

# Making Recommender Systems Forget: Learning and Unlearning for Erasable Recommendation

Yuyuan Li<sup>1</sup>, Xiaolin Zheng<sup>1</sup>, Chaochao Chen<sup>1\*</sup> and Junlin Liu<sup>1</sup>

<sup>1</sup>Zhejiang University

{11821022, xlzheng, zjucce, junlin}@zju.edu.cn,

## Abstract

Privacy laws and regulations enforce data-driven systems, e.g., recommender systems, to erase the data that concern individuals. As machine learning models potentially memorize the training data, data erasure should also unlearn the data lineage in models, which raises increasing interest in the problem of Machine Unlearning (MU). However, existing MU methods cannot be directly applied into recommendation. The basic idea of most recommender systems is collaborative filtering, but existing MU methods ignore the *collaborative information* across users and items. In this paper, we propose a general erasable recommendation framework, namely LASER, which consists of Group module and SeqTrain module. Firstly, Group module partitions users into balanced groups based on their similarity of collaborative embedding learned via hypergraph. Then SeqTrain module trains the model sequentially on all groups with curriculum learning. Both theoretical analysis and experiments on two real-world datasets demonstrate that LASER can not only achieve efficient unlearning, but also outperform the state-of-the-art unlearning framework in terms of model utility.

## 1 Introduction

Recommender Systems (RSs) are typically built by analyzing the data collected from users, such as users' ratings on items. Existing regulations, e.g., the General Data Protection Regulation [EU, 2014], enforce the ability of erasing the personal data that concern individuals, which raises the concerns of privacy in RSs. Due to the fact that machine learning models, which have been ubiquitously applied in RSs, potentially memorize the training data [Bourtole *et al.*, 2021]. Besides simply deleting the target data, data erasure is also required to unlearning the data lineage in RS models. In general, it is beneficial for both users and recommendation platforms to build an erasable RS that supports unlearning in addition to learning. On the one hand, an erasable RS can preserve the

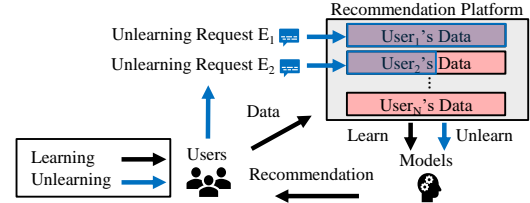


Figure 1: A schematic view of learning and unlearning in recommendation.

privacy of users. On the other hand, it can also enhance recommendation performance by active unlearning. This is because the performance of RSs is sensitive to the training data which can be easily polluted by accidental mistakes or poisoned by intentional attack [Schafer *et al.*, 2007].

Figure 1 illustrates the schema of learning and unlearning in recommendation. In the learning process (black arrows), the platform trains recommendation models with the data collected from users. In the unlearning process (blue arrows), the platform unlearns the target data lineage according to the unlearning requests from users. There are basically two types of unlearning requests, (i) *user-wise erasure* ( $E_1$ ) which unlearns all of the data from the user(s), and (ii) *data-wise erasure* ( $E_2$ ) which unlearns a portion of user data. In order to evaluate unlearning methods, we summarize three goals of unlearning, i.e., G1: *Unlearn Completeness*, G2: *Computational Efficiency*, and G3: *Model Utility*.

Recently, researches have studied the problem of Machine Unlearning (MU) which aims to unlearn the target data lineage in machine learning models. A straightforward MU method is retraining the model from scratch on the new dataset after deleting the target data, which perfectly achieves G1 and G3, but fails G2. Due to tremendous computational overhead of recommendation models, efficiency is one of the key concerns for unlearning. To overcome the issue of inefficiency, two classes of MU methods, i.e., *retrain unlearning* and *reverse unlearning*, have been proposed [Bourtole *et al.*, 2021; Schelter *et al.*, 2021; Sekhari *et al.*, 2021]. Retrain unlearning basically divides the original dataset into subsets and retrains the model on the target subset(s) to reduce computational overhead. In contrast, reverse unlearning focuses on the target data and executes reverse operations.

\*Contact Author

Unfortunately, the above MU methods cannot be directly applied into recommendation. The reason is that the basic idea of most RSs is Collaborative Filtering (CF) [Shi *et al.*, 2014], but these MU methods fail to consider the *collaborative information* across users and items. On the one hand, retrain unlearning trains each model on a subset of the original dataset, which means that each model can only utilize collaborative information within the subset. Consequently, retrain unlearning reduces total model utility (G3). On the other hand, reverse unlearning only executes reverse operations on the target data ignoring the collaborative effect with associated data. Thus, reverse unlearning cannot unlearn with completeness (G1).

In this paper, we focus on user-wise erasure problem and propose LASER for it. Following the idea of retrain unlearning, LASER naturally achieves G1. Specifically, LASER consists of two modules, i.e., *Group module* and *Sequential Train (SeqTrain) module*, which are constructed not only to achieve efficient unlearning (G2), but also to enhance model utility (G3) by preserving collaborative information. Firstly, Group module divides the original data into balanced groups for efficient retraining. These groups are generated based on user similarity, which is measured by the distance of collaborative embedding learned via hypergraph. Then, SeqTrain module trains the model on all groups sequentially to aggregate collaborative information. To further improve model performance in such a sequential training manner, SeqTrain module trains the groups in an easy-to-hard order, which imitates the learning style in human curricula. Our theoretical analysis and empirical study demonstrate that sequential training can improve the performance of models. Through these two modules, LASER achieves efficient unlearning and boosts model utility for CF-based recommendation. We summarize the main contributions of this paper as follows:

- We focus on user-wise erasure and propose an erasable recommendation framework (LASER) for CF models.
- We encode the high-order collaborations via hypergraph and propose a balanced grouping method to enhance unlearning efficiency.
- We propose collaborative cohesion to measure the volume of collaborative information, as well as the learning difficulty. Our theoretical analysis reveals that the easy-to-hard learning order improves model utility.
- We conduct extensive experiments on two real-world datasets to demonstrate that LASER supports efficient unlearning and outperforms the state-of-the-art unlearning framework in model utility.

