Recommendation Unlearning via Matrix Correction

Jiahao Liu*, Dongsheng Li†, Hansu Gu, Tun Lu*, Jiongran Wu*, Peng Zhang*, Li Shang* and Ning Gu*
* *School of Computer Science, Fudan University,* Shanghai, China
† *Microsoft Research Asia,* Shanghai, China
jiahaoliu21@m.fudan.edu.cn, dongsli@microsoft.com, hansug@acm.org, lutun@fudan.edu.cn
23212010033@m.fudan.edu.cn, {zhangpeng_,lishang,ninggu}@fudan.edu.cn

***Abstract*—Recommender systems are important for providing personalized services to users, but the vast amount of collected user data has raised concerns about privacy (e.g., sensitive data), security (e.g., malicious data) and utility (e.g., toxic data). To address these challenges, recommendation unlearning has emerged as a promising approach, which allows specific data and models to be forgotten, mitigating the risks of sensitive/malicious/toxic user data. However, existing methods often struggle to balance completeness, utility, and efficiency, i.e., compromising one for the other, leading to suboptimal recommendation unlearning. In this paper, we propose an Interaction and Mapping Matrices Correction (IMCorrect) method for recommendation unlearning. Firstly, we reveal that many collaborative filtering (CF) algorithms can be formulated as mapping-based approach, in which the recommendation results can be obtained by multiplying the user-item interaction matrix with a mapping matrix. Then, IMCorrect can achieve efficient recommendation unlearning by correcting the interaction matrix and enhance the completeness and utility by correcting the mapping matrix, all without costly model retraining. Unlike existing methods, IMCorrect is a whitebox model that offers greater flexibility in handling various recommendation unlearning scenarios. Additionally, it has the unique capability of incrementally learning from new data, which further enhances its practicality. We conducted comprehensive experiments to validate the effectiveness of IMCorrect and the results demonstrate that IMCorrect is superior in completeness, utility, and efficiency, and is applicable in many recommendation unlearning scenarios.**

***Index Terms*—recommendation unlearning, recommender systems, collaborative filtering, machine unlearning**

I. Introduction

Recommender systems play a vital role in providing personalized services to users by learning their preferences through collected user data [1]. However, a major concern with these systems is the potential privacy breach since they retain the training data [2]–[4]. As a result, recommender systems may inadvertently leak user privacy [5]–[7]. To address this issue, users may want to delete specific interactions, preferences, or even all of their data, exercising their right to be forgotten as granted by recent regulations such as GDPR[1], CCPA[2], and PIPEDA[3]. This necessitates the ability of recommender systems to forget the knowledge they have acquired, a process known as *recommendation unlearning* [8]. Moreover, recommender systems are sensitive to data quality [9], and the presence of malicious, toxic, out-of-date, or out-of-distribution data can hurt their performance [2], [10], [11]. Removing such data can significantly improve the usability of the recommender system.

For recommendation unlearning to be effective, it should not only involve the deletion of collected user data but also the removal of the part of the model trained using that data. The process should adhere to three fundamental principles [8]:

- *Completeness*. This principle requires completely eliminating the influence of the target data on the model, ensuring that no traces of the forgotten information remain.
- *Utility*. The accuracy and performance of the model after the unlearning process should be comparable to its performance before the forgetting process.
- *Efficiency*. The unlearning process should be efficient in terms of time and computational resources.

One straightforward method to achieve both completeness and utility is to retrain the model from scratch. However, this approach is impractical due to its high training cost, making it inefficient for real-world applications. Fig. 1 shows the process of recommendation unlearning.

To ensure the efficiency of forgetting, existing recommendation unlearning methods have made compromises between completeness or utility and efficiency. RecEraser [8] is based on the divide-aggregate architecture. It first divides the data into several non-overlapping shards and then trains corresponding sub-models using data from each shard. Finally, the results of each sub-model are aggregated. When data needs to be forgotten, only the sub-model corresponding to the shard containing the target data and the aggregation layer need to be retrained. On one hand, by avoiding the need to retrain the model on the entire dataset, the efficiency of forgetting is improved. On the other hand, since dividing data into shards inevitably leads to the loss of collaborative information, the utility of the model is compromised. This type of method, which improves the retrain efficiency while ensuring complete forgetting, is known as *exact recommendation unlearning*. Several other methods [12]–[14] achieve unlearning using fine-tuning without changing the original workflow. When data needs to be forgotten, they use influence functions to quantify the impact of the target data on the model and further eliminate

[1]https://gdpr-info.eu
[2]https://oag.ca.gov/privacy/ccpa
[3]https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda

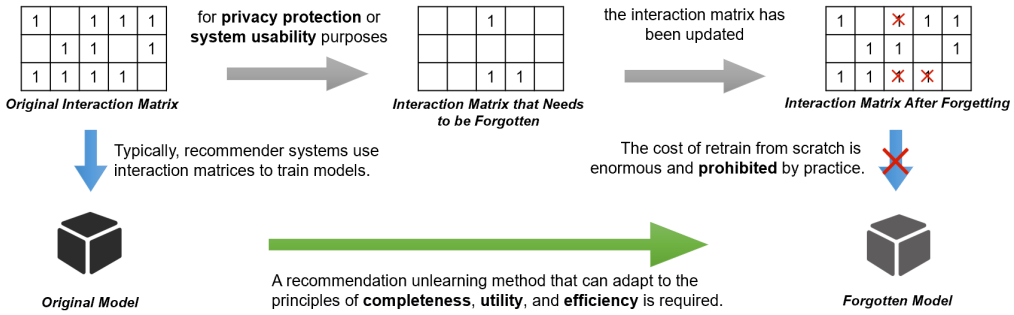

Fig. 1. The process of recommendation unlearning.

this influence. Due to the need of reducing the number of updated parameters in such methods to make the calculation of the Hessian matrix feasible, it may raise concerns about completeness. This type of method, which avoids retraining, and sacrifices the completeness of forgetting while improving efficiency, is called *approximate recommendation unlearning*.

Collaborative Filtering (CF) has been widely used to provide personalized recommendations to users by modeling their interactions with items [15]–[17]. In this paper, we focus on the form of obtaining prediction matrices in CF models and classify them into two categories: *embedding-based CF methods* and *mapping-based CF methods*. In the embedding-based CF methods, embeddings for users and items are first obtained, and then scores for users and items are often calculated based on the similarity between these embeddings to generate the prediction matrix, e.g., dot product. On the other hand, mapping-based CF methods directly map the interaction matrix to the prediction matrix via a mapping matrix. We conduct a thorough analysis of the mapping-based CF methods and reveal that the mapping matrix holds the essence of the similarity matrix between items, encompassing global structural information. Moreover, we show that the embedding-based CF methods can be naturally formulated as mapping-based CF methods, decoupling user interaction data from item similarity.

Based on the mapping matrix formulation, we propose a model-agnostic recommendation unlearning method named **IMCorrect** (**I**nteraction and **M**apping Matrices **Correct**ion). IMCorrect achieves highly efficient forgetting by correcting only the interaction matrix without the need for costly retraining, while preserving the model's performance. Additionally, we analyze how the target data influences the mapping matrix and subsequently correct it to eliminate this influence, enhancing the completeness and utility of unlearning. The whitebox nature of IMCorrect makes it a flexible approach compared to existing methods, enabling it to address various recommendation unlearning scenarios effectively.

Specifically, we analyze IMCorrect in three different application scenarios: 1) handling out-of-distribution data, 2) handling out-of-date data, and 3) addressing attack data. We also explore how IMCorrect can be applied to recommendation unlearning while adhering to privacy protection requirements. A noteworthy advantage of IMCorrect is its capability to not only forget existing interaction data (decremental learning) but also learn from new interaction data (incremental learning), accommodating recommender systems that continuously update their models due to evolving user preferences [18]–[21].

We instantiate IMCorrect on SLIM [22], GF-CF [23], and MF [15] and demonstrate its superiority through comprehensive experiments. Firstly, we validate the general capabilities (completeness, utility, and efficiency) of IMCorrect under three types of attacks. Then, we perform case studies on recommendation unlearning tasks in various real-world scenarios to demonstrate the abilities of IMCorrect in different situations. Experimental results show that IMCorrect achieves outstanding performance in completeness, utility, and efficiency, while effectively handling real-world unlearning tasks.

Our main contributions can be summarized as follows:

- By analyzing the mapping matrix and its implications in CF methods, we shed light on the inner workings of mapping-based CF methods and their connections to embedding-based CF methods.
- We proposed a new model-agnostic method for approximate recommendation unlearning, IMCorrect, which is a whitebox model that offers greater flexibility, making it easily applicable to various recommendation scenarios.
- We instantiated IMCorrect on various CF methods and conducted comprehensive experiments to validate its superiority in terms of general capabilities and scenario-specific abilities, showcasing its practical value in real-world applications.

II. Related Work

In this section, we introduce the related works on collaborative filtering and recommendation unlearning.

