Would You Like Your Data to Be Trained? A User Controllable Recommendation Framework

Lei Wang[1], **Xu Chen**[1,2*], **Zhenhua Dong**[3], **Quanyu Dai**[3]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods
[3]Noah's Ark Lab, Huawei
wanglei154@ruc.edu.cn, xu.chen@ruc.edu.cn, dongzhenhua@huawei.com, daiquanyu@huawei.com

Abstract

Recommender systems have a significant impact on various real-world applications, shaping people's daily lives and enhancing productivity. Traditional recommender models aim to collect extensive user information to accurately estimate user preferences. However, in practical scenarios, users may not want all their behaviors to be included in the model training process. This paper introduces a novel recommendation paradigm that allows users to indicate their "willingness" regarding which data should contribute to model training. The models are then optimized to maximize utility, which considers the trade-off between recommendation performance and respecting user preferences. The recommendation problem is formulated as a multiplayer game, with each user acting as a player and using a selection vector to indicate their willingness to include specific interacted items in training. To efficiently solve this game, an influence function-based model is proposed to approximate recommendation performances for different actions without re-optimizing the model. Furthermore, an enhanced model leveraging multiple anchor actions for the influence function is introduced to improve performance approximation accuracy. The convergence rate of the algorithm is theoretically analyzed, and the advantages of incorporating multiple anchor actions are demonstrated. Extensive experiments on both simulated and real-world datasets validate the effectiveness of the proposed models in balancing recommendation quality and user willingness. To promote this research direction, we have released our project at https://paitesanshi.github.io/IFRQE/.

Introduction

Recommender systems are critical components in real-world applications, providing significant business value and user convenience. Collaborative filtering (CF) has been a fundamental assumption in recommender systems, assuming that users who behaved similarly in the past will continue to behave similarly in the future. To accurately estimate user similarities, recommender systems traditionally collect extensive user behavior data. However, in real-world scenarios, users may not want all their data to be used for model training. As exampled in Figure 1, a male user may occasionally click on items that do not reflect his true preferences, and a female user may purchase items for others that are not indicative of her own preferences. In such cases, users may prefer that these irrelevant items not be used for training the model. This motivates the need for a recommender system that allows users to actively indicate their willingness regarding which items should be leveraged for training.

To address this challenge, we propose a novel, user-controllable recommendation paradigm. In this paradigm, each user specifies a willingness vector, which indicates their preference for excluding certain items from model training. We formulate the problem as a multiplayer game, where each player represents a user and their action is a selection vector representing a subset of their interacted items. The recommender model is then optimized based on the selected items from different users, striking a balance between recommendation performance and user preferences. However, solving this game requires repeated optimization of the recommender model to compute rewards for different actions, which may not be feasible in practice.

Faced with the aforementioned challenge, we utilize the concept of influence functions to approximate recommendation performance without re-optimizing the model. Specifically, we start by setting an anchor selection vector to train the recommender model. Then, for different actions, we directly compute the recommendation performance using the influence function (Koh and Liang 2017) and the previously obtained model trained with anchor selection vector. Our method is referred to as **i**nfluence **f**unction based **r**ecommendation **q**uality **e**xploration (called IFRQE for short). To achieve more accurate performance approximation, we extend this approach by employing multiple anchor selection vectors. In addition to these models, we provide a theoretical analysis of the convergence rate of our learning algorithm and demonstrate the advantages of increasing the number of anchor selection vectors. Extensive experiments are conducted using both synthetic and real-world datasets to validate the effectiveness of our models.

It is worth noting that there have been previous works on federated recommendation (Yang et al. 2020), recommendation unlearning (Chen et al. 2022a), and controllable recommendation. However, these studies differ fundamentally from our approach as they neither allow users to actively indicate data nor incorporate user willingness into the optimization target. Furthermore, while (Chen et al. 2022b) pro-

*Corresponding author

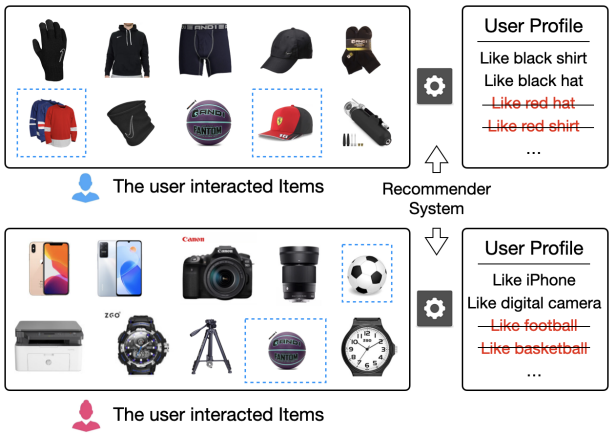

Figure 1: Motivating examples. In the upper case, the user does not want to leverage the occasionally clicked items to train the model. In the bottom case, the items purchased for the other people are expected to be omitted by the model.

posed a model for balancing user privacy and recommendation performance, we improve upon it by introducing more flexible user willingness and utilizing influence functions to enhance optimization efficiency.

The main contributions of this paper are summarized as follows: (1) We introduce a novel recommendation paradigm where users can explicitly indicate their willingness to leverage different items for model training. (2) To address this problem, we formulate the recommendation task as a multiplayer game and propose two influence function-based models for efficient game solving. (3) We provide theoretical analysis by deriving the convergence rate of our learning algorithm and showcasing the superiority of multiple anchor selection vectors. (4) Extensive experiments are conducted on synthetic and real-world datasets to demonstrate the effectiveness of our models.

Problem Formulation

Suppose we have a user set $\mathcal{U}$ and an item set $\mathcal{V}$. Let $\mathcal{O}^u$ be the set of items interacted by user u, and we separate it into a training set $\mathcal{S}^u$, a validation set $\mathcal{T}^u$ and a testing set $\mathcal{D}^u$. In our problem, each user u can specify a willingness vector $\boldsymbol{\beta}^u = \{\beta_1^u, \beta_2^u, ..., \beta_{|\mathcal{S}^u|}^u\}$. Intuitively, for different items, the users may have various willingness on leveraging them to train the model. Thus, we allow the users to indicate a continuous value to describe their willingness flexibly, that is, $\beta_k^u \in [0,1]$, $\forall k \in [1, |\mathcal{S}^u|]$. Suppose $\mathcal{S}^u = \{s_1^u, s_2^u, ..., s_{|\mathcal{S}^u|}^u\}$, then the larger β_k^u is, the more the user do not want s_k^u to join into the model training process. Our primary task involves selecting an appropriate subset from $\mathcal{S} = \mathcal{S}^u | u \in \mathcal{U}$ for training the recommender model, ensuring both satisfactory recommendation performance and adherence to user willingness. Increasing the number of selected items can enhance user understanding and improve recommendation performance. However, this approach may violate user willingness, as items with larger β_k^u values may be included. On the other hand, selecting only a few items to train the model may better satisfy user willingness but potentially lead to reduced recommendation performance. The key challenge lies in effectively balancing recommendation performance and user willingness.