## 2 Problem Formulation

### 2.1 Unlearning

The main problem of building an erasable RS is unlearning and we formulate the process of unlearning as follows: letting  $[N] = \{1, \dots, N\}$  denotes a user set, we assume that the recommendation platform learns a model  $\mathcal{M}$  on the dataset  $\mathcal{D} = \{d_i, i \in [N]\}$  where  $d_i$  denotes the data collected from user  $i$ . As it is illustrated by blue arrows in Figure 1, any user  $i \in [N]$  can submit an unlearning request  $E \subset d_i$  to unlearn the target data. For user-wise erasure,  $E = d_i$ , and typically

$|\mathcal{D}| \gg |E|$ . In practice, the unlearning requests are often submitted sequentially. For conciseness, we assume that the platform processes a batch of unlearning requests together. Finally, the platform unlearns the model  $\mathcal{M}$  based on request  $E$ , and produces an unlearned model  $\mathcal{M}_u$ . Let  $\mathcal{M}_u^*$  denote the model that is retrained on  $\mathcal{D} \setminus E$  from scratch. Although retraining from scratch is extremely inefficient,  $\mathcal{M}_u^*$  is the ground-truth unlearned model. Thus, the three goals of unlearning is equivalent to efficiently producing a model whose distribution is close to that of  $\mathcal{M}_u^*$ .

### 2.2 Goals of Unlearning

**G1: Unlearn Completeness.** Completely unlearning the target data lineage is one of the most fundamental goals of unlearning, which means fully erasing the influence on model parameters and making it impossible to recover.

**G2: Unlearn Efficiency.** Due to the considerable computational overhead of practical recommendation models, unlearn efficiency, especially time efficiency, is an essential goal of unlearning.

**G3: Model Utility.** Although unlearning too much data lineage will inevitably reduce the model utility, an adequate unlearning method should achieve comparable performance with the ground-truth model  $\mathcal{M}_u^*$ .

### 2.3 Choice of Recommendation Models

Our proposed LASER can be generalized to most existing CF models. For conciseness, in this paper, we only consider rating data and choose two well-known CF models, i.e., Deep Matrix Factorization (DMF) [Xue *et al.*, 2017] and Neural Matrix Factorization (NMF) [He *et al.*, 2017]. Basically, the two models decompose the user-item interaction matrix  $R$  into two low-rank embedding matrices, i.e.,  $\alpha$  and  $\beta$ , which represent user features and item features respectively. The two models predict unknown ratings as follows:

$$\hat{R}_{i,j}^{\text{DMF}} = \cos(\sigma(\alpha_i), \sigma(\beta_j)), \quad (1)$$

$$\hat{R}_{i,j}^{\text{NMF}} = \sigma(F_{\text{GMF}}(i, j) \oplus F_{\text{MLP}}(i, j)), \quad (2)$$

where  $F_{\text{GMF}}(i, j) = \sigma(\alpha_i \odot \beta_j)$ ,  $F_{\text{MLP}}(i, j) = \sigma(\alpha_i \oplus \beta_j)$ ,  $\sigma$  denotes layer operations,  $\oplus$  and  $\odot$  denote vector concatenation and element-wise product respectively. Following the original papers, we adopt normalized binary cross entropy and Adam to train the above models.

## 3 LASER Framework

Generally speaking, LASER follows the idea of retraining unlearning to achieve G1 and G2, and maintains model utility (G3) by preserving collaborative information.

### 3.1 Overview

Figure 2 illustrates the structure of LASER framework, which consists of Group module and SeqTrain module. We introduce the learning and unlearning pipelines as follows:

**Learning:** As the black arrows show in Figure 2, firstly, Group module divides  $\mathcal{D}$  into disjoint groups based on the similarity of collaborative embedding. Secondly, SeqTrain module sorts the groups based on the learning difficulty, then

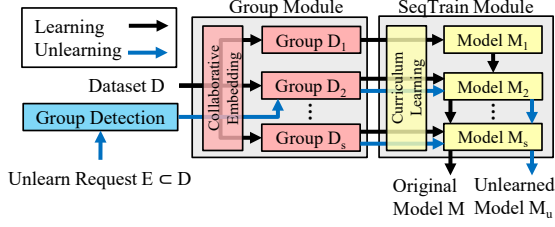


Figure 2: Overview of LASER framework.

trains the model sequentially in an easy-to-hard order and finally saves these models. In this way, we can train the model on the whole dataset to preserve collaborative information.

**Unlearning:** As the blue arrows show, once an unlearning request  $E$  is received, Group module detects in which group  $E$  is located and SeqTrain module sequentially retrains the model using the previously saved models. In this way, we can retrain the model efficiently during unlearning.

### 3.2 Group Module

Group module aims to partition a dataset  $\mathcal{D}$  into  $S$  groups such that  $\bigcap_{i \in [S]} \mathcal{D}_i = \emptyset$  and  $\bigcup_{i \in [S]} \mathcal{D}_i = \mathcal{D}$ . This paper focuses on user-wise erasure which means the above dataset partition is equivalent to dividing the user set  $[N]$  into  $S$  disjoint groups. Partitioning unlabeled data into groups is a classic unsupervised learning problem named clustering. Thus, a straightforward idea is to cluster users based on the observed data. There are two grouping principles, i.e., *collaboration* and *balance*, for effective and efficient unlearning in RSs.

#### Collaboration Principle

**Motivation** Collaboration is of great importance to enhancing RS model utility. There are two main challenges in collaborative grouping. Firstly, in the context of RS, directly clustering users suffers from data sparsity. Since each user only interacts with few items, most elements of the user rating vector, i.e., each row of the user-item interaction matrix  $R$ , are empty. Secondly, the original  $R$  or its corresponding bipartite graph cannot sufficiently represent the user-item collaboration, especially high-order relations [Ji *et al.*, 2020].

**Method** In order to encode the sparse ratings and enrich the collaborative information, we learn the hidden collaboration via *hypergraph*. Figure 3(a) depicts a traditional bipartite graph where an edge connects two vertices (a user and an item). However, Figure 3(b) shows the high-order relations of  $u_1$  which consists of multiple vertices. Hypergraph is a generalized structure for relation modeling, in which a hyper-edge connects two or more vertices. Due to this property, hypergraph can sufficiently model the high-order relations that cannot be directly represented by a traditional graph.

Specifically, we take the following three steps to learn collaborative embedding: (i) using  $R$  to build the corresponding hypergraph; (ii) for each user, performing random walk to obtain its relations, which transforms the task to sequence embedding; (iii) applying the sequence embedding technique, e.g., Word2Vec [Church, 2017], to learn the collaborative embedding. Due to page limit, we summarize the above process in Algorithm 1 and present the details in Appendix A.