A. Collaborative Filtering

Collaborative filtering is a kind of widely used recommendation algorithm that extracts users' preferences and item attributes from user-item interaction data to provide personalized recommendations [24]–[26]. Recommender systems effectively alleviate the problem of information overload, and collaborative filtering algorithms have continuously evolved since their inception.

User-based Collaborative Filtering (User-CF) [27] and Item-based Collaborative Filtering (Item-CF) [28] calculate Pearson correlation coefficients between users and items based on their interaction records to generate recommendations. To address the sparsity problem caused by incomplete data, Matrix Factorization (MF) techniques have been proposed and applied in collaborative filtering [15], [16], [29]. MF models user interests and item attributes as latent vectors in the same latent space, composed of several latent factors, and measures the interaction scores between users and items through dot products. The latent vectors can be obtained through Singular Value Decomposition (SVD) [30], reconstruction loss optimization [16], or Bayesian Personalized Ranking (BPR) [17]. SLIM [22] and EASEˆR [31] efficiently and accurately provide top-K recommendations through sparse linear methods.

In recent years, collaborative filtering algorithms combined with deep learning have become very popular. AutoRec [32] predicts users' interests using an autoencoder architecture, where the encoder and decoder are implemented through multi-layer perceptrons (MLP). NCF [33] introduces non-linear mechanisms in neural networks to enhance the expressive power of recommendation models. NGCF [34] further considers multi-hop information in the user-item interaction graph using graph neural networks. LightGCN [35] removes some unnecessary modules from NGCF to achieve more efficient and accurate predictions.

B. Recommendation Unlearning

Machine unlearning [4], also known as selective forgetting [36] or data removal/deletion [37], refers to the process of removing the influence of specific data from a trained model. Existing machine unlearning works often focus on tasks in computer vision or natural language processing [4], [38], [39]. Due to architectural differences, these methods cannot be directly applied to recommendation tasks [12]. Unlike differential privacy [40]–[42], which focuses on protecting rather than deleting private information, recommendation unlearning requires the recommender system to be able to delete user data and its associated model. According to the degree of achievement regarding the completeness principle, recommendation unlearning can be divided into exact recommendation unlearning and approximate recommendation unlearning.

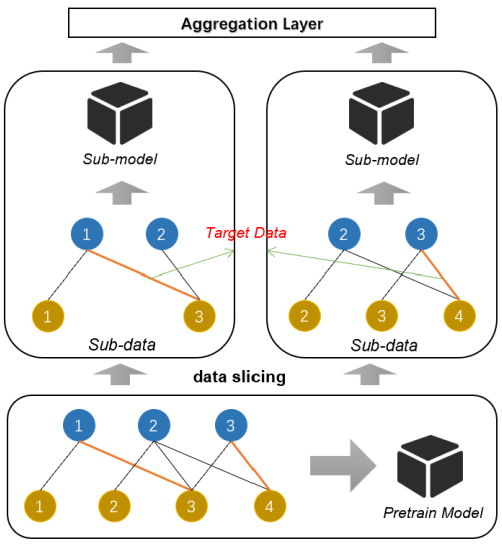

Fig. 2. The process of RecEraser.

1) Exact Recommendation Unlearning: Exact unlearning refers to methods that can guarantee complete forgetting of data (completeness) [4], [43], [44]. These methods often use a divide-aggregate architecture to improve retraining efficiency by partitioning the model [38], [45] and dividing the data [4], [46], [47]. SISA (Sharded, Isolated, Sliced, and Aggregated) [4] reduces the amount of data involved in retraining by dividing the data into several disjoint shards. For graph data, GraphEraser [4] uses a balanced graph partition method during sharding and employs a learning-based method for static weighting during aggregation.

Following the concepts of SISA and GraphEraser, RecEraser [8] introduces three novel data partition algorithms to divide the training data into balanced groups, ensuring that similar users and/or items are grouped together and protects collaborative information to a certain extent, which is crucial for collaborative filtering algorithms. RecEraser then uses sub-models to train each group separately. Finally, considering that different subgroups contribute differently to different user-item pairs, RecEraser proposes an adaptive attention-based aggregation method to further enhance model utility. When data needs to be forgotten, only the sub-model corresponding to the shard containing the target data needs to be retrained. Since it involves retraining, completeness can be ensured. LASER [48] is concurrent work with RecEraser, following a similar approach but with different specific details.

We acknowledge the contributions of RecEraser, but there are some shortcomings that need to be addressed. For ease of description, we have illustrated the process of data slicing, sub-model training, aggregation, and forgetting in RecEraser, as shown in Fig. 2.

- Collaborative filtering, as an association-sensitive task, contains a significant amount of collaborative information in the data [49]. Although RecEraser tries to preserve collaborative information as much as possible through carefully designed sharding and aggregation, the partitioning operation inevitably disrupts collaborative information, leading to a degradation of model utility. As shown in Fig. 2, interactions of the same user or item may be stored in different shards, making it challenging for sub-models to fully capture user interaction patterns.
- RecEraser can only handle individual forgetting requests, and when requests arrive in batch form, the efficiency gained from sharding for forgetting may diminish. As shown in Fig. 2, when multiple target data need to be forgotten simultaneously, the system may be required to retrain multiple sub-models simultaneously.
- RecEraser does not completely avoid retraining, which reduces its efficiency. As shown in Fig. 2, both the

pretrain phase and the aggregation phase require training with all the data, and each forgetting process requires retraining the sub-model and the aggregation layer.
- The divide-aggregate architecture on which RecEraser relies has its own drawbacks: 1) it disrupts the original workflow and is difficult to deploy; 2) we found in experiments that the performance of sub-models varies significantly, resulting in high variance; 3) as shown in Fig. 2, the architecture may store multiple sets of parameters for the same user or item, leading to a substantial increase in parameter size.

2) Approximate Recommendation Unlearning: Due to the aforementioned shortcomings, approximate recommendation unlearning has become a hot topic in current research. Approximate unlearning refers to methods that can sacrifice some completeness to improve the efficiency of forgetting and the utility of the model, achieving only statistical forgetting [50]–[52]. These methods often directly eliminate the influence of target data on the model through inverse gradient-based update strategies to achieve forgetting without retraining. The tool used to measure the influence of target data on a trained model is called the influence function, which is a classic concept originating from robust statistics [53]–[55]. Since calculating the Hessian matrix is required, existing work that applies influence functions for machine unlearning needs to optimize for efficiency [56]–[58]. The influence function has been widely used in recommendation tasks, for example, some works use influence functions to achieve recommendation attacks [59]–[62], recommendation explanations [63], and recommendation debiasing [64].

Existing recommendation unlearning works have adapted the use of influence functions from classification or regression tasks [36], [37], [65], [66] to the collaborative filtering task. Their approaches involve utilizing influence functions with 2nd-order optimization methods, such as Newton and quasi-Newton methods, which are advantageous due to their fast final convergence, to accelerate the finetuning process [67]. The differences between these works mainly lie in how they optimize to speed up the computation of the Hessian matrix. AltEraser [12] decomposes the large optimization problem into several smaller independent sub-problems using alternating optimization [68], [69]. Alternating optimization allows each sub-problem to be solvable and they can be solved in parallel, thus improving optimization efficiency. In the specific implementation, AltEraser also utilizes optimization techniques like Hessian-free Newton [70], Ad Hoc Newton [65], and others. SCIF [13] selectively updates user embeddings to reduce the number of parameters that need to be updated and preserves the collaborative information across the remaining users and items to enhance the model's utility. SCIF computes the Hessian-Vector Product following [71] and [54] to reduce computational overhead. IFRU [14] extends the influence function to quantify the influence of unusable data in the computational graph aspect and proposes an importance-based pruning algorithm to reduce the cost of the influence function.

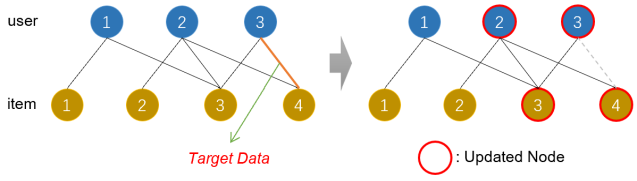

Fig. 3. The recommendation unlearning method based on the influence function only updates the target data and its neighbors.

Yuan et al. [72] proposed an unlearning method for federated recommendation, which removes a user's contribution by rolling back and introduces a small-sized negative sampling method to reduce the number of item embedding updates, along with an importance-based update selection mechanism to store only important model updates. Unlearn-ALS [73] modifies the fine-tuning procedure under Alternating Least Squares optimization for bi-linear recommendation models. Approximate recommendation unlearning cannot guarantee completeness, so they use membership inference [5], [74]–[77], a binary classification problem that determines whether the target data appears in the training set of the model after forgetting, to check the completeness of forgetting.