To solve the above task, we regard each user as a player, and formulate the recommendation problem as a multiplayer game. For each user u, the action is a binary selection vector $\boldsymbol{o}^u = \{o_1^u, ..., o_{|\mathcal{S}^u|}^u\}$, where $o_k^u = 1$ means s_k^u is selected to train the model, otherwise $o_k^u = 0$. The reward for user u is designed as follows:

$$\overline{z}_u(\boldsymbol{o}^u, \boldsymbol{o}^{-u}) = -L_f(\mathcal{T}^u, \hat{\boldsymbol{\theta}}(\boldsymbol{o}^u, \boldsymbol{o}^{-u})) - \lambda \sum_{k=1}^{|\mathcal{S}^u|} \boldsymbol{o}_k^u \beta_k^u, \quad (1)$$

where $\boldsymbol{o}^{-u} = \{\boldsymbol{o}^1, ..., \boldsymbol{o}^{u-1}, \boldsymbol{o}^{u+1}, ..., \boldsymbol{o}^N\}$ is the joint selection vectors of all the user except u. f is a recommender model. $\hat{\boldsymbol{\theta}}(\boldsymbol{o}^u, \boldsymbol{o}^{-u})$ are the parameters of f learned based on the training samples selected by $\{\boldsymbol{o}^u, \boldsymbol{o}^{-u}\}$. $-L_f(\mathcal{T}^u, \hat{\boldsymbol{\theta}}(\boldsymbol{o}^u, \boldsymbol{o}^{-u}))$ is the negative validation loss based on $\hat{\boldsymbol{\theta}}(\boldsymbol{o}^u, \boldsymbol{o}^{-u})$, which is leveraged to measure the recommendation performance. The second term evaluates the violation of the user willingness. If the items that the user want to omit (*e.g.*, β_k^u is large) are selected to train the model, then $o_k^u \beta_k^u$ is large, which lowers the reward. λ is a pre-defined balancing parameter.

Instead of learning the specific selection vectors, we formulate our problem in a more general manner, where we assume that there is potential distribution (or called strategy) on the selection vectors for each user. Let the strategy of user u be $\boldsymbol{\alpha}^u = [\boldsymbol{\alpha}_{\boldsymbol{o}^u}^u] \in \triangle(2^{|\mathcal{S}^u|})$, which is a discrete distribution, and $\boldsymbol{\alpha}_{\boldsymbol{o}^u}^u$ is the probability of leveraging the items indicated by $\boldsymbol{o}^u$ to train the model. For example, suppose the training set is $\mathcal{S}^u = \{0, 1, 2\}$, then $\boldsymbol{\alpha}^u$ is a distribution on $\{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\}$, where A_1 denotes $\{0, 0, 0\}$, A_2 denotes $\{0, 0, 1\}$, and so on. Furthermore, A_5 denotes $\{0, 1, 1\}$ and $\boldsymbol{\alpha}_{A_5}^u$ represents the probability of using items $\{1, 2\}$ for model training.

We aim to learn the optimal joint strategy $\boldsymbol{\alpha}^* = \{\boldsymbol{\alpha}^{1*}, \boldsymbol{\alpha}^{2*}, ..., \boldsymbol{\alpha}^{N*}\}$, such that the corresponding expected reward of each user u is the largest when the other user strategies are fixed, that is:

$$z_u(\boldsymbol{\alpha}^{u*}, \boldsymbol{\alpha}^{-u*}) \geq z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u*}), \ \forall u \in [N], \boldsymbol{\alpha}^u \in \triangle(2^{|\mathcal{S}^u|}), \quad (2)$$

where $[N] = [1, 2, ..., N]$. $\boldsymbol{\alpha}^{-u}$ is the joint strategy of all the users except u. $z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u})$ is the expected reward for user u under $\{\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u}\}$, that is,

$$z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u}) = E_{\boldsymbol{o}^u \sim \boldsymbol{\alpha}^u, \, \boldsymbol{o}^{-u} \sim \boldsymbol{\alpha}^{-u}}[\overline{z}_u(\boldsymbol{o}^u, \boldsymbol{o}^{-u})]. \quad (3)$$

After obtaining the optimal strategy $\boldsymbol{\alpha}^*$, we firstly sample the selection vectors $\boldsymbol{o} = \{\boldsymbol{o}^u, \boldsymbol{o}^{-u}\}$ from $\boldsymbol{\alpha}^*$. Then, the training samples are generated by filtering $\mathcal{S}$ with $\boldsymbol{o}$, which are leveraged to optimize the final recommender model.

Remark. (i) In the above formulation, we do not impose any assumption on f and L_f, which makes our framework applicable for any recommender model and loss function. (ii) Although the action space in objective (2) may appear large, potentially impacting training efficiency, there are strategies

to mitigate this issue. In our experiments, we initialize $\boldsymbol{\alpha}$ with an informative prior and focus the search for the optimal solution around this prior to accelerate the training process. Additionally, in practice, one can assign the same value o_k^u to items in the same category or limit user indications to the most important or recent parts of their interactions.

The IFRQE Model

In this section, we firstly introduce a basic model for solving the above game based on influence function. Then, we design an improved model to enhance the accuracy for approximating the recommendation performance. At last, we theoretically analyze the learning algorithm of our model. In the following, we describe our models more in detail.

The Basic Model

To solve the game defined based on (1) and (2), one has to derive the validation loss $L_f(\mathcal{T}^u, \hat{\boldsymbol{\theta}}(\boldsymbol{o}))$ for different $\boldsymbol{o}$'s. A straightforward method is firstly training f for each $\boldsymbol{o}$ to obtain the corresponding parameter $\hat{\boldsymbol{\theta}}(\boldsymbol{o})$, and then the loss is computed based on the validation set $\mathcal{T}^u$ and $\hat{\boldsymbol{\theta}}(\boldsymbol{o})$. However, such method is infeasible, since repeatedly training the recommender model is quite time-consuming. Fortunately, the previous studies on influence function (Koh and Liang 2017) may shed some lights on approximating the validation loss without retraining the model. In specific, we first define an anchor selection vector $\widetilde{\boldsymbol{o}} = \{\widetilde{\boldsymbol{o}}^1, ..., \widetilde{\boldsymbol{o}}^N\}$, and then train the recommender model f based on $\widetilde{\boldsymbol{o}}$ to obtain the parameters $\widetilde{\boldsymbol{\theta}}$. At last, for a selection vector $\boldsymbol{o}$, we approximate $L_f(\mathcal{T}^u, \hat{\boldsymbol{\theta}}(\boldsymbol{o}))$ based on the following theory:

Theorem 1. *Given the model parameters $\boldsymbol{\theta}$, we define the validation loss by $L_f(\mathcal{T}^u, \boldsymbol{\theta}) = \sum_{y \in \mathcal{T}^u} l_f(y, \boldsymbol{\theta})$, where $l_f(y, \boldsymbol{\theta})$ is the loss of sample y based on $\boldsymbol{\theta}$. Suppose $| \bigtriangledown^2 l_f(s_k^v, \widetilde{\boldsymbol{\theta}})| \leq B$ and $\sum_v \sum_{k=1}^{|\mathcal{S}^v|}(\widetilde{o}_k^v - o_k^v)B$ is a small value, then the validation loss for $\boldsymbol{o}$ is:*

$$L_f(\mathcal{T}^u, \hat{\boldsymbol{\theta}}(\boldsymbol{o})) \approx L_f(\mathcal{T}^u, \widetilde{\boldsymbol{\theta}}) - \frac{1}{Z} \sum_{y \in \mathcal{T}^u} \sum_{v \in \mathcal{U}} \sum_{k=1}^{|\mathcal{S}^v|} o_k^v \bigtriangledown l_f(y, \widetilde{\boldsymbol{\theta}}) H_{\widetilde{\boldsymbol{\theta}}}^{-1} \bigtriangledown l_f(s_k^v, \widetilde{\boldsymbol{\theta}}) \tag{4}$$

where $\widetilde{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{Z} \sum_{v \in \mathcal{U}} \sum_{k=1}^{|\mathcal{S}^v|} \widetilde{o}_k^v l_f(s_k^v, \boldsymbol{\theta})$ are the parameters learned based on the anchor selection vector. Z is the total number of training samples. $\bigtriangledown l_f(\cdot)$ is the gradient of a sample loss. $H_{\widetilde{\boldsymbol{\theta}}} = \frac{1}{Z} \sum_v \sum_{k=1}^{|\mathcal{S}^v|} \widetilde{o}_k^v \bigtriangledown^2 l_f(s_k^v, \widetilde{\boldsymbol{\theta}})$ is the Hessian matrix of the training loss.