Figure 3: An illustration of an user-item bipartite graph and the high-order relations of  $u_1$  where  $u$  and  $i$  denote user and item respectively.

#### Algorithm 1 Collaborative Embedding via Hypergraph

**Input:** User-item interaction matrix:  $R$

**Parameter:** Embedding dimension:  $M$

**Output:** User embedding matrix:  $B \in \mathcal{R}^{N \times M}$

**Procedure:**

- 1: Build hypergraph:  $H = \text{BuildHypergraph}(R)$ ;
- 2: Sample user sequence:  $F = \text{RandWalk}(H)$ ;
- 3: Generate user embedding:  $B = \text{SeqEmbedding}(F)$ ;

### Balance Principle

**Motivation** After learning the collaborative embedding, the next step is clustering users based on their embeddings. However, traditional clustering results in unbalanced grouping, which means that the user number in each group, i.e., group size, varies drastically. We conduct  $k$ -means [Kanungo *et al.*, 2002] clustering on two real-world datasets and report the result in Appendix B.1. From it, we observe that the grouping distribution is highly unbalanced. Since we have no prior knowledge about the distribution of unlearning requests, it is better to assume that users submit unlearning requests with equal probability. The balance principle is making the group size evenly distributed so that it can maximize unlearning efficiency (see Appendix B.2 for theoretical explanation).

**Method** To achieve balanced grouping, we propose a general method that can be applied to most existing clustering algorithms, including but not limited to  $k$ -means clustering, label propagation algorithm [Wang and Zhang, 2007], and Gaussian mixture models [Reynolds, 2009]. We present the proposed balanced grouping method in Algorithm 2. To be specific, we set the ceiling size  $c_i = \lceil N/S \rceil$  for each group and build a *priority list*  $P$  to store the priority of each user-group pair (line 3 in Algorithm 2). For example, in  $k$ -means clustering, the pair with a smaller Euclidean distance is considered to have a larger priority. During every grouping iteration, we allocate the user-group pairs (i) according to their priority, and (ii) if the group has not reached the ceiling size (line 5-8 in Algorithm 2). We demonstrate balanced  $k$ -means (BKM) clustering as an example in Appendix B.3.

### 3.3 SeqTrain Module

Instead of training models on each group in isolation and aggregating them, SeqTrain module trains the model on the whole dataset to preserve collaborative information. We achieve this by training the model sequentially on all groups, and we further study the effect of training order.

#### Training Order

Based on the theory of Curriculum Learning [Wang *et al.*, 2021], training order can make a huge difference in model

**Algorithm 2** Balanced Grouping Method

**Input:** User embedding matrix:  $B$ , grouping number:  $S$ , maximal iteration:  $\tau$

**Output:** Group label of  $N$  users:  $\Lambda = (\lambda_1, \dots, \lambda_N)$ ,  $\lambda_i \in [S]$

**Procedure:**

```

1:  $t = 0$ ,  $c_i = \lceil N/S \rceil$  for  $i \in [S]$ , randomly allocate  $\Lambda$ ;
2: while True do
3:    $P = \text{ComputeSimilarity}(B, \Lambda)$ ;
4:    $P = \text{Sort}(P, \text{order} = \text{descend})$ ;
5:   for  $\text{priority}(i, j)$  in  $P$  do
6:     if  $c_j > 0$  then
7:        $\lambda_i = j$ ,  $c_j = c_j - 1$ ;
8:     end if
9:   end for
10:   $t = t + 1$ ;
11:  if  $t > \tau$  or  $\Lambda$  do not change then
12:    break;
13:  end if
14: end while

```

performance. We regard each group as a learning task. Then we aim to train the model from easier tasks to harder ones, which imitates the learning order in human curricula. A task can be considered as easy if the model has a relatively low loss on it. The loss can only be obtained either after training or during training. However, we have to decide the training order before training and cannot regroup the users during training, which means we have to rely on predefined information to measure the learning difficulty of each group.

For CF-based recommendation, collaborative information contributes to better model performance, i.e., lower loss. It is reasonable to use collaborative information as a difficulty measurer. In order to measure the volume of collaborative information, we propose the concept of *collaborative cohesion*, which calculates the cohesion of a group. The higher the collaborative cohesion, the easier it is for the recommendation model to learn. The cohesion is defined by the average similarity, i.e., collaboration, of users within the group. Formally, the collaborative cohesion  $\rho$  is computed as:

$$\rho = \frac{\sum_{x, y \in g} \text{sim}(x, y)}{|g|}, \quad (3)$$

where  $g$  denotes a group and  $\text{sim}(x, y)$  computes the similarity between two users. In this paper, we set  $\text{sim}(x, y) = 1/\text{dist}(x, y)$ , where  $\text{dist}(x, y)$  denotes Euclidean distance of two users. We will empirically study the validity of collaborative cohesion as a difficulty measurer in Section 4.5.

**Theoretical Analysis**

Let  $\theta$  denote the model parameter vector and  $L_\theta(g_i)$  denote the loss of the model when given the data of  $i$ -th group. Adopting the widely used empirical risk minimization framework, we have the empirical loss

$$\mathcal{L}(\theta) = \hat{\mathbb{E}}[L_\theta] = \frac{1}{S} \sum_{i=1}^S L_\theta(g_i). \quad (4)$$

Minimizing the empirical loss can be regarded as maximizing model utility [Hacohen and Weinshall, 2019], which is

Dataset	User #	Item #	Rating #	Sparsity
ML	6,040	3,706	1,000,209	95.532%
AM	478,235	266,414	836,006	99.999%

Table 1: Summary of datasets.

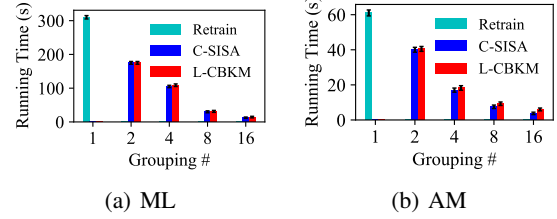


Figure 4: Comparison results of running time.

defined as:

$$\mathcal{U}(\theta) = \hat{\mathbb{E}}(U_\theta) = \frac{1}{S} \sum_{i=1}^S U_\theta(g_i) \triangleq \frac{1}{S} \sum_{i=1}^S e^{-L_\theta(g_i)}. \quad (5)$$

As collaborative cohesion  $\rho$  indicates the learning difficulty of each group, introducing  $\rho$  can be interpreted as providing a Bayesian prior  $p$  for model utility.

