Federated Matrix Factorization with Privacy Guarantee

Zitao Li
Purdue University
li2490@purdue.edu

Bolin Ding
Alibaba Group
bolin.ding@alibaba-inc.com

Ce Zhang
ETH Zürich
ce.zhang@inf.ethz.ch

Ninghui Li
Purdue University
ninghui@purdue.edu

Jingren Zhou
Alibaba Group
jingren.zhou@alibaba-inc.com

ABSTRACT

Matrix factorization (MF) approximates unobserved ratings in a rating matrix, whose rows correspond to users and columns correspond to items to be rated, and has been serving as a fundamental building block in recommendation systems. This paper comprehensively studies the problem of matrix factorization in different *federated learning* (FL) settings, where a set of parties want to cooperate in training but refuse to share data directly. We first propose a generic algorithmic framework for various settings of federated matrix factorization (FMF) and provide a theoretical convergence guarantee. We then systematically characterize privacy-leakage risks in data collection, training, and publishing stages for three different settings and introduce privacy notions to provide end-to-end privacy protections. The first one is *vertical federated learning* (VFL), where multiple parties have the ratings from the same set of users but on disjoint sets of items. The second one is *horizontal federated learning* (HFL), where parties have ratings from different sets of users but on the same set of items. The third setting is *local federated learning* (LFL), where the ratings of the users are only stored on their local devices. We introduce adapted versions of FMF with the privacy notions guaranteed in the three settings. In particular, a new private learning technique called *embedding clipping* is introduced and used in all the three settings to ensure differential privacy. For the LFL setting, we combine differential privacy with secure aggregation to protect the communication between user devices and the server with a strength similar to the local differential privacy model, but much better accuracy. We perform experiments to demonstrate the effectiveness of our approaches.

PVLDB Reference Format:
Zitao Li, Bolin Ding, Ce Zhang, Ninghui Li, and Jingren Zhou. Federated Matrix Factorization with Privacy Guarantee. PVLDB, 15(4): 900 - 913, 2022.
doi:10.14778/3503585.3503598

1 INTRODUCTION

Federated learning (FL) has been applied as a paradigm for multiple parties to collaboratively train a model without directly sharing their data. FL can be categorized according to the data partition as summarized in [26]. Based on whether the data are partitioned by features or users, there are *vertical FL* (VFL) and *horizontal FL* (HFL), respectively. Based on how much users' data each party can access, the HFL setting can be extended from the cross-silo setting to the cross-device setting, which we call *local FL* (LFL) in this paper.

This paper investigates the *matrix factorization* problem (e.g., [18, 30, 45, 51]) under the above three settings of federated learning. Matrix factorization is an important building block in recommendation systems. In its simplest form, with a rating matrix as the input, it learns to represent users (rows) and items (columns or features) as low-dim vectors, called *user embeddings* and *item embeddings*, respectively, so that the dot-product of a user embedding and an item embedding measures how the user prefers the item. The embeddings, preferably learned in a privacy-preserving way, can be used in downstream applications [4, 52].

1.1 Coordination and Trust Models

For federated matrix factorization, we assume that the coordination pattern is the classic server-client model [26, 50]. There are three types of entities: *n users*, totally *s parties* holding users' data, and a *coordination server*. We assume that users' ratings for items are fixed and stored on parties beforehand; each party does not trust the server or other parties, and wants to protect its users' privacy.

It is challenging in FL to prevent privacy leakage. As shown in [39], users in the anonymized rating matrices can still be identified. Without sharing the ratings directly, training samples can still be recovered from shared gradients in FL [54]. Even if no local data are shared explicitly and the secure multiparty computation techniques [6, 49] are applied in communication between parties, the final models are still vulnerable to inference attacks [12]. To provide provable resistance to such attacks, efforts have been made to protect federate learning with differential privacy (DP) [13].