Due to the inability of approximate recommendation unlearning methods to efficiently compute the Hessian matrix, they each reduce the number of parameters that need to be updated in different ways. While this approach can improve the efficiency of forgetting, it also means that **these methods are unable to completely eliminate the influence of target data**, especially in non-convex models like deep neural networks. As shown in Fig. 3, to reduce the number of updated parameters, these methods only update the embeddings of the target data and its limited neighbors, which results in an incomplete eradication of the influence of target data on the model.

III. Analysis

In this section, we first introduce four representative collaborative filtering methods. Then, we analyze the essence of the mapping matrix in mapping-based CF methods. Finally, we explain how to compute the mapping matrix for embedding-based CF methods to formulate them as mapping-based CF methods. Assume that there are m users and n items, and the interaction matrix $R \in \{0,1\}^{m\times n}$ indicates the interactions between users and items, where the value 1 represents an interaction, and 0 represents no interaction. We have listed some important symbols and their definitions in the TABLE I.

A. Details of SLIM, GF-CF, AutoRec and MF

In this section, we first introduce four representative collaborative filtering methods: SLIM [22], GF-CF [23], AutoRec [32], and MF [15], [30]. For simplicity of exposition, we omit regularization terms for all methods.

1) SLIM learns a mapping matrix $W_{SLIM} \in \mathbb{R}^{n\times n}$ through the following optimization problem:

$$\begin{aligned} W_{SLIM} &= \arg\min_{W} \|R - RW\|_2, \\ &\text{s.t. } W \geq 0 \text{ and } diag(W) = 0. \end{aligned} \tag{1}$$

TABLE I
THE NOTATIONS USED IN THIS PAPER

Symbol	Definition
R	original interaction matrix
W	original mapping matrix
$\bar{R}$	interaction matrix that needs to be forgotten
$\bar{W}$	mapping matrix that needs to be forgotten
$\tilde{R}$	interaction matrix after forgetting
$\tilde{W}$	mapping matrix after forgetting
$\hat{R}$	predicted interaction matrix
P	user embedding matrix
Q	item embedding matrix
$\mathbf{a}_{i\cdot}$	the i-th row of matrix A
$\mathbf{a}_{\cdot j}$	the j-th column of matrix A
a_{ij}	the element of the i-th row and j-th column of matrix A

Then, the prediction matrix is computed as follows:

$$\hat{R}_{SLIM} = RW_{SLIM}. \tag{2}$$

2) GF-CF first performs truncated singular value decomposition (SVD) on R:

$$U, \Sigma, V = svd_k(R), \tag{3}$$

where $U \in \mathbb{R}^{m\times k}$, $\Sigma \in \mathbb{R}^{k\times k}$, and $V \in \mathbb{R}^{n\times k}$. Then, it constructs a low-pass graph filter $W_{GF-CF} \in \mathbb{R}^{n\times n}$ as follows:

$$W_{GF-CF} = VV^\top. \tag{4}$$

Finally, the prediction matrix is computed as follows:

$$\hat{R}_{GF-CF} = RW_{GF-CF} = RVV^\top. \tag{5}$$

Here, R multiplied by V in the first step represents mapping the interaction matrix to the Fourier space and keeping only the low-frequency components (Fourier transformation), and the subsequent multiplication by $V^\top$ represents mapping back the low-frequency signals in the Fourier space to the original space (inverse Fourier transformation).

3) AutoRec utilizes an autoencoder $W_{AutoRec} : \mathbb{R}^{z^+\times n} \rightarrow \mathbb{R}^{z^+\times n}$ to compress and reconstruct R:

$$W_{AutoRec}(R) = d(e(R)), \tag{6}$$

where $e : \mathbb{R}^{z^+\times n} \rightarrow \mathbb{R}^{z^+\times k}$ and $d : \mathbb{R}^{z^+\times k} \rightarrow \mathbb{R}^{z^+\times n}$ are encoding and decoding functions realized by multi-layer perceptrons (MLPs), respectively, and z^+ represents any positive integer. By minimizing the reconstruction loss, the optimal autoencoder $W^*_{AutoRec}$ is obtained:

$$W^*_{AutoRec} = \arg\min_{W_{AutoRec}} \|R - W_{AutoRec}(R)\|_2. \tag{7}$$

Finally, the prediction matrix is computed as follows:

$$\hat{R}_{AutoRec} = W^*_{AutoRec}(R). \tag{8}$$

4) Matrix factorization (MF) can be implemented in various ways. After obtaining U, Σ, and V through truncated SVD using (3), Sarwar et al. [30] computed user embeddings P_{SVD} and item embeddings Q_{SVD} as follows:

$$P_{SVD} = U\Sigma^{1/2}, Q_{SVD} = V\Sigma^{1/2}. \tag{9}$$

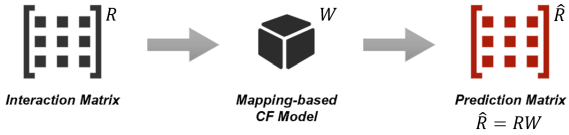

Fig. 4. The process of obtaining the prediction matrix by mapping-based CF methods.

Then, the prediction matrix is computed as follows:

$$\hat{R}_{SVD} = P_{SVD}Q^\top_{SVD} \tag{10}$$

Hu et al. [15], on the other hand, obtain user embeddings $P_{Re} \in \mathbb{R}^{m\times k}$ and item embeddings $Q_{Re} \in \mathbb{R}^{n\times k}$ by minimizing the reconstruction loss as follows:

$$P_{Re}, Q_{Re} = \arg\min_{P,Q} \|R - PQ^\top\|_2. \tag{11}$$

Then, the prediction matrix is computed as follows:

$$\hat{R}_{Re} = P_{Re}Q^\top_{Re}. \tag{12}$$

In fact, $\hat{R}_{SVD}$ represents the best rank-k approximation of R under the squared reconstruction loss.

We have introduced classic and representative methods, while many recent works can be seen as extensions of them. For example, NCF [33] and LightGCN [35] can be viewed as improvements to MF, differing only in the way they obtain user embeddings and item embeddings. However, recent research has shown that some of these newly proposed methods do not always outperform these classic methods, and mapping-based CF methods and embedding-based CF methods each have their advantages and disadvantages on different datasets [78].

B. Mapping Matrix of Mapping-based CF Methods

As shown in Fig. 4, for SLIM, GF-CF, or AutoRec, they can all be unified in the form as the result of multiplying the interaction matrix R with a mapping matrix $W \in \mathbb{R}^{n\times n}$, to yield the prediction matrix:

$$\hat{R} = RW. \tag{13}$$

In the case of AutoRec, its encoder and decoder use MLPs to approximate the Fourier transform and inverse transform processes in GF-CF.

Equation (13) shows that each column of $\hat{R}$ is a linear combination of all columns of R, where the combination coefficients are given by the corresponding column of W. For example, for the j-th column of $\hat{R}$, we have:

$$\hat{\mathbf{r}}_{\cdot j} = \sum_{i=1}^{n} w_{ij}\mathbf{r}_{\cdot i}. \tag{14}$$

Essentially, all mapping-based CF methods minimize the reconstruction loss, aiming to make R and RW to be as close as possible. In the case of GF-CF, W is obtained through numerical methods, while SLIM and AutoRec optimize the reconstruction loss using (1) and (7) to obtain W respectively. It is evident that the reconstruction loss is minimized when

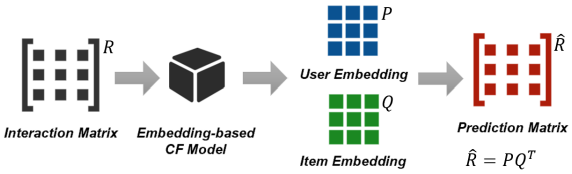

Fig. 5. The process of obtaining the prediction matrix by embedding-based CF methods.

W is an identity matrix. To avoid finding trivial solutions, SLIM imposes a constraint that the diagonal elements of W are set to zero, meaning that a column cannot participate in its own reconstruction. On the other hand, AutoRec and GF-CF impose rank constraints on W.

Generally, the more similar (i.e., higher co-occurrence frequency) the interaction patterns between two items, the larger the corresponding weight (combination coefficient) should be when reconstructing through linear combinations. Since the reconstruction loss is related to the interaction patterns of items and is a global measure, **the essence of W reflects the similarity between items with the ability to capture global structural information**. Additionally, the spectral domain operations in GF-CF enable the consideration of global structural information in the spatial domain, further supporting the interpretation of W as a measure of item similarity with global structural information.

C. Mapping Matrix in Embedding-based CF Methods

After recognizing that the essence of the mapping matrix is a measure of item similarity that considers global structural information, we can compute it from the embedding-based CF methods. Specifically, as shown in Fig. 5, embedding-based CF methods first obtain user embeddings P and item embeddings Q, which also contain global structural information. This means that users and items with similar interaction patterns have similar embeddings. We can compute the mapping matrix as follows:

$$W = QQ^T, \tag{15}$$

and then use (13) to calculate the prediction matrix $\hat{R}$.