**Theorem 1.** *Given a Bayesian prior  $p$  for the model parameters  $\theta$ , we have:*

$$\mathcal{U}_p(\theta) = \mathcal{U}(\theta) + \hat{\text{Cov}}[U_\theta, p]. \quad (6)$$

*Proof.* The proof can be found in Appendix C.  $\square$

Theorem 2 reveals that as long as  $\hat{\text{Cov}}[U_\theta, p] > 0$ , SeqTrain module can improve the original model utility from  $\mathcal{U}(\theta)$  to  $\mathcal{U}_p(\theta)$ . Our empirical study in Section 4.5 shows that  $\rho$  is positively related to  $U_\theta$  which confirms that collaborative cohesion is a proper difficulty measurer.

**4 Experiments****4.1 Dataset**

We evaluate LASER on two publicly accessible datasets: MovieLens 1M (ML)<sup>1</sup> and Amazon Digital Music (AM)<sup>2</sup>. The ML and AM datasets are widely used to evaluate CF algorithms [Harper and Konstan, 2015; He and McAuley, 2016]. We filter out the users and items that have less than 5 interactions. We use 90% of ratings for training and the rest for testing. Table 1 summarizes the statistics of two datasets.

**4.2 Experimental Settings**

We test our proposed framework on two well-known CF models, i.e., DMF [Xue *et al.*, 2017] and NMF [He *et al.*, 2017]. Specifically, we set learning rate to 0.001, total training epochs  $T = 50$  for ML dataset and 20 for AM dataset. The dimension of user (item) embedding matrix is 16 and network structures in both DMF and NMF are set as two layers (64,

<sup>1</sup><https://grouplens.org/datasets/movielens/>

<sup>2</sup><http://jmcauley.ucsd.edu/data/amazon/>

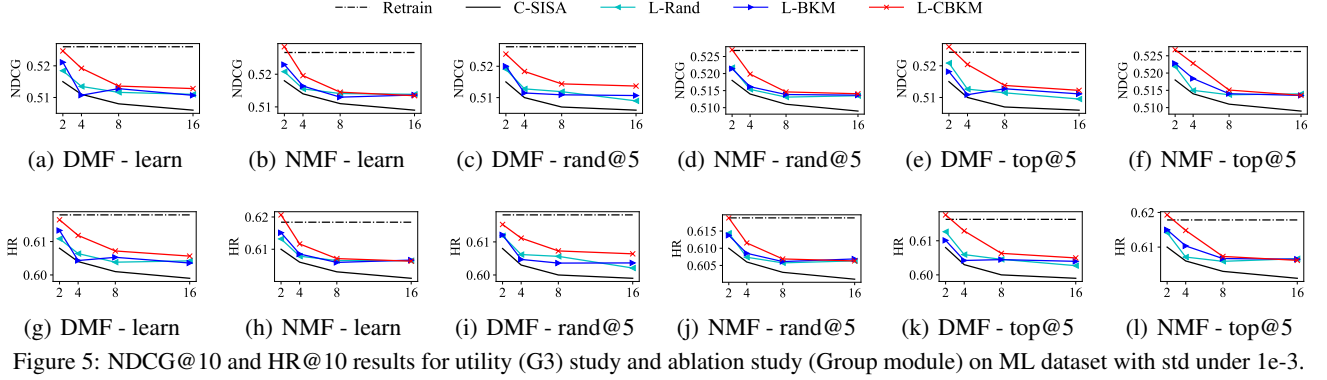


Figure 5: NDCG@10 and HR@10 results for utility (G3) study and ablation study (Group module) on ML dataset with std under 1e-3.

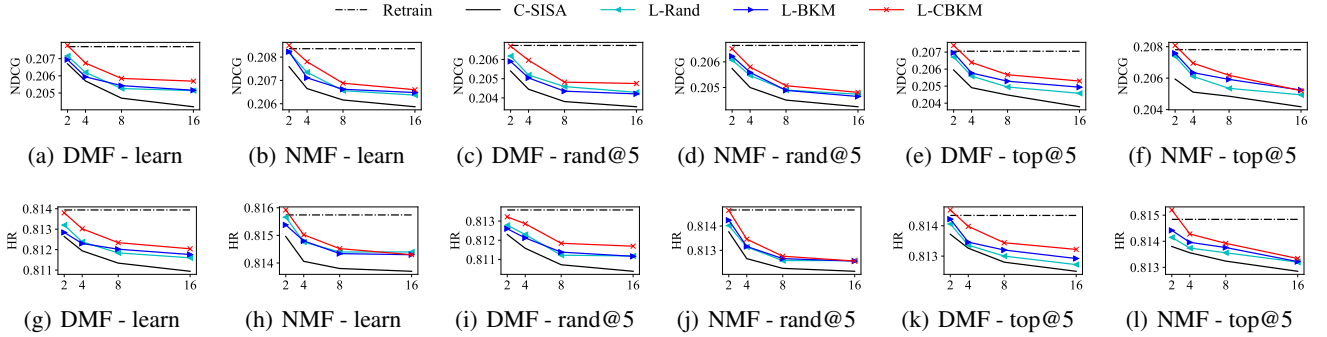


Figure 6: NDCG@10 and HR@10 results for utility (G3) study and ablation study (Group module) on AM dataset with std under 1e-3.

32). We initialize the model parameters with a Gaussian distribution  $\mathcal{N}(0, 0.01^2)$ . All unlearning methods for comparison are listed as follows:

**Retrain:** Retraining from scratch, i.e.,  $\mathcal{M}_u^*$ .

**C-SISA:** We modify the State-Of-The-Art (SOTA) unlearning framework, i.e., SISA [Bourtoule *et al.*, 2021], to fit recommendation setting. Since the original SISA trains one model on each user group in isolation, it can only learn the user embeddings within the group. Thus, one has to concatenate fragmentary user embedding matrices to obtain the final user embedding matrix  $\alpha$ . We name this method as Concatenated-SISA and C-SISA for short.

**L-Rand:** LASER via balanced random grouping.

**L-BKM:** LASER via BKM with rating data.

**L-CBKM:** LASER via BKM with collaborative embedding.

All models and algorithms are implemented with Python 3.8 and PyTorch 1.9. We run all experiments on the same Ubuntu 20.04 LTS System server with 48-core CPU (Intel Xeon Gold 5118, 2.3GHz), 256GB RAM and NVIDIA GeForce RTX 3090 GPU. We ran all models for 10 times and report the means and standard deviations (std).