The proof of this theorem is presented in the Appendix[1]. Based on this theory, we can compute $L_f(\mathcal{T}^u, \hat{\boldsymbol{\theta}}(\boldsymbol{o}))$ without retraining the model via $\hat{\boldsymbol{\theta}}(\boldsymbol{o}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{Z} \sum_v \sum_{k=1}^{|\mathcal{S}^v|} o_k^{t,v} l_f(s_k^v, \boldsymbol{\theta})$. By bringing (4) into (1), $\overline{z}_u(\boldsymbol{o}^u, \boldsymbol{o}^{-u})$ can be written as:

$$\overline{z}_u(\boldsymbol{o}^u, \boldsymbol{o}^{-u}) = -L_f(\mathcal{T}^u, \widetilde{\boldsymbol{\theta}}) + \frac{1}{Z} \sum_{y \in \mathcal{T}^u} \sum_{v \in \mathcal{U}} \sum_{k=1}^{|\mathcal{S}^v|} o_k^v \bigtriangledown l_f(y, \widetilde{\boldsymbol{\theta}}) H_{\widetilde{\boldsymbol{\theta}}}^{-1} \bigtriangledown l_f(s_k^v, \widetilde{\boldsymbol{\theta}}) - \lambda \sum_{k=1}^{|\mathcal{S}^u|} o_k^u \beta_k^u. \tag{5}$$

Accordingly, the expected reward $z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u})$ is:

Theorem 2. *Let $\boldsymbol{g}_y^v = [g(s_1^v, y), ..., g(s_{|\mathcal{S}^v|}^v, y)]$, where $g(s_k^v, y) = \bigtriangledown_\theta l_f(y, \widetilde{\boldsymbol{\theta}})^T H_{\widetilde{\boldsymbol{\theta}}}^{-1} \bigtriangledown_\theta l_f(s_k^v, \widetilde{\boldsymbol{\theta}})$, then:*

$$z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u}) = \sum_{\boldsymbol{o}^u} \alpha_{\boldsymbol{o}^u}^u [(\sum_{y \in \mathcal{T}^u} \frac{\boldsymbol{g}_y^u}{N} - \lambda \boldsymbol{\beta}^u)^T \boldsymbol{o}^u] + C, \tag{6}$$

where $C = -L(\mathcal{T}^u, \widetilde{\boldsymbol{\theta}}) + \sum_{v \neq u} E_{\boldsymbol{o}}[(\boldsymbol{o}^v)^T \sum_{y \in \mathcal{T}^u} \frac{\boldsymbol{g}_y^v}{Z}]$.

The deriving process of the above theorem is presented in the Appendix. To solve objective (2), we need to find an optimal $\boldsymbol{\alpha}^u$ for the following optimization problem:

$$\max_{\boldsymbol{\alpha}^u} \sum_{\boldsymbol{o}^u} \alpha_{\boldsymbol{o}^u}^u A(\boldsymbol{o}^u), s.t. \sum_{\boldsymbol{o}^u} \alpha_{\boldsymbol{o}^u}^u = 1, \ \alpha_{\boldsymbol{o}^u}^u \geq 0 \tag{7}$$

where we denote $(\sum_{y \in \mathcal{T}^u} \frac{\boldsymbol{g}_y^u}{N} - \lambda \boldsymbol{\beta}^u)^T \boldsymbol{o}^u$ by $A(\boldsymbol{o}^u)$.

The Improved Model

The key of the above method lies in the accurate approximation of $L_f(\mathcal{T}^u, \hat{\boldsymbol{\theta}}(\boldsymbol{o}))$. However, if the current selection vector $\boldsymbol{o}$ is too much different from the anchor vector $\widetilde{\boldsymbol{o}}$, then the assumption $\sum_v \sum_{k=1}^{|\mathcal{S}^v|}(\widetilde{o}_k^v - o_k^v)B$ is a small value in theory 1 may not hold, which can lead to larger approximation errors. To alleviate this problem, we propose to set multiple anchor vectors $\{\widetilde{\boldsymbol{o}}^1, \widetilde{\boldsymbol{o}}^2, ...\widetilde{\boldsymbol{o}}^T\}$, where $\widetilde{\boldsymbol{o}}^t = \{\widetilde{\boldsymbol{o}}^{t,1}, ..., \widetilde{\boldsymbol{o}}^{t,N}\}$ is the tth anchor vector and $\widetilde{o}_k^{t,v}$ is the kth element of $\widetilde{\boldsymbol{o}}^{t,v}$. For approximating $L_f(\mathcal{T}^u, \hat{\boldsymbol{\theta}}(\boldsymbol{o}))$, we select the anchor vector nearest to $\boldsymbol{o}$, where we have the following theory.

Theorem 3. *For a candidate selection vector $\boldsymbol{o}$, suppose $t = \arg\min_{i \in [1,T]} \sum_{v=1}^N D(\widetilde{\boldsymbol{o}}^{i,v}, \boldsymbol{o}^v)$, where D is the hamming distance between two vectors, and $\widetilde{\boldsymbol{\theta}}^t = \arg\min_{\boldsymbol{\theta}} \frac{1}{Z} \sum_v \sum_{k=1}^{|\mathcal{S}^v|} \widetilde{o}_k^{t,v} l_f(s_k^v, \boldsymbol{\theta})$, then:*

$$L_f(\mathcal{T}^u, \hat{\boldsymbol{\theta}}(\boldsymbol{o})) \approx L_f(\mathcal{T}^u, \widetilde{\boldsymbol{\theta}}^t) - \frac{1}{Z} \sum_{y \in \mathcal{T}^u} \sum_{v \in \mathcal{U}} \sum_{k=1}^{|\mathcal{S}^v|} o_k^v \bigtriangledown l_f(y, \widetilde{\boldsymbol{\theta}}^t) H_{\widetilde{\boldsymbol{\theta}}^t}^{-1} \bigtriangledown l_f(s_k^v, \widetilde{\boldsymbol{\theta}}^t), \tag{8}$$

where $H_{\widetilde{\boldsymbol{\theta}}^t} = \frac{1}{Z} \sum_v \sum_{k=1}^{|\mathcal{S}^v|} \widetilde{o}_k^{t,v} \bigtriangledown^2 l_f(s_k^v, \widetilde{\boldsymbol{\theta}}^t)$.

Based on (8), the expected reward can be derived based on the following theory.

