presented to the user at each time step. Thus, from a logged history of user and a current slate recommendation, $\phi^\star$ maps (s, a) to the latent representation of user. In reality, the latent space $\mathcal{U}$ would be a compact representation space that contains the most important information that impacts user decisions. Learning such representation and interpreting would be meaningful in real life applications.

In episodic online learning, the goal is to learn a stationary policy $\hat{\pi}$ that maximize $V_{P^\star,r}^{\hat{\pi}}$, where $P^\star$ is the ground truth transition. We can only reset at initial distribution d_0, which emphasize the restricted nature of recommender systems and the challenge for exploration. The sampling of state s from visitation d_P^π is done by following the *roll-in* procedure: from starting state $s_0 \sim d_0$, at every time step t, with probability $1 - \gamma$ we terminate, otherwise we execute $a_t \sim \pi(s_t)$ and transit to $s_{t+1} \sim P(\cdot|s_t, a_t)$.

Function approximation for representation learning. Since $\phi^\star$ and $\mu^\star$ are unknown, we use function classes to capture them. These model classes can be learned using the existing myopic methods for user behavior prediction.

Assumption 2.2. (Realizability) We assume to have access to model classes $\phi^\star \in \Phi$ and $\mu^\star \in \Psi$.

We assume the function approximators also follow the normalization as $\phi^\star$ and $\mu^\star$, i.e., for any $\phi \in \Phi$ and $\mu \in \Psi$, $\|\phi(s, a)\|_2 \leq 1$ for all (s, a), $\| \int \mu(s) f(s) d(s) \|_2 \leq \sqrt{d}$ for all $f : \mathcal{S} \to [0, 1]$, and $\int \mu^\top(s') \phi(s, a) d(s') = 1$ for all (s, a).

We learn the functions via supervised learning oracle, which should be computationally efficient given the task.

Definition 2.3. (Maximum Likelihood Estimator) The MLE oracle takes a dataset of (s, a, s') tuples, and returns

$$\hat{P} := (\hat{\mu}, \hat{\phi}) = \arg \max_{(\mu,\phi) \in \mathcal{M}} \mathbb{E}_{\mathcal{D}_n + \mathcal{D}'_n} [\ln \mu^\top(s') \phi(s, a)].$$

The assumption above is achievable by the myopic estimators of user behavior models within recommendation context.

User choice model.

We introduce an assumption that is developed to reasonably structuralize the user behavior, and lead to effective reduction on the action space (Ie et al., 2019).

Assumption 2.4. (Reward / transition dependence on selection) We assume that $r(s, a)$ and $P(s'|s, a)$ depend only on the item $i \in a$ on slate that is consumed by the user (including the null item).

The original action space under our MDP is $\binom{|\mathcal{I}|}{k}$ with unordered k-sets over $\mathcal{I}$. With large item space, effective exploration is impossible. Luckily, the nature of human preference and structure of recommender systems poses the possibility for reduction. With the prior assumption on user choice setup, where user select zero (null) or one item from slate, we formalize the properties as follows:

$$r(s,a) = \sum_{i \in a} P(i|s,a) r(s,a,i),$$
$$P(s'|s,a) = \sum_{i \in a} P(i|s,a) P(s'|s,a,i).$$
$$\forall a, a' \text{ containing } i, \quad r(s,a,i) = r(s,a',i) = r(s,i),$$
$$P(s'|s,a,i) = P(s'|s,a',i) = P(s'|s,i), \forall a, a'.$$

Now, notice that the transition can be described as $P(s'|s,a) = \sum_{i \in a} P(i|s,a) P(s'|s,i)$, where $P(i|s,a)$ represents the user's choice on item i given a slate, and $P(s'|s,i)$ is the user state transition when i is consumed. This helps us to redefine uniform action in Section 3 and obtain complexity bound independent of the combinatorial action space.

Similar to the reward $r(s,a)$, we assume the user choice given a (s,a)-pair is known, relying on the success of existing myopic, item-level recommender (Covington et al., 2016). Thus, the low-rank MDP realizability assumption here is used to describe $P(s'|s,i)$.

3. Main results

We now define uniform action $U(\mathcal{A})$ within the above setup. Remind that the uniform action is used to encourage user transition to some novel states. As user transition is only dependent on consumed item i, we use the following definition for sufficient exploration.

Definition 3.1. The uniform action for a slate recommendation $U(\mathcal{A})$ is defined as the following:

1. randomly pick an item i from $\mathcal{I}$.
2. fill the remainder of the slates using the items least-likely be selected by the user.