### 4.3 Computational Efficiency (G2)

We evaluate the efficiency by computing the running time of three methods, i.e., Retrain, C-SISA, and L-CBKM (on behalf of LASER). Note that we only report the time of unlearning, because the main problem of an erasable RS is unlearning and the three methods are theoretically similar in learning time. We randomly unlearn 5% of users in the last group for

each dataset. In order to fully exploit the efficiency of C-SISA, we run each model in parallel for C-SISA. As DMF and NMF cost comparable training time, we report the running time of DMF in Figure 4 for conciseness. We can observe consistent results in both datasets, i.e., (i) our proposed L-CBKM achieves almost comparable running time with paralleled C-SISA, and (ii) C-SISA and L-CBKM cost less unlearning time as the number of groups increases. The results show the efficiency of LASER, especially when the group size is large which usually comes with utility loss, as we will report later.

### 4.4 Model Utility (G3)

We use two common metrics, i.e., Normalized Discounted Cumulative Gain (NDCG) and Hit Ratio (HR), to evaluate the performance of CF models [He *et al.*, 2015; Xue *et al.*, 2017]. We truncate the ranked list at 10 for both metrics and report NDCG@10 and HR@10 during both learning and unlearning processes. In order to fully study the effect of unlearning, we define two types of user-wise unlearning request, i.e.,  $\text{rand}@K$  and  $\text{top}@K$ , which denote unlearning  $K\%$  random users and the top  $K\%$  users (w.r.t. the number of ratings), respectively. We vary  $K$  in  $\{2.5, 5\}$  for both datasets.

We compare LASER (L-Rand, L-BKM and L-CBKM) with the SOTA MU framework C-SISA, and report the results in Figure 5 and 6, where we vary grouping number  $S$  in  $\{1, 2, 4, 8, 16\}$  and set  $K$  to 5. From them, we observe that (i) even if the users are randomly grouped (L-Rand), LASER outperforms C-SISA in all testing cases, which means that, in the context of recommendation, LASER can enhance perfor-



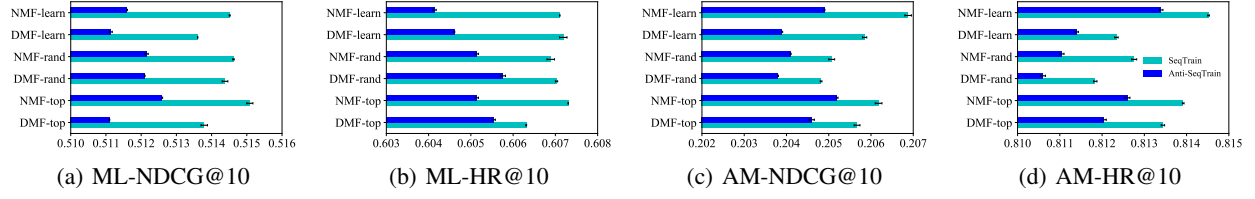


Figure 7: Results of ablation study (SeqTrain module), where (a), (b), (c), and (d) share the same legend.

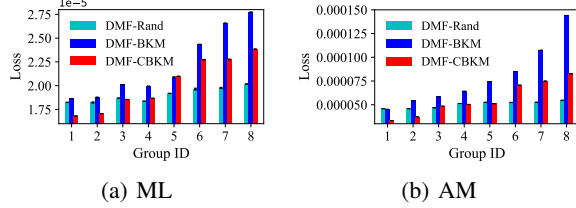


Figure 8: Validation of collaborative cohesion.

mance by preserving collaborative information, (ii) the difference between Retrain and other testing methods tends to be stable as  $S$  increases, which indicates that LASER is robust with large grouping numbers, and (iii) good performance in the learning process leads to good performance in the unlearning process. Due to page limit, we report the results with  $K = 2.5$  in Appendix D.1.

## 4.5 Ablation Study

### Group Module

We compare the effectiveness of different grouping methods (Rand, BKM, and CBKM) and report the results in Figure 5 and 6. For DMF, CBKM outperforms other grouping methods in most of testing cases, while BKM achieves similar performance with Rand, because it cannot properly group the users with the sparse data. For NMF, the results are consistent with DMF, except that all grouping methods achieve similar performance when  $S = 16$ . This is probably because NMF’s strong learning ability helps it to better capture user feature when  $S$  is large, which weakens the effect of different grouping methods [Rendle *et al.*, 2020; Xu *et al.*, 2021]. Additionally, to further compare the effect of sparse data and collaborative embedding, we report their collaborative cohesion  $\rho$  in Appendix D.2. In summary, CBKM groups users effectively in most cases on both datasets.

### SeqTrain Module

In order to study the influence of the SeqTrain module, we verify (i) the effectiveness of training order, and (ii) the validity of collaborative cohesion.

**Training Order** We compare our proposed SeqTrain (easy-to-hard) with anti-SeqTrain (hard-to-easy) which trains all groups in a reverse order. We set  $S$  to 8 and  $K$  to 5 for both datasets. Figure 7 clearly shows that SeqTrain outperforms anti-SeqTrain in all testing cases, which verifies the effectiveness of our proposed SeqTrain module.

**Collaborative Cohesion** We verify the validity of  $\rho$  by studying the relation between testing loss and  $\rho$ . We set  $S$

to 8, sort the groups in descending order according to  $\rho$ , and train DMF on each group from scratch for  $T$  epochs. We report the testing loss in Figure 8. From it, we see that the loss generally grows with the order of groups, which means that  $\rho$  is negatively related with testing loss. That is, the larger the collaborative cohesion, the easier the model can be trained. In other words,  $\rho$  is a valid difficulty measurer which is positively related with model utility. The result of NMF is consistent with DMF, which is reported in Appendix D.3.

## 5 Related Work

**Retrain Unlearning** This approach retrains the model on  $D \setminus E$  in efficient ways, rather than from scratch. Cao and Yang [2015] firstly formed the task of MU and proposed a statistical query learning based unlearning framework. The basic idea is to transform training samples into a reduced number of summations. Thus, the models can retrain more efficiently on these summations. Ginart *et al.* [2019] proposed efficient unlearning algorithms for  $k$ -means clustering with bounds of unlearning time complexity. Lately, Bourtole *et al.* [2021] proposed a general unlearning framework, i.e., SISA, for most of existing machine learning models. SISA divides the dataset into several disjoint subsets and trains one model on each subset. Finally, SISA aggregates all models via majority vote. When SISA receives an unlearning request, it only needs to retrain a subset of data.