Theorem 4. *Let $A_t = \{\boldsymbol{o} | \sum_{v=1}^N D(\widetilde{\boldsymbol{o}}^{t,v}, \boldsymbol{o}^v) \leq \sum_{v=1}^N D(\widetilde{\boldsymbol{o}}^{t',v}, \boldsymbol{o}^v), \ \forall t' \neq t\}$, $\boldsymbol{g}_y^{t,v} = [g(s_k^v, y, t)]_{k=1}^{|\mathcal{S}^v|}$, where $g(s_k^v, y, t) = \bigtriangledown l_f(y, \widetilde{\boldsymbol{\theta}}^t) H_{\widetilde{\boldsymbol{\theta}}^t}^{-1} \bigtriangledown l_f(s_k^v, \widetilde{\boldsymbol{\theta}}^t)$ and $\boldsymbol{g}^{t,v} = \sum_{y \in \mathcal{T}^u} \boldsymbol{g}_y^{t,v}$. We define $\overline{z}_u(\boldsymbol{o}^u, \boldsymbol{o}^{-u}, t) = -L_f(\mathcal{T}^u, \widetilde{\boldsymbol{\theta}}^t) +$*

[1]Refer to Appendix: Proof of Theorem 1. Our Appendix is available at https://github.com/Paitesanshi/IFRQE.

Algorithm 1: Learning algorithm for $\boldsymbol{\alpha}_m^u$

Indicate the learning rate γ.
Let $\boldsymbol{\alpha}^{-u} = \boldsymbol{\alpha}_{m-1}^{-u}$ and $\boldsymbol{\alpha}_1^u = \boldsymbol{\alpha}_{m-1}^u$.
for *l in [1, L]* **do**
  Sample $\boldsymbol{o}^u, \boldsymbol{o}^{-u}$ according to $\boldsymbol{\alpha}_l^u, \boldsymbol{\alpha}^{-u}$.
  Compute the gradient $\hat{g}_{\boldsymbol{o}^u}$ based on (11).
  Let $[\hat{\boldsymbol{g}}]_s = 1(s = \boldsymbol{o}^u)\hat{g}_{\boldsymbol{o}^u}$.
  Update $\boldsymbol{\alpha}_{l+1}^u = \Pi_{\triangle}[\boldsymbol{\alpha}_l^u + \gamma\hat{\boldsymbol{g}}]$, where $\Pi_{\triangle}$ means projecting a vector into a simplex.
end
Return $\boldsymbol{\alpha}^u = \frac{1}{L}\sum_{l=1}^{L}\boldsymbol{\alpha}_l^u$.

$\frac{1}{Z}\sum_{y\in\mathcal{T}^u}\sum_{v\in\mathcal{U}}\sum_{k=1}^{|\mathcal{S}^v|} o_k^v \bigtriangledown l_f(y, \widetilde{\boldsymbol{\theta}}^t)H_{\widetilde{\boldsymbol{\theta}}^t}^{-1} \bigtriangledown l_f(s_k^v, \widetilde{\boldsymbol{\theta}}^t) - \lambda\sum_{k=1}^{|\mathcal{S}^u|} o_k^u\beta_k^u$, *Then, we have:*

$$
\begin{aligned}
&z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u}) = E_{\boldsymbol{o}}[\sum_{t=1}^{T} 1(\boldsymbol{o}\in A_t)\overline{z}_u(\boldsymbol{o}^u, \boldsymbol{o}^{-u}, t)] \\
&= \sum_{t=1}^{T}\sum_{\boldsymbol{o}} 1(\boldsymbol{o}\in A_t)\alpha_{\boldsymbol{o}^u}^u\alpha_{\boldsymbol{o}^{-u}}^{-u}\{-L(\mathcal{T}^u, \widetilde{\boldsymbol{\theta}}^t) + \frac{1}{Z}\sum_{v\neq u}(\boldsymbol{o}^v)^T\boldsymbol{g}^{t,v} \\
&+ \frac{1}{Z}(\boldsymbol{o}^u)^T\boldsymbol{g}^{t,u} - \lambda(\boldsymbol{o}^u)^T\boldsymbol{\beta}^u\},
\end{aligned}
\tag{9}
$$

where $1(c) = 1$ *if the condition* c *is true, otherwise* $1(c) = 0$. *And* $\alpha_{\boldsymbol{o}^{-u}}^{-u} = \alpha_{\boldsymbol{o}^1}^1 ... \alpha_{\boldsymbol{o}^{u-1}}^{(u-1)}\alpha_{\boldsymbol{o}^{u+1}}^{(u+1)} ... \alpha_{\boldsymbol{o}^N}^N$.

The proof of the above theorem is presented in the Appendix.

Finally, we derive $\boldsymbol{\alpha}^u$ by solving the following problem:

$$
\begin{aligned}
&\max_{\alpha^u}\sum_{t=1}^{T}\sum_{\boldsymbol{o}} 1(\boldsymbol{o}\in A_t)\alpha_{\boldsymbol{o}^u}^u\alpha_{\boldsymbol{o}^{-u}}^{-u}B(\boldsymbol{o}^u, \boldsymbol{o}^{-u}, t), \\
&s.t.\ \sum_{\boldsymbol{o}^u}\alpha^u(\boldsymbol{o}^u) = 1,\ \ \alpha^u(\boldsymbol{o}^u) > 0
\end{aligned}
\tag{10}
$$

where the notation $B(\boldsymbol{o}^u, \boldsymbol{o}^{-u}, t) = -L(\mathcal{T}^u, \widetilde{\boldsymbol{\theta}}^t) + \frac{1}{Z}\sum_{v\neq u}(\boldsymbol{o}^v)^T\boldsymbol{g}^{t,v} + \frac{1}{Z}(\boldsymbol{o}^u)^T\boldsymbol{g}^{t,u} - \lambda(\boldsymbol{o}^u)^T\boldsymbol{\beta}^u$.

Since it is hard to efficiently obtain a closed-form solution for $\boldsymbol{\alpha}^u$, we learn it based on the projected gradient descent method (Calamai and Moré 1987). More specifically, the gradient of $z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u})$ w.r.t $\alpha_{\boldsymbol{o}^u}^u$ is:

$$
\begin{aligned}
&\frac{\partial z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u})}{\partial\alpha_{\boldsymbol{o}^u}^u} \overset{\text{def}}{=} g_{\boldsymbol{o}^u} \\
&= E_{\boldsymbol{o}^{-u}}[\sum_{t=1}^{T} B(\boldsymbol{o}^u, \boldsymbol{o}^{-u}, t)\mathbf{1}(\boldsymbol{o}^{-u}\in S_t(\boldsymbol{o}^u))].
\end{aligned}
\tag{11}
$$

where $S_t(\boldsymbol{o}^u) = \{\boldsymbol{o}^{-u} | [\boldsymbol{o}^u, \boldsymbol{o}^{-u}] \in A_t\}$. Let $\boldsymbol{g} = [g_{\boldsymbol{o}^u}]_{\boldsymbol{o}^u}$ be the gradient of $z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u})$ w.r.t $\boldsymbol{\alpha}^u$, which is a $2^{|\mathcal{S}^u|}$ dimensional vector. $\hat{g}_{\boldsymbol{o}^u}$ is the stochastic gradient by sampling $\boldsymbol{o}^{-u}$ from $\boldsymbol{\alpha}^{-u}$. Then the complete training process can be seen in Algorithm 2.