We further show that under this definition, the effective uniform action space is $O(k|\mathcal{I}|)$.

Lemma 3.2. *The user choose the selected item i with probability at least $1/k$.*

We use the basic properties of probability and pigeonhole principle to further arrive at the proposition.

Proposition 3.3. *Efficient exploration action space is achieved in $O(k|\mathcal{I}|)$.*

Proof. By product rule of probability, the user select an item i with probability at least $1/(k|\mathcal{I}|)$ under the uniform action $U(\mathcal{A})$ as defined above.

By pigeonhole principle, every $k|\mathcal{I}| + 1$ uniform actions lead to at least one duplicate action in this space. □

Algorithm 1 UCB-driven representation learning for recommendation (Rep-UCB-Rec)

```
Input: Regularizer $\lambda_n$, parameter $\alpha_n$, Models $\mathcal{M} = \{(\mu, \phi) : \mu \in \Psi, \phi \in \Phi\}$
Initialize $\pi_0(\cdot|s)$ to be uniform; set $\mathcal{D}_0 = \emptyset$, $\mathcal{D}'_0 = \emptyset$.
for episode $n = 1, ..., N$ do
    Collect a tuple $(s, a, s', a', \tilde{s})$ with
        $s \sim d^{\pi_{n-1}}_{P^\star}, a \sim U(\mathcal{A}),$
        $s' \sim P^\star(\cdot|s,a), a' \sim U(\mathcal{A}), \tilde{s} \sim P^\star(\cdot|s,a)$
    Update datasets
        $\mathcal{D}_n = \mathcal{D}_{n-1} + (s,a,s'), \mathcal{D}'_n = \mathcal{D}'_{n-1} + (s',a',\tilde{s})$
    Learn representation via ERM (i.e., MLE)
        $\hat{P}_n := (\hat{\mu}_n, \hat{\phi}_n)$
        $= \arg\max_{(\mu,\phi) \in \mathcal{M}} \mathbb{E}_{\mathcal{D}_n + \mathcal{D}'_n} \left[ \ln \sum_{i \in a} \mu^\top(s') P(i|s,a) \phi(s,i) \right]$
    Update empirical covariance matrix
        $\hat{\Sigma}_n = \sum_{s,a \in \mathcal{D}} \hat{\varphi}_n(s,a) \hat{\varphi}_n(s,a)^\top + \lambda_n I$
        where $\hat{\varphi}_n(s,a) := \sum_{i \in a} P(i|s,a) \hat{\phi}_n(s,i)$
    Set the exploration bonus
        $\hat{b}_n(s,a) := \min \left( \alpha_n \sqrt{\hat{\varphi}_n(s,a)^\top \hat{\Sigma}_n^{-1} \hat{\varphi}_n(s,a)}, 2 \right)$
    Update policy
        $\pi_n = \arg\max_\pi V^\pi_{\hat{P}_n, r+\hat{b}_n}$
end for
```

Note that the above change in distribution of action space definition is introduced by asymmetry of uniform distributions. More specifically, by the principle of indifference, we choose our objective on user state transition instead of naive action space (White, 2010).

3.1. Algorithm

The algorithm is based on Rep-UCB (Uehara et al., 2021), with specified definition on state space $\mathcal{S}$, action space $\mathcal{A}$, the data collection process, and the uniform action $U(\mathcal{A})$ under the context of recommender system.

A state s consist of static user features and a history of past recommendations and user responses. An action a provides a slate recommendation. During the data collection process, for every iteration, we do one rollout under current policy π. We assume the states sampled from the initial distribution d_0 follows the prior Theorem 2.2 on model classes. The sampling procedure for $s \sim d_{P^\star}^{\pi_{n-1}}$ begins with $s_0 \sim d_0$. At every time step t, we terminate with probability $1-\gamma$, or execute $a_t \sim \pi(s_t)$ and observe $s_{t+1} \sim P^\star(\cdot|s_t, a_t)$. We sample uniform action $U(\mathcal{A})$ defined by Theorem 3.1. It means the recommender would do recommendations based on planning until termination, then do two uniform recommendations by the end of data collection process. We collect the tuple $(s, a, s', a', \tilde{s})$ and update datasets.

Representation learning, building the empirical covariance matrix, and calculating the bonus are done sequentially after datasets updates. The final step within the episode is planning using the learned estimation of transition with added exploration bonus.