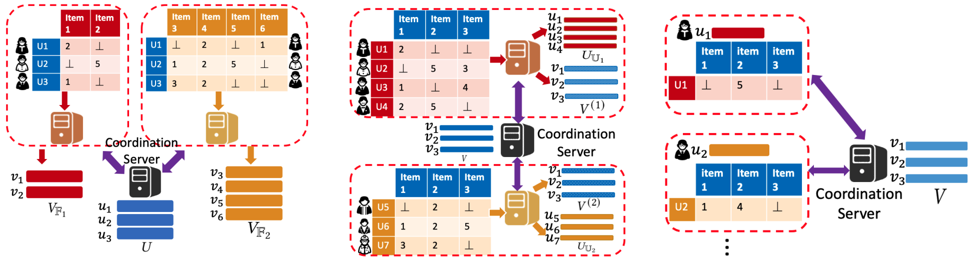
In all following settings, we aim to ensure that all the information sent out by one party satisfies DP relative to its user's data. That is, the *privacy boundary* for each party is directly surrounding that party. In other words, each party does not need to trust any other party in order to protect the data it has. We identify whether a certain kind of information (e.g., user/item embeddings) cross such privacy boundaries depending on the settings below.

We give an overview of the coordination and trust models proposed in this paper for different FL settings in Figures 1(a)-1(c).

Vertical FL (VFL). The rating matrix is partitioned on columns across different parties. All parties share the same set of users, but each owns a different subset of items. A typical application scenario for VFL is that two companies, e.g., an online content platform and a local retailer, have the same set of users, but each has different user-behavior information; they want to cooperate in training a recommendation model, which learns from both users' behavior on online contents (items such as videos) and items in local stores.

During the learning procedure, different parties exchange user embeddings with the coordination server; the shared user embeddings are aggregated at the coordination server and sent back to each party to refine local user/item embeddings for the next iterations. In the trust model of this setting, the communication between

Proceedings of the VLDB Endowment, Vol. 15, No. 4 ISSN 2150-8097.
doi:10.14778/3503585.3503598

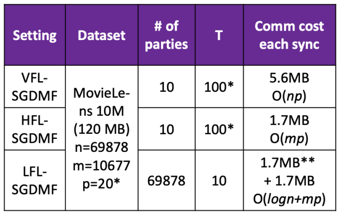


(a) VFL setting

(b) HFL setting

(c) LFL setting

Figure 1: Different FL settings (a)-(c) with sensitive information exchanged on privacy boundaries (the dashed lines). Each u_i is a user embeddings, v_j is an item embedding.



Setting	Dataset	# of parties	T	Comm cost each sync
VFL-SGDMF	MovieLens 10M (120 MB) n=69878 m=10677 p=20*	10	100*	5.6MB O(*np*)
HFL-SGDMF		10	100*	1.7MB O(*mp*)
LFL-SGDMF		69878	10	1.7MB** + 1.7MB O(*logn+mp*)

* Our default experiment setting is sync iterations T=100, local iterations T'=10 and embedding dimension p=20 in this paper.
** LFL setting relies on Secure Aggregation, which needs additional O(log n) for decoding.

Figure 2: Communication cost

parties and the coordination server (**purple** arrows) and the output of final user/item embeddings (**red**/**orange** arrows) cross the privacy boundary, and they need to be protected under DP. Note that since one user's data is split among multiple parties, the joint output satisfies DP relative to a user if all parties follow the protocol.

Horizontal FL (HFL). In HFL, each party has a subset of users and all their ratings, so the rating matrix is partitioned on rows across different parties. Different from distributed learning, there is no i.i.d. assumption about the partitioning in HFL. A typical scenario for HFL is that different hospitals have records for different groups of patients (users) about the same set of diseases (items); hospitals want to train a patient-disease prediction model cooperatively.

The coordination for HFL is symmetric to the one for VFL. As different parties have different sets of users, a coordination server synchronizes all updates on item embeddings periodically, and sends them back to each party to refine user embeddings learned for the users in this party. The final user embeddings do not have to be published (within the privacy boundary, following the similar assumptions as [24, 33]); thus, only the shared updates on item embeddings (**purple** arrows), as well as the aggregated item embeddings if to be published, need to be protected with DP.