Algorithm 2: Learning Algorithm with Multiple θ_t

Initialize $\{\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2, ..., \boldsymbol{\alpha}^N\}$ and let $\boldsymbol{\alpha}_0^u = \boldsymbol{\alpha}^u$ $(u \in [1, N])$.
Indicate the max iteration number M and threshold κ.
Indicate the anchor vectors $\{\widetilde{\boldsymbol{o}}^1, \widetilde{\boldsymbol{o}}^2, ...\widetilde{\boldsymbol{o}}^T\}$.
Obtain the model parameters $\widetilde{\boldsymbol{\theta}}^t$ for each $\widetilde{\boldsymbol{o}}^t$.
for *m in [1, M]* **do**
  for *u in [1, N]* **do**
    Let $\boldsymbol{\alpha}_{m-1}^{-u} = \{\boldsymbol{\alpha}_{m-1}^1, ..., \boldsymbol{\alpha}_{m-1}^{u-1}, \boldsymbol{\alpha}_{m-1}^{u+1}, ..., \boldsymbol{\alpha}_{m-1}^N\}$.
    Obtain $\boldsymbol{\alpha}_m^u$ by inputting $\boldsymbol{\alpha}_{m-1}^{-u}$ and $\boldsymbol{\alpha}_{m-1}^u$ into Algorithm 1.
  end
  if $|\boldsymbol{\alpha}_m^u - \boldsymbol{\alpha}_{m-1}^u| < \kappa\ , \forall u \in [1, N]$ **then**
    Break.
  end
end
Output $\boldsymbol{\alpha}^{u*} = \boldsymbol{\alpha}_m^u$ $(u \in [1, N])$.

Theoretical Analysis

In this section, we provide theoretical analysis on the convergence of our algorithm and also present the advantage of multiple anchor selection vectors comparing with the single one in terms of the validation loss approximation. For the convergence analysis, we have the following theory:

Theorem 5. *Based on the definition of* $\boldsymbol{g}$*, we have* $z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u}) = (\boldsymbol{\alpha}^u)^T\boldsymbol{g}$*. Suppose* $\hat{\boldsymbol{g}}$ *is an unbiased estimation of* $\boldsymbol{g}$*, and* $||\hat{\boldsymbol{g}}(\boldsymbol{\alpha}^u)||_2^2 \leq G$*, then the solution obtained from Algorithm 1 is larger than the optimal solution minus a bounded value, that is:*

$$
E[z_u(\hat{\boldsymbol{\alpha}}^u, \boldsymbol{\alpha}^{-u})] \geq \max_{\boldsymbol{\alpha}^u} E[z_u(\boldsymbol{\alpha}^u, \boldsymbol{\alpha}^{-u})] - (\frac{1}{L\gamma} + \gamma^2G^2),
\tag{12}
$$

where $\hat{\boldsymbol{\alpha}}^u$ *is obtained based on Algorithm 1.*

We prove this theory in Appendix. It provides foundations for Algorithm 1, which demonstrates that the learned strategies are nearly optimal. For the advantage of using multiple anchor selection vectors, we have the following theory:

Theorem 6. *For two sets of anchor selection vectors* $\boldsymbol{P} = \{\widetilde{\boldsymbol{o}}_P^1, \widetilde{\boldsymbol{o}}_P^2, ...\widetilde{\boldsymbol{o}}_P^{T_P}\}$ *and* $\boldsymbol{Q} = \{\widetilde{\boldsymbol{o}}_Q^1, \widetilde{\boldsymbol{o}}_Q^2, ...\widetilde{\boldsymbol{o}}_Q^{T_Q}\}$*, suppose* $A_t^P = \{\boldsymbol{o} | \sum_{v=1}^N D(\widetilde{\boldsymbol{o}}^{t,v}, \boldsymbol{o}^v) \leq \sum_{v=1}^N D(\widetilde{\boldsymbol{o}}^{t',v}, \boldsymbol{o}^v), \forall\widetilde{\boldsymbol{o}}^{t'} \in \boldsymbol{P}\}$, $A_t^Q = \{\boldsymbol{o} | \sum_{v=1}^N D(\widetilde{\boldsymbol{o}}^{t,v}, \boldsymbol{o}^v) \leq \sum_{v=1}^N D(\widetilde{\boldsymbol{o}}^{t',v}, \boldsymbol{o}^v), \forall\widetilde{\boldsymbol{o}}^{t'} \in \boldsymbol{Q}\}$. *Based on theory 1, we consider the following upper bounds of the approximation error for the validation loss:*

$$
\begin{aligned}
err(\boldsymbol{P}) &= \sum_{t=1}^{T_P}\sum_{\boldsymbol{o}\in A_t^P}[\sum_v D(\widetilde{\boldsymbol{o}}^{t,v}, \boldsymbol{o}^v)]B, \\
err(\boldsymbol{Q}) &= \sum_{t=1}^{T_Q}\sum_{\boldsymbol{o}\in A_t^Q}[\sum_v D(\widetilde{\boldsymbol{o}}^{t,v}, \boldsymbol{o}^v)]B.
\end{aligned}
\tag{13}
$$

We have if $\boldsymbol{P} \subseteq \boldsymbol{Q}$*, then* $err(\boldsymbol{P}) \geq err(\boldsymbol{Q})$.

The proof of this theorem is provided in the Appendix, indicating that including the single anchor vector in the multiple anchor vectors reduces the upper bound of the approximation error for the validation loss. However, it is important to note that introducing more anchor vectors inevitably requires additional retraining of the recommender model, consequently resulting in a decrease in efficiency. Nevertheless, the improved model allows for a balance between approximation accuracy and model efficiency.

Related Work

Recommender system is a rapidly developing research field, with a large amount of models developed each year (Ricci, Rokach, and Shapira 2015; Wang et al. 2023). Early recommender models primarily relied on matrix factorization techniques (Koren, Bell, and Volinsky 2009). However, with the advancements in deep learning, numerous deep recommender models have emerged. These models encompass various approaches, such as sequential (Kang and McAuley 2018; Chen et al. 2018; Wang et al. 2021) and graph-based algorithms (Wang et al. 2019; He et al. 2020). In an effort to protect user privacy, federated recommender models have been proposed (Lin et al. 2020; Wu et al. 2021; Yang et al. 2020; Chen et al. 2020). These models aim to shift the training process from the server to the clients to prevent access to raw user behaviors and ensure privacy. While our model can also contribute to privacy protection, it does not assume an adversarial server. Instead, our focus is on preventing the leakage of privacy information through recommendation results. Another relevant area is recommendation unlearning, which seeks to eliminate the influences of user-item interactions from trained recommender models (Chen et al. 2022a; Li et al. 2022). However, these models do not account for user active disclosure behaviors or incorporate user willingness into optimization targets. Controllable recommendation has also received attention in recent studies (Wang et al. 2022; Parra and Brusilovsky 2015). However, these approaches primarily aim to break filter bubbles rather than aligning with user-indicated willingness. Game theory-based recommender models have also been explored (Ben-Porat and Tennenholtz 2018; Xu et al. 2018; Halkidi and Koutsopoulos 2011). Although our work shares some similarities with these models, we distinguish ourselves by constructing the game from the user's perspective, not limiting our framework to rating behaviors, and leveraging influence functions to enhance recommendation performance estimation, a novel approach in this context.

Experiments

In this section, we conduct comprehensive experiments to demonstrate the effectiveness of our models, where we focus on the following research questions: **RQ1**: Whether our framework can achieve better trade-off between the recommendation performance and user willingness? **RQ2**: Whether the validation loss can be properly approximated by our framework? **RQ3**: What is the influence of the number of anchor selection vectors? **RQ4**: How does the dataset sparsity influence the performance of our framework? **RQ5**: Whether our framework is effective for different weighting parameter λ's? **RQ6**: Whether our framework is effective for the user willingness with different characters? **RQ7**: How does our framework perform if the recommender model is trained from scratch without using the influence function? In the following, we first introduce the experiment setup, followed by a presentation and analysis of the empirical results to address the above questions.