**Reverse Unlearning** This approach aims to withdraw the target data lineage from a learned model through reverse operations on  $E$ . Baumhauer, Schöttle, and Zeppelzauer [2020] applied linear filtration on corresponding parameters to unlearn the entire class for logit-based classification. Sekhari *et al.* [2021] focused on convex loss and proposed an unlearning algorithm based on reverse gradient operations with theoretical guarantee.

## 6 Conclusion

In this paper, we propose an erasable recommendation (LASER) framework which consists of *Group module* and *SeqTrain module*. The Group module partitions users into balanced groups according to their collaborative embedding. The SeqTrain module sequentially trains all groups in an easy-to-hard learning order with *collaborative cohesion* as the difficulty measurer. Our theoretical analysis reveals that SeqTrain module can improve model utility. Extensive experiments on two real-world recommendation datasets demonstrate that LASER can not only achieve efficient unlearning, but also outperform the state-of-the-art unlearning models in terms of model utility.

## A Collaboration Embedding via Hypergraph

There are three steps to learn collaborative embedding via hypergraph, i.e., hypergraph building, random walk to sample user sequence, and user sequence embedding. In this section, we first introduce the preliminaries of hypergraph, then present the details of the above three steps.

### A.1 Preliminary of Hypergraph

Agarwal et al. [2006] showed that the hypergraph with edge-independent vertex weight can be reduced to either clique graph or star graph. In order to sufficiently represent the high-order relations in recommendation, we learn collaborative embedding via the hypergraph with edge-dependent vertex weight [Chitra and Raphael, 2019]. For conciseness, we refer to ‘hypergraph with edge-dependent vertex weight’ as ‘hypergraph’ in the rest of this paper.

A hypergraph can be formulated as  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ , where  $\mathcal{V}$  denotes the vertex set  $\{v_i\}$ ,  $\mathcal{E}$  denotes the hyperedge set  $\{e_i\}$  ( $e_i \in 2^{\mathcal{V}}$ ), and  $\mathcal{W}$  denotes the weight set  $\{w(i, j) | \text{for } \forall i, j \text{ that } e_i \in \mathcal{E}, v_j \in e_i\}$ .

### A.2 Hypergraph Building

In order to build a hypergraph, we need association rules to connect a hypergraph with the original bipartite graph which is equivalent to the user-item interaction matrix. There are three rules for building vertex, hyperedge, and weight respectively.

**Vertex Building Rule** We define each user as a vertex in hypergraph, which means  $|\mathcal{V}| = N$ .

**Definition 1** (User’s  $l$ -order reachable neighbors). *In a user-item bipartite graph,  $user_i(user_j)$  is  $user_j(user_i)$ ’s  $l$ -order reachable neighbor if there exists a sequence of adjacent vertices, i.e., path, between  $user_i$  and  $user_j$ , and the number of vertices in this path is smaller than  $l$ .*

**Hyperedge Building Rule** We define the user’s  $l$ -order reachable neighbors, which is represented as a hyperedge in hypergraph, to model the high-order relations. Correspondingly, the users, i.e., neighbors, within the neighborhood are vertices that connected by this hyperedge. Following [Ji et al., 2020], we set  $l = 4$  in this paper.

**Weight Building Rule** We define  $w(i, j)$  as the average rating of  $user_j$  within  $user_i$ ’s  $l$ -order reachable neighbors.

### A.3 Random Walk

A random walk on a hypergraph is typically defined as follows [Chitra and Raphael, 2019]. We modify the traditional random walk to suit recommendation setting and present it in Algorithm 3. Our proposed random walker migrates back and forth between vertices and hyperedge to sample user sequences. From a vertex, the walker choose a hyperedge based on its size, i.e., number of vertices it connects (line 5 in Algorithm 3). From a hyperedge, the walker choose a vertex based on its associated weight (line 6 in Algorithm 3).

Obviously, walk repetition and walk depth are two hyperparameters in random walk. According to our empirical study, we set walk repetition as 4 and walk depth as 8 in this paper.

---

### Algorithm 3 Random Walk on Hypergraph

---

**Input:** Hypergraph:  $\mathcal{G}$ , walk repetition:  $rep$ , walk depth:  $dep$

**Output:** User sequences:  $F \in \mathbb{R}^{(N \cdot rep) \times dep}$

**Procedure:**

```

1: for  $v_i$  in  $\mathcal{V}$  do
2:   for  $j$  in  $[rep]$  do
3:     Start a user sequence with  $v_i, v_{i,1} = v_i$ ;
4:     for  $k$  in  $[dep]$  do
5:       Select a hyperedge  $e$  containing  $v_{i,k}$  with probability proportional to  $|e|$ ;
6:       Select a vertex  $v$  from  $e$  with probability proportional to  $w(e, v)$ ;
7:       Add  $v$  into user sequence,  $v_{i,k+1} = v$ ;
8:     end for
9:   Complete the user sequence and add it to  $F$ ;
10:  end for
11: end for

```

---

### A.4 Sequence Embedding

After random walk, we have  $N \cdot rep$  user sequences. Regarding the sequence of users as the sentence of words, we can apply sequence embedding techniques in natural language processing field to learn the collaborative embedding. In this paper, we apply the widely used Word2Vec [Church, 2017] to learn the embedding from sampled user sequences. The choice of sequence embedding techniques is not a focus of this paper, please refer to the original paper for more details.

## B Balanced Grouping Method

### B.1 Unbalanced Phenomenon

We conduct  $k$ -means clustering on two real-world datasets with  $k = 2, 4, 8, 16$  and report the result in Figure 9. We find that the grouping distributions are highly unbalanced on both datasets.

### B.2 Efficiency Analysis

We use  $c_i$  to denote the training time cost of  $i$ -th group which is in proportion to the group size. Our proposed LASER adopts a sequential training manner, therefore the retraining time cost from  $i$ -th group is  $\sum_{j=i}^S c_j$ . Since we assume that users submit unlearning requests with equal probability, the probability of a request locating in  $i$ -th group is  $c_i/Z$  where  $Z = \sum_i c_i$ . Thus, the expectation of retraining time cost can be written as:

$$\mathbb{E}(C) = \sum_{i=1}^S \left( \left( \sum_{j=i}^S c_j \right) c_i / Z \right). \quad (7)$$

We prove that  $\mathbb{E}(C)$  reaches the minimum when  $c_i = Z/n$ , which means balanced grouping can maximize the unlearning efficiency.

