LightFR: Lightweight Federated Recommendation with Privacy-preserving Matrix Factorization

HONGLEI ZHANG, School of Computer and Information Technology, Beijing Jiaotong University, China
FANGYUAN LUO, School of Computer and Information Technology, Beijing Jiaotong University, China
JUN WU, School of Computer and Information Technology, Beijing Jiaotong University, China
XIANGNAN HE, University of Science and Technology of China, China
YIDONG LI*, School of Computer and Information Technology, Beijing Jiaotong University, China

Federated recommender system (FRS), which enables many local devices to train a shared model jointly without transmitting local raw data, has become a prevalent recommendation paradigm with privacy-preserving advantages. However, previous work on FRS performs similarity search via inner product in continuous embedding space, which causes an efficiency bottleneck when the scale of items is extremely large. We argue that such a scheme in federated settings ignores the limited capacities in resource-constrained user devices (*i.e.*, storage space, computational overhead, and communication bandwidth), and makes it harder to be deployed in large-scale recommender systems. Besides, it has been shown that transmitting local gradients in real-valued form between server and clients may leak users' private information. To this end, we propose a lightweight federated recommendation framework with privacy-preserving matrix factorization, *LightFR*, that is able to generate high-quality binary codes by exploiting learning to hash technique under federated settings, and thus enjoys both fast online inference and economic memory consumption. Moreover, we devise an efficient federated discrete optimization algorithm to collaboratively train model parameters between the server and clients, which can effectively prevent real-valued gradient attacks from malicious parties. Through extensive experiments on four real-world datasets, we show that our LightFR model outperforms several state-of-the-art FRS methods in terms of recommendation accuracy, inference efficiency and data privacy.

CCS Concepts: • **Information systems → Collaborative filtering**; • **Security and privacy → Privacy protections**.

Additional Key Words and Phrases: Federated Recommender System, Matrix Factorization, Privacy Preservation, Learning to Hash

1 INTRODUCTION

Recommender system (RS) is an effective functionality for alleviating information overload [35], with the rapid growth of online user interaction data. The significance of RS cannot be overstated, regarding their widespread utilization in industry, such as web search and e-commerce platforms [8, 50], and their potential to surmount obstacles with user modeling in academia [35, 47]. However, such a scheme that all behavior data is collected

*Corresponding author.

Authors' addresses: Honglei Zhang, honglei.zhang@bjtu.edu.cn, School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China; Fangyuan Luo, fangyuanluo@bjtu.edu.cn, School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China; Jun Wu, wuj@bjtu.edu.cn, School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China; Xiangnan He, hexn@ustc.edu.cn, University of Science and Technology of China, Hefei, China; Yidong Li, School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China, ydli@bjtu.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.
1046-8188/2022/12-ART $15.00
https://doi.org/10.1145/3578361

Just Accepted

in a centralized manner, will inevitably result in the leakage of private user information [29, 53]. Thus, privacy concerns in RS arise. Considering the sensitivity of user personal data, regulations such as General Data Protection Regulation (GDPR)[1], have been put into effect to restrict the centralized collection of users' private data. Such actions lead to the occurrence of data isolation trend, which aggravates the data sparsity issue in RS scenarios.

Focusing on this dilemma, federated recommender system (FRS) has received widespread attention [45], due to the advantages of privacy protection and considerable performance. In FRS, a global model in the server can be aggregated and updated from user-specific local models with the collaboration of the server and clients, ensuring that users' private interaction data never leaves their devices. Among them, the more prominent work is Federated Collaborative Filtering (FCF) [2], where each user latent vector is updated locally and the item latent matrix is transmitted and updated collaboratively between the server and clients. Subsequently, FedFast [28] improves the client sampling strategy and the active aggregation mechanism on the shoulders of FCF, and speeds up the convergence efficiency while guaranteeing the model efficacy. Although the above methods realize high-quality federated recommendations, it requires transmitting a full amount of item latent matrix between the server and clients, which brings a huge carbon footprint and is unaffordable for cost-conscious clients.

Following that, some work explores reducing the payload of the entire item latent matrix by meta learning techniques [22, 39]. For example, MetaMF [22] adopts the meta recommender to deploy smaller models on the client to reduce memory consumption and PrivRec [39] introduces a first-order meta-learning model to enable fast adaptation on local devices. Besides, there are also some attempts to utilize knowledge distillation mechanisms concentrating on transferring knowledge from a heavy model to a light one to achieve the purpose of designing lightweight recommender models on clients [38, 43]. Whereas these methods alleviate the storage and communication overheads to some extent, they can not shorten the inference time on clients since they should perform forward propagation in dense embedding space, which is unacceptable on computing-sensitive clients when the number of candidates is quite large. Besides, such solutions need to transmit the original real-valued gradient data, and it has been proved that the transmission of local gradients between the server and clients in continuous embedding space may leak users' private information [6]. Consequently, it is a crucial challenge for existing FRS techniques to enhance the capability of preserving users' privacy.

Along with this research line, several endeavors are devoted to facilitating the privacy of FRS [6, 21]. For instance, FedRec [21] employs a hybrid filling strategy to randomly sample some virtual items to protect the actual gradients, and FMF-LDP [27] adopts the differential privacy mechanism and a proxy network to reduce the fingerprint surface for implicit data. Although these mechanisms can protect users' privacy to a certain level, they require huge memory, calculation and communication overheads, which are often not applicable to resource-constrained clients under federated settings. We argue that previous work on FRS generally provides top-K recommendations among all existing items via inner product in dense vector space, which is the leading factor for the challenge of high cost of resources in large-scale recommendation scenarios. Intuitive results can be observed from Fig. 1, which shows that as the number of items increases, the storage cost, communication overhead, and inference time on the user terminal devices will rise dramatically. Note that different from centralized RS in a data server, FRS that requires to be deployed on local clients with low-resource settings, has more stringent restrictions for model scale. Hence, a lightweight recommendation model is even more urgent in FRS. In summary, none of these methods take into account both the issues of *efficiency* and *privacy*, which are the two primary challenges for real-world FRS.

Considering the shortcomings of existing work, we believe it is essential to develop a lightweight and privacy-preserving FRS, which not only benefits from the low cost of resources, but also increases the capability of privacy protection. To achieve this, we resort to the notion of learning to hash to obtain the binary representations of users and items so that the efficiency and privacy issues can be effectively addressed. However, solving the discrete

[1]https://gdpr-info.eu/

Just Accepted

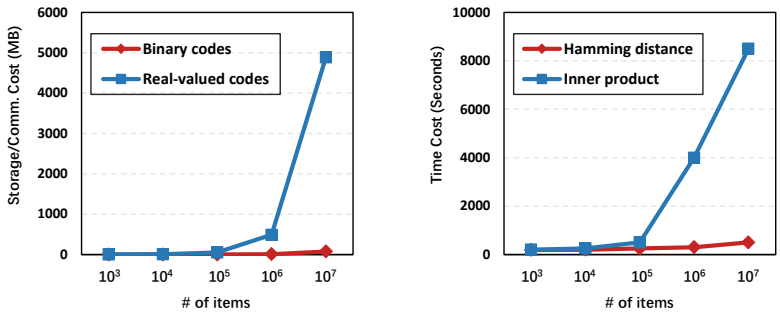


Fig. 1. Comparisons on storage, communication cost (on left panel) and inference time (on right panel) for Hamming distances (binary codes) and inner products (real-valued codes) on various-scaled items. The experiments are conducted by randomly generating binary codes and real-valued vectors with a length of 64 on 10^5 users and $10^3 - 10^7$ items, and we report the average results over 5 repetitions[2]. Since the number of items has a wide range, we display the x-axis evenly in the form of exponential intervals.

Just Accepted

optimization in federated settings is not trivial, since it is infeasible to utilize the straightforward heuristics because it is generally an NP-hard problem which involves exponential combinatorial searches for the binary codes. Hence, it is imperative to design an efficient federated discrete algorithm between the server and clients, which can embed the preferences of users into the discrete Hamming space, and meanwhile reduce resource utilization on both the server and clients in a privacy-persevering manner.

To remedy these issues in a unified way, we present *LightFR*, a principled lightweight and privacy-preserving framework for FRS built on learning to hash and federated learning techniques. Specifically, we introduce a federated discrete optimization algorithm that can solve the above-mentioned issue in a computationally tractable fashion in federated settings, which can produce suitable binary user representation on local clients and binary item representations on the server side. By introducing learning to hash into LightFR, it can kill three birds with one stone. Firstly, by encoding real-valued data vectors into compact binary codes, hashing makes efficient in-memory storage of massive data feasible in resource-limited user devices, and in an analogous way, the utilization of binary codes can reduce communication overheads as well. Secondly, as similarity calculation by inner product in a continuous vector space is replaced by bit operations in a discrete Hamming space, the time complexity of linear search is significantly reduced. Thirdly, by encoding the continuous real-valued vector into discrete binary codes, we prove that our proposed LightFR model is capable of effectively avoiding the leakage of user's sensitive information. Overall, the main contributions of this work are listed as follows:

- We tackle the problems of efficiency and privacy toward FRS in a unified way, *i.e.*, the heavy parameterization inherited from real-valued representations in Euclidean space. In light of this, we seek solutions from the Hamming space by exploiting learning to hash technique, in which high-quality binary codes are obtained in the server and clients. To the best of our knowledge, this work represents the first effort towards this target in FRS.
- To effectively train the discrete parameters in federated settings, we propose an efficient federated discrete optimization strategy between the central server and distributed clients, which facilitates both efficient and effective retrieval in terminal devices. Besides, we discuss the superiority of its multiple beyond-accuracy metrics, *i.e.*, storage/communication efficiency, inference efficiency, and privacy preservation from the theoretical perspective.

[2]We perform the similarity search on a machine with 2.0GHz Intel Xeon E5-2640 processor.

- Extensive experiments on four real-world datasets with different volumes demonstrate the advantages of our model on effectiveness, efficiency and privacy over several state-of-the-art FRS techniques.

The rest of this work is organized as follows. The related work *w.r.t.* matrix factorization, learning to hash and federated recommender system, is comprehensively discussed in Section 2. In Section 3, we illustrate the proposed LightFR in detail and discuss its superiority from multiple perspectives, *i.e.*, storage/communication efficiency, inference efficiency, and privacy preservation. Experimental settings and model performance evaluation are shown in Section 4. Finally, we conclude this work and put forward future work.

2 RELATED WORK

In this section, we briefly review three relevant areas to this work, *i.e.*, matrix factorization, learning to hash and federated recommender system. For a more comprehensive summary of the corresponding directions, please refer to the survey papers [35, 37, 45].

2.1 Matrix Factorization

Matrix factorization (MF), also known as latent factor model, has become a popular direction for collaborative filtering family in recommender systems [13, 17]. The goal of MF is to map original users and items into a common latent subspace, in which the similarities between users and items are calculated by inner products using their latent vectors [31]. Formally, assume that there are n users, m items and a user-item rating matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$ in a website and the latent vector of user u and item i is denoted as f-dimensional embeddings, $\mathbf{p}_u \in \mathbb{R}^f$ and $\mathbf{q}_i \in \mathbb{R}^f$, so the observed rating r_{ui} of user u on item i is estimated by the inner product of respective latent vectors, *i.e.*, $\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$. A general objective is to minimize the following squared loss with regularization term:

$$\min_{\mathbf{p}_u, \mathbf{q}_i} \sum_{(u,i,r_{ui}) \in \Omega} \left(r_{ui} - \mathbf{p}_u^T \mathbf{q}_i \right)^2 + \lambda \left(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2 \right) \tag{1}$$

where $\mathbf{P} \in \mathbb{R}^{f \times n}$ and $\mathbf{Q} \in \mathbb{R}^{f \times m}$ are the user and item latent matrix composed of all user and item latent vectors, respectively. Besides, Ω is a set of triplets of observed entries and $\lambda > 0$ is a trade-off hyper parameter to avoid the over-fitting problem. The above loss function can be solved by (stochastic) gradient descent or alternating least square algorithms.

Owing to its high capability and flexibility, MF has attracted a lot of attention for many years. Early studies mainly focused on how to fuse side information via traditional mechanisms to improve recommendation performance [14, 48]. Koren proposed SVD++ model by incorporating implicit feedback into MF method which only exploits explicit ratings [16]. Hu et al. introduced the influence of geographical neighbors, business's review and category information into the delicate matrix factorization [14], and again proved its high flexibility. Apart from fusing more information, MF can also be seamlessly integrated with other advanced models [1, 12, 18]. Agarwal et al proposed to introduce latent dirichlet allocation (LDA) into MF framework [1], where the use of an LDA prior is to regularize item factors and the combination of them can provide interpretable user factors as affinities to latent item topics. He et al. proposed neural collaborative filtering (NCF) [12], which incorporates a multi-layer perceptron into MF and can better model the user-item interactions with non-linear transformations. In short, a number of studies have investigated the superiority of fusing side information [35] or complicated models [40] to enhance vanilla MF.