Experiment Setup

Datasets. We base our experiments on both synthetic and real-world datasets. For the synthetic dataset, we generated it based on a well known recommendation simulator called RecSim[2]. In specific, we generate 1000 users and 1000 items. For each user or item, we simulate its profiles by four random features, which is denoted by $\boldsymbol{e}_u \in \mathbb{R}^4$ or $\boldsymbol{e}_v \in \mathbb{R}^4$. The dataset is generated according to the observational parameter $\tau_{uv} = \mathbf{1}(\sigma(\boldsymbol{e}_u^T \boldsymbol{e}_v) \geq \eta)$, where $\mathbf{1}(\cdot)$ is the indicator function and η is a threshold for setting the hardness of generating the interaction. If $\tau_{uv} = 1$, then the user-item pair (u, v) is observed in the dataset. For the willingness of user u on item v, we simulate it by: $\beta_v^u = \sigma(s_u g(\boldsymbol{e}_u, \boldsymbol{e}_v) + b_u)$, where s_u is sampled from a uniform distribution in the range of $[a_1, a_2]$, controlling the sensitivity of the user willingness. Smaller s_u means that the user has similar willingness on leveraging different items to train the model, while larger s_u means that the user willingness on different items are more diverse. b_u is sampled from a Gaussian distribution $\mathcal{N}(0, a_3)$, indicating the average level of the user willingness. g is a one-layer fully connected neural network with randomly assigned parameters. In the experiments, the threshold η and (a_1, a_2, a_3) are initially set as 0.5 and $(0.5, 1, 1)$, respectively. Then, we tune them to study the influence of different dataset sparsities and user willingness characters on our framework performance. For the real-world experiments, we evaluate different models based on three publicly available datasets from different domains, including **Diginetica**[3], **Steam**[4] and **Amazon Video**[5]. Diginetica and Amazon Video are e-commerce datasets, where we are provided with the user-item purchasing records. Steam is a game dataset, which includes the interactions between the users and games. The statistics of the above datasets are concluded in Appendix.

Baselines. To demonstrate the generality of our idea, we implement the base model f with the following recommender algorithms[6]: **MF** (Koren, Bell, and Volinsky 2009) is the traditional matrix factorization method. **NeuMF** (He et al. 2017) is a well known neural collaborative filtering model. **LightGCN** (He et al. 2020) is a graph-based recommender model. We compare our framework to various methods. Firstly, **Random** uses user behaviors randomly in training (*i.e.*, selection vector $\boldsymbol{o}$ is randomly sampled). Sec-

[2]https://github.com/google-research/recsim

[3]https://darel13712.github.io/rs_datasets/Datasets/diginetica/

[4]https://steam.internet.byu.edu/

[5]https://jmcauley.ucsd.edu/data/amazon/

[6]Experiments with more other base models can be found in the Appendix.

Dataset	Simulation			Diginetica			Steam			Amazon Video		
Metric	$F_1 \uparrow$	$wv \downarrow$	reward↑	$F_1 \uparrow$	$wv \downarrow$	reward↑	$F_1 \uparrow$	$wv \downarrow$	reward↑	$F_1 \uparrow$	$wv \downarrow$	reward↑
MF	**1.75**$_{(.022)}$	2.12$_{(.007)}$	-2.25$_{(.032)}$	4.72$_{(.044)}$	2.01$_{(.053)}$	-2.65$_{(.072)}$	**10.4**$_{(.043)}$	2.62$_{(.014)}$	-2.99$_{(.093)}$	1.70$_{(.012)}$	2.36$_{(.021)}$	-2.43$_{(.022)}$
w/ Random	1.73$_{(.031)}$	2.01$_{(.006)}$	-2.13$_{(.014)}$	4.18$_{(.014)}$	1.96$_{(.013)}$	-2.04$_{(.006)}$	9.40$_{(.009)}$	2.61$_{(.017)}$	-2.82$_{(.015)}$	1.53$_{(.011)}$	2.24$_{(.022)}$	-2.30$_{(.023)}$
w/ Threshold	1.70$_{(.036)}$	2.01$_{(.032)}$	-2.20$_{(.021)}$	4.70$_{(.012)}$	1.97$_{(.011)}$	-2.06$_{(.040)}$	9.25$_{(.048)}$	2.35$_{(.023)}$	-2.57$_{(.017)}$	1.50$_{(.026)}$	2.12$_{(.027)}$	-2.20$_{(.021)}$
w/ Proactive	1.23$_{(.025)}$	**1.56**$_{(.013)}$	-2.25$_{(.007)}$	2.21$_{(.018)}$	**1.30**$_{(.024)}$	-1.99$_{(.016)}$	9.93$_{(.019)}$	**2.08**$_{(.022)}$	-2.43$_{(.025)}$	1.55$_{(.035)}$	1.94$_{(.026)}$	-2.07$_{(.005)}$
w/ IFRQE	1.50$_{(.015)}$	1.97$_{(.003)}$	-2.09$_{(.017)}$	**5.23**$_{(.008)}$	2.01$_{(.014)}$	-2.18$_{(.006)}$	**10.4**$_{(.012)}$	2.14$_{(.012)}$	-2.43$_{(.015)}$	1.57$_{(.015)}$	**1.70**$_{(.016)}$	-2.10$_{(.005)}$
w/ IFRQE++	1.63$_{(.007)}$	1.82$_{(.003)}$	**-1.92**$_{(.012)}$	4.02$_{(.011)}$	1.79$_{(.019)}$	**-1.92**$_{(.022)}$	9.83$_{(.017)}$	2.13$_{(.032)}$	**-2.42**$_{(.014)}$	**1.72**$_{(.012)}$	1.87$_{(.032)}$	**-1.98**$_{(.056)}$
NeuMF	0.87$_{(.017)}$	2.13$_{(.013)}$	-2.32$_{(.011)}$	**4.22**$_{(.013)}$	2.01$_{(.015)}$	-2.11$_{(.003)}$	**10.2**$_{(.007)}$	2.62$_{(.012)}$	-2.76$_{(.007)}$	1.65$_{(.014)}$	2.36$_{(.013)}$	-2.47$_{(.010)}$
w/ Random	0.89$_{(.009)}$	2.02$_{(.005)}$	-2.19$_{(.011)}$	2.38$_{(.018)}$	1.91$_{(.011)}$	-2.11$_{(.016)}$	7.73$_{(.005)}$	2.47$_{(.019)}$	-2.59$_{(.011)}$	1.80$_{(.014)}$	2.31$_{(.006)}$	-2.41$_{(.010)}$
w/ Threshold	1.50$_{(.032)}$	2.01$_{(.048)}$	-2.18$_{(.031)}$	4.01$_{(.012)}$	1.97$_{(.044)}$	-2.13$_{(.006)}$	7.50$_{(.019)}$	2.35$_{(.020)}$	-2.86$_{(.010)}$	1.55$_{(.002)}$	2.12$_{(.018)}$	-2.21$_{(.017)}$
w/ Proactive	**1.83**$_{(.005)}$	2.01$_{(.023)}$	-2.18$_{(.017)}$	1.30$_{(.008)}$	**1.69**$_{(.011)}$	-2.14$_{(.012)}$	9.55$_{(.025)}$	2.22$_{(.032)}$	-2.42$_{(.005)}$	1.43$_{(.015)}$	**1.96**$_{(.006)}$	-2.41$_{(.015)}$
w/ IFRQE	1.40$_{(.014)}$	1.91$_{(.013)}$	-2.48$_{(.013)}$	2.92$_{(.137)}$	2.01$_{(.017)}$	-2.17$_{(.019)}$	8.17$_{(.033)}$	**2.16**$_{(.019)}$	-2.45$_{(.027)}$	**1.83**$_{(.015)}$	1.99$_{(.014)}$	-2.24$_{(.012)}$
w/ IFRQE++	0.70$_{(.015)}$	**1.80**$_{(.003)}$	**-2.11**$_{(.011)}$	3.00$_{(.017)}$	1.91$_{(.013)}$	**-2.08**$_{(.023)}$	8.87$_{(.021)}$	2.21$_{(.014)}$	**-2.37**$_{(.027)}$	1.22$_{(.031)}$	2.00$_{(.014)}$	**-2.17**$_{(.011)}$
LightGCN	0.90$_{(.005)}$	2.13$_{(.008)}$	-2.82$_{(.012)}$	**7.50**$_{(.009)}$	2.01$_{(.012)}$	-2.71$_{(.009)}$	10.1$_{(.010)}$	2.62$_{(.009)}$	-3.13$_{(.014)}$	**2.15**$_{(.021)}$	2.36$_{(.008)}$	-3.05$_{(.007)}$
w/ Random	0.82$_{(.013)}$	2.02$_{(.013)}$	-2.71$_{(.022)}$	7.48$_{(.011)}$	1.95$_{(.005)}$	-2.64$_{(.007)}$	10.0$_{(.027)}$	2.49$_{(.026)}$	-3.07$_{(.019)}$	1.75$_{(.003)}$	2.24$_{(.009)}$	-2.93$_{(.015)}$
w/ Threshold	0.96$_{(.038)}$	2.01$_{(.008)}$	-2.70$_{(.003)}$	7.13$_{(.017)}$	1.97$_{(.040)}$	-2.60$_{(.011)}$	9.40$_{(.048)}$	2.35$_{(.027)}$	-2.82$_{(.046)}$	1.53$_{(.033)}$	2.12$_{(.022)}$	-2.30$_{(.048)}$
w/ Proactive	**1.01**$_{(.013)}$	1.78$_{(.011)}$	-2.48$_{(.017)}$	5.51$_{(.038)}$	**1.42**$_{(.007)}$	-2.11$_{(.018)}$	**10.3**$_{(.021)}$	**2.12**$_{(.022)}$	-2.81$_{(.015)}$	1.28$_{(.013)}$	1.95$_{(.016)}$	-2.64$_{(.018)}$
w/ IFRQE	0.97$_{(.008)}$	2.12$_{(.010)}$	-2.82$_{(.014)}$	4.88$_{(.019)}$	1.47$_{(.011)}$	-2.16$_{(.013)}$	9.02$_{(.016)}$	2.49$_{(.025)}$	-2.82$_{(.008)}$	1.90$_{(.007)}$	**1.75**$_{(.018)}$	-2.13$_{(.013)}$
w/ IFRQE++	0.75$_{(.022)}$	**1.75**$_{(.008)}$	**-2.44**$_{(.016)}$	5.26$_{(.012)}$	1.32$_{(.014)}$	**-2.01**$_{(.009)}$	8.90$_{(.018)}$	2.48$_{(.018)}$	**-2.80**$_{(.005)}$	1.62$_{(.013)}$	**1.75**$_{(.009)}$	**-2.03**$_{(.023)}$