This implies that embedding-based CF methods can be naturally transformed into mapping-based CF methods. Although the resulting prediction matrix may differ from $PQ^\top$ to some extent, this transformation has significant advantages for recommendation unlearning. In mapping-based CF methods, the personal interaction records of users (sensitive information) are decoupled from the item similarity information. On the other hand, in the embedding-based CF methods, user embeddings and item embeddings are semantically coupled together, which can cause trouble for recommendation unlearning.

IV. Method

In this section, we first describe the recommendation unlearning problem, then introduce our proposed method, IMCorrect, and finally explain how IMCorrect can be applied in different scenarios.

A. Problem Description

Assuming that we have obtained the original mapping matrix W based on the original interaction matrix R using the methods described in Section III and have obtained the original prediction matrix $\hat{R}$ according to (13). Now, a small portion of elements in R, denoted as $\bar{R} \in \{0,1\}^{m\times n}$, has been flipped to create a new interaction matrix $\tilde{R} \in \{0,1\}^{m\times n}$. The problem is how to quickly obtain the new prediction matrix $\hat{\tilde{R}}$.

Since IMCorrect has the ability to both forget existing interaction data (decremental learning) and learn new interaction data (incremental learning), the flipping can be either a negative flip (from 1 to 0) or a positive flip (from 0 to 1). We assume that the interactions in $\bar{R}$ are either all part of positive flips or all part of negative flips.

If an element is updated in the negative flips, it means that its corresponding interaction needs to be unlearned, which refers to recommendation unlearning. In this case, we have:

$$\tilde{R} = R - \bar{R}. \tag{16}$$

If an element is updated in the positive flips, it means that its corresponding interaction needs to be learned, which refers to incremental recommendation. In this case, we have:

$$\tilde{R} = R + \bar{R}. \tag{17}$$

This paper mainly focuses on the performance of IMCorrect in the recommendation unlearning task, but IMCorrect can be easily extended to incremental learning.

B. IMCorrect

In IMCorrect, we can achieve recommendation unlearning by correcting the interaction matrix and/or the mapping matrix.

1) Correction of Interaction Matrix: In Section III-B, we explained the meaning of the mapping matrix W in mapping-based CF methods from a reconstruction perspective during the learning phase. In the inference phase, after learning the mapping matrix W, each row of the prediction matrix in (13) can be considered as the result of aggregating a user's interaction history using W, where the u-th row of $\hat{R}$ is:

$$\begin{aligned}\hat{\mathbf{r}}_{u\cdot} = \mathbf{r}_{u\cdot}W &= \mathbf{r}_{u\cdot}(\mathbf{w}_{\cdot 1}, ..., \mathbf{w}_{\cdot n}) \\ &= (\mathbf{r}_{u\cdot}\cdot\mathbf{w}_{\cdot 1}, ..., \mathbf{r}_{u\cdot}\cdot\mathbf{w}_{\cdot n}) \\ &= (\Sigma_{i=1}^{n} r_{ui}w_{i1}, ..., \Sigma_{i=1}^{n} r_{ui}w_{in}).\end{aligned} \tag{18}$$

This implies that mapping-based CF methods essentially use a matrix W, which contains global structural information of item-item similarities, to aggregate the interaction matrix and achieve smoothing of the original interaction matrix R, resulting in the prediction matrix $\hat{R}$.

In (18), if a user u has interacted with item i ($r_{ui} = 1$), then item i will contribute to the aggregation score for user u, with the weights being the similarity between item i and the corresponding items. On the other hand, if a user u has not interacted with item i ($r_{ui} = 0$), then item i will not contribute to the aggregation score for user u. This indicates that, for a trained mapping matrix, users' interaction history

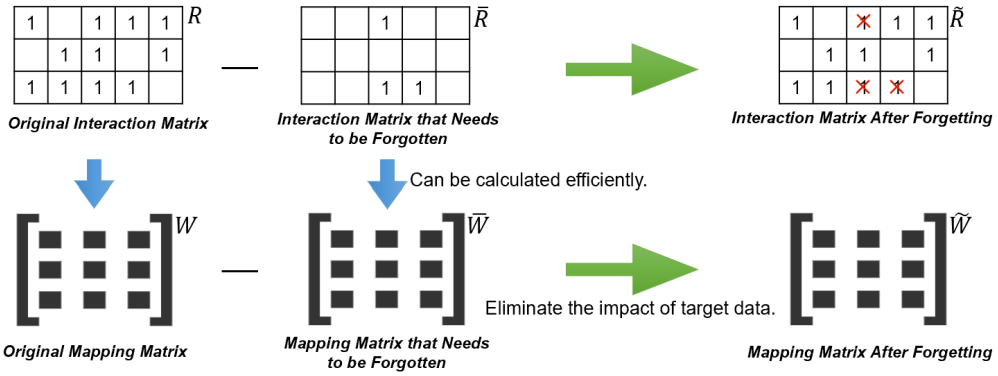

Fig. 6. The unlearning process of IMCorrect.

(the rows in the interaction matrix) completely determines their recommendation results. Therefore, we can correct the interaction matrix to achieve recommendation unlearning (or incremental recommendation):

$$\hat{\tilde{R}} = \tilde{R}W. \tag{19}$$

This means that with no additional training cost, we can achieve recommendation unlearning (or incremental recommendation).

2) Correction of Mapping Matrix: The slight changes in the interaction matrix will obviously lead to slight changes in the mapping matrix W. While (19) is already capable of handling some recommendation unlearning tasks, in order to correct the prediction matrix more accurately and thoroughly, it is necessary to calculate a new mapping matrix $\tilde{W}$ based on the new interaction matrix $\tilde{R}$. However, instead of obtaining a new mapping matrix $\tilde{W}$ through retraining as described in Section III, which is highly inefficient, we achieve this by correcting the original mapping matrix W.

The original mapping matrix W is learned from the original interaction matrix R using the method described in Section III, which considers both the new interaction matrix $\tilde{R}$ and interaction matrix $\bar{R}$ that needs to be forgotten. As we have recognized, the essence of the mapping matrix is a similarity matrix, and in the collaborative filtering scenario, item-item similarities are determined by the topological relationships in the bipartite graph composed of users and items. Therefore, **the essence of the problem in correcting the mapping matrix is how to quantify the impact of updates in the link relationships between nodes in the bipartite graph on item-item similarities and eliminate this impact**.

Taking negative flip as an example, it refers to the process of forgetting $\bar{R}$, the interactions that need to be updated from the original interaction matrix R, resulting in a new interaction matrix $\tilde{R}$.

On one hand, this process reduces the co-occurrence frequency between items in $\bar{R}$ and other items, leading to a decrease in their similarity. The extent of reduction depends on the number of times an item appears in $\bar{R}$, and we represent this effect as follows:

$$W' = (\frac{c_1 - \bar{c}_1}{c_1}\mathbf{w}_{\cdot 1}, ..., \frac{c_n - \bar{c}_n}{c_n}\mathbf{w}_{\cdot n}), \tag{20}$$

where $c_i = \Sigma_{u=1}^{m} r_{ui}$ and $\bar{c}_i = \Sigma_{u=1}^{m} \bar{r}_{ui}$ are the number of interactions of item i in the original interaction matrix R and the interaction matrix $\bar{R}$ that needs to be forgotten, respectively. For positive flip, a similar conclusion can be drawn, where the newly added interactions in $\bar{R}$ increase the similarity between items, resulting in

$$W' = (\frac{c_1 + \bar{c}_1}{c_1}\mathbf{w}_{\cdot 1}, ..., \frac{c_n + \bar{c}_n}{c_n}\mathbf{w}_{\cdot n}). \tag{21}$$

Equations (20) and (21) indicate that the more times an item appears in $\bar{R}$, the greater its impact on the similarity with other items.