2.2 Learning to Hash

Recently, hashing has gained increasing attention due to its great efficiency in retrieving relevant items from massive data. The goal of hashing is to construct a mapping function to index each data point into a compact binary code, where the Hamming distances of similar objects are minimized and that of dissimilar ones are

Just Accepted

maximized. There are two main kinds of hashing-based methods, *i.e.*, locality sensitive hashing (LSH) [15] and learning to hash (L2H) [37], where the formers are data-independent and use predefined hash functions without considering the underlying dataset, while the latters are data-dependent and learn tailored hash functions for specific datasets. Despite an extra training process, recent work showed that L2H greatly surpasses LSH in querying efficiency.

Studies in L2H have proceeded along two dimensions: two-stage approaches [24, 52] and learning hash codes directly [34, 49]. For this research line of two-stage approaches, the first stage is to learn continuous representations for data, which are subsequently binarized into hashing codes using *sign* threshold as a separate post-processing step. For instance, Zhou et al. learned user-item features with traditional CF and then rotated their learned features by running Iterative Quantization (ITQ) to acquire hash codes [52]. However, such two-stage approaches are well-known to suffer from a large quantization loss, which is one of the main reasons why researchers are turning to the investigation of learning hash codes directly, where the binary codes are optimized straightforwardly rather than through a two-step approach. For example, Zhang et al. learned hash codes of users and items directly and further investigated additional constraints to improve generalization by better utilizing the Hamming space [49]. Following that, Zhang et al. proposed a hashing based deep learning framework to unify the user-item interactions and the item content data to overcome the issues of data sparsity and cold-start, while improving the efficiency of online recommendation [51]. However, the training process of these hashing methods mentioned above is usually conducted on centralized data. Hence, they are heavily not suitable for FRS scenario, where it has distinct advantages on privacy protection over centrally stored recommender systems, which is exactly the main motivation of our work.

2.3 Federated Recommender System

Federated Learning (FL) is a promising machine learning paradigm in recent years since it can enable collaborative learning across a variety of clients without sharing local private data [26, 46]. In general, there are two major components in the standard FL framework, where one is the client which trains the local models on their private user data independently, and the other is the server which aggregates the local models (gradients) uploaded from the clients to the global one. As a result of its role in ensuring privacy protection, there are many efforts to improve the basic FL framework, such as FedAvg [26], FedProx [19] and FedRep [7]. In recommendation scenario, user private information, *e.g.*, user's attribute and behavior interactions with items, is considerably sensitive information and probably cause identity information leakage if attacked by malicious parties [29, 41]. Hence, some recent endeavors have developed federated recommender system (FRS) for user privacy preservation while still maintaining considerable performance [23, 45]. Federated Collaborative Filtering (FCF) [2] and FedRec [21] are two pioneering *privacy-by-design* works establishing a novel federated learning framework to learn the user and item embeddings on top of matrix factorization and the former is designed for implicit feedback, while the latter is for explicit feedback. However, Chai et al. argue that the model updates sent to the server in the original real-valued form as the aforementioned approaches do, may contain sensitive information to uncover raw data [6]. Along this path of research, Li et al. proposed to employ differential privacy to limit the exposure of the data in FRS [20]. Besides, FedMF introduced homomorphic encryption into the FCF to ensure the confidentiality of parameter transmission [6].

Aside from the privacy issue in FRS framework, there exists a great efficiency challenge on storing the global model in clients and transmitting the whole parameters between server and clients. From this research line, some efforts adopt meta learning mechanisms to reduce the payload on the clients [22, 39]. For instance, Wang et al proposed to employ the approximated first-order gradients for one-stage meta learning, thereby reducing computational burden while maintaining a comparable performance [39]. Besides, Lin et al. proposed MetaMF to deploy a big meta network into the server while deploying a small recommender model into the device to perform

Just Accepted

Table 1. Comparison of different FRS methods relating to memory efficiency, inference efficiency, communication efficiency and privacy enhancement. The Eff. and Enh. denote Efficiency and Enhancement, respectively.

Models	Memory Eff.	Inference Eff.	Communication Eff.	Privacy Enh.
FCF[2]	×	×	×	×
FedMF[6]	×	×	×	✓
FedRec[21]	×	×	×	✓
MetaMF[22]	✓	×	✓	×
PrivRec[39]	×	×	✓	✓
LightFR	✓	✓	✓	✓

rating prediction [22]. However, MetaMF may pose some privacy concerns since it necessitates the procedure of initializing the embeddings for all users on the server side. Additionally, there is also outstanding work that leverages knowledge distillation methodology to achieve the goal of deploying lightweight recommender models on user devices [38, 43]. Specifically, Wang et al. introduced LLRec framework, whose efficiency and robustness are maintained via the teacher-student training protocol, and then to perform the next point-of-interest recommendation task locally on resource-constrained clients [38]. However, all these methods fail to take into account the challenges of efficiency (*i.e.*, memory, calculation and communication) and privacy at the same time, which are the two major issues for real-world FRS. To better demonstrate the advantages of our approach, we summarize the qualitative comparisons between LightFR and existing FRS methods on efficiency and privacy in Table 1. As for the quantitative analysis of the aforementioned aspects, please refer to Fig. 3 in Section 4.2.1 to see the related experimental results.

3 THE PROPOSED LIGHTFR FRAMEWORK

In this section, we first formally describe the preliminaries, and then introduce the details of our proposed LightFR framework for efficient and privacy-preserving recommendation, followed by the federated discrete optimization algorithm delicately designed for FRS. Finally, we thoroughly discuss its superiority on multiple beyond-accuracy metrics from a theoretical perspective. For clarity, we list some notations frequently used throughout the work in Table 2.

3.1 Preliminary

Unlike fully centralized recommender systems, FRS hardly establishes a complete user-item rating matrix $\mathbf{R}^{n\times m}$ in the server since it no longer retains the fully observed dataset Ω for the sake of users' privacy. Specifically, we assume that there are a set of independent users $\mathcal{U}$, and a set of items $\mathcal{I}$ stored in the central server[3]. Following the FL principles, each user $u \in \mathcal{U}$ owns a local private dataset Ω_u consisting of some feedback tuples $(u, i, r_{ui} | i \in \mathcal{I}_u)$, where $\mathcal{I}_u$ represents observed items of client u. The goal of FRS is to predict the rating of user u to each unseen item $i \in \mathcal{I} \backslash \mathcal{I}_u$ and then recommend the top ranked items to the target user. For federated training process, the gradients of users and items are calculated locally with the Eq.(1). Specifically, the local update for their own user embedding $\mathbf{p}_u$ is performed independently without requiring any other user's private data:

$$\mathbf{p}_u = \mathbf{p}_u - 2\eta \left(\sum\nolimits_{i \in \mathcal{I}_u} (\mathbf{p}_u^T \mathbf{q}_i - r_{ui}) \mathbf{q}_i + \lambda \mathbf{p}_u \right) \tag{2}$$

[3]The meaning of the terms "user", "client", and "device" is same under federated settings, and we use them interchangeably.

Just Accepted

Table 2. The list of notations and corresponding explanations.

Notation	Explanation
$\mathcal{U}$; $\mathcal{I}$	user (client) set; item set
n; m	total number of users; total number of items
$\mathbf{R}$; r_{ui}	rating matrix whose element r_{ui} denotes the rating score for user u on item i
u, v; i, j	the specific user u, $v \in \mathcal{U}$; the specific item i, $j \in \mathcal{I}$
Ω_u; Ω_i	local private dataset in client u; global feedback set interacted with item i
$\mathcal{I}_u$; $\mathcal{U}_i$	the local observed item set of client u; global user set who interacts with item i
$\mathbf{p}_u$; $\mathbf{q}_i$	the real-valued embedding of user u; real-valued embedding of item i
$\Delta\mathbf{q}_i^u$	the gradient towards the item embedding vector $\mathbf{q}_i$ from the client u
$\mathbf{P}$; $\mathbf{Q}$	the user real-valued embedding matrix; item real-valued embedding matrix
$\mathbf{b}_u$; $\mathbf{d}_i$	the binary vector of user u; binary vector of item i
b_{uk}; $b_{u\bar{k}}$	the k-th bit code of the user binary vector $\mathbf{b}_u$; the rest codes of $\mathbf{b}_u$ excluding the b_{uk}
Δd_{ik}^u	the gradient towards the k-th bit of item binary vector $\mathbf{d}_i$ from the client u
$\mathbf{B}$; $\mathbf{D}$	the user binary embedding matrix; item binary embedding matrix
$\Delta\mathbf{D}_t^u$	the gradient matrix towards item binary matrix $\mathbf{D}$ uploaded from client u at iteration t
$\mathcal{U}_s$	a subset of clients randomly selected by the coordinated server
η; λ	the learning rate; regularization parameter
T; E	the number of training rounds between the server and clients; number of local training epochs
p	the fraction of clients to participate in current training round

where η is the learning rate. Conversely, the item embedding $\mathbf{q}_i$ is updated globally on the server by aggregating local item gradient $\Delta\mathbf{q}_i^u$ uploaded from each client u:

$$\mathbf{q}_i = \mathbf{q}_i - 2\eta\left(\sum\nolimits_{u\in\mathcal{U}_i}\Delta\mathbf{q}_i^u\right) \tag{3}$$

where the item gradient $\Delta\mathbf{q}_i^u = (\mathbf{p}_u^T\mathbf{q}_i - r_{ui})\mathbf{p}_u + \lambda\mathbf{q}_i$ is calculated by the local client u, and then the updated item matrix $\mathbf{Q}$ is sent down to all clients. The federated process repeats until convergence. For federated testing period, the client u downloads the up-to-date item embedding matrix $\mathbf{Q}$ from FRS server and estimates ratings by inner product, *i.e.*, $r_u = \mathbf{p}_u^T\mathbf{Q}$ in local device.

3.2 The LightFR Model

As mentioned earlier, to fulfill the training process of FRS, the item embedding matrix $\mathbf{Q}$ and corresponding gradient matrix $\Delta\mathbf{Q} = [\Delta\mathbf{q}_1, \Delta\mathbf{q}_2, \cdots, \Delta\mathbf{q}_m]$ are exchanged between the server and clients. We emphasize that the parameters scale linearly in Euclidean space with the increasing number of items (as shown in Fig. 1), posing a significant efficiency bottleneck in terms of storage, communication and inference time in resource-constrained local devices. Besides, transmitting gradient information in its raw real-valued form could result in privacy issues. In light of these challenges, we present our efficient and privacy-preserving method, with the assistance of binary codes generated by learning to hash technique in Hamming space, to the point where it is suitable for FRS when deployed in production.

An overview of the proposed framework is shown in Fig. 2. In the beginning, the server randomly initializes an item binary matrix $\mathbf{D} = [\mathbf{d}_1, \cdots, \mathbf{d}_m] \in \{\pm 1\}^{f\times m}$ and clients initialize their own user binary vector $\mathbf{b}_u \in \{\pm 1\}^f$.

[4]For simplicity, we omit the subscript t which denotes the t-th iteration in image caption part.

Just Accepted

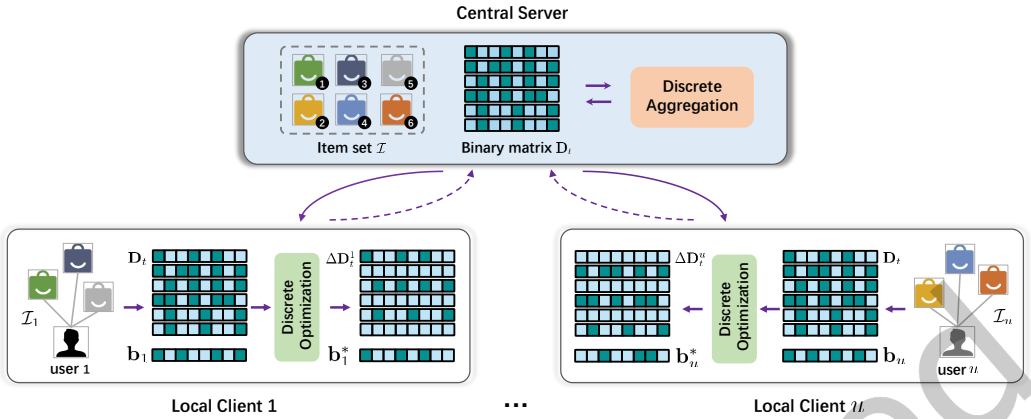


Fig. 2. The framework of our proposed LightFR approach. Firstly, the global item binary matrix $\mathbf{D}$ is initialized in the central server[4], and then delivered to the distributed clients for local updates. Subsequently, each local client receives the latest item binary matrix $\mathbf{D}$ and (in the first round) initializes his own private binary vector $\mathbf{b}_u$. Later on, each client updates his own binary vector $\mathbf{b}_u$ and calculates the item gradient matrix $\Delta\mathbf{D}^u$ to be uploaded through the (local) discrete optimization module. After receiving the gradient matrix uploaded by all selected clients, the update process is conducted via the (global) discrete aggregation module on the server side. Finally, the latest item binary matrix is distributed to each client for next-step local optimization.