3.2. Analysis

The PAC bound for REP-UCB (Uehara et al., 2021) provides us a good starting point. Our anlaysis focuses on reducing the naive combinatorial action space $|\mathcal{A}|$ to be polynomial in slate size k and item space size $|\mathcal{I}|$.

Theorem 3.4. *(PAC Bound for* REP-UCB-REC*) Fix* $\delta \in (0,1)$, $\epsilon \in (0,1)$. *Let* $\hat{\pi}$ *be a uniform mixture of* $\pi_1, ..., \pi_N$ *and* $\pi^\star := \arg\max_\pi V^\pi_{P^\star,r}$ *be optimal policy. Set the parameters as follows:*

$$\alpha_n = O\left(\sqrt{(k|\mathcal{I}| + d^2)\gamma \ln(|\mathcal{M}|n/\delta)}\right),$$

$$\lambda_n = O(d \ln(|\mathcal{M}|n/\delta)),$$

with probability at least $1-\delta$, *we have*

$$V^{\pi^\star}_{P^\star,r} - V^{\hat{\pi}}_{P^\star,r} \leq \epsilon.$$

The number of collected samples is at most

$$O\left(\frac{d^4 k^2 |\mathcal{I}|^2 \ln(|\mathcal{M}|/\delta)^2}{(1-\gamma)^5 \epsilon^2} \cdot \nu\right)$$

where

$$\nu := O\left(\ln\left(\frac{\iota}{\delta}\ln^2(1+\iota)\right) \cdot \ln^2(1+\iota)\right)$$

and

$$\iota := \frac{d^4 k^2 |\mathcal{I}|^2 \ln(|\mathcal{M}|/\delta)^2}{(1-\gamma)^5 \epsilon^2}.$$

Proof. The upper bound dependency on $|\mathcal{A}|$ is introduced by the importance weighing of the policy π to uniform action, where $\max_{(s,a)} \frac{\pi(a|s)}{u(a)} \leq |\mathcal{A}|$. Under our definition of $U(\mathcal{A})$ under slate recommendation context, we naturally change the upper bound to $k|\mathcal{I}|$. Note that this dependency is then interchangeable throughout the proof. □

4. Simulations

We discuss the simulation environment setup and the algorithm construction in this section. The simulation uses *Recsim NG* (Mladenov et al., 2021). A graph illustration is shown in Appendix A.

4.1. Simulation environment

Item class. A static set of items are sampled at the beginning of the simulation. Each item is represented by a T-dimensional vector $\mathbf{i} \in [-1,1]^T$, where each dimension represents a topic. Each item has a length $l(\mathbf{i}) \in [0,1]$ (e.g. length of a video, music or an article), and a quality $q(\mathbf{i}) \in [-1,1]$ that is unobserved from the user.

User interest. Users $\mathbf{u} \in U$ have various degrees of interests in each topics. Each user $\mathbf{u}$ is represented by an interest vector $\mathbf{u} \in [-1,1]^T$. The user interest towards a certain item is calculated by inner product $\mathbf{i}^\top \mathbf{u}$. The user interest vector $\mathbf{u}$ and the mechanism of user interest towards items are unobserved to recommenders.

User choice. Given a slate of k items, a user choose to consume one item from the slate with

$$P(\mathbf{i}|a, \mathbf{u}) = \frac{e^{\mathbf{i}^\top \mathbf{u}}}{\sum_{\mathbf{j}\in a} e^{\mathbf{j}^\top \mathbf{u}}}.$$

This choice model is called multinomial logit model (Louviere et al., 2000). For the null item (no choice), the item is simply represented by a T-dimensional zeros vector with length and quality zeros.

User dynamics. The internal transition of user interest vector allows the environment to capture Markovian transition and allow RL methods to do meaningful planning. After the consumption on item $\mathbf{i}$ at time step t, a user $\mathbf{u}$ follows

$$\mathbf{u}_t = c_0 \mathbf{u}_{t-1} + c_1 q(\mathbf{i})(\mathbf{i} - \mathbf{u}) + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, c_3)$. The constants are for normalization. Under this update, the user transits to favor the topics in $\mathbf{i}$ more if the quality of $\mathbf{i}$ is positive.

Reward. The reward is reflected by the user consumption time of a chosen item $\mathbf{i}$. It is linear with respect to the length and the user interest of item $\mathbf{i}$.

4.2. Algorithm construction

The state is a history of length h that the user interacts with the recommender. The history contains the slate recommendations and user responses at each time step $t-h, ..., t-1$. We construct the sampling procedure introduced by REP-UCB-REC. At each episode, the recommender follows the current learned policy and observe user responses. Until the rollin procedure terminates, the recommender further does two step uniform recommendation.