*Proof.* Based on Multinomial Theorem, we can rewrite (7)

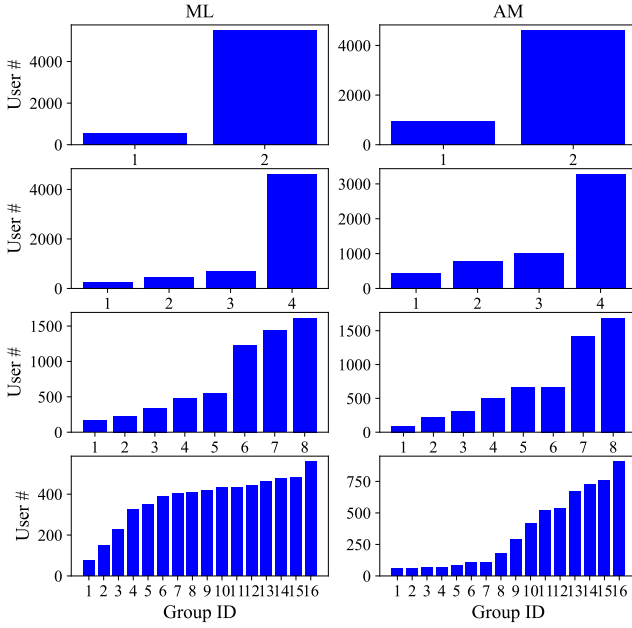


Figure 9: Distributions of group size (user number within the group) via  $k$ -means clustering on two datasets, i.e., MovieLens 1M (ML) and Amazon Digital Music (AM), with  $k = 2, 4, 8, 16$ .

as:

$$\begin{aligned}\mathbb{E}(C) &= \frac{1}{2Z} \left( \left( \sum_{i=1}^n c_i \right)^2 + \sum_{i=1}^n c_i^2 \right) \\ &= \frac{1}{2Z} \left( Z^2 + \sum_{i=1}^n c_i^2 \right).\end{aligned}\quad (8)$$

According to Cauchy-Schwarz Inequality [Bhatia and Davis, 1995], for  $n$  random variables  $x_i, y_i$ , we have:

$$\sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i^2 \geq \left( \sum_{i=1}^n x_i y_i \right)^2. \quad (9)$$

Setting  $y_i = 1$  for all  $i$ , we can get a lower bound of  $\sum_i x_i^2$  as:

$$\sum_{i=1}^n x_i^2 \geq \frac{(\sum_{i=1}^n x_i)^2}{n}. \quad (10)$$

Taking it into (8), we have:

$$\mathbb{E}(C) \geq \frac{Z}{2} \left( 1 + \frac{1}{n} \right) \quad (11)$$

We can easily compute that setting  $c_i = Z/n$  achieves this lower bound of  $\mathbb{E}(C)$ .  $\square$

### B.3 Balanced $k$ -means Clustering

Our proposed balanced grouping method (Algorithm 2) can be applied to a wide range of clustering algorithms. We take balanced  $k$ -means clustering as an example and present the details in Algorithm 4.

---

#### Algorithm 4 *ComputeSimilarity* ( $k$ -means version)

---

**Input:** User embedding matrix:  $B$ , group label:  $\Lambda$

**Output:** Priority list  $P$

**Procedure:**

- 1:  $P =$  empty list;
  - 2: compute  $N$  centroids with respect to  $\Lambda$ ;
  - 3: **for**  $(i, j)$  in all user-centroid pairs **do**
  - 4:   append  $\text{dist}(i, j)$  to  $P$ ;
  - 5: **end for**
- 

## C Proof of Theorem 2

**Theorem 2.** Given a Bayesian prior  $p$  for the model parameters  $\theta$ , we have:

$$\mathcal{U}_p(\theta) = \mathcal{U}(\theta) + \hat{\text{Cov}}[U_\theta, p]. \quad (12)$$

**Recall** Let  $\theta$  denotes the model parameter vector and  $L_\theta(g_i)$  denotes the loss of the model of  $i$ -th group. Adopting empirical risk minimization framework, we have the empirical loss

$$\mathcal{L}(\theta) = \hat{\mathbb{E}}[L_\theta] = \frac{1}{S} \sum_{i=1}^S L_\theta(g_i), \quad \theta^* = \arg \min_{\theta} \mathcal{L}(\theta). \quad (13)$$

Minimizing the empirical loss can be regarded as maximizing model utility [Hacohen and Weinshall, 2019], which is defined as:

$$\begin{aligned}\mathcal{U}(\theta) &= \hat{\mathbb{E}}(U_\theta) = \frac{1}{S} \sum_{i=1}^S U_\theta(g_i) \triangleq \frac{1}{S} \sum_{i=1}^S e^{-L_\theta(g_i)}, \\ \theta^* &= \arg \max_{\theta} \mathcal{U}(\theta).\end{aligned}\quad (14)$$

*Proof.* With the help of the Bayesian prior  $p$ , we can formulate (14) as:

$$\begin{aligned}\mathcal{U}_p(\theta) &= \hat{\mathbb{E}}_p[U_\theta] = \sum_{i=1}^S U_\theta(g_i) p_i = \sum_{i=1}^S e^{-L_\theta(g_i)} p_i, \\ \theta^* &= \arg \max_{\theta} \mathcal{U}_p(\theta),\end{aligned}\quad (15)$$

where  $p_i$  denotes the prior probability of  $i$ -th group. As  $U_\theta$  and  $p$  are two random variables in  $\mathcal{U}_p(\theta)$ , we can rewrite (15) as follows:

$$\begin{aligned}\mathcal{U}_p(\theta) &= S \hat{\mathbb{E}}[U_\theta] \hat{\mathbb{E}}[p] + \sum_{i=1}^S (U_\theta(g_i) - \hat{\mathbb{E}}[U_\theta]) (p_i - \hat{\mathbb{E}}[p]) \\ &= \mathcal{U}(\theta) + \hat{\text{Cov}}[U_\theta, p].\end{aligned}\quad (16)$$

$\square$

## D More Empirical Results

### D.1 Utility (G3)

We report the results of utility metrics with two unlearning request types (rand@2.5 and top@2.5) in Figure 10 and Figure 11. We observe that the results are consistent with that of rand@5 and top@5.