Just Accepted

Note that the server holds the entire item set $\mathcal{I}$, while each user u exclusively has his/her private consumed item set $\mathcal{I}_u$. At iteration t, a subset of users $\mathcal{U}_s$ is randomly selected, and then each user $u \in \mathcal{U}_s$ downloads the latest global model (*i.e.*, the item binary matrix $\mathbf{D}_t$) to the local device. Subsequently, the private user binary vector $\mathbf{b}_u$ is updated to $\mathbf{b}_u^*$ and the gradient of the item binary matrix $\Delta\mathbf{D}_t^u$ is calculated by (local) discrete optimization module using private local dataset Ω_u. After the central server receives all local gradients submitted by users $\mathcal{U}_s$, it aggregates the collected gradients to facilitate the global model update by (global) discrete aggregation module. Finally, the server sends the latest item binary matrix to each client for the next round of optimization. The federated discrete optimization process is repeated until it converges. By jointly optimizing the pre-defined hashing-based loss functions between the server and clients, we can obtain the well-trained private user binary vectors in each client and the resulting item binary matrix in the server. Next we will introduce in detail the loss function of learning binary codes for users and items in FRS scenario.

The objective of LightFR is to tackle a joint discrete optimization task in federated settings, instead of solving the continuous optimization problem widely explored in traditional FRS. As a result, the similarities of two binary vectors cannot be measured directly by the inner product operations which will result in significant efficiency bottlenecks in the case of large-scale items, and hence the Hamming similarity is utilized to assess the proximity between the two ones. Instead of the Euclidean space, our proposed method aims to find binary codes in Hamming space, guaranteeing efficient storage and fast similarity search across users and items in federated

scenarios. Generally, the similarity between user u and item i in Hamming space is defined as:

$$\begin{aligned}
\mathrm{sim}(\mathbf{b}_u, \mathbf{d}_i) &= \frac{1}{f}\sum_{k=1}^{f}\mathbb{I}(b_{uk} = d_{ik}) \\
&= \frac{1}{2f}\left(\sum_{k=1}^{f}\mathbb{I}(b_{uk} = d_{ik}) + f - \sum_{k=1}^{f}\mathbb{I}(b_{uk} \neq d_{ik})\right) \\
&= \frac{1}{2f}\left(f + \sum_{k=1}^{f} b_{uk}d_{ik}\right) \\
&= \frac{1}{2} + \frac{1}{2f}\mathbf{b}_u^T\mathbf{d}_i
\end{aligned} \tag{4}$$

where b_{uk} and d_{ik} denotes the k-th bit of the user and item binary codes $\mathbf{b}_u$ and $\mathbf{d}_i$, respectively. Besides, $\mathbb{I}(\cdot)$ represents the indicator function that yields 1 if the statement is true and 0 otherwise. Apparently, the value range of the Hamming similarity is ranging from 0 to 1, which just satisfies the basic requirements for similarity and $sim(\mathbf{b}_u, \mathbf{d}_i) = 0$ if all the bits of $\mathbf{b}_u$ and $\mathbf{d}_i$ are totally different and $sim(\mathbf{b}_u, \mathbf{d}_i) = 1$ if $\mathbf{b}_u = \mathbf{d}_i$. Note that the Hamming similarity has a highly efficient hardware-level implementation, which allows us to find similar items in time that is independent to the total number of items [33].

Similar to the problem of conventional MF in Eq.(1), the preferences between users and items should be preserved by the above similarities with their respective binary codes, and the user-item rating matrix should be reconstructed by that as well. Therefore, the objective of our proposed LightFR built on FedAvg [26] is formulated as follows:

$$\begin{aligned}
\mathcal{L} = \sum_{u\in\mathcal{U}}\frac{|\mathcal{I}_u|}{N}\underbrace{\sum_{i\in\Omega_u}(r_{ui} - \mathrm{sim}(\mathbf{b}_u, \mathbf{d}_i))^2}_{\text{local-specific loss}} + \underbrace{\lambda(||\sum_u \mathbf{b}_u||^2 + ||\sum_i \mathbf{d}_i||^2)}_{\text{balanced constraint term}} \\
s.t.\ \mathbf{b}_u \in \{\pm 1\}^f, \mathbf{d}_i \in \{\pm 1\}^f
\end{aligned} \tag{5}$$

where N is the total number of instances over all clients, and $|\mathcal{I}_u|$ denotes the length of local samples on client u. Note that, due to the binary constraints above, the conventional regularization term $\|\mathbf{B}\|_F^2 + \|\mathbf{D}\|_F^2$ used in Eq.(1) becomes constant and hence is removed in Eq.(5). However, in order to obtain the informative binary representations, a balanced constraint is utilized to maximize the information entropy of each bit, given in the form of constraint term in Eq.(5), and the trade-off parameter λ controls the proportion between minimizing the squared loss and the balanced constraints. Following that, we will detail our proposed federated discrete optimization method for optimizing the loss function Eq.(5) in FRS scenario.

3.3 Federated Discrete Optimization

The goal of our federated discrete optimization is to find appropriate binary codes for users and items in federated settings such that the user preference over items is accurately preserved in Hamming space with their respective binary codes. However, solving the discrete optimization problem in Eq.(5) by straightforward heuristics is challenging since it is generally an NP-hard problem which involves $O(2^{(m+n)f})$ combinatorial searches for the binary codes. To this end, we introduce a collaborative alternating optimization method that can solve the above-mentioned question in a computationally tractable fashion in federated settings. Specifically, the proposed optimization algorithm mainly consists of two modules, *i.e.*, (local) discrete optimization in each client and (global) discrete aggregation in the central server. Thereinto, the local discrete optimization module is primarily

Just Accepted

responsible for updating their respective user binary vectors and calculating the gradients of the binary item matrix to be uploaded using local data at the client; while the principal mission of the global discrete optimization module is to aggregate the gradients of items from multiple clients for updating the discrete item latent matrix. Next, we will elaborate on each module at length.

3.3.1 (Local) Discrete Optimization. In this part, we will introduce how to update the private user binary vector and calculate the gradients for item binary matrix with their private data leaving locally. We employ an alternating optimization strategy for solving the federated discrete optimization problem shown in Eq.(5) that iterates the following two steps: (1) minimization with regard to each $\mathbf{b}_u$ with $\mathbf{d}_i$ fixed in clients; and (2) minimization with regard to $\mathbf{d}_i$ with $\mathbf{b}_u$ fixed by computing the gradients in clients and aggregating them in the server. Concretely, we in turn calculate the gradients for $\mathbf{b}_u$ and $\mathbf{d}_i$, given another one fixed, and then update the private user binary vector using the locally computed user gradients and upload the item binary gradients to the server for aggregation.

First, we aim to optimize the private user binary vector $\mathbf{b}_u$ via fixing the item binary vector $\mathbf{d}_i$ in his/her own client without accessing any data from other clients. We update the user binary vector $\mathbf{b}_u$ of each client in parallel according to the following expanded formulation:

$$\underset{\mathbf{b}_u \in \{\pm 1\}^f}{\arg\min}\, \mathcal{L}^u_{local} = \frac{1}{4f^2}\left(\sum_{i\in\Omega_u} \mathbf{d}_i^T \mathbf{b}_u\right)^2 - \left(\sum_{i\in\Omega_u} \frac{2r_{ui}-1}{2f}\mathbf{d}_i^T\right)\mathbf{b}_u + \lambda \left\|\mathbf{b}_u\right\|^2 \tag{6}$$

where Ω_u denotes the private observed ratings for local client u and the constant term containing $\mathbf{d}_i$ is omitted. We can easily verify that only the user's local data can be utilized to update the user's discrete representations so as to achieve the aim of privacy protection. Since the balanced constraint is plugged into the basic discrete loss function in Eq.(5), the aforementioned minimization issue is generally NP-hard, and hence we employ the Discrete Coordinate Descent (DCD) algorithm [34] to update each bit of the user binary vector $\mathbf{b}_u$. Specifically, let b_{uk} denote the k-th bit of the user binary vector $\mathbf{b}_u$ and $\mathbf{b}_{u\bar{k}}$ be the rest binary codes excluding the k-th bit b_{uk} of user u. Without loss of generality, assume $\mathbf{b}_u = [b_{uk}\ \mathbf{b}_{u\bar{k}}]$ and $\mathbf{d}_i = [d_{ik}\ \mathbf{d}_{i\bar{k}}]$, and the quadratic term of Eq.(6) with regard to b_{uk} can be represented as:

$$\frac{1}{4f^2}\sum_{i\in\Omega_u}\left(\mathbf{d}_i^T\mathbf{b}_u\right)^2 = \underbrace{\frac{1}{4f^2}\sum_{i\in\Omega_u}\left(\left(\mathbf{d}_{i\bar{k}}^T\mathbf{b}_{u\bar{k}}\right)^2 + (d_{ik}b_{uk})^2\right)}_{\text{constant}} + \frac{1}{2f^2}b_{uk}\sum_{i\in\Omega_u}\left(\mathbf{d}_{i\bar{k}}^T\mathbf{b}_{u\bar{k}}d_{ik}\right) \tag{7}$$

where the constant part in Eq.(7) can be omitted. Besides, the rest terms in Eq.(6) with regard to b_{uk} can be written as:

$$\begin{aligned}&\left(\sum_{i\in\Omega_u}\frac{2r_{ui}-1}{2f}\mathbf{d}_i^T\right)\mathbf{b}_u - \lambda\left\|\mathbf{b}_u\right\|^2\\ &= \underbrace{\sum_{i\in\Omega_u}\frac{2r_{ui}-1}{2f}\mathbf{d}_{i\bar{k}}^T\mathbf{b}_{u\bar{k}} + b_{uk}^2 + \left(\sum_{k'} b_{uk'}\right)^2}_{\text{constant}}\\ &+ b_{uk}\sum_{i\in\Omega_u}\frac{2r_{ui}-1}{2f}d_{ik} - 2\lambda b_{uk}\sum_{k'} b_{uk'}\end{aligned} \tag{8}$$

Just Accepted

where $b_{uk'}$ represents the k'-th bit of the $\mathbf{b}_{u\bar{k}}$. By bringing Eq.(7) and Eq.(8) into Eq.(6) and omitting the constant parts, therefore, we can derive a series of bit-wise minimization problems:

$$\begin{aligned}\underset{b_{uk}\in\{\pm1\}}{\arg\min}\ \mathcal{L}^{u}_{local} &= \sum_{i\in\Omega_u}\frac{b_{uk}}{f}\left(\frac{1}{2f}\mathbf{d}^T_{i\bar{k}}\mathbf{b}_{u\bar{k}}+\frac{1}{2}-r_{ui}\right)d_{ik}+2\lambda b_{uk}\sum_{k'}b_{uk'}\\ &= -b_{uk}\cdot\left[\sum_{i\in\Omega_u}\frac{1}{f}\left(r_{ui}-\frac{1}{2}-\frac{1}{2f}\mathbf{d}^T_{i\bar{k}}\mathbf{b}_{u\bar{k}}\right)d_{ik}-2\lambda\sum_{k'}b_{uk'}\right]\\ &= -b_{uk}b^*_{uk}\end{aligned} \tag{9}$$

where $b^*_{uk}=\sum_{i\in\Omega_u}\frac{1}{f}\left(r_{ui}-\frac{1}{2}-\frac{1}{2f}\mathbf{d}^T_{i\bar{k}}\mathbf{b}_{u\bar{k}}\right)d_{ik}-2\lambda\sum_{k'}b_{uk'}$. We can clearly check that the optimized b_{uk} should be the *sign* operation of b^*_{uk}, according to the DCD update protocol where it will update b_{uk} with $b_{u\bar{k}}$ fixed. Hence, the user binary vector $\mathbf{b}_u$ can be derived *bit by bit* with the following update rule:

$$b_{uk}=\text{sign}\left(F\left(b^*_{uk},b_{uk}\right)\right) \tag{10}$$

where $F(\cdot)$ is a custom function that $F(b^*_{uk},b_{uk})=b^*_{uk}$ if $b^*_{uk}\neq 0$ and $F(b^*_{uk},b_{uk})=b_{uk}$ otherwise. In other words, we do not update b_{uk} when $b^*_{uk}=0$. In this way, the user binary vector $\mathbf{b}_u$ could be iteratively updated using their own local data until there is no change for each bit.