Table 1: Overall comparison between different models. We use bold fonts to label the best performance for each dataset, evaluation metric and base model. "()" indicates the standard error. The results of F_1 are percentage values with "%" omitted. For the metrics, ↑ means the larger the better, while ↓ means the lower the better. The performance improvements of our model against the baselines are significant under paired t-test.

ondly, **Threshold** sets $o_u^k = 0$ if $\beta_u^k > 0.5$. We also compare with **Proactive**(Chen et al. 2022b), a model designed to consider both recommendation accuracy and user privacy. For the sake of fairness, we use the same parameter λ to balance these two aspects in objective. Our basic and improved methods are named **IFRQE** and **IFRQE++**, respectively.

Implementation Details. For each user, we use 20% and 10% of interactions for testing and validation, respectively, with the rest for training. All models employ binary cross entropy as the loss function. Recommendation performance is evaluated using $\boldsymbol{F_1}$ (Chicco and Jurman 2020) with five items recommended for comparison to the ground truth. To assess adherence to user willingness, we calculate overall willingness violation, $wv = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\sum_{k=1}^{|\mathcal{S}^u|} o_k^u\beta_k^u$, lower wv indicating higher satisfaction. Model performance in balancing recommendation quality and user willingness is compared via **reward** (eq. 1). Our framework and baselines are optimized for the same reward, testing different λ's. Anchor selection vectors in IFRQE and IFRQE++ are randomly selected[7]. Model hyperparameters are set via grid search. For instance, the number of anchor selection vectors is searched in $[1, 2, ..., 10]$, learning rate and batch size in $[0.0001, 0.001, 0.005, 0.01]$ and $[1024, 2048, 4096]$, respectively. Anchor selection vectors are binomially distributed, with a mean of 0.9. For real-world datasets, we simulate user willingness with random assignments. To control for randomness, we repeat experiments ten times, reporting average and standard error. More implementation details[8] and complete experiment results can be found in the Appendix.

Overall Comparison

The overall comparison results are presented in Table 1. From the user willingness perspective, our model consistently outperforms the base model and heuristic methods, indicating a finer consideration of user preferences. This improvement is expected, as the base model utilizes all available items for training, while the heuristic methods adopt a coarser approach to user willingness. In terms of recommendation performance, our models exhibit only a moderate decrease compared to the base model, with an average drop of approximately 8.02%. Notably, in certain cases, such as with the Diginetica and Amazon Video datasets using MF as the base model, our approach even demonstrates improved performance. This observation suggests that the removal of items with higher β_k^u values, which may not be essential for predicting items in the testing set, leads to higher rewards by aligning with user willingness and enhances performance by focusing on more informative training samples. From the reward perspective, our framework consistently achieves the best performances across most datasets and base models, illustrating the effectiveness and generalizability of our approach in striking a balance between recommendation quality and user willingness. On average, our IFRQE++ method improves the base model by approximately 12.4%, 11.1%, 14.6%, and 21.3% on the simulation, Diginetica, Steam, and Amazon Video datasets, respectively. The superior performance of our framework compared to the random method underscores the non-trivial nature of the problem and the limitations of random selection. When compared to Proactive, our framework demonstrates comparable performance

[7]IFRQE++'s anchor selection vectors include those of IFRQE

[8]MindSpore implementation at https://github.com/mindspore-lab/models/tree/master/research/huawei-noah/IFRQE.

Figure 2: (a) Approximation error on the validation loss. (b) Influence of T.

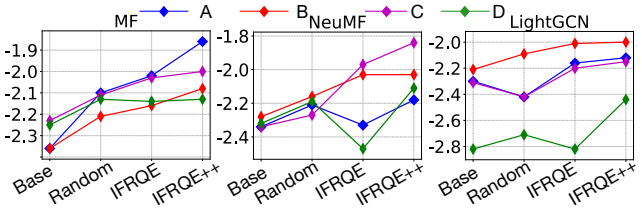

Figure 3: Model comparisons on the datasets with differently characterized user willingness.

improvements, while significantly reducing the training time required. Specifically, our framework achieves similar results to Proactive in a fraction of the time, suggesting its practicality and feasibility in real-world scenarios. Furthermore, in the comparison between IFRQE and IFRQE++, the latter consistently outperforms the former in a variety of cases. We hypothesize that the incorporation of multiple anchor vectors enables a more accurate approximation of the validation loss, facilitating the learning of optimal strategies and resulting in better rewards.