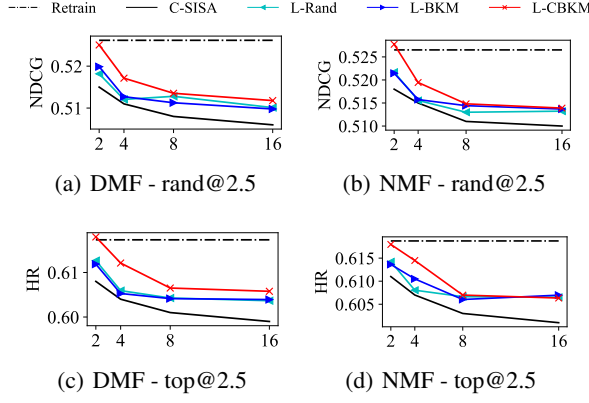


Figure 10: NDCG@10 and HR@10 results for ML dataset with std under  $1e-3$ .

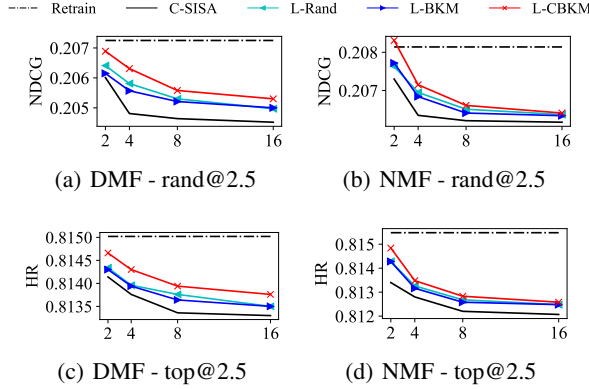


Figure 11: NDCG@10 and HR@10 results for AM dataset with std under  $1e-3$ .

## D.2 Comparison of Sparse Data and Collaborative Embedding

We compare the effect of sparse data (BKM) and collaborative embedding (CBKM) in terms of collaborative cohesion and report the result in Figure 12. We find that our proposed collaborative embedding can achieve much higher collaborative cohesion than the original sparse data.

## D.3 Verification of Collaborative Cohesion

To verify the validity of  $\rho$ , we report the testing loss of NMF in Figure 8. We observe the consistent results with DMF.

## References

[Agarwal *et al.*, 2006] Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pages 17–24, 2006.

[Baumhauer *et al.*, 2020] Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine unlearning: Linear filtration for logit-based classifiers. *arXiv preprint arXiv:2002.02730*, 2020.

[Bhatia and Davis, 1995] Rajendra Bhatia and Chandler Davis. A cauchy-schwarz inequality for operators with applications. *Linear Algebra Appl*, 223:119–129, 1995.

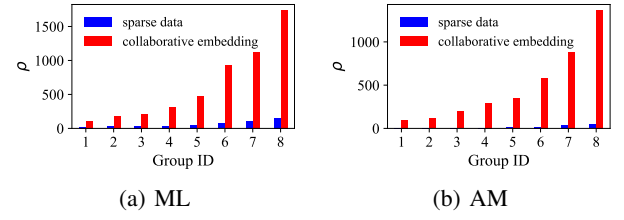


Figure 12: Comparison of sparse data and collaborative embedding.

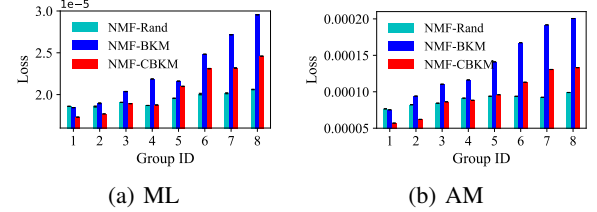


Figure 13: Validation of collaborative cohesion.

[Bourtole *et al.*, 2021] Lucas Bourtole, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *Proceedings in the 42nd IEEE Symposium on Security and Privacy (SP)*, 2021.

[Cao and Yang, 2015] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *Proceedings in the 36th IEEE Symposium on Security and Privacy (SP)*, pages 463–480, 2015.

[Chitra and Raphael, 2019] Uthsav Chitra and Benjamin Raphael. Random walks on hypergraphs with edge-dependent vertex weights. In *International Conference on Machine Learning (ICML)*, pages 1172–1181, 2019.

[Church, 2017] Kenneth Ward Church. Word2vec. *Natural Language Engineering*, 23(1):155–162, 2017.

[EU, 2014] Council EU. Council regulation (eu) on 2012/0011. Website, 2014. <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52012PC0011>.

[Ginart *et al.*, 2019] Tony Ginart, Melody Guan, Gerg Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. In *Advances in 32nd Neural Information Processing Systems (NeurIPS)*, 2019.

[Hacohen and Weinshall, 2019] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 2535–2544, 2019.

[Harper and Konstan, 2015] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm Transactions on Interactive Intelligent Systems (TIIS)*, 5(4):1–19, 2015.

[He and McAuley, 2016] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceed-*

- ings of the 25th International Conference on World Wide Web (WWW), pages 507–517, 2016.
- [He *et al.*, 2015] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1661–1670, 2015.
- [He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, pages 173–182, 2017.
- [Ji *et al.*, 2020] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. Dual channel hypergraph collaborative filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2020–2029, 2020.
- [Kanungo *et al.*, 2002] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(7):881–892, 2002.
- [Rendle *et al.*, 2020] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. Neural collaborative filtering vs. matrix factorization revisited. In *Fourteenth ACM Conference on Recommender Systems*, pages 240–248, 2020.
- [Reynolds, 2009] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741:659–663, 2009.
- [Schafer *et al.*, 2007] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.
- [Schelter *et al.*, 2021] Sebastian Schelter, Stefan Grafberger, and Ted Dunning. Hedgecut: Maintaining randomised trees for low-latency machine unlearning. In *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*, pages 1545–1557, 2021.
- [Sekhari *et al.*, 2021] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In *Advances in 34th Neural Information Processing Systems (NeurIPS)*, 2021.
- [Shi *et al.*, 2014] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):1–45, 2014.
- [Wang and Zhang, 2007] Fei Wang and Changshui Zhang. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67, 2007.
- [Wang *et al.*, 2021] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.
- [Xu *et al.*, 2021] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Rethinking neural vs. matrix-factorization collaborative filtering: the theoretical perspectives. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 11514–11524, 2021.
- [Xue *et al.*, 2017] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 17, pages 3203–3209, 2017.