Local FL (LFL). LFL is a special case of HFL, where a user as one party by her/himself has all her/his ratings. It is a common scenario in mobile applications: each user holds the information about which items s/he visits only on her/his own mobile device, and does not want to share it directly with other parties; LFL enables the service provider to train a recommender without collecting the sensitive information about exact interactions between users and items.

Each user's device exchanges updates on item embeddings with the coordination server who aggregates them for the shared item embeddings; afterwards, the shared item embeddings are broadcast to each user's device to refine user embedding which is kept only locally. Note that under the trust model similar to [41], the user embedding does not have to be published, as the ranking procedure, e.g., calculating the dot-product of user embedding and item embedding for a specific user, can be done locally in users device. Only the shared updates on item embeddings (**purple** arrows) and the aggregated item embeddings, need to be protected with DP.

Formal description about the coordination and trust models in these three FL settings will be given in Sections 3-6.

1.2 Our Contributions

This paper comprehensively studies the matrix factorization problem under the three FL settings (i.e., HFL, VFL and LFL). Our contributions in this paper can be summarized as the following:

- We introduce an abstracted computation paradigm in Section 3 for solving the matrix factorization problem in FL settings. This paradigm is applicable for the three settings. We prove the convergence of our proposed paradigm with limited communication and coordination between parties and no assumption on whether the data are distributed uniformly across parties or not.
- Different FL settings confront different potential privacy-leakage risks. In the context of matrix factorization, we systematically characterize privacy-leakage risks setting by setting, and propose corresponding notions of privacy in different settings to prevent such risks. Based on DP and secure aggregation, these privacy notions provide end-to-end privacy guarantees on the information across the privacy boundary. Depending on the protection granularity and strength needed, there are two versions for each privacy notion, per-rating version (protecting each rating at one time) and per-user version (protecting each user's behavior as a whole). We design FL algorithms based on these privacy notions for different settings.
- Gradient clipping is a standard technique in private optimization under DP when solving problems with unbounded gradients, but it also has drawbacks as introducing additional bias and requiring complicated hyper-parameter tuning [3, 10]. Instead, we propose using *embedding clipping* to bound the sensitivity of the embedding updates: (intermediate) item/user embeddings are projected into a subspace so that their norms and thus the sensitivity is bounded by a threshold, which is dependent on the range of the ratings in the task and does not need to be tuned. We introduce our embedding clipping technique under the VFL setting in Section 4, and will repeatedly use it in other settings as well. We show with experiments that embedding clipping can have more accurate updates than gradient clipping and other approaches.
- Based on our proposed FL paradigm and embedding clipping, we design `VFL-SGDMF` algorithm for matrix factorization under the VFL setting in Section 4, providing privacy protection for both user embeddings and item embeddings in both intermediate and final outputs. Compared with training only locally with no communication, our experiments show that our `VFL-SGDMF` algorithm can provide high accuracy in predictions by absorbing heterogeneous information owned by different parties.

- We propose HFL-SGDMF algorithm for matrix factorization under the HFL setting in Section 5. In this setting, the communication between different parties is protected by DP, but the parties can fine-tune the final results to obtain more accurate models. Our experiments show VFL-SGDMF outperforms the non-private local training method and a strawman method based on DPSGD [1, 42] synchronizing privatized gradient from all parties in every iteration.
- Extending from the cross-silo HFL setting to cross-device LFL setting in Section 6, we combine secure aggregation with differential privacy and propose LFL-SGDMF algorithm to ensure the server can only obtain aggregated and privatized sum of gradients. A novel two-round aggregation approach is introduced to ensure that our algorithm tolerates user dropouts (disconnection of users' devices) during training without introducing too much excessive DP noise. With similar strength of privacy protection, our algorithm provides more accurate predictions on the testing set than the approach purely based on local differential privacy.