The supervised learning for representation is done in an offline manner, where we combine two model classes and train them together to make predictions from (s, a) to s'. Note that by definiton of states, we only predict the user next response towards current action, and slide up one time step the time window of history. It eases the complexity of large state space to $O(k)$ and reflects on the special structure behind recommender systems. After calculating the emperical covariance matrix and exploration bonus, we utilize a costomized simulator where transition and reward are estimated. Standard policy gradient methods are used to calculate the updated policy within the this environment under $\hat{P}$ and $(r + \hat{b})$.

5. Conclusion

In this work, we propose a sample-efficient representation learning algorithm, using the standard slate recommendation setup, to treat this as an online RL problem with low-rank Markov decision processes (MDPs). We show that the sample complexity for learning near-optimal policy is $O(d^4k^2|\mathcal{I}|^2/\epsilon^2(1-\gamma)^5)$ where k is the slate size and $\mathcal{I}$ is the item space. We further show the detailed construction of a recommender simulation environment with the proposed setup and sampling method.

References

Afsar, M. M., Crump, T., and Far, B. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

Agarwal, A., Kakade, S., Krishnamurthy, A., and Sun, W. Flambe: Structural complexity and representation learning of low rank mdps. *Advances in neural information processing systems*, 33:20095–20107, 2020.

Breese, J. S., Heckerman, D., and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*, 2013.

Chen, M., Chang, B., Xu, C., and Chi, E. H. User response models to improve a reinforce recommender system. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 121–129, 2021.

Covington, P., Adams, J., and Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pp. 191–198, 2016.

Deshpande, M. and Karypis, G. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

Efroni, Y., Jin, C., Krishnamurthy, A., and Miryoosefi, S. Provable reinforcement learning with a short-term memory. In *International Conference on Machine Learning*, pp. 5832–5850. PMLR, 2022.

Gauci, J., Conti, E., Liang, Y., Virochsiri, K., He, Y., Kaden, Z., Narayanan, V., Ye, X., Chen, Z., and Fujimoto, S. Horizon: Facebook's open source applied reinforcement learning platform. *arXiv preprint arXiv:1811.00260*, 2018.

Gomez-Uribe, C. A. and Hunt, N. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015.

He, R. and McAuley, J. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th international conference on data mining (ICDM)*, pp. 191–200. IEEE, 2016.

Ie, E., Jain, V., Wang, J., Narvekar, S., Agarwal, R., Wu, R., Cheng, H.-T., Lustman, M., Gatto, V., Covington, P., et al. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767*, 2019.

Jacobson, K., Murali, V., Newett, E., Whitman, B., and Yon, R. Music personalization at spotify. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 373–373, 2016.

Jin, C., Yang, Z., Wang, Z., and Jordan, M. I. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pp. 2137–2143. PMLR, 2020.

Joachims, T. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142, 2002.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37, 2009.

Krestel, R., Fankhauser, P., and Nejdl, W. Latent dirichlet allocation for tag recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pp. 61–68, 2009.

Liu, F., Guo, H., Li, X., Tang, R., Ye, Y., and He, X. End-to-end deep reinforcement learning based recommendation with supervised embedding. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 384–392, 2020.

Louviere, J. J., Hensher, D. A., and Swait, J. D. *Stated choice methods: analysis and applications*. Cambridge university press, 2000.

Luce, R. D. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

Mehrotra, R., Lalmas, M., Kenney, D., Lim-Meng, T., and Hashemian, G. Jointly leveraging intent and interaction signals to predict user satisfaction with slate recommendations. In *The World Wide Web Conference*, pp. 1256–1267, 2019.

Misra, D., Henaff, M., Krishnamurthy, A., and Langford, J. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *International conference on machine learning*, pp. 6961–6971. PMLR, 2020.

Mladenov, M., Hsu, C.-W., Jain, V., Ie, E., Colby, C., Mayoraz, N., Pham, H., Tran, D., Vendrov, I., and Boutilier, C. Recsim ng: Toward principled uncertainty modeling for recommender ecosystems. *arXiv preprint arXiv:2103.08057*, 2021.

Mnih, A. and Salakhutdinov, R. R. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20, 2007.

Moshfeghi, Y., Piwowarski, B., and Jose, J. M. Handling data sparsity in collaborative filtering using emotion and semantic based features. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 625–634, 2011.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 811–820, 2010.