Secondly, we aim to calculate the gradients towards the item binary vector $\mathbf{d}_i$ via fixing the user binary vector $\mathbf{b}_u$, and then send it to the server for the preparation of global discrete aggregation. Note that different from the problem of updating user binary codes, the client u only interacts with the item i once in its local private data and can not access the feedback about item i from other clients, hence we can update $\mathbf{d}_i$ according to the following expanded formulation:

$$\underset{\mathbf{d}_i\in\{\pm1\}^f}{\arg\min}\ \mathcal{L}^{i}_{local}=\frac{1}{4f^2}\left(\mathbf{b}^T_u\mathbf{d}_i\right)^2-\left(\frac{2r_{ui}-1}{2f}\mathbf{d}^T_i\right)\mathbf{d}_i+\lambda\left\|\mathbf{d}_i\right\|^2 \tag{11}$$

where we only focus on the specific client u in the process of local discrete optimization, and hence the sum operation (*i.e.*, $\sum_u\in\Omega_i$) is prohibited in its local device. Finally, we can derive a series of bit-wise minimization problems on each bit of the item binary vector:

$$\begin{aligned}\underset{d_{ik}\in\{\pm1\}}{\arg\min}\ \mathcal{L}^{i}_{local} &= \frac{d_{ik}}{f}\left(\frac{1}{2f}\mathbf{b}^T_{u\bar{k}}\mathbf{d}_{ik}+\frac{1}{2}-r_{ui}\right)b_{uk}+2\lambda d_{ik}\sum_{k'}d_{ik'}\\ &= -d_{ik}\cdot\left[\frac{1}{f}\left(r_{ui}-\frac{1}{2}-\frac{1}{2f}\mathbf{b}^T_{u\bar{k}}\mathbf{d}_{ik}\right)b_{uk}-2\lambda\sum_{k'}d_{ik'}\right]\\ &= -d_{ik}\cdot\left[\frac{1}{f}\Delta d^u_{ik}-2\lambda\sum_{k'}d_{ik'}\right]\end{aligned} \tag{12}$$

where we define $\Delta d^u_{ik}=\left(r_{ui}-\frac{1}{2}-\frac{1}{2f}\mathbf{b}^T_{u\bar{k}}\mathbf{d}_{i\bar{k}}\right)b_{uk}$ as the gradient of the k-th bit of the item binary vector $\mathbf{d}_i$ from the client u, and then upload it to the central server for aggregation. So far, the update of user binary vectors and the calculation of the gradients towards the item binary vectors have been completed in local discrete optimization module. Next, we will introduce the global discrete aggregation module to update the item binary vectors in the central server.

Just Accepted

3.3.2 (Global) Discrete aggregation. In this part, we will illustrate how to update the item binary matrix $\mathbf{D}$ using the gradients uploaded from the subset of clients $\mathcal{U}_s$ in the central server. Specifically, the loss function in the form of aggregation for the item binary vector $\mathbf{d}_i$ is as follows:

$$\underset{\mathbf{d}_i \in \{\pm 1\}^f}{\arg\min}\, \mathcal{L}^i_{global} = \frac{1}{4f^2}\left(\sum_{u\in\Omega_i} \mathbf{b}_u^T \mathbf{d}_i\right)^2 - \left(\sum_{u\in\Omega_i} \frac{2r_{ui}-1}{2f}\mathbf{d}_i^T\right)\mathbf{d}_i + \lambda \left\|\mathbf{d}_i\right\|^2 \tag{13}$$

where Ω_i denotes the client set who has interacted with item i, which demonstrates that the same item will be associated with multiple clients. It's worth noting that this step involves data aggregation across different clients, so it needs to be executed on the server. Similar to the derivation process of updating the user binary vector, we can get a set of problems involving bit-wise minimization:

$$\begin{aligned}
\underset{d_{ik} \in \{\pm 1\}}{\arg\min}\, \mathcal{L}^i_{global} &= \sum_{u\in\Omega_i} \frac{d_{ik}}{f}\left(\frac{1}{2f}\mathbf{b}_{u\bar{k}}^T \mathbf{d}_{ik} + \frac{1}{2} - r_{ui}\right) b_{uk} + 2\lambda d_{ik} \sum_{k'} d_{ik'} \\
&= -d_{ik} \cdot \left[\sum_{u\in\Omega_i} \frac{1}{f}\left(r_{ui} - \frac{1}{2} - \frac{1}{2f}\mathbf{b}_{u\bar{k}}^T \mathbf{d}_{ik}\right) b_{uk} - 2\lambda \sum_{k'} d_{ik'}\right] \\
&= -d_{ik} \cdot \left[\sum_{u\in\Omega_i} \frac{1}{f}\Delta d^u_{ik} - 2\lambda \sum_{k'} d_{ik'}\right] = -d_{ik} d^*_{ik}
\end{aligned} \tag{14}$$

where Δd^u_{ik} denotes the gradients towards the k-th bit of the item binary vector $\mathbf{d}_i$ uploaded from the client u. It can be observed from Eq.(14) that the update of global item binary vectors can be completely performed by the aggregation of the gradients uploaded from the clients. After aggregating the gradients Δd^u_{ik} from the selected clients $\mathcal{U}_s$, the update of the item binary vector $\mathbf{d}_i$ can be performed *bit by bit* with the following protocol:

$$d_{ik} = \text{sign}\left(F\left(d^*_{ik}, d_{ik}\right)\right) \tag{15}$$

Following aggregation of the gradient data from all clients, the server conducts the sign operation on them, and finally obtains the updated item binary matrix $\mathbf{D}$, which will be distributed to the clients for the next round of optimization until convergence. To offer a holistic view of discrete aggregation and facilitate batch implementation, we define $\Delta\mathbf{D}^u$ as the gradient matrix from the client u where $\Delta\mathbf{D}^u = [\Delta\mathbf{d}^u_1, \cdots, \Delta\mathbf{d}^u_m]$ and $\Delta\mathbf{d}^u_m = [\Delta d^u_{m1}, \cdots, \Delta d^u_{mk}]$. Hence, we can provide the matrix form of the aggregation rule based on the uploaded *gradients* from the clients:

$$\text{Agg}_{grad} : \mathbf{D} = \text{sign}(\frac{1}{f}\sum_{u=1}^{n} \Delta\mathbf{D}^u - 2\lambda\mathbf{D}') \tag{16}$$

where $\mathbf{D}' = [\mathbf{d}'_1, \cdots, \mathbf{d}'_m]$ and $\mathbf{d}'_m = [d'_{m1}, \cdots, d'_{mk}]$. Besides, the element d'_{mk} is defined as $d'_{mk} = \sum_{k'} d_{mk'}$ and $d_{mk'}$ represents the k'-th bit of the rest codes $\mathbf{d}_{m\bar{k}}$ exceeding the d_{mk}. We name this aggregation mechanism Agg_{grad}. Apart from the aggregation of the gradients from clients, we can also directly aggregate the item binary matrix which has been locally updated on the client, and the aggregation mechanism based on the uploaded discrete *parameters* is as follows:

$$\text{Agg}_{para} : \mathbf{D} = \text{sign}(\sum_{u=1}^{n} \mathbf{D}^u) \tag{17}$$

where $\mathbf{D}^u$ denotes the item binary matrix which has been updated locally using the Eq.(11) in the client u, and then is uploaded to the server for aggregation. Similarly, we refer to this aggregation mechanism as Agg_{para}. We

Just Accepted

Algorithm 1: Federated Discrete Optimization Algorithm

Input : Total number of clients n; Total number of items m
The code length f; The trade-off parameter λ; The number of selected clients c
The global training rounds T; The local training epochs E

Output : Global item binary matrix $\mathbf{D}$ at server; local user binary vector $\mathbf{b}_u$ in each client u

1 **Server executes:**
2 **Initialization:** The item binary matrix $\mathbf{D}_t \in \mathbb{R}^{f \times m}$, where $t = 0$;
3 **for** *each round* $t = 1, \cdots, T$ **do**
4 $\quad \mathcal{U}_s \leftarrow$ randomly select a subset of clients with the ratio of $p = c/n$;
5 $\quad$ **for** *each client* $u \in \mathcal{U}_s$ in parallel **do**
6 $\quad\quad \Delta\mathbf{D}_t^u \leftarrow$ ClientUpdate($\mathbf{D}_t$; u; t); `// (Local) Discrete Optimization`
7 $\quad$ **end**
8 $\quad \mathbf{D}_{t+1} \leftarrow$ Eq.(16); `// (Global) Discrete Aggregation`
9 **end**
10 **Client executes:**
11 **Function** ClientUpdate($\mathbf{D}_t$; u; t) :
12 $\quad$ downloading latest $\mathbf{D}_t$ from the central server;
13 $\quad$ **for** *each epoch* $e = 1, \cdots, E$ **do**
14 $\quad\quad$ **for** $k = 1, \cdots, f$ **do** `// Update private user binary vector`
15 $\quad\quad\quad b_{uk} \leftarrow$ Eq.(10);
16 $\quad\quad$ **end**
17 $\quad\quad$ **for** $(i, r_{ui}) \in \Omega_u$ **do**
18 $\quad\quad\quad$ **for** $k = 1, \cdots, f$ **do** `// Computing gradients for item binary matirx`
19 $\quad\quad\quad\quad \Delta d_{ik}^u \leftarrow \left(r_{ui} - \frac{1}{2} - \frac{1}{2f}\mathbf{b}_{u\bar{k}}^T \mathbf{d}_{i\bar{k}}\right) b_{uk}$;
20 $\quad\quad\quad\quad \Delta\mathbf{D}_t[k; i] \leftarrow \Delta d_{ik}^u$;
21 $\quad\quad\quad$ **end**
22 $\quad\quad$ **end**
23 $\quad$ **end**
24 $\quad$ return $\Delta\mathbf{D}_t$;
25 **end**

will examine the two aggregation mechanisms in the experimental part. Note that the storage/communication efficiency and privacy can be improved by exchanging the binary matrix instead of the real-valued one. Next, we will logically present further details concerning these steps between the distributed clients and the coordinated server.

The pseudo-code of the algorithm for Federated Discrete Optimization is presented in Algorithm 1. The input consists of the number of clients and items, n and m respectively, and the training hyper-parameters such as the code length f, global training rounds T and local training epochs E. The target is to output the well-trained global item binary matrix in the central server and the private user binary vector in each client. In the algorithm, the Line 3 to the Line 9 is the loop operated on the server, which sends parameters to clients and collects their gradients for aggregation. The function ClientUpdate() is the operation on local devices where the Line 14 to Line 16 is the procedure for updating private user binary vector and the Line 17 to Line 22 is to calculate the

Just Accepted

gradients towards item binary matrix. Finally, this function returns the gradients (Line 24) to the server for global aggregation. The above training process, *i.e.*, the outer loop (Line 3 to Line 9) will repeat until its convergence, *e.g.*, the achievement of the preset training rounds or predefined thresholds.

Through the analysis of the above algorithm, we can conclude that its total time complexity is $O(T \times n \times E \times (f^2 \times |\Omega_u| + m \times f^2))$. Note that, the optimization between clients can be easily performed in parallel under federated settings, so the number of clients n here can be omitted. Therefore, the final time complexity of our proposed algorithm can be expressed as $O(T \times E \times f^2 \times (|\Omega_u| + m))$. Because T, E and f are usually small hyper-parameters in our work and fixed during the training stage, we can see that the training time complexity (*a.k.a* the encoding time for the binary codes) is linear with the number of rated items per client (i.e., $|\Omega_u|$) and the number of items (i.e., m). In summary, training our discrete optimization algorithm is efficient under federated scenarios.

3.3.3 Cold-start Scenario. Necessarily, the cold-start issue, in which few or even no prior interactions (*e.g.*, ratings or clicks) are known for certain users or items, is an inherited challenging problem in traditional collaborative filtering paradigms. Similarly, it is also required to account for the existence of new clients (*a.k.a.* cold-start clients) or new items (*a.k.a.* cold-start items) in federated recommendation scenarios. Hence, in this part, we will introduce how to deal with the situation of new clients (items) in our federated discrete optimization algorithm.

Apparently, it is expensive to train the whole algorithm from scratch to obtain binary codes for these cold-start samples, when new users (clients) or items arrive. Therefore, a feasible countermeasure is to learn temporary binary codes for new coming samples online and then retrain the entire data offline when possible. Note that we focus on the cold-start situation which allows for the existence of a few interactions with users or items. As for the scenario where there is entirely no interactions, it must necessitate the assistance of some side information and warm-up techniques [4, 54], which is beyond the research scope of our work.