On the other hand, for negative flip, since the interactions in $\bar{R}$ are removed, the pairs of items that have high similarity due to similar interaction patterns in $\bar{R}$ should have lower similarity in $\tilde{W}$ than in W'. For positive flip, the pairs of items that have high similarity in $\bar{R}$ due to similar interaction patterns should have higher similarity in $\tilde{W}$ than in W', as the interactions in $\bar{R}$ are added to R. Since the number of interactions in $\bar{R}$ is much lower than that in R and $\tilde{R}$, the method introduced in Section III can quickly learn the mapping matrix $\bar{W}$ corresponding to $\bar{R}$. After obtaining $\bar{W}$, we perform column-wise weighting on it as follows:

$$\bar{W}' = (\frac{\bar{c}_1}{c_1}\bar{\mathbf{w}}_{\cdot 1}, ..., \frac{\bar{c}_n}{c_n}\bar{\mathbf{w}}_{\cdot n}). \tag{22}$$

Since the number of interactions in $\bar{R}$ is much lower than that of R, the reliability of similarity relationships in $\bar{W}$ obtained based on $\bar{R}$ should be much lower than that in W' obtained based on R. The column-wise weighting takes into account the difference in reliability due to the number of item interactions. We use $\bar{W}'$ to quantify the impact of the updated link relationships $\bar{R}$ on item-item similarities. For negative flip, item pairs with high similarity in $\bar{W}'$ should be weakened in $\tilde{W}$ as follows:

$$\tilde{W} = W' - \bar{W}'. \tag{23}$$

TABLE II
THREE APPLICATION SCENARIOS OF IMCORRECT FOR IMPROVING SYSTEM USABILITY

Scenario	Out-of-distribution data	Out-of-date data	Attack data
Explanation	Interactions outside a user's true interests.	Interactions caused by outdated short-term user interests.	Malicious interactions initiated by users.
Case	A man who is not interested in female products buys a lipstick as a birthday gift for his girlfriend on an e-commerce platform. As a result, the e-commerce platform keeps recommending products related to lipsticks based on this purchase record. In this case, the man may want the system to forget this lipstick purchase interaction.	A user has already purchased a smartphone on an e-commerce platform and no longer intends to make any more purchases. However, the recommender system continues to suggest smartphones to this user based on their recent search and browsing history. In this scenario, the system should initiate forgetting of the user's recent behavior.	On a short video platform, some authors may generate fake views to make their videos appear popular, tricking the recommender system into promoting these videos to more users. Once the system identifies such interactions, it should promptly forget them to reduce the likelihood of promoting such videos to other users.
Usage	Both the interaction matrix and the mapping matrix need to be updated.	Only the interaction matrix needs to be updated.	Only the mapping matrix needs to be updated.

For positive flip, item pairs with high similarity in $\bar{W}'$ should be enhanced in $\tilde{W}$ as follows:

$$\tilde{W} = W' + \bar{W}'. \tag{24}$$

Finally, we obtain the new prediction matrix as follows:

$$\hat{\tilde{R}} = \tilde{R}\tilde{W}. \tag{25}$$

Fig. 6 shows the unlearning process of IMCorrect. Note that IMCorrect only requires a model that can provide a mapping matrix, without caring about specific implementation methods.

C. Application Scenarios

In the previous section, we have introduced two ways to achieve recommendation unlearning for the mapping-based CF methods. The first approach involves correcting only the interaction matrix as (19), while the second approach involves correcting both the interaction matrix and the mapping matrix simultaneously as (25). In fact, in some cases, we can achieve recommendation unlearning by only correcting the mapping matrix:

$$\hat{\tilde{R}} = R\tilde{W}. \tag{26}$$

A User's recommendation results are influenced by both her/his interaction history and the mapping matrix. **Correcting a user's interaction records (a row in the interaction matrix) will only affect the user's recommendation results, while correcting the mapping matrix will affect the recommendation results of all users.** Whether to correct the mapping matrix should consider whether a recommendation unlearning should have an impact on the recommendation results of other users.

In this section, we present some typical scenarios in which IMCorrect can be used to achieve recommendation unlearning, taking into account the requirements for usability and privacy protection.

1) Usability: As shown in TABLE II, we have listed three scenarios for recommendation unlearning based on considerations of system usability: out-of-distribution data, out-of-date data, and attack data.

For out-of-distribution data, interactions that fall outside of a user's true interests often do not exhibit specific patterns and may have a certain degree of randomness, which can affect the calculation of item similarities in the mapping matrix. Therefore, when users require recommendation unlearning for this purpose, both the interaction matrix and the mapping matrix need to be corrected.

For out-of-date data, some platforms capture users' short-term interests to provide better services, which is crucial. However, in some cases, users' short-term interests can vanish instantly due to certain events. The effect of IMCorrect is immediate, which is very beneficial for some special scenarios. Compared to the examples in TABLE II, there are some examples that may be more urgent. For example, a person may have just experienced a failed relationship and may not want the recommender systems to recommend memory-evoking items. IMCorrect can make quick changes to this situation. Since out-of-date data is not harmful data, when users request recommendation unlearning for this purpose, there is no need to correct the mapping matrix, and only the interaction records of the target user need to be corrected.

For attack data, they often exhibit certain patterns that can severely affect the calculation of item similarities in the mapping matrix, thus impacting the overall effectiveness of the recommender system. When one requests recommendation unlearning for this purpose, the main goal is to mitigate the impact of the attack data on the overall recommendation performance. Therefore, there is no need to modify users' interaction records, and only the mapping matrix needs to be corrected.

In summary, depending on the specific purpose and requirements of recommendation unlearning, different strategies can be adopted, including correcting both the interaction matrix and the mapping matrix, correcting only the interaction matrix, or correcting only the mapping matrix. IMCorrect provides the flexibility to address various recommendation unlearning scenarios based on specific needs.

2) Privacy: When users request recommendation unlearning for privacy protection purposes, there are three levels of forgetting: 1) interaction level, which means the user's request is to forget a single interaction in the system; 2) preference level, which means the user's request is to forget all inter-

TABLE III
DATASET STATISTICS

Dataset	# User	# Item	# Interaction	Density
ML-1M	5741	2803	824857	5.1%
Gowalla	5992	5639	281412	0.8%
Yelp	18728	15295	888154	0.3%

actions highly related to a specific interaction (for mapping-based CF methods, identifying highly related interactions is straightforward due to the stored similarity relationships); and 3) account level, which means the user's request is to delete all of her/his interactions from the system.

In practice, mapping-based CF methods decouple users' sensitive data (interaction matrix) from model parameters (mapping matrix), so deleting the corresponding interaction records can ensure user privacy without the need to modify the mapping matrix. Additionally, the data removed for privacy protection purposes is often not harmful data (e.g., out-of-distribution data, attack data), so deleting the model trained with such data might impact the recommendation performance. However, when recommendation unlearning is requested for privacy protection purposes, the system is required by regulations to delete the model trained with the forgotten data.

We believe that when users request recommendation unlearning for privacy protection, they should be given the option to choose whether to keep the model trained with their interaction data or not, in addition to deleting the interaction records. This may depend on further legal considerations. In practice, some recommendation unlearning methods already adopt this approach. For example, SCIF [13] only updates the target user's embedding without considering the impact of forgetting the target data on the embeddings of other users and items. This means the post-forgetting model retains the knowledge learned from the target data. In contrast, IMCorrect provides users with the option to choose whether they want the model to be trained with their data or not.

V. EXPERIMENT

This section validates the superiority of IMCorrect through comprehensive experiments. Specifically, we first introduce the experimental settings, and then address the following two research questions (RQs):

- RQ1: How does IMCorrect perform in terms of completeness, utility, and efficiency?
- RQ2: How does IMCorrect perform in different unlearning scenarios?

A. Setting

In this section, we introduce the experimental datasets, the backbones used to instantiate IMCorrect, the compared baselines, and the hyperparameters of the models.

1) Datasets: We conducted experiments on three publicly available datasets:

- ML-1M[4]: A movie rating dataset collected from the MovieLens website [79], where an interaction is considered to have occurred if a user rated a movie.
- Gowalla[5]: A location-based social networking website where users share their locations by checking in [80], and an interaction is considered to have occurred if a user checked in at a certain location.
- Yelp[6]: A user review dataset in which users post reviews on the restaurants they visited. Each user review is considered an interaction.

Their statistical information is shown in TABLE III, representing different data sizes and densities. If a dataset contains ratings, we only retain interactions with ratings greater than or equal to 3. For each dataset, we adopt the 20-core setting [81], which means that we only keep users and items with more than 20 interactions. We divided the training set, validation set, and test set by the ratio of 7:1:2.

2) Backbones: IMCorrect is a model-agnostic method. As mentioned earlier, any mapping-based CF method can be used as the backbone for instantiating IMCorrect, and any embedding-based CF method can be first formulated as a mapping-based CF method and then used as the backbone for IMCorrect. We instantiate IMCorrect using two mapping-based CF methods (SLIM [22] and GF-CF [23]) and one embedding-based method (MF [16]), which we refer to as IMCorrect-SLIM, IMCorrect-GF-CF, and IMCorrect-MF, respectively. The descriptions and details of these three methods have been introduced in Sections III-A.

The selected backbone methods are representative. On the one hand, GF-CF obtains the mapping matrix through numerical calculations, while SLIM obtains the mapping matrix by optimizing the reconstruction loss. On the other hand, GF-CF separates compression (or encoding, Fourier transform) and decompression (or decoding, inverse Fourier transform) into two processes, whereas SLIM combines these two processes together. Therefore, we consider GF-CF and SLIM to represent the majority of mapping-based CF methods. MF, on the other hand, can be viewed as the initial form of all subsequent embedding-based CF methods.

3) Baselines: We choose retrain from scratch (referred to as *Retrain*) and RecEraser [8] as the baseline methods for comparison, as both of them can achieve complete recommendation unlearning. Regarding SISA [4] and GraphEraser [4], they can be considered as variants of RecEraser and have been shown to under-perform RecEraser in collaborative filtering tasks, so we do not include them in our comparison. As far as we know, RecEraser is currently the only open-source recommendation unlearning method.