1.3 Related Work

The majority of the work enforcing DP in FL is under the HFL setting [26, 50], and they can be categorized into two classes. One class of works, such as DP-FedSGD [36, 48], are extensions based on the DPSGD [1, 42], in which a server aggregates the (privatized) gradients from each party and updates models centrally. Another class of techniques, like DP-FedAvg [35, 47], performs model average periodically, where local parties send their (privatized) updated local models to the server, and the server updates the centralized model by averaging those local models. DP-FedAvg has less communication cost than DP-FedSGD, by avoiding sharing gradients in every iteration. Our paper has the following differences in the HFL setting compared with those existing works. 1) Different from [35, 36], we assume that there is no fully trusted coordinator, and all the information shared by the parties must be differentially private. 2) Compared with [47, 48], MF is a non-convex optimization problem with unbounded norm of gradient. 3) Different from the DPSGD [1], we propose embedding clipping for this problem. Embedding clipping can maintain the aggregated gradient information better than the gradient clipping approaches, as shown in Section 4. There are also results under non-private HFL setting [28, 29].

Existing work about other problems in the VFL setting includes learning tree models with secure multiparty computation techniques [32, 49] and training composed models [23]. A recent work [22] studies the generalized linear model with distributed features under the ADMM framework. Another paper [9] discussing asynchronous supervised learning with VFL assumes the labels are public accessible. According to our knowledge, this is the first paper to discuss the MF problem in VFL setting with a privacy guarantee.

Existing work of privacy-preserving learning under the LFL setting aligns with the local differential privacy (LDP), such as [11, 15, 16, 19]. There are cryptographic methods in the LFL setting called secure aggregation [5, 6] in parallel. The authors of [41] proposed a algorithm to learn item embeddings with the local differential privacy (LDP). However, we show that their algorithm is not better than training only the user embeddings locally.

Some work about matrix factorization with DP in a centralized setting includes [24, 25, 33, 40]. Also, the MF problem was studied with homomorphic encryption [8]. Without privacy protection, federated matrix factorization was studied as a multi-view learning problem in [17]. Some other work focuses on the tensor factorization in HFL, like [27] with non-private aggregation and [34] solving the problem with EASGD and output perturbation for DP.

2 BACKGROUND

2.1 Matrix Factorization

The input to the *matrix factorization* (MF) [30] problem is a matrix $\mathbf{X} \in (\mathbb{R} \cup \{\perp\})^{n \times m}$ consisting of ratings from n users on m items (e.g., movies). An element $\mathbf{X}_{ij} \in \mathbb{R}$ is the *observed* rating from user i on item j, and $\mathbf{X}_{ij} = \perp$ means that the rating is *unobserved.* Although n and m can be very large, the matrix is sparse in the sense that only a small fraction, e.g., 1%, of ratings, are observed. We denote the set of indices of observed ratings as $\Omega = \{(i,j) | \mathbf{X}_{ij} \neq \perp\}$. With the assumption that the rating matrix can be approximated by the inner product of two low rank matrices, $U \in \mathbb{R}^{n \times p}$ and $V \in \mathbb{R}^{m \times p}$ where $p \ll n, m$, the matrix factorization problem can be formalized as minimizing the loss function $\mathcal{L}(\mathbf{X}, U, V)$:

$$\frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} \mathcal{L}_{i,j}(\mathbf{X}, U, V) = \frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} \left(\mathbf{X}_{ij} - \langle u_i, v_j \rangle\right)^2, \qquad (1)$$

where u_i and v_j are the row vectors of U and V, which are called *user embeddings* and *item embeddings*, respectively. For a specific user i, ratings on items $j \in [m]$ can be approximated by the inner product of user embedding and item embeddings $\langle u_i, v_j \rangle$. We will formalize the federated matrix factorization problem in Section 3.

2.2 Differential Privacy

In the federated learning settings, any non-trivial solution requires parties (or user devices in the LFL setting) to share information distilled from local datasets with others. Recent research results [38, 54] have shown that sharing gradients directly can leak user privacy. In this paper, we employ DP [13] to protect user's information from privacy leakage because DP is more powerful in resisting membership inference attacks, re-identification or reconstruction attacks than other privacy notations, such as k-anonymity [14, 31].