Approximation of the Validation Loss

The key of our framework for speeding up the training efficiency lies in the approximation of the validation loss. In this section, we study how large is approximation error induced by our framework. We present the performance of MF on datasets Diginetica and Amazon Video, with the complete results provided in the Appendix. We firstly generate ten selection vectors and train the base model to obtain the parameters and the corresponding real validation loss. Then, we approximate the validation loss by equation (4) and (8), where we use two anchor vectors in **IFRQE++**. At last, the (approximated) validation losses are reported by averaging over all the selection vectors, and we compute the approximation error by $\frac{|\text{real validation loss} - \text{approximated validation loss}|}{\text{real validation loss}}$. The results are presented in Figure 2(a), where we can see: the approximation accuracy of IFRQE and IFRQE++ are satisfied. Especially, on Amazon Video, the approximation error of **IFRQE++** is only 1.2%. By leveraging more anchor vectors, **IFRQE++** is more accurate than **IFRQE** on both datasets. In specific, the approximation error is reduced from 19.2% to 15.2% and 9.4% to 1.2% on the datasets of Diginetica and Amazon Video, respectively.

Influence of the Number of Anchor Vectors

In this section, we investigate the influence of parameter T on the final performance by tuning its value within the range of $[1, 10]$. Notably, $T = 1$ corresponds to the method IFRQE. Following the experimental settings described earlier, we present the reward and time cost for each T in Figure 2(b), with comprehensive results provided in the Appendix. From the results, we observe that the reward achieved by our framework is sub-optimal when $T = 1$, which aligns with the findings presented in Table 1. By increasing the number of anchor vectors, the reward improves. In most cases, as T increases, the reward initially rises before reaching an optimal point where it stabilizes. We speculate that initially, additional anchor vectors contribute to more accurate validation approximation, leading to better rewards. However, once a sufficient number of anchor vectors are included, further increasing T may not provide significant improvements as the validation approximation primarily relies on key anchor vectors. Regarding efficiency, as T increases, our model incurs a longer time cost. This outcome is expected since larger values of T necessitate more rounds of base recommender model retraining. Notably, these observations indicate that IFRQE++ offers an opportunity to balance the effectiveness and efficiency of the framework.

Influence of Different User Willingness

Here we study whether our model is always effective on differently characterized user willingness. We base the experiments on the simulation datasets, where the willingness of a user is controlled by the sensitive parameter s_u and average willingness level b_u. Considering that the distributions of s_u and b_u are determined by a_1, a_2 and a_3, we build four datasets by specifying (a_1, a_2, a_3) with $A = (0.6, 2, 1.2)$, $B = (0.4, 2, 0.8)$ and $C = (0.6, 1, 1.5)$ and $D = (0.5, 1, 1)$, respectively, to simulate different user willingness characters. We present the comparisons of different models on these datasets in Figure 3, where we can see: the random method is less competitive, and sometimes, it is even worse than the base model (*e.g.*, Datasets A and B with LightGCN as the base model). For different datasets and base models, IFRQE and IFRQE++ can usually obtain the second best and best performances. This result agrees with the observations in Table 1 and manifests that our methods are generally effective for different user willingness characters.

Conclusion and Future Work

This paper improved traditional system-centered recommendation paradigm by allowing users to participate into the model optimization process via specifying their willingness. We built two models based on influence function to efficiently solve the above problem, and also provided theoretical guarantees. Extensive experiments were conducted to demonstrate the effectiveness of our models. However, there are some limitations can be studied in the future. We do not consider the dynamic nature of the recommender system, therefore, an interesting direction is to study the temporal user willingness, for example, under online settings. Additionally, we assume that the user overall utility is a linear combination between the recommendation quality and user willingness. One can extend it to the non-linear case, where more efforts are needed to solve the multiplayer game.

Acknowledgements

This work is supported in part by National Natural Science Foundation of China (No. 62102420), Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, Public Computing Cloud, Renmin University of China, fund for building world-class universities (disciplines) of Renmin University of China, Intelligent Social Governance Platform. The work is sponsored by Huawei Innovation Research Programs. We gratefully acknowledge the support from Mindspore[9], CANN (Compute Architecture for Neural Networks) and Ascend Ai Processor used for this research.

References

Ben-Porat, O.; and Tennenholtz, M. 2018. A game-theoretic approach to recommendation systems with strategic content providers. *Advances in Neural Information Processing Systems*, 31.

Calamai, P. H.; and Moré, J. J. 1987. Projected gradient methods for linearly constrained problems. *Mathematical programming*, 39(1): 93–116.

Chen, C.; Sun, F.; Zhang, M.; and Ding, B. 2022a. Recommendation Unlearning. *arXiv preprint arXiv:2201.06820*.

Chen, C.; Zhang, J.; Tung, A. K.; Kankanhalli, M.; and Chen, G. 2020. Robust federated recommendation system. *arXiv preprint arXiv:2006.08259*.

Chen, X.; Xu, H.; Zhang, Y.; Tang, J.; Cao, Y.; Qin, Z.; and Zha, H. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*, 108–116.

Chen, Z.; Sun, F.; Tang, Y.; Chen, H.; Gao, J.; and Ding, B. 2022b. Proactively Control Privacy in Recommender Systems. *arXiv preprint arXiv:2204.00279*.

Chicco, D.; and Jurman, G. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1): 1–13.

Halkidi, M.; and Koutsopoulos, I. 2011. A game theoretic framework for data privacy preservation in recommender systems. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 629–644. Springer.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 639–648.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 173–182.

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, 197–206. IEEE.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, 1885–1894. PMLR.

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

Li, Y.; Zheng, X.; Chen, C.; and Liu, J. 2022. Making Recommender Systems Forget: Learning and Unlearning for Erasable Recommendation. *arXiv preprint arXiv:2203.11491*.

Lin, G.; Liang, F.; Pan, W.; and Ming, Z. 2020. Fedrec: Federated recommendation with explicit feedback. *IEEE Intelligent Systems*, 36(5): 21–30.

Parra, D.; and Brusilovsky, P. 2015. User-controllable personalization: A case study with SetFusion. *International Journal of Human-Computer Studies*, 78: 43–67.

Ricci, F.; Rokach, L.; and Shapira, B. 2015. Recommender systems: introduction and challenges. In *Recommender systems handbook*, 1–34. Springer.

Wang, L.; Zhang, J.; Chen, X.; Lin, Y.; Song, R.; Zhao, W. X.; and Wen, J.-R. 2023. RecAgent: A Novel Simulation Paradigm for Recommender Systems. *arXiv preprint arXiv:2306.02552*.

Wang, W.; Feng, F.; Nie, L.; and Chua, T.-S. 2022. User-controllable Recommendation Against Filter Bubbles. *arXiv preprint arXiv:2204.13844*.

Wang, X.; He, X.; Cao, Y.; Liu, M.; and Chua, T.-S. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 950–958.

Wang, Z.; Zhang, J.; Xu, H.; Chen, X.; Zhang, Y.; Zhao, W. X.; and Wen, J.-R. 2021. Counterfactual data-augmented sequential recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, 347–356.

Wu, C.; Wu, F.; Cao, Y.; Huang, Y.; and Xie, X. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925*.

Xu, L.; Jiang, C.; Chen, Y.; Ren, Y.; and Liu, K. R. 2018. User participation in collaborative filtering-based recommendation systems: a game theoretic approach. *IEEE transactions on cybernetics*, 49(4): 1339–1352.

Yang, L.; Tan, B.; Zheng, V. W.; Chen, K.; and Yang, Q. 2020. Federated recommendation systems. In *Federated Learning*, 225–239. Springer.

[9]https://www.mindspore.cn