RecEraser is also a model-agnostic method, but the original paper only provided the implementation for embedding-based CF methods. Here, we extend RecEraser to mapping-based CF methods. The process of pretraining, slicing, and training sub-models for RecEraser with mapping-based CF methods is the

[4]https://grouplens.org/datasets/movielens/
[5]http://snap.stanford.edu/data/loc-gowalla.html
[6]https://www.yelp.com/dataset

same as that for embedding-based CF methods, with the only difference lying in the aggregation layer. In the aggregation layer of embedding-based CF models, it requires aggregating user embeddings and item embeddings from different sub-models. Since the semantic space of embeddings from different sub-models may differ, a multi-layer perceptron (MLP) is used for feature transformation, which is the motivation behind adding the MLP in the original RecEraser. However, in the aggregation layer of mapping-based CF models, we need to aggregate the mapping matrices from different sub-models, and each dimension of the mapping matrix has the same meaning, eliminating the need for feature transformation. Therefore, for mapping-based CF models, we omit the MLP.

We instantiated RecEraser on the mapping-based CF method SLIM and the embedding-based CF method MF for comparison with IMCorrect. GF-CF cannot be used to instantiate RecEraser because GF-CF is a numerical method and cannot optimize the aggregation layer of RecEraser.

4) Hyperparameters: Since IMCorrect itself does not have hyperparameters, we will only introduce the hyperparameters used for the backbones and baselines. We determine the optimal hyperparameters by performing grid search and selecting the ones that perform the best on the validation set. For SLIM, we set both the L1 regularization coefficient and the L2 regularization coefficient to 1. For GF-CF, we set the rank of truncated singular values to 64. For MF, we set the dimension of embeddings to 64 and the regularization coefficient to 0.001. For RecEraser, we use MF for pretraining and set the number of shards to 10. The maximum number of iterations in the slicing process is set to 50, and the dimension of the attention matrix in the aggregation layer is set to 32. The hyperparameters for the sub-models of RecEraser, RecEraser-SLIM and RecEraser-MF, were the same as those used for IMCorrect-SLIM, and IMCorrect-MF. Additionally, to ensure reproducibility, we fix the random seed to 2024 for all experiments.

B. General Performance

To evaluate the general performance of a recommendation unlearning method, it should be assessed from three dimensions: completeness, utility, and efficiency.

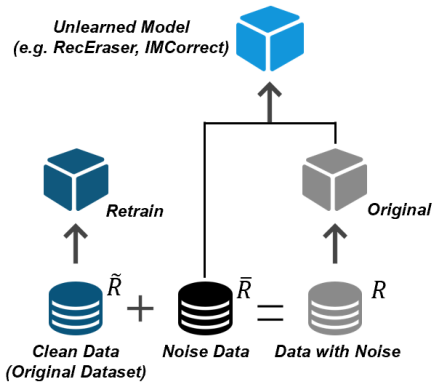

Fig. 7. The process of dataset generation.

1) Evaluation Methods: Efficiency can be directly evaluated by comparing the running time of different methods. As for completeness and utility, they can be assessed by comparing the differences in recommendation accuracy between different methods and *Retrain*. However, directly deleting clean interaction data from the training set would lead to a decrease in performance, making it challenging to identify the source of these differences. Completeness and utility are intertwined, making it difficult to effectively evaluate them. For instance, a method may exhibit good utility (small difference from *Retrain*) because it does not achieve high completeness.

To address this issue, three noisy datasets are constructed, and it is assumed that these noisy datasets are the original interaction matrix R, with the noisy interactions representing the interactions to be forgotten in the matrix $\bar{R}$. By forgetting these noisy interactions, the original dataset was obtained, corresponding to $\tilde{R}$, the interaction matrix after forgetting. The three methods for constructing noisy datasets are as follows:

- The "inert" dataset, where interactions are randomly added to each user based on the number of interactions in the training set, with proportions ranging from 1% to 50%.
- The "delete" dataset, where interactions are randomly deleted from each user based on the number of interactions in the training set, with proportions ranging from 1% to 50%.
- The "update" dataset, where interactions are both randomly added and deleted from each user based on the number of interactions in the training set, with proportions ranging from 1% to 50%.

As shown in Fig. 7, the results of the backbone model running on these noisy datasets are denoted as "*Original*", while the results of the backbone model running on the original clean datasets are denoted as "*Retrain*". In this setting, forgetting noisy interactions will lead to an improvement in performance because the data in $\bar{R}$ is considered toxic data. This way, completeness and utility are unified. In other words, the better the utility of a method, the better its completeness. This evaluation method is consistent with [14] and [72]. Recall is used as the evaluation metric to compare the performance of different methods.

2) Completeness and Utility Comparison: Due to space limitation, we present the results of implementing different recommendation unlearning methods using SLIM as the backbone on all datasets, and present the results achieved using MF and GF-CF only on the Yelp dataset due to consistent conclusions. Fig. 8-12 respectively show these results, each with "insert", "delete", and "update" attacks, with the abscissa representing the proportion of attack data to the total training set. From the results, we have the following observations:

- The performance of *Original* is worse than *Retrain* in all settings. This is because *Original* represents the performance of the backbone model on the data with noise, while *Retrain* represents the performance of the backbone

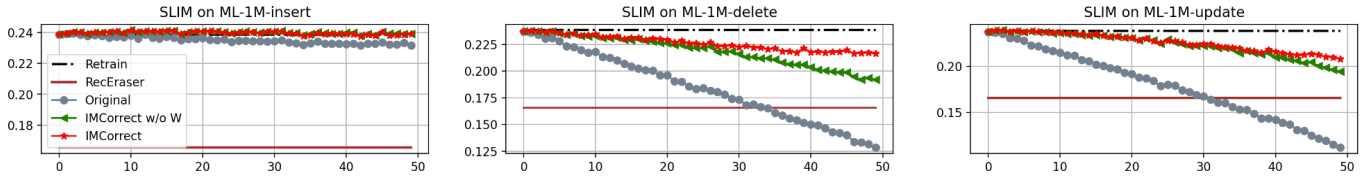

Fig. 8. Performance comparison on the ML-1M dataset using SLIM as the backbone.

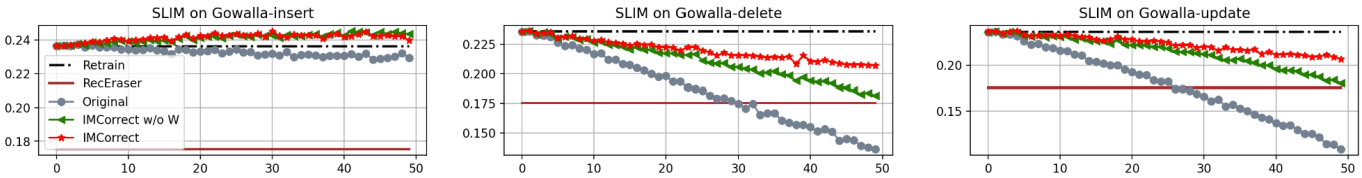

Fig. 9. Performance comparison on the Gowalla dataset using SLIM as the backbone.

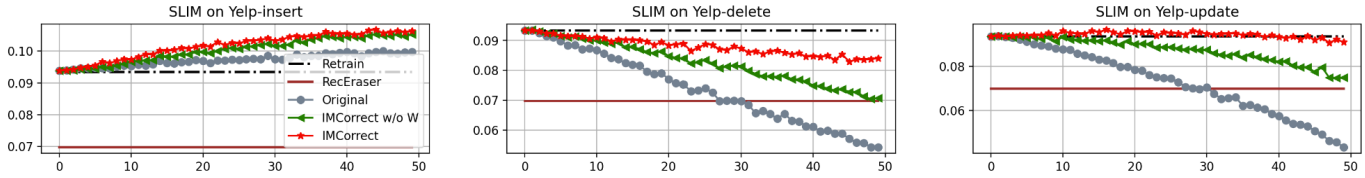

Fig. 10. Performance comparison on the Yelp dataset using SLIM as the backbone.

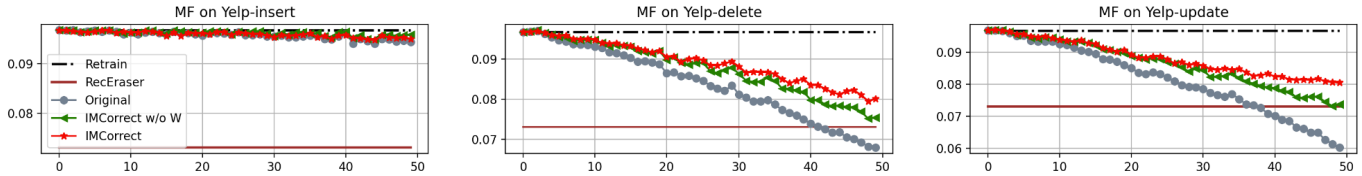

Fig. 11. Performance comparison on the Yelp dataset using MF as the backbone.

model on the data after forgetting the noises. Since the noisy interactions are toxic data, forgetting them leads to performance improvement.