Firstly, we will explore the case when a new client arrives. Without loss of generality, let $\{r_{ui} | i \in \Omega_u\}$ be the set of local observed private interactions for existing items in the new client u and its binary codes is $\mathbf{b}_u$. It is worth mentioning that it's unnecessary to impose the global balanced constraint as described in Eq.(5) for a single user. Hence, we should only concentrate on minimizing the squared error loss in each client in the following way:

$$\underset{\mathbf{b}_u \in \{\pm 1\}}{\arg\min}\, \mathcal{L}^u_{cold} = \sum_{i \in \Omega_u} (r_{ui} - \mathrm{sim}(\mathbf{b}_u, \mathbf{d}_i))^2 \tag{18}$$

where $\mathrm{sim}(\mathbf{b}_u, \mathbf{d}_i) = \frac{1}{2} + \frac{1}{2f}\mathbf{b}_u^T \mathbf{d}_i$. We can easily observe that Eq.(18) is a particular form of Eq.(6) with removing the regularization term. So we can quickly learn the k-th bit b_{uk} of $\mathbf{b}_u$ by the DCD optimization protocol $b_{uk} = \mathrm{sign}\left(F\left(b^*_{uk}, b_{uk}\right)\right)$, and b^*_{uk} is derived from the following formulation:

$$b^*_{uk} = \sum_{i \in \Omega_u} \frac{1}{f}\left(r_{ui} - \frac{1}{2} - \frac{1}{2f}\mathbf{d}^T_{i\bar{k}}\mathbf{b}_{u\bar{k}}\right) d_{ik} \tag{19}$$

where $\mathbf{b}_{u\bar{k}}$ denotes the rest codes of the user binary vectors $\mathbf{b}_u$ excluding the k-th bit b_{uk}. We can see that for the arrival of new users, we can update the user binary vectors locally via the discrete optimization module in their terminal devices without retraining a large number of existing user binary vectors.

Secondly, we will explore the case when a new item arrives. Similar to the procedure of cold-start clients, the global balanced constraint specified in Eq.(13) is ignored for a new coming item, and the following loss function in the server can be modified as:

$$\underset{\mathbf{d}_i \in \{\pm 1\}}{\arg\min}\, \mathcal{L}^i_{cold} = \sum_{u \in \Omega_i} (r_{ui} - \mathrm{sim}(\mathbf{b}_u, \mathbf{d}_i))^2 \tag{20}$$

Just Accepted

where Ω_i denotes the set of global observed interactions for existing clients on target item i, which indicates that the new coming item will be interacted across multiple clients. Hence, we will perform the gradient aggregation process by the discrete aggregation module in the server and conduct the gradient calculation procedure via the discrete optimization module on each client. By expanding and simplifying the above Eq.(20), we can acquire the aggregation form on the server as follows:

$$\begin{aligned}\arg\min_{\mathbf{d}_i \in \{\pm 1\}} \mathcal{L}^i_{cold} &= \sum_{u \in \Omega_i} \frac{d_{ik}}{f} \left(\frac{1}{2f} \mathbf{b}^T_{u\bar{k}} \mathbf{d}_{ik} + \frac{1}{2} - r_{ui} \right) b_{uk} \\ &= -d_{ik} \cdot \sum_{u \in \Omega_i} \frac{1}{f} \left(r_{ui} - \frac{1}{2} - \frac{1}{2f} \mathbf{b}^T_{u\bar{k}} \mathbf{d}_{ik} \right) b_{uk} \\ &= -d_{ik} \cdot \left[\sum_{u \in \Omega_i} \frac{1}{f} \Delta d^u_{ik} \right] = -d_{ik} d^*_{ik}\end{aligned} \tag{21}$$

where d^*_{ik} denotes the gradient aggregation process in the server and Δd^u_{ik} denotes the gradient calculation procedure towards the k-th bit of the item binary vector $\mathbf{d}_i$ uploaded from the client u. By performing the gradient calculation to the new items on each client and uploading them to the server for aggregation, the near-optimal update of the binary codes towards those cold-start items is achieved on the premise of protecting users' local privacy information.

3.4 Discussion

In this section, we theoretically discuss the superiority of our proposed LightFR from three beyond-accuracy perspectives: storage/communication efficiency, inference efficiency, and privacy preserving.

3.4.1 Storage/Communication Efficiency. In federated settings, the storage overhead in local clients and communication consumption between the server and clients are always an unneglectable issue [5]. As for the storage overhead in each client, it is mainly composed of the private user embedding and the global item embedding matrix. Considering the traditional Euclidean space which is widely applied in many FRS methods and a 64-bit floating point precision, the simple formulation of storage overhead estimation is exactly: $((1 + m) \times f \times 64)/8$ bytes, which increases linearly with the ever-increasing number of items. For example, we assume the number of items m is 10 million and the dimension f is 128, and it will take over 10.2 GB of storage space, which is hard to be deployed into general devices with limited memory. Apparently, by storing the binary representations of the user and items in Hamming space, the memory consumption will not exceed 1.3 GB, which is acceptable for common mobile devices. As for the communication overhead which is exchanged between the server and users, it primarily depends on the number of items m to recommend. The requirement to transmit huge parameters (*i.e.*, the item embedding matrix) between the FL server and users over several communication rounds imposes strict limitations for both the server and clients. In Euclidean space, the formula used to estimate communication overhead is $(m \times f \times 64)/8$ while the approximated formula in Hamming space is only $m \times f$. Similar to the calculation process of storage overhead, the communication consumption in Euclidean space is 8 times than that in Hamming space. To be more precise, the summarized formulas in Table 6 and experimental results in Fig. 3 comparing with other classical federated recommender systems in storage and communication aspects clearly show the efficiency of our method. As a result, by transmitting the binary item matrix produced by our LightFR model, the payload of data exchanged can be considerably reduced, and thus allowing user devices to utilize lower bandwidth resources.

3.4.2 Inference Efficiency. Unlike many existing FRS approaches, which estimate the correlation scores via inner product or cosine similarity in a continuous Euclidean space between user and item representations, our proposed

Just Accepted

LightFR model eventually in each client generates their own private binary vectors and acquires the latest global item binary matrix which is downloaded from the server, and then efficiently perform the similarity search in Hamming space using the binary ones at their local devices. Generally, given f-dimensional representation of m items in Euclidean space and the results of top-k recommendations will incur an inference inefficiency with the time complexity of $O(mf + k \log k)$, which scales approximately linearly with the number of items. Not surprisingly, our proposed method adopts bit operations in a proper Hamming space, so the time complexity of linear search is greatly decreased and even constant time scan is possible [42]. Besides, the Hamming similarity has a highly efficient hardware-level implementation, allowing us to locate relevant items in time that is independent to the number of items [33]. To be specific, if items are represented by the f-dimensional double-precision float embeddings, and the inner product of them in Euclidean space requires f times of floating-point multiplications, while the similarity calculation with f-dimensional binary vectors in Hamming space needs only one XOR operation and one time of sum operation. The experimental results quantified in the third panel in Fig. 3 verify the efficiency of our method in inference time compared with other federated recommender systems. In a nutshell, the similarity search in Hamming space is more significantly efficient than that in Euclidean space, and it is more urgent and suitable for resource-constrained clients in the FRS scenarios.

3.4.3 Privacy Preserving. It is critical to preserve and enhance privacy in FRS scenarios, since previous work has proved that the original rating data is likely to be leaked when transmitting the gradients or model parameters in real-valued forms [29]. The proposed LightFR requires the exchange of binary representations in discrete Hamming space rather than the real-valued ones in continuous Euclidean space between the server and clients, which is the key property that brings benefits to FRS in terms of privacy enhancement. As FedMF [6] states, given the real-valued gradients of a user u towards the item i at iterations t and $t+1$ uploaded in two continuous steps *i.e.*, g_i^t and g_i^{t+1}, and the corresponding real-valued user latent embedding $\mathbf{p}_u$, it can infer the user's rating information r_{ui} according to the following formulations:

$$\frac{g_{ik}^t}{p_{uk}^t} - \frac{g_{ik}^{t+1}}{p_{uk}^t + \frac{\alpha_k}{p_{uk}^t}} = \frac{p_{uk}^t}{g_{ik}^t}\beta_i + \frac{g_{ik}^t}{p_{uk}^t}\gamma_i \tag{22}$$

$$r_{ui} = \frac{g_{ik}^t}{p_{uk}^t} + \sum_{d=1}^{f} p_{ud}^t q_{id}^t \tag{23}$$

where g_{ik}^t denotes the k-th dimension of the uploaded gradient g_i^t about the item i at iteration t, and p_{ud}^t and q_{id}^t represent the d-th dimension of the user embedding $\mathbf{p}_u$ and item embedding $\mathbf{q}_i$, respectively. Besides, we can treat β_i, γ_i and α_k as the constants. Hence, the premise of inferring the rating information r_{ui}, *i.e.*, the Eq.(23), is to solve the variable p_{uk}^t. We can easily confirm that there must be one real-valued scalar of p_{uk}^t in Euclidean space that satisfies the Eq.(22), and it can be solved by some iterative methods to compute a numeric solution, *e.g.*, gradient descent optimization methods. Notably, our proposed method assumes that the user's latent embedding p_{uk}^t is discrete in Hamming space, which violates the premise of the continuous real-valued solutions in Eq.(23). Besides, the non-differentiable and discontinuous operation *sign*, which is an irreversible process, makes it tough or even impossible to solve the Eq.(22). Thus, the users' private rating data on their local devices can not be easily inferred.

In addition, we provide a theoretical analysis to demonstrate that our proposed LightFR model is able to enhance users' privacy. Assuming d_{ik}^{t+1} is the k-th bit of item i to be uploaded from user u to the server at time

Just Accepted

$t+1$, according to Eq.(15), we have

$$d_{ik}^{t+1} = sign(F((d_{ik}^{*})^{t}, d_{ik}^{t}))$$
$$(d_{ik}^{*})^{t} = \frac{1}{f}(r_{ui} - \frac{1}{2} - \frac{1}{2f}(d_{i\bar{k}}^{T})^{t} b_{u\bar{k}}^{t})b_{uk}^{t} \tag{24}$$

where $F(x,y)$is a function that $F(x,y) = x$ if $x \neq 0$ and $F(x,y) = y$ otherwise. Thus, we will discuss the effectiveness of preserving users' privacy according to the following three cases.

(1) if $d_{ik}^{t+1} \neq d_{ik}^{t}$ and $(d_{ik}^{*})^{t} \neq 0$, we can determine whether $(d_{ik}^{*})^{t}$ is positive or negative by d_{ik}^{t+1}. When we get sign$((d_{ik}^{*})^{t})$ and $(d_{i\bar{k}}^{T})^{t}$, we have no idea to determine r_{ui} from Eq.(24) since $\mathbf{b}_{u}^{t}$ is unkonwn.
(2) if $d_{ik}^{t+1} = d_{ik}^{t}$ and $(d_{ik}^{*})^{t} \neq 0$, similar with (1), we can only get sign$((d_{ik}^{*})^{t})$. So the value of r_{ui} is also undetermined.
(3) if $d_{ik}^{t+1} = d_{ik}^{t}$ and $(d_{ik}^{*})^{t} = 0$, we have

$$(d_{ik}^{*})^{t} = \frac{1}{f}(r_{ui} - \frac{1}{2} - \frac{1}{2f}(d_{i\bar{k}}^{T})^{t} b_{u\bar{k}}^{t})b_{uk}^{t} \tag{25}$$
$$= \frac{1}{f}(r_{ui} - \frac{1}{2} - \frac{1}{2f}((\mathbf{d}_{i}^{T})^{t}\mathbf{b}_{u}^{t})b_{uk}^{t} + \frac{1}{2f^{2}}d_{ik}^{t} = 0 \tag{26}$$

Following that, we expand the Eq.(25) and Eq.(26) by the dimension f, and we can derive r_{ui} from the following two sets of equations:

$$\begin{cases} \frac{1}{f}(r_{ui} - \frac{1}{2} - \frac{1}{2f}((\mathbf{d}_{i}^{T})^{t}\mathbf{b}_{u}^{t})b_{u1}^{t} + \frac{1}{2f^{2}}d_{i1}^{t} = 0, \\ \vdots \\ \frac{1}{f}(r_{ui} - \frac{1}{2} - \frac{1}{2f}((\mathbf{d}_{i}^{T})^{t}\mathbf{b}_{u}^{t})b_{uk}^{t} + \frac{1}{2f^{2}}d_{ik}^{t} = 0, \\ \vdots \\ \frac{1}{f}(r_{ui} - \frac{1}{2} - \frac{1}{2f}((\mathbf{d}_{i}^{T})^{t}\mathbf{b}_{u}^{t})b_{uf}^{t} + \frac{1}{2f^{2}}d_{if}^{t} = 0 \end{cases} \tag{27}$$

$$\begin{cases} \frac{1}{f}(r_{ui} - \frac{1}{2} - \frac{1}{2f}((\mathbf{d}_{i}^{T})^{t+1}\mathbf{b}_{u}^{t+1})b_{u1}^{t+1} + \frac{1}{2f^{2}}d_{i1}^{t+1} = 0, \\ \vdots \\ \frac{1}{f}(r_{ui} - \frac{1}{2} - \frac{1}{2f}((\mathbf{d}_{i}^{T})^{t+1}\mathbf{b}_{u}^{t+1})b_{uk}^{t+1} + \frac{1}{2f^{2}}d_{ik}^{t+1} = 0, \\ \vdots \\ \frac{1}{f}(r_{ui} - \frac{1}{2} - \frac{1}{2f}((\mathbf{d}_{i}^{T})^{t+1}\mathbf{b}_{u}^{t+1})b_{uf}^{t+1} + \frac{1}{2f^{2}}d_{if}^{t+1} = 0 \end{cases} \tag{28}$$

It is worth noting that the premise of Eq.(27) and Eq.(28) hold on is that $(d_{ik}^{*})^{t} = 0, (d_{ik}^{*})^{t+1} = 0, \forall k \in \{1, 2, \cdots, f\}$. Such a premise illustrates the invalidity of the classical discrete coordinate descend optimization algorithm [34]. Thus, when $d_{ik}^{t+1} = d_{ik}^{t}$ and $(d_{ik}^{*})^{t} = 0$, the server cannot derive the value of r_{ui}. In summary, we cannot infer the sensitive rating data of client u based on the uploaded information.