Definition 1 ((ϵ, δ)-DP). *A randomized mechanism* M *is differentially private iff given any pair of neighboring datasets* $\mathbf{X}$ *and* $\mathbf{X}'$, *the following holds for all possible output* o: $\Pr[\mathsf{M}(\mathbf{X}) = o] \leq e^{\epsilon} \Pr[\mathsf{M}(\mathbf{X}') = o] + \delta$.

Sequential composition: moments accountant. The private dataset may be accessed multiple times. Moments accountant [1] can be used to provide tight analysis for the privacy loss of a sequence of adaptive mechanisms. In general, let $\mathsf{M}(\mathbf{X}, \mathsf{aux})$ be a function that returns outputs from the dataset $\mathbf{X}$ and an auxiliary input aux. The auxiliary input aux can include all intermediate results from previous steps. The logarithm of the moment generating function (of privacy loss in M) evaluated at the value λ is defined as

$$\alpha_{\mathsf{M}}(\lambda; \mathbf{X}, \mathbf{X}', \mathsf{aux}) = \log \mathbb{E}_{o \sim \mathsf{M}(\mathbf{X}, \mathsf{aux})} \left[\exp(\lambda \log \frac{\Pr[\mathsf{M}(\mathbf{X}, \mathsf{aux}) = o]}{\Pr[\mathsf{M}(\mathbf{X}', \mathsf{aux}) = o]}) \right],$$

which is upper bounded by $\alpha_{\mathsf{M}}(\lambda) = \max_{\mathbf{X}, \mathbf{X}', \mathsf{aux}} \alpha_{\mathsf{M}}(\lambda; \mathbf{X}, \mathbf{X}', \mathsf{aux})$. Suppose there is a sequence of such adaptive mechanisms $\mathsf{M}_1, \ldots, \mathsf{M}_T$, where each M_t takes the dataset $\mathbf{X} \in \mathcal{X}$ and the outputs of all previous $t-1$ mechanisms as input. The moment accountant technique bounds the total privacy loss in $\mathsf{M}_1, \ldots, \mathsf{M}_T$ via composing the logarithms of the moment generating functions, $\alpha_{\mathsf{M}_t}(\lambda)$'s for $t \in [T]$.

THEOREM 1 (MOMENT ACCOUNTANTS (MA) [1]). *Let a mechanism $\mathtt{M}$ be a sequence of adaptive mechanisms $\mathtt{M}_1, \dots, \mathtt{M}_T$ defined as above. Then we have 1) [composition] for any λ, $\alpha_{\mathtt{M}}(\lambda) \leq \sum_{t=1}^{T} \alpha_{\mathtt{M}_t}(\lambda)$; and 2) [tail bound] for any $\epsilon > 0$, the mechanism $\mathtt{M}$ is (ϵ, δ)-differentially privacy for $\delta = \min_\lambda \exp(\alpha_{\mathtt{M}}(\lambda) - \lambda\epsilon)$.*

This paper applies MA to analyze the privacy guarantee of our solutions to federated MF. In the experiments, we calculate the exact privacy loss using the package [44].

Parallel composition. Other than sequential composition, parallel composition is another useful property of DP in FL. When the parties have disjoint datasets, the total privacy loss in the FL process is equal to the maximum privacy loss of an individual party.

PROPOSITION 1 (PARALLEL COMPOSITION [37]). *If mechanisms $\mathtt{M}_1, \dots, \mathtt{M}_T$ are $(\epsilon^{(1)}, \delta^{(1)}), \dots, (\epsilon^{(s)}, \delta^{(s)})$ differentially private and they are computed on disjoint datasets $\mathbf{X}_1, \dots, \mathbf{X}_s$, then the overall privacy loss will be $(\max_k\{\epsilon^{(k)}\}, \max_k\{\delta^{(k)}\})$.*

2.2.1 Differential Privacy in Federated Matrix Factorization. When we enforce DP on the MF problem in FL settings, some details need to be specified as the following.