- IMCorrect performs better than IMCorrect w/o W (without mapping matrix correction), and IMCorrect w/o W performs better than *Original*. This indicates that efficient recommendation unlearning can be achieved by only correcting the interaction matrix, and correcting the mapping matrix can further enhance completeness and utility.
- As RecEraser involves retraining, its performance remains consistent for any proportion of noisy data. Although the performance of IMCorrect tends to decline with the increase of noise, even in the most severely attacked scenario where 50% of interactions are corrupted, IMCorrect (even IMCorrect w/o W) still outperforms RecEraser. RecEraser's performance is even worse than *Original* until the noise proportion in the "delete" or "update" dataset exceeds approximately 25%. This suggests that RecEraser's utility is severely compromised due to the partitioning process.
- Even when random interactions are added for each user with a proportion of 50% of their total interactions in the "insert" dataset, the performance of *Original* remains nearly unchanged. This indicates that the model itself has sufficient robustness against attacks of adding random interactions. As mentioned earlier, mapping-based CF models smooth the interaction matrix to obtain the prediction matrix, acting like a low-pass graph filter. The added random interactions lack specific patterns and represent high-frequency information, so *Original* can achieve decent results.
- In contrast to the limited impact of the "insert" dataset, random deletion of interactions in the "delete" dataset has a significant impact on the model performance. This is because the deleted interactions are often the result of users' self-selected choices based on their interests, so they have certain patterns and deleting them will result in a decrease in similarity confidence due to increased sparsity of training data.
- An interesting phenomenon is observed where, when using SLIM as the backbone, as the noise increases on "insert" data, the performance of IMCorrect and IMCorrect w/o W outperforms *Retrain*. On the Yelp dataset, even *Original* can surpass *Retrain*. This phenomenon

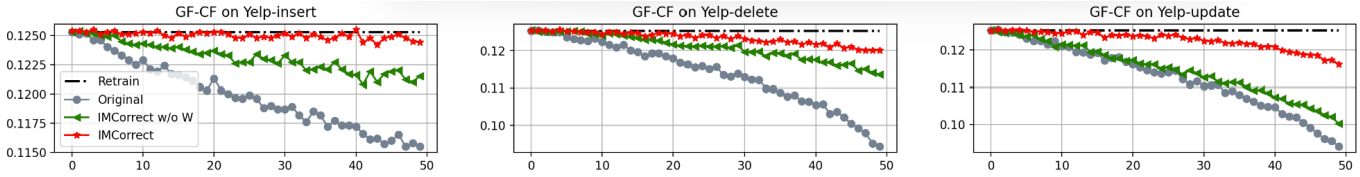

Fig. 12. Performance comparison on the Gowalla dataset using GF-CF as the backbone. Note that RecEraser cannot be applied to GF-CF.

becomes more pronounced as the sparsity of the dataset increases. A similar phenomenon was observed in [14]. Although the discussion of this phenomenon is not the focus of this paper, we mention it as it also implies that IMCorrect is particularly effective on sparse datasets.

3) Efficiency Comparison: In TABLE IV, we compare the efficiency of different methods on the "delete" dataset with 5% corrupted data, where the total time of RecEraser is measured by summing the average time of sub-model training and the time of the aggregation layer. Note that we only consider SLIM and MF as the backbones in the comparison because RecEraser cannot be applied to GF-CF. As shown in the results, IMCorrect can be fitted quickly, while the aggregation layer of RecEraser requires training on entire datasets, indicating that IMCorrect can achieve much higher efficiency than *Retrain* and RecEraser.

TABLE IV
RUNNING TIME COMPARISON ("M" STANDS FOR MINUTE AND "S" STANDS FOR SECOND)

Model	ML-1M	Gowalla	Yelp
Retrain-SLIM	13.0m	5.1m	8.5m
RecEraser-SLIM (sub-model+aggregate)	2.1m+11.7m	1.2m+6.6m	2.3m+27.4m
IMCorrect-SLIM	30s	15s	1.6m
Retrain-MF	24.7m	16.8m	32.7m
RecEraser-MF (sub-model+aggregate)	5.4m+12.1m	3.2m+7.1m	8.4m+32.2m
IMCorrect-MF	2.2m	1.9m	5.6m

C. Application Scenario Analysis

1) Usability: There are three specific scenarios for recommendation unlearning, each corresponding to modifying the interaction matrix or the mapping matrix, or both. For the out-of-distribution data scenario, it aligns with the noise setting used in Section V-B to validate the general performance of IMCorrect, where specific interaction patterns are often absent. Since we have already verified the capability of IMCorrect in this scenario, we will not conduct a case study for it. As for out-of-date data, it essentially involves forgetting specific interactions or preferences, similar to user-initiated forgetting for privacy protection purposes. The difference lies in that out-of-date data forgetting is often initiated by the system, while forgetting preferences is initiated by users. Therefore, we only present the results of the case study on attack data.

The experiments were conducted on the ML-1M dataset. We selected item 2024 and found that it had 55 interactions in the clean original training dataset. Moreover, we found that when the model was trained using the original dataset, item 2024 appeared in the recommendation lists of all users only 1 time. Next, we simulated malicious interactions by adding 1000 interactions for item 2024. At this point, we observed that the occurrences of this item in the recommendation lists increased to 106 times. Finally, we applied IMCorrect to remove the influence of the malicious interactions, and found that the number of recommendations for item 2024 was restored to 0. This indicates that IMCorrect can effectively ensure the usability of the model against attacks.

2) Privacy: When users initiate recommendation unlearning for privacy protection purposes, there are three levels: interaction level, preference level, and account level. Both interaction level and account level can be considered as special cases of preference level. Therefore, we conducted a case study specifically for the preference level, where we simultaneously correct the interaction matrix and mapping matrix.

The experiment was conducted on the ML-1M dataset. We selected user 2024 and recorded the recommendations obtained from the model trained on the clean original dataset. For constructing the forgotten preference, we first randomly select a target item that user 2024 had interacted with, then select 10 most similar items from user 2024's historical interactions, and finally let IMCorrect forget the 11 items. We have repeated the above process 100 times and compared the recommendation lists before and after forgetting. In the results, the average similarity between the recommended items and the target item is 0.0136 before forgetting and 0.0063 after forgetting, i.e., the recommended items are much less similar to the target items after forgetting, indicating the effectiveness of IMCorrect in forgetting user preferences (i.e., protecting user privacy) from the recommendation models.

VI. CONCLUSIONS

In this paper, we propose a model-agnostic recommendation unlearning method named IMCorrect. IMCorrect achieves high efficiency in forgetting by only correcting the interaction matrix, and can further achieve high completeness and utility by correcting the mapping matrix. Detailed experiments and case studies show that IMCorrect can achieve high completeness, utility, and efficiency, and can be applied to many real-world recommendation unlearning scenarios. This paper primarily focuses on validating its decremental learning capability. It should be noted that IMCorrect has the ability for incremental learning, we will further explore the performance of IMCorrect in incremental recommendation scenarios in future works.

REFERENCES

[1] S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, and Y. Gao, "Dual channel hypergraph collaborative filtering," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 2020–2029.

[2] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 267–284.

[3] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[4] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 141–159.

[5] M. Zhang, Z. Ren, Z. Wang, P. Ren, Z. Chen, P. Hu, and Y. Zhang, "Membership inference attacks against recommender systems," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 864–879.

[6] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2633–2650.

[7] S. Zanella-Béguelin, L. Wutschitz, S. Tople, V. Rühle, A. Paverd, O. Ohrimenko, B. Köpf, and M. Brockschmidt, "Analyzing information leakage of updates to natural language models," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 363–375.

[8] C. Chen, F. Sun, M. Zhang, and B. Ding, "Recommendation unlearning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2768–2777.

[9] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web: methods and strategies of web personalization*. Springer, 2007, pp. 291–324.

[10] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," *Advances in neural information processing systems*, vol. 29, 2016.

[11] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 19–35.

[12] W. Liu, J. Wan, X. Wang, W. Zhang, D. Zhang, and H. Li, "Forgetting fast in recommender systems," *arXiv preprint arXiv:2208.06875*, 2022.

[13] Y. Li, C. Chen, X. Zheng, Y. Zhang, B. Gong, and J. Wang, "Selective and collaborative influence function for efficient recommendation unlearning," *arXiv preprint arXiv:2304.10199*, 2023.

[14] Y. Zhang, Z. Hu, Y. Bai, F. Feng, J. Wu, Q. Wang, and X. He, "Recommendation unlearning via influence function," *arXiv preprint arXiv:2307.02147*, 2023.

[15] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *2008 Eighth IEEE international conference on data mining*. Ieee, 2008, pp. 263–272.