Therefore, to some extent, our LightFR framework can theoretically prevent malicious attackers from inferring the sensitive rating information of local clients, thereby achieving the purpose of enhancing the capacity of preserving privacy.

4 EXPERIMENTS

In this section, we first introduce our experimental settings in detail, and then present the extensive experimental results and in-depth analysis that validate the effectiveness of our proposed LightFR framework from multiple aspects.

Just Accepted

4.1 Experimental Settings

First, we introduce the details of the adopted datasets in our work, and then elaborate on the evaluation metrics utilized to verify the effectiveness and efficiency of our proposed model. Besides, we list several comparison recommendation methods based on centralized storage and some privacy-preserving ones based on federated learning, and finally, we detail some other implementation details to ensure reproducibility and fair comparison of the experiments.

4.1.1 Datasets. For a comprehensive comparison, we adopt four commonly used public datasets with various scales to conduct experimental analyses, which are MovieLens-1M[5], Filmtrust[6], Douban-Movie[7] and Ciao[8]. Specifically, MovieLens-1M dataset originally contains approximately 1 million ratings of 3,952 movies from 6,040 users, and the data density is 4.19% and the average number of user ratings is 166. Filmtrust dataset is crawled from online rating website FilmTrust, which originally contains about 35 thousand ratings, from 1,508 users on 2,071 films and its data density is 1.14% and each user has 24 ratings in average, which has the least interactions among them. Douban-Movie dataset is built from online sharing website Douban which provides user rating, review and recommendation services for movies, books and music, and it has nearly 894 thousand ratings from 2,964 users of 39,695 movies and its data density is only 0.76% and the average number of ratings per user is 302 which is the most interactions among them. Ciao dataset is crawled from the popular product review sites Ciao in the month of May, 2011, and it contains about 282 thousand ratings on 105,096 movies from 7,375 users, which is the most sparse dataset with only the density ratio of 0.04% and the average number of user ratings is merely 38. The rating scale of MovieLens-1M, Douban-Movie and Ciao ranges from 1 to 5 in 1 increment, while Filmtrust ranges from 0.5 to 4 in 0.5 increments. For each user, we first sort the positive samples by timestamp in chronological order and then separate them into three chunks: 80% as the training set, 10% as the validation set and 10% as the test set. For the validation and testing stage in federated settings, we randomly sample a fixed number of items as negatives for each positive item in each local client. The detailed statistics of these datasets are summarized in Table 3, where # Average means the average number of user ratings, which reflects the data density of clients in each dataset. Importantly, experiments conducted on the above four datasets with varying scales and sparsity can comprehensively reflect the performance of the model. In federated settings, each user is regarded as a local client, and the user's data is locally stored on the device.

Table 3. Statistics of the utilized datasets in evaluation.

Datasets	# Users	# Items	# Ratings	# Average	Rating Range	Data Density
MovieLens-1M [11]	6,040	3,952	1,000,209	166	$[1,2,\cdots,5]$	4.19%
Filmtrust [9]	1,508	2,071	35,497	24	$[0.5,1,\cdots,4]$	1.14%
Douban-Movie [25]	2,964	39,695	894,888	302	$[1,2,\cdots,5]$	0.76%
Ciao [36]	7,375	105,096	282,619	38	$[1,2,\cdots,5]$	0.04%

4.1.2 Evaluation Metrics. To evaluate the performance and verify the effectiveness of our model, we utilize two commonly used evaluation metrics, *i.e.*, Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG), and both of them are widely adopted for item ranking task. The above two metrics are usually truncated at

[5]https://grouplens.org/datasets/movielens/1M/
[6]https://guoguibing.github.io/librec/datasets.html
[7]https://www.cse.cuhk.edu.hk/irwin.king.new/pub/data/douban
[8]https://www.cse.msu.edu/ tangjili/datasetcode/truststudy.htm

Just Accepted

a particular rank level (*e.g.* the first k ranked items) to emphasize the importance of the first retrieved items. Specifically, HR@k, recall at a cutoff k, is used to count the number of occurrences of the testing item in the predicted ranked item set, and NDCG@k, which truncates the ranked list at k, measures the ranking quality which assigns higher scores to hit at the top position ranks while the positive items at bottom positions of the ranking list contribute less to the final result. Intuitively, the HR metric measures whether the test item is present on the top-k ranked list or not, and the NDCG metric measures the ranking quality which comprehensively considers both the positions of ratings and the ranking precision.

4.1.3 Comparison Models. We adopt two kinds of benchmarks for comprehensive comparisons, *i.e.,* classical Matrix Factorization (MF) based methods and recent federated MF approaches. The classical MF-based methods are based on centralized storage settings, which are not capable of protecting user privacy. Most existing federated MF methods essentially perform similarity search via inner product in Euclidean space, which are not able to efficiently handle the rating data in a privacy enhancement manner.

Classical MF-based models

- PMF [32]: a canonical probabilistic latent factor model which factorizes both users and items into a common subspace, in which the similarity between users and items can be measured by inner product in Euclidean space.
- SVD++ [16]: another latent factor model which explores the biases of users and items, and incorporates the user implicit feedback into PMF framework.
- DDL [51]: a hashing-based MF model which adds deep learning technique into the discrete collaborative filtering framework by exploiting rating and item content data.
- NCF [12]: the state-of-the-art deep learning based MF method that combines generalized matrix factorization and multi-layer perceptron (MLP) to model user-item interactions.

Federated MF models

- FCF [2]: a pioneering privacy-preserving federated collaborative filtering method which formulates the updating rules of collaborative filtering to suit the FL settings.
- FedMF [6]: a privacy-enhanced matrix factorization approach based on secure homomorphic encryption under federated settings.
- FedRec [21]: it is another privacy-enhanced model with non-cryptographic techniques, in which some unrated items are randomly sampled and assigned with some virtual ratings.
- MetaMF [22]: a novel federated MF model that deploys a big meta network into the server while deploying a small model into the device to perform rating prediction task.
- PrivRec [39]: a fast-adapting federated recommender model that adopts a meta-learning strategy to enable fast convergence on local devices in a privacy-preserving way[9].

4.1.4 Implementation Details. In our experiments, the dimension of user and item embedding f is set to 32 for all the real-valued MF methods and 64 for the hashing-based models. The significance of setting the dimensions in this way is to achieve the recommendation performance comparable to the real-valued models as much as possible on the premise of saving resources compared with those dense models. Therefore, the threshold of our LightFR being effective lies in the length of binary codes. For the centralized MF-based models, we set the training epoch E to 50 and adopt the early stopping technique, that is, if the performance of five consecutive epochs is not improved, it will stop running, and set the batch-size B to 512. For the federated MF-based comparison methods, we set the global training rounds T to 50, the local epoch E to 1. Besides, we set the ratio of selected clients p to 0.6. Moreover, we specify the length of public key $l = 1024$ used in FedMF [6], the sampling parameter $\rho = 3$ in

[9]For fair comparison, we instantiate PrivRec equipped with the first-order meta learning and differential privacy components, which is built on the MF backbone as our proposed model does.

Just Accepted

FedRec [21], the length of hidden layers $L = 2$, the number of hidden units $h = 8$ and the size of low-dimensional item embeddings $s = 8$ in MetaMF [22], and the noise scale $z = 1$, the clipping bound $S = 50$ in PrivRec [39]. The detailed parameter settings are listed in Table 4. All the hyper parameters are searched and tuned according to the performance on the validation dataset.

Table 4. The detailed parameter settings on comparison models and the proposed LightFR.

Methods	**FCF (Common parameters)**	**FedMF**	**FedRec**	**MetaMF**	**PrivRec**	**LightFR**
Parameters	$p = 0.6$, $E = 1$, $T = 50$	$l = 1024$	$\rho = 3$	$L = 2$, $h = 8$, $s = 8$	$z = 1$, $s = 50$	$\lambda = 0.6$

4.2 Experimental Results

In this section, we present the extensive experimental results of LightFR *w.r.t.* state-of-the-art centralized and federated baselines. In detail, firstly we will compare our approach with the other nine recent MF-based methods in overall recommendation performance, and then the ablation study is provided to analyze the contribution of our proposed method. Finally, the sensitivity analysis is further given to explore the effects of different hyperparameters on our model.

4.2.1 Overall Performance. We conduct the overall comparison of different models including classical MF-based methods and federated MF models, where the first four methods are centralized manner and the last five methods are in federated settings. Table 5 summarizes the experimental results of HR@10 and NDCG@10 on the four widely used datasets. Firstly, we deliver the analysis of the classical MF-based models under the centralized storage paradigm. From such results, we have the following observations.

- Among the centralized classical MF models, the fact that SVD++ model outperforms PMF shows the advantage of incorporating implicit data into the explicit feedback on top of the basic MF framework. Besides, the reason why DDL is nearly comparable to PMF is that the deep neural model with item content data is introduced on the basis of discrete collaborative filtering algorithm, which is beneficial to extract efficient binary representations.
- In addition, thanks to the effective non-linear feature transformation and high-order feature extraction capability of multi-layer perceptron (MLP), NCF surpasses the other two canonical MF models. Note that such improvement increases along with the increasing of data scale, where the datasets are arranged in the order of increasing data scale, which demonstrates that the deep neural models represented by MLP require a big quantity of data to work, which is resource-intensive to run on the client in the federated learning environment.

Next, we focus on the experimental analysis of the federated MF baselines and our proposed model. From the experimental results in Table 5, the following noteworthy findings are drawn.

- The five federated MF baselines (such as FCF, FedMF, FedRec, MetaMF and PrivRec), marginally impair the performance compared with the centralized MF models (such as PMF, SVD++, DDL and NCF). More specifically, the recommendation performance of the FedMF and FedRec methods is often inferior to that of the centralized MF models, and the performance of the FCF, MetaMF and PrivRec models are comparable to that of the centralized ones. This is mainly because, in the federated settings, in order to achieve privacy protection and prevent the original data from leaving the local clients, the global parameters are updated by aggregating the gradient information or model parameters of the clients, and thus the gradient noises and model losses are inevitably introduced during the aggregation process.

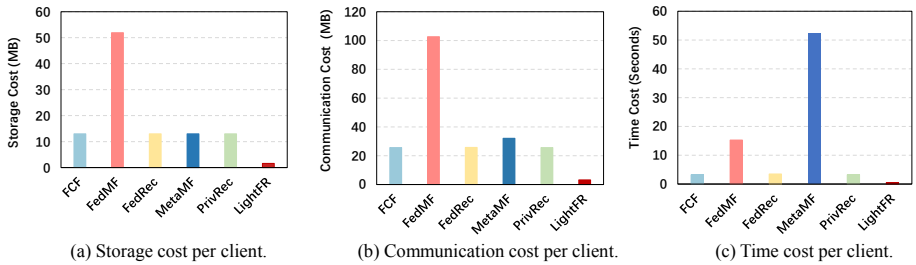
Just Accepted

Table 5. Comparison results of LightFR and the baselines on the four datasets. The best federated learning results are in bold and the second best federated method is with asterisks, and the best results for centralized learning methods are underlined.