Ricci, F., Rokach, L., and Shapira, B. Introduction to recommender systems handbook. In *Recommender systems handbook*, pp. 1–35. Springer, 2010.

Shani, G., Heckerman, D., Brafman, R. I., and Boutilier, C. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(9), 2005.

Srebro, N., Rennie, J., and Jaakkola, T. Maximum-margin matrix factorization. *Advances in neural information processing systems*, 17, 2004.

Swaminathan, A., Krishnamurthy, A., Agarwal, A., Dudik, M., Langford, J., Jose, D., and Zitouni, I. Off-policy evaluation for slate recommendation. *Advances in Neural Information Processing Systems*, 30, 2017.

Uehara, M., Zhang, X., and Sun, W. Representation learning for online and offline rl in low-rank mdps. *arXiv preprint arXiv:2110.04652*, 2021.

Van den Oord, A., Dieleman, S., and Schrauwen, B. Deep content-based music recommendation. *Advances in neural information processing systems*, 26, 2013.

Van Meteren, R. and Van Someren, M. Using content-based filtering for recommendation. In *Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop*, volume 30, pp. 47–56. Barcelona, 2000.

Viappiani, P. and Boutilier, C. Optimal bayesian recommendation sets and myopically optimal choice query sets. *Advances in neural information processing systems*, 23, 2010.

White, R. Evidential symmetry and mushy credence. 2010.

Xin, X., Karatzoglou, A., Arapakis, I., and Jose, J. M. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 931–940, 2020.

Zhou, K., Wang, H., Zhao, W. X., Zhu, Y., Wang, S., Zhang, F., Wang, Z., and Wen, J.-R. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pp. 1893–1902, 2020.

A. Graph illustration of the simulation environment.

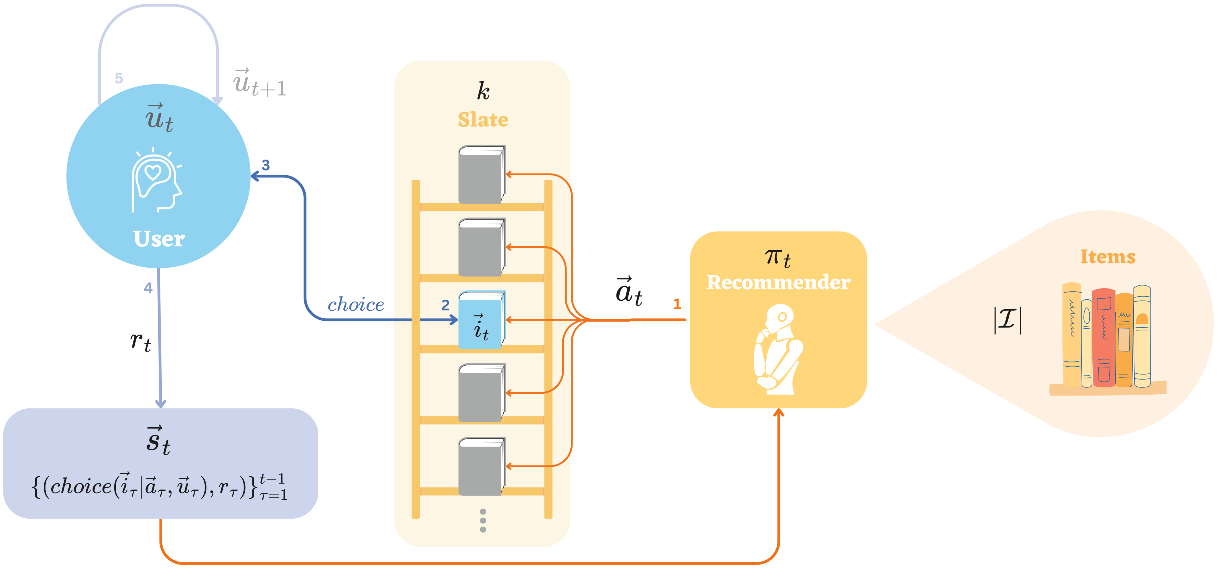

Figure 2. This flowchart describes the sequential decision making process of the recommender system setup we focus on. More specifically, the user choose an item $\vec{i}_t$ from the slate-based recommendation, transit to a new user internal state $\vec{u}_{t+1}$, and emit a reward r_t and new observable state $\vec{s}_{t+1}$. The recommender learns to use the observed state to predict user internal state and plan with exploration and exploitation.