[16] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[17] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[18] C. Wang, W. Ma, M. Zhang, C. Chen, Y. Liu, and S. Ma, "Toward dynamic user intention: Temporal evolutionary effects of item relations in sequential recommendation," *ACM Transactions on Information Systems (TOIS)*, vol. 39, no. 2, pp. 1–33, 2020.

[19] J. Liu, D. Li, H. Gu, T. Lu, P. Zhang, and N. Gu, "Parameter-free dynamic graph embedding for link prediction," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 623–27 635, 2022.

[20] J. Liu, D. Li, H. Gu, T. Lu, P. Zhang, L. Shang, and N. Gu, "Triple structural information modelling for accurate, explainable and interactive recommendation," *arXiv preprint arXiv:2304.11528*, 2023.

[21] J. Xia, D. Li, H. Gu, J. Liu, T. Lu, and N. Gu, "Fire: Fast incremental recommendation with graph signal processing," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2360–2369.

[22] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *2011 IEEE 11th international conference on data mining*. IEEE, 2011, pp. 497–506.

[23] Y. Shen, Y. Wu, Y. Zhang, C. Shan, J. Zhang, B. K. Letaief, and D. Li, "How powerful is graph convolution for recommendation?" in *Proceedings of the 30th ACM international conference on information & knowledge management*, 2021, pp. 1619–1629.

[24] Y. Koren, S. Rendle, and R. Bell, "Advances in collaborative filtering," *Recommender systems handbook*, pp. 91–142, 2021.

[25] J. Liu, D. Li, H. Gu, T. Lu, P. Zhang, L. Shang, and N. Gu, "Personalized graph signal processing for collaborative filtering," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 1264–1272.

[26] C. Chen, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Efficient neural matrix factorization without sampling for recommendation," *ACM Transactions on Information Systems (TOIS)*, vol. 38, no. 2, pp. 1–28, 2020.

[27] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999, pp. 230–237.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[29] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 426–434.

[30] B. Sarwar, G. Karypis, J. Konstan, and J. T. Riedl, "Application of dimensionality reduction in recommender system-a case study," *Computer Science & Engineering (CS&E) Technical Reports*, 2000.

[31] H. Steck, "Embarrassingly shallow autoencoders for sparse data," in *The World Wide Web Conference*, 2019, pp. 3251–3257.

[32] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th international conference on World Wide Web*, 2015, pp. 111–112.

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

[34] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[35] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "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*, 2020, pp. 639–648.

[36] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9304–9312.

[37] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten, "Certified data removal from machine learning models," *arXiv preprint arXiv:1911.03030*, 2019.

[38] S. Schelter, S. Grafberger, and T. Dunning, "Hedgecut: Maintaining randomised trees for low-latency machine unlearning," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1545–1557.

[39] T. Baumhauer, P. Schöttle, and M. Zeppelzauer, "Machine unlearning: Linear filtration for logit-based classifiers," *Machine Learning*, vol. 111, no. 9, pp. 3203–3226, 2022.

[40] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[41] C. Gao, C. Huang, D. Lin, D. Jin, and Y. Li, "Dplcf: differentially private local collaborative filtering," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 961–970.

[42] C. Dwork, "Differential privacy," in *International colloquium on automata, languages, and programming*. Springer, 2006, pp. 1–12.

[43] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "Graph unlearning," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 499–513.

[44] Y. He, G. Meng, K. Chen, J. He, and X. Hu, "Deepobliviate: a powerful charm for erasing data residual memory in deep neural networks," *arXiv preprint arXiv:2105.06209*, 2021.

[45] A. Ginart, M. Guan, G. Valiant, and J. Y. Zou, "Making ai forget you: Data deletion in machine learning," *Advances in neural information processing systems*, vol. 32, 2019.

[46] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 463–480.

[47] H. Yan, X. Li, Z. Guo, H. Li, F. Li, and X. Lin, "Arcane: An efficient architecture for exact machine unlearning," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 2022, pp. 4006–4013.

[48] Y. Li, X. Zheng, C. Chen, and J. Liu, "Making recommender systems forget: Learning and unlearning for erasable recommendation," *arXiv preprint arXiv:2203.11491*, 2022.

[49] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, pp. 1–45, 2014.

[50] Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou, "Approximate data deletion from machine learning models," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 2008–2016.

[51] S. Neel, A. Roth, and S. Sharifi-Malvajerdi, "Descent-to-delete: Gradient-based methods for machine unlearning," in *Algorithmic Learning Theory*. PMLR, 2021, pp. 931–962.

[52] A. Golatkar, A. Achille, and S. Soatto, "Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*. Springer, 2020, pp. 383–398.

[53] F. R. Hampel, "The influence curve and its role in robust estimation," *Journal of the american statistical association*, vol. 69, no. 346, pp. 383–393, 1974.

[54] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International conference on machine learning*. PMLR, 2017, pp. 1885–1894.

[55] P. W. W. Koh, K.-S. Ang, H. Teo, and P. S. Liang, "On the accuracy of influence functions for measuring group effects," *Advances in neural information processing systems*, vol. 32, 2019.

[56] G. Wu, M. Hashemi, and C. Srinivasa, "Puma: Performance unchanged model augmentation for training data removal," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8675–8682.

[57] R. Mehta, S. Pal, V. Singh, and S. N. Ravi, "Deep unlearning via randomized conditionally independent hessians," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 422–10 431.

[58] Y. Wu, E. Dobriban, and S. Davidson, "Deltagrad: Rapid retraining of machine learning models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 355–10 366.

[59] H. Yi, F. Zhang, and J. Lan, "A robust collaborative recommendation algorithm based on k-distance and tukey m-estimator," *China Communications*, vol. 11, no. 9, pp. 112–123, 2014.

[60] M. Fang, N. Z. Gong, and J. Liu, "Influence function based data poisoning attacks to top-n recommender systems," in *Proceedings of The Web Conference 2020*, 2020, pp. 3019–3025.

[61] C. Wu, D. Lian, Y. Ge, Z. Zhu, and E. Chen, "Triple adversarial learning for influence based poisoning attack in recommender systems," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1830–1840.

[62] H. Zhang, Y. Li, B. Ding, and J. Gao, "Practical data poisoning attack against next-item recommendation," in *Proceedings of The Web Conference 2020*, 2020, pp. 2458–2464.

[63] W. Cheng, Y. Shen, L. Huang, and Y. Zhu, "Incorporating interpretability into latent factor models via fast influence analysis," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 885–893.

[64] J. Yu, H. Zhu, C.-Y. Chang, X. Feng, B. Yuan, X. He, and Z. Dong, "Influence function for unbiased recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1929–1932.

[65] A. Mahadevan and M. Mathioudakis, "Certifiable machine unlearning for linear models," *arXiv preprint arXiv:2106.15093*, 2021.

[66] A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh, "Remember what you want to forget: Algorithms for machine unlearning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 075–18 086, 2021.

[67] C.-H. Tsai, C.-Y. Lin, and C.-J. Lin, "Incremental and decremental training for linear classification," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 343–352.

[68] J. C. Bezdek and R. J. Hathaway, "Convergence of alternating optimization," *Neural, Parallel & Scientific Computations*, vol. 11, no. 4, pp. 351–368, 2003.

[69] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Algorithmic Aspects in Information and Management: 4th International Conference, AAIM 2008, Shanghai, China, June 23-25, 2008. Proceedings 4*. Springer, 2008, pp. 337–348.

[70] J. Martens *et al.*, "Deep learning via hessian-free optimization." in *ICML*, vol. 27, 2010, pp. 735–742.

[71] S. Basu, X. You, and S. Feizi, "On second-order group influence functions for black-box predictions," in *International Conference on Machine Learning*. PMLR, 2020, pp. 715–724.

[72] W. Yuan, H. Yin, F. Wu, S. Zhang, T. He, and H. Wang, "Federated unlearning for on-device recommendation," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 393–401.

[73] M. Xu, J. Sun, X. Yang, K. Yao, and C. Wang, "Netflix and forget: Efficient and exact machine unlearning from bi-linear recommendations," *arXiv preprint arXiv:2302.06676*, 2023.

[74] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[75] D. Yu, H. Zhang, W. Chen, J. Yin, and T.-Y. Liu, "How does data augmentation affect privacy in machine learning?" in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 746–10 753.

[76] B. Wu, C. Chen, S. Zhao, C. Chen, Y. Yao, G. Sun, L. Wang, X. Zhang, and J. Zhou, "Characterizing membership privacy in stochastic gradient langevin dynamics," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6372–6379.

[77] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

[78] J. Zhu, Q. Dai, L. Su, R. Ma, J. Liu, G. Cai, X. Xiao, and R. Zhang, "Bars: Towards open benchmarking for recommender systems," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 2912–2923.

[79] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[80] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 1082–1090.

[81] R. He and J. McAuley, "Vbpr: visual bayesian personalized ranking from implicit feedback," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.