Methods	MovieLens-1M		Filmtrust		Douban-Movie		Ciao	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
PMF[32]	0.5124	0.2768	0.8704	0.6610	0.3011	0.1678	0.4636	0.2434
SVD++[16]	0.5291	0.2826	0.8793	0.6777	0.3118	0.1886	0.4692	0.2463
DDL[51]	0.5101	0.2743	0.8630	0.6579	0.2967	0.1671	0.4566	0.2429
NCF[12]	0.5342	0.2901	0.8827	0.6907	0.3223	0.2097	0.4732	0.2513
FCF[2]	0.4945	0.2625	0.8543	0.6376	0.2921	0.1615	0.4492	0.2304
FedMF[6]	0.4836	0.2534	0.8601	0.6563	0.2786	0.1443	0.4461	0.2272
FedRec[21]	0.4893	0.2612	0.8503	0.6304	0.2832	0.1593	0.4474	0.2301
MetaMF[22]	0.4994*	0.2691*	0.8566	0.6389	**0.2937**	**0.1669**	0.4503	0.2402
PrivRec[39]	0.4989	0.2688	0.8605*	0.6563*	0.2931	0.1666	0.4534*	0.2411*
LightFR	**0.5014**	**0.2709**	**0.8615**	**0.6565**	0.2934*	0.1665*	**0.4556**	**0.2413**

- Considering that the performance of FCF method is significantly better than that of the FedMF and FedRec methods. There are two reasons: on the one hand, the loss function of FCF is modeled based on implicit feedback data, whereas the FedMF and FedRec directly model the explicit feedback data. On the other hand, FedMF can be seen as introducing a homomorphic encryption mechanism on the basis of FCF, which often results in a small performance drop but strengthens the privacy protection ability of the FRS framework, while FedRec can be seen as introducing a hybrid filling strategy based on FCF, which means that it well address the privacy issue but introduce some noise to the raw data.
- Among the five federated MF baselines, PrivRec model is superior to other baselines in most cases since it introduces a first-order meta-learning method that enables fast convergence and few communication rounds with only a few data points in local devices, which is particularly evident in Filmtrust and Ciao datasets. Besides, the performance of MetaMF is optimal on Douban-Movie dataset, since it introduces a meta network on top of collaborative filtering methods to capture the collaborative information, and its performance improves gradually with the rise of data volume, which shows that the sufficient training data is the cornerstone of high performance in deep neural models but it is not realistic on the local clients in federated scenario. It should be noted that the performance of MetaMF in Filmtrust dataset is worse than that of FedMF, which could be owing to the over-fitting issue caused by deep neural models in the case of little amounts of training data. Although these methods can achieve comparable performance, the transfer of the original model parameters between the server and clients may make it subject to privacy leakage attacks.
- Our proposed model LightFR outperforms the federated baselines in most cases. Specifically, LightFR is superior to the five federated MF baselines (*i.e.*, FCF, FedMF, FedRec, MetaMF and PrivRec) in terms of every metric on MovieLens-1M, Filmtrust and Ciao datasets, and the performance on Douban-Movie dataset is comparable to MetaMF. Moreover, the performance of our method is comparable to that of the centralized MF models such as PMF, and our model can be strengthened when more complex modeling techniques are introduced (such as SVD++ model with implicit feedback information and NCF model with high-order extracted features). Although our method is slightly worse than MetaMF method on Douban-Movie dataset,

Just Accepted



(a) Storage cost per client. (b) Communication cost per client. (c) Time cost per client.

Fig. 3. Comparisons with baselines in terms of storage cost, communication cost and inference time cost on each local client.

our method can achieve the purpose of less inference time, less memory occupation and fewer bandwidth resources under the premise of considerable performance, which can be easily migrated and deployed to mobile terminal devices under the federated settings, while MetaMF still performs nearest neighbor search via inner products with the real-valued embeddings in Euclidean space, which will result in significant storage consumption and calculation overhead.

It's worth noting that the recommendation performance of our proposed LightFR model can be on par with that of the real-valued models in most cases. The intriguing experimental results can be supported by the dimension of the user (item) latent vectors f. As mentioned above in the experimental settings, we set the dimension of the latent embeddings $f = 32$ in real-valued MF models and $f = 64$ for the binary MF models. Note that, our main motivation is to explore the lightweight and privacy enhancement mechanisms in federated recommendation scenario, so that the resource-constrained clients can occupy less memory and communication overheads, and shorten the inference time under the strict data privacy protections. Therefore, the purpose of this setting is to narrow the gap in recommendation accuracy between the binary codes and that of the real-valued ones on the premise of saving resources. In order to make a fair comparison, we still maintain the same experimental settings to measure the storage, communication overheads and inference time between the real-valued models and the binary models in the next part. Through extensive experiments, we show that our binary model can greatly reduce the storage, communication overheads and inference time while keeping the recommendation performance comparable. In future work, we will consider further improving the accuracy of our binary model when it uses the same code length as the real-valued models, which is beyond the scope of this current work.

In this part, we compare our model with five federated MF baselines on several beyond-accuracy metrics, *i.e.*, storage (memory) cost, communication cost and inference time cost on the local client, so as to verify the comprehensive performance of our method. Specifically, we take the large-scale Ciao dataset as an example, and calculate the storage overhead, communication consumption which includes uploads and downloads, and test the inference time cost on the client side. As for the calculation of memory and communication overheads, we identify and set the default parameters as follows: the number of items in Ciao dataset $m = 105,096$, the average number of items interacted by user in Ciao dataset $\mathcal{I}_u = 38$. Table 6 summarizes the formula list of the comparison methods (FCF, FedMF, FedRec, MetaMF and PrivRec) and our LightFR model to estimate the model space complexity, storage and communication overheads. And the first two panels of the Fig. 3 show the experimental results in terms of storage and communication costs on the client side. When it comes to the cost of inference time, we perform the similarity search on a local client with 16GB of RAM and 2.30GHz 8-core processor. The third panel of Fig. 3 demonstrates the inference time cost of the five federated MF baselines and our proposed method.

From the results of Fig. 3, we can draw the following conclusions. First, our LightFR consistently outperforms the state-of-the-art federated MF baselines in terms of memory, communication and inference efficiency. More

Just Accepted

Table 6. Summary of formulas for estimating the space complexity, storage and communication cost.

Methods	Space Complexity	Storage Cost (Bytes)	Communication Cost (Bytes)
FCF[2]	$O(\lvert\mathcal{I}_u\rvert + m)$	$[(1+m)\times f \times 64]/8$	$[(\lvert\mathcal{I}_u\rvert + m)\times f \times 64]/8$
FedMF[6]	$O(\lvert\mathcal{I}_u\rvert + m)$	$[(1+m)\times f \times 1024]/8$	$[(\lvert\mathcal{I}_u\rvert + m)\times f \times 1024]/8$
FedRec[21]	$O(\lvert\mathcal{I}_u\rvert(1+\rho) + m)$	$[(1+m)\times f \times 64]/8$	$\{[\lvert\mathcal{I}_u\rvert(1+\rho)+m]\times f \times 64\}/8$
MetaMF[22]	$O(m + L\cdot h)$	$[(m+L\cdot h)\times f \times 64]/8$	$[(m\times f + 2\cdot L\cdot h + f\times s + s\times m)\times 64]/8$
PrivRec[39]	$O(\lvert\mathcal{I}_u\rvert + m)$	$[(1+m)\times f \times 64]/8$	$[(\lvert\mathcal{I}_u\rvert + m)\times f \times 64]/8$
LightFR	$O(\lvert\mathcal{I}_u\rvert + m)$	$(1+m)\times f$	$(\lvert\mathcal{I}_u\rvert + m)\times f$

specifically, as for storage efficiency, the occupancy of LightFR is about 3.1% of that in FedMF method, about 8.1% of that in MetaMF model, and exactly 8% of that in FCF, FedRec and PrivRec models in Fig. 3 (a). This is mainly because FedMF needs an encryption process, which expands the dimension of the original vector from 32 to 256; MetaMF needs to store the additional parameters of private Rating Prediction (RP) module, in addition to the item embeddings; And yet, FCF, FedRec, PrivRec and LightFR just need to retain the item embedding matrix, whereas the first three methods require the real-valued forms and our method only requires the binary ones. When it comes to communication efficiency in Fig. 3 (b), the advantage of LightFR is similar to the tread in storage efficiency. The LightFR only requires to transmit the lightweight binary item embeddings between the server and clients, while FCF needs to transmit the encrypted item embeddings which are larger than the original ones, and FCF, FedRec, MetaMF and PrivRec require to transfer the real-valued item embeddings or model parameters. In terms of search efficiency, LightFR incurs more significant speedup than the federated MF baselines, which is about 100 times faster than MetaMF, around 30 times faster than FedMF, and 7 times faster than FCF, FedRec and PrivRec models in Fig. 3 (c). Since MetaMF requires a forward propagation process to generate the predicted ratings for target user on some items, which is the main reason for its slow inference speed. Besides, FedMF must execute the further decryption process for the encrypted item embedding matrix before performing the similar search for all the existing items on the local client. Moreover, our LightFR is faster than FCF, FedRec and PrivRec, since the former performs the XOR operation employing the binary codes in Hamming space, whereas the latter two approaches utilize the real-valued embeddings to execute the inner product operations in Euclidean space. As a result, our LightFR model outperforms the other state-of-the-art federated baselines in terms of memory cost, communication overhead and inference time, while leads to negligible accuracy degradation.

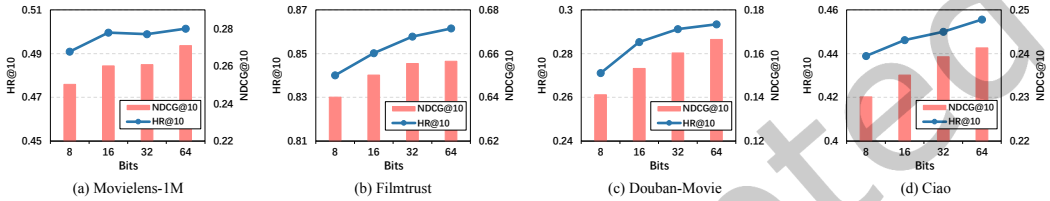
4.2.2 Ablation Study. To better understand the contribution of our proposed federated discrete optimization algorithm, we evaluate the performance gain of our method over several variants in Table 7. We denote the full model as LightFR which performs fine-grained gradient aggregation process in federated discrete aggregation module. **LightFR**$_{para}$ represents the direct discrete parameter aggregation mechanism. **LightFR**$_{init}$ obtains binary codes by directly conducting median quantization on real-valued features learned by MF without the collaborative optimization procedure between the server and clients. **Random** means that the binary codes of the users and items are randomly generated at the clients and the similarity search is conducted in the Hamming space.

From the experimental results, we can draw some important conclusions. Firstly, the fact that our method is clearly superior to the **Random** method, which confirms its supremacy of our proposed discrete optimization algorithm in the federated settings. Besides, we find that the method **LightFR**$_{init}$ of discretizing the latent

Just Accepted

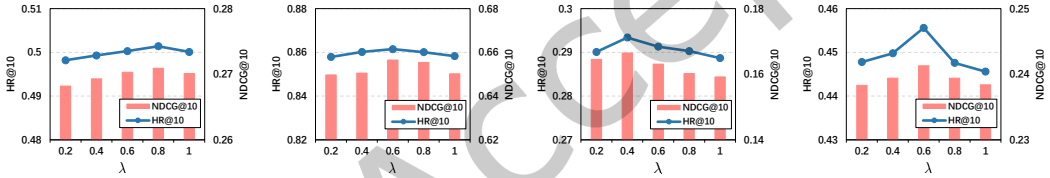
Table 7. Ablation study results towards HR@10 and NDCG@10 on the four datasets.

Methods	MovieLens-1M		Filmtrust		Douban-Movie		Ciao	
	HR@10	**NDCG@10**	**HR@10**	**NDCG@10**	**HR@10**	**NDCG@10**	**HR@10**	**NDCG@10**
Random	0.2419	0.1043	0.5793	0.3531	0.1408	0.0893	0.2882	0.1254
LightFR$_{init}$	0.3110	0.1542	0.6129	0.4932	0.1832	0.1193	0.3105	0.1632
LightFR$_{para}$	0.4942	0.2618	0.8589	0.6554	0.2913	0.1619	0.4489	0.2267
LightFR	**0.5014**	**0.2709**	**0.8615**	**0.6565**	**0.2934**	**0.1665**	**0.4556**	**0.2413**



(a) Movielens-1M (b) Filmtrust (c) Douban-Movie (d) Ciao

Fig. 4. Performance of LightFR with various code lengths f evaluated on four datasets.



(a) Movielens-1M (b) Filmtrust (c) Douban-Movie (d) Ciao

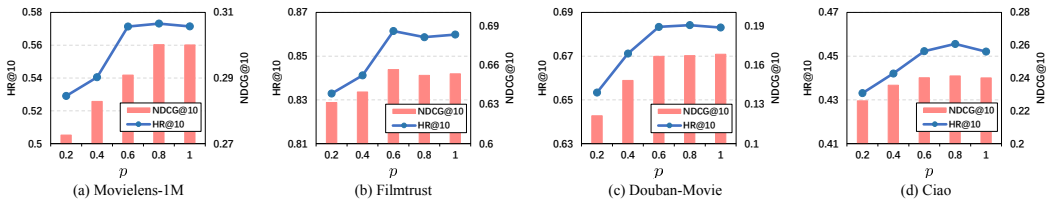
Fig. 5. Performance of LightFR with different values of trade-off parameter λ evaluated on four datasets.

Just Accepted

features of the pre-trained MF model is also less effective than our LightFR method, which verifies the superiority of our proposed method in the collaborative optimization between the server and local clients. Finally, the LightFR model, which performs gradient aggregation process on the server, is marginally better than the **LightFR**$_{para}$ which directly adopts parameter aggregation mechanism in the discrete aggregation module. The results can be ascribed to the fact that the use of direct aggregation of discrete parameters causes more information loss than the well-designed gradient aggregation mechanism in our approach.

4.2.3 Sensitivity Analysis. In this part, we analyze the performance fluctuations of our proposed LightFR with varying hyper parameters including the length of binary codes f, the trade-off parameter λ, and the ratio of selected clients p.

Firstly, the impact of the length of binary codes f on performance is studied. According to the experimental results shown in Fig. 4, as f increases from 8 to 64, significant performance improvements of LightFR are observed on the four datasets. Although with the increase of dimensions, the growth rate of performance gradually slows down. According to the above experimental results, the following insights can be obtained, that is, in the case of restricted storage capacity on the local client, the length of binary representations of users and items can be large as much as feasible, which can fully represent the structural properties of the original data.



(a) Movielens-1M (b) Filmtrust (c) Douban-Movie (d) Ciao

Fig. 6. Performance of LightFR with different ratios of selected clients p evaluated on four datasets.

Then, we experiment on a series of different values of trade-off parameter $\lambda \in \{0.2, 0.4, 0.6, 0.8, 1\}$. As shown in Fig. 5, when the value of the trade-off parameter λ increases, the HR@10 and NDCG@10 of the model show a upward trend, indicating that the balanced constraints are of positive significance to the discrete modeling of user preferences and item attributes. However, when the value of the trade-off parameter continues to increase, the model performance does not show a corresponding improvement, indicating that too large the value of the trade-off parameter is not conducive to the binary representations of the users and items but may lead to the appearance of overfitting issue. As a result, with the increase of the trade-off parameter λ, the recommendation accuracy increases at first. Then, the accuracy will gradually decrease as the λ continues to increase further. Besides, the change of the trade-off parameter λ has little fluctuations in the overall performance, which shows that our method is somewhat insensitive to the hyper parameter of λ.

Lastly, the effect of the ratio of selected clients p is discussed. The variable p is a hyper parameter which controls the proportion of local clients selected to participate in a round of global training. Intuitively, the higher the proportion of clients selected for each round of global training, the better the recommendation accuracy will be. We evaluate the impact of different ratios of selected clients $p \in \{0.2, 0.4, 0.6, 0.8, 1\}$, and we can derive some insightful observations from such results in Fig. 6. As the ratio of selected clients p increases from 0.2 to 1, there is generally an upward trend in HR@10 and NDCG@10 of our LightFR, but the improvement tends to stop when p is larger than 0.8 on Movielens-1M, Douban-Movie and Ciao datasets, 0.6 on Filmtrust dataset, respectively. We take the Movielens-1M dataset as an example, since our method has a similar tendency to the performance impact of p on the above four datasets. As the ratio of selected clients is increased from 0.2 to 0.8, the HR@10 and NDCG@10 have experienced a noticeable rise. This is mainly because, with more aggregation from different clients about the computed gradients or model parameters, the global model can obtain richer information to capture the preferences of users and the attributes of items more accurately. However, as the ratio of selected clients p continues to rise, the quality of recommendation list for users becomes slightly worse since aggregating the gradient information from more different clients means that more noise and unnecessary bias may be introduced. Furthermore, it will take a longer time to train the model since the server needs to wait for more clients to perform local training and aggregate their corresponding training results. Based on these findings, it is crucial for choosing the suitable parameter of p to achieve a good balance between the recommendation quality and the computing efficiency.

5 CONCLUSION

In this work, we propose a lightweight and privacy-preserving federated matrix factorization framework, *LightFR*, which enjoys both fast online inference and economic memory and communication consumption in federated settings. It decentralizes data storage compared with existing hashing based recommender systems. We alleviate the four challenges in designing this framework with learning to hash technique, *i.e.*, the huge memory occupation, the large communication bandwidth and the heavy calculation overheads on the local resource-constrained clients, and privacy protection for parameters transmitting between the server and clients. Besides, we design a

Just Accepted

federated discrete optimization algorithm between the central server and distributed clients, which can employ collaborative discrete optimization in federated scenarios to produce superior binary user representation on the local client and suitable binary item representations on the server side. Furthermore, we comprehensively discuss the superiority of our model on storage/communication efficiency, inference efficiency, and privacy enhancement from theoretical perspectives. We further conduct extensive experiments and the overall comparing experiments demonstrate that our framework significantly outperforms state-of-the-art FRS methods in terms of recommendation accuracy, resource savings and data privacy. Lastly, detailed sensitivity analysis regarding the hyper parameters further justifies the efficacy of our proposed model integrating learning to hash technique into canonical MF backbone in federated settings.

Despite the effectiveness and efficiency of our LightFR, there are still a few future directions to explore. Firstly, we essentially make a preliminary attempt to introduce the fundamental learning to hash technique into FL framework in recommendation scenario. Therefore, the binary user and item representations could be substantially enhanced by integrating extra side information to obtain more accurate and efficient discrete representations in federated settings [10, 44]. Secondly, LightFR exclusively designs discrete representation learning on top of vanilla MF model in its current version. In future, with the high flexibility of our proposed framework, we may explore the federated discrete representation learning mechanism for more advanced user modeling algorithms, such as factorization machines [30] and graph neural networks [3], so as to learn more compact and informative binary representations of users and items.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their helpful comments and suggestions. This work was partially supported by the National Natural Science Foundation of China (No. U1934220) and the Fundamental Research Funds for the Central Universities (No. 2021JBZ104).

REFERENCES

[1] Deepak Agarwal and Bee-Chung Chen. 2010. fLDA: matrix factorization through latent dirichlet allocation. In *WSDM*. 91–100.
[2] Muhammad Ammad-Ud-Din, Elena Ivannikova, Suleiman A Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. 2019. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888* (2019).
[3] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2018).
[4] Desheng Cai, Shengsheng Qian, Quan Fang, Jun Hu, and Changsheng Xu. 2022. User Cold-Start Recommendation via Inductive Heterogeneous Graph Neural Network. *ACM Trans. Inf. Syst.* (2022). Just Accepted.
[5] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. 2018. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210* (2018).
[6] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2020. Secure federated matrix factorization. *IEEE Intell. Syst.* 36, 5 (2020), 11–20.
[7] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. 2021. Exploiting shared representations for personalized federated learning. In *ICML*. 2089–2099.
[8] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*. 191–198.
[9] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2013. A Novel Bayesian Similarity Measure for Recommender Systems.. In *IJCAI*. 2619–2625.
[10] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. 2020. Content-Aware Neural Hashing for Cold-Start Recommendation. In *SIGIR*. 971–980.
[11] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2015), 1–19.
[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[13] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.* 22, 1 (2004), 5–53.
[14] Longke Hu, Aixin Sun, and Yong Liu. 2014. Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction. In *SIGIR*. 345–354.

Just Accepted

[15] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *STOC*. 604–613.
[16] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*. 426–434.
[17] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.
[18] Muyang Li, Xiangyu Zhao, Chuan Lyu, Minghao Zhao, Runze Wu, and Ruocheng Guo. 2022. MLP4Rec: A Pure MLP Architecture for Sequential Recommendations. In *IJCAI*. 2138–2144.
[19] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. In *MLSys*. 429–450.
[20] Tan Li, Linqi Song, and Christina Fragouli. 2020. Federated recommendation system via differential privacy. In *ISIT*. 2592–2597.
[21] Guanyu Lin, Feng Liang, Weike Pan, and Zhong Ming. 2020. Fedrec: Federated recommendation with explicit feedback. *IEEE Intell. Syst.* 36, 5 (2020), 21–30.
[22] Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Dongxiao Yu, Jun Ma, Maarten de Rijke, and Xiuzhen Cheng. 2020. Meta Matrix Factorization for Federated Rating Predictions. In *SIGIR*. 981–990.
[23] Zhaohao Lin, Weike Pan, Qiang Yang, and Zhong Ming. 2022. A Generic Federated Recommendation Framework via Fake Marks and Secret Sharing. *ACM Trans. Inf. Syst.* (2022). Just Accepted.
[24] Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. 2014. Collaborative Hashing. In *CVPR*. 2147–2154.
[25] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *WSDM*. 287–296.
[26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTAT*. 1273–1282.
[27] Lorenzo Minto, Moritz Haller, Benjamin Livshits, and Hamed Haddadi. 2021. Stronger privacy for federated collaborative filtering with implicit feedback. In *RecSys*. 342–350.
[28] Khalil Muhammad, Qinqin Wang, Diarmuid O'Reilly-Morgan, Elias Tragos, Barry Smyth, Neil Hurley, and Geraci. 2020. Fedfast: Going beyond average for faster training of federated recommender systems. In *SIGKDD*. 1234–1242.
[29] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust De-anonymization of Large Sparse Datasets. In *S&P*. 111–125.
[30] Steffen Rendle. 2010. Factorization machines. In *ICDM*. 995–1000.
[31] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *RecSys*. 240–248.
[32] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *NeurIPS*. 1257–1264.
[33] Ying Shan, Jian Jiao, Jie Zhu, and JC Mao. 2018. Recurrent binary embedding for gpu-enabled exhaustive retrieval from billion-scale semantic vectors. In *SIGKDD*. 2170–2179.
[34] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised Discrete Hashing. In *CVPR*. 37–45.
[35] Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges. *ACM Comput. Surv.* 47, 1 (2014), 1–45.
[36] Jiliang Tang, Huiji Gao, Xia Hu, and Huan Liu. 2013. Exploiting homophily effect for trust prediction. In *WSDM*. 53–62.
[37] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. 2017. A survey on learning to hash. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 4 (2017), 769–790.
[38] Qinyong Wang, Hongzhi Yin, Tong Chen, Zi Huang, Hao Wang, Yanchang Zhao, and Nguyen Quoc Viet Hung. 2020. Next point-of-interest recommendation on resource-constrained mobile devices. In *WWW*. 906–916.
[39] Qinyong Wang, Hongzhi Yin, Tong Chen, Junliang Yu, Alexander Zhou, and Xiangliang Zhang. 2021. Fast-adapting and privacy-preserving federated recommender system. *The VLDB Journal* (2021), 1–20.
[40] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z. Sheng, Mehmet A. Orgun, Longbing Cao, Francesco Ricci, and Philip S. Yu. 2021. Graph Learning based Recommender Systems: A Review. In *IJCAI*. 4644–4652.
[41] Shoujin Wang, Xiuzhen Zhang, Yan Wang, Huan Liu, and Francesco Ricci. 2022. Trustworthy Recommender Systems. *arXiv preprint arXiv:2208.06265* (2022).
[42] Yair Weiss, Antonio Torralba, and Rob Fergus. 2008. Spectral hashing. In *NeurIPS*. 1753–1760.
[43] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. 2022. Communication-efficient federated learning via knowledge distillation. *Nature communications* 13, 1 (2022), 1–8.
[44] Jun Wu, Fangyuan Luo, Yujia Zhang, and Haishuai Wang. 2020. Semi-discrete matrix factorization. *IEEE Intell. Syst.* 35, 5 (2020), 73–83.
[45] Liu Yang, Ben Tan, Vincent W Zheng, Kai Chen, and Qiang Yang. 2020. Federated recommendation systems. In *Federated Learning*. 225–239.
[46] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Trans. Intel. Syst. Tec.* 10, 2 (2019), 1–19.

Just Accepted

[47] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation. In *SIGIR*. 1469–1478.
[48] Honglei Zhang, Gangdu Liu, and Jun Wu. 2018. Social collaborative filtering ensemble. In *PRICAI*. 1005–1017.
[49] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete Collaborative Filtering. In *SIGIR*. 325–334.
[50] Ruizhe Zhang, Xiaohui Xie, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2021. Constructing a comparison-based click model for web search. In *WWW*. 270–283.
[51] Yan Zhang, Hongzhi Yin, Zi Huang, Xingzhong Du, Guowu Yang, and Defu Lian. 2018. Discrete deep learning for fast content-aware recommendation. In *WSDM*. 717–726.
[52] Ke Zhou and Hongyuan Zha. 2012. Learning Binary Codes for Collaborative Filtering. In *SIGKDD*. 498–506.
[53] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *NeurIPS*. 14747–14756.
[54] Yongchun Zhu, Ruobing Xie, Fuzhen Zhuang, Kaikai Ge, Ying Sun, Xu Zhang, Leyu Lin, and Juan Cao. 2021. Learning to warm up cold item embeddings for cold-start recommendation with meta scaling and shifting networks. In *SIGIR*. 1167–1176.

Just Accepted