Privacy definitions related to MF. Different definitions of "neighboring datasets" imply different privacy guarantees. In this paper, we consider two types of guarantees, *per-rating privacy* and *per-user privacy*. We formally define them as follows.

- *Per-rating privacy.* To provide per-rating privacy, two rating matrices $\mathbf{X}$ and $\mathbf{X}'$ are neighboring datasets if there is an index (i, j), such that either $\mathbf{X}_{ij} = \perp$ is unobserved and $\mathbf{X}'_{ij} \neq \perp$ is observed, or the reverse; and for any other index $(i', j') \neq (i, j)$, $\mathbf{X}_{i'j'} = \mathbf{X}'_{i'j'}$. Thus, per-rating privacy protects every single rating given by each user. The protection provided by per-rating privacy on the overall behavior of a user can be weak if a user contributes a large number of ratings due to the sequential composition of differential privacy.
- *Per-user privacy.* To provide *per-user privacy*, two rating matrices $\mathbf{X}$ and $\mathbf{X}'$ are neighboring datasets if and only if there exists a user i such that i) for any user $i' \neq i$, $\mathbf{X}_{i'\cdot} = \mathbf{X}'_{i'\cdot}$, and ii) either $\mathbf{X}_{ij} = \perp$ for all items $j \in [m]$ or $\mathbf{X}'_{ij} = \perp$ for all items $j \in [m]$. The same definition has been used in [25]. This definition helps prevent the adversary from distinguishing any individual user i's ratings from not rating anything. The "dummy row" is only used to define neighboring rating matrix and sensitivity calculation, but it is never added in the computation process. We use the above per-user definition in this paper because it works with moment accountants more naturally for composition. An alternative is to define $\mathbf{X}$ and $\mathbf{X}'$ as neighboring if they have the same number of rows and differ in at most one row. This definition helps prevent the adversary from distinguishing any individual user i's ratings from any other behavior. Satisfy (ϵ, δ)-DP using the former satisfies $(2\epsilon, 2\delta)$-DP under the latter.

Bounding sensitivity. In DP, the effect of adding/removing one record is called sensitivity. Gradient clipping and trimming are two techniques to bound the sensitivities in MF.

- *Gradient clipping.* In differentially private optimization, when the gradient-based method is applied (e.g., DPSGD) but the gradients are unbounded, gradient clipping is the technique to bound the sensitivity of gradients [1]. For example, for per-rating privacy, the gradient for the user embedding from a rating of user i on item j, $\nabla_U \mathcal{L}_{i,j}(U, V)$, is not bounded if there are no additional constraints. Thus, the gradient has to be clipped as $\frac{\nabla_U \mathcal{L}_{i,j}(U,V)}{\max\{1, \|\nabla_U \mathcal{L}_{i,j}(U,V)\|_2 / C\}}$.

  However, gradient clipping has several disadvantages in practice. Firstly, gradient clipping limits the effect of each rating. So gradient clipping limits the effects of gradients with large magnitudes in aggregation, the useful updates may be canceled out. Secondly, gradient clipping requires extract space to store the gradients from each rating/user. In this paper, we propose to use *embedding clipping* (in Section 4) to bound the sensitivity of the gradient.
- *Trimming [33].* The bounded gradients from a single rating are sufficient to derive the sensitivity of per-rating privacy. However, a user can have many ratings, and the sensitivity of per-user privacy is linear to the maximum number of ratings a user can have. An excessive noise is required to provide per-user privacy because of the high sensitivity. Thus, when per-user privacy is enforced, as the first step, each user's record keeps at most $\theta^{(k)}$ ratings in the local rating matrix of party k, and turns the rest of ratings to $\perp$.
- *Privacy budget composition.* Solving the federated MF problem requires multiple accesses to the dataset. Based on different data partitions and privacy settings, both parallel composition and sequential composition can be used in this paper. The privacy losses are bounded for the three different settings in Section 4 to 6.

3 FEDERATED MATRIX FACTORIZATION

The goal of federated matrix factorization is to let the involved parties learn some common components cooperatively. In the VFL setting, the parties want to learn the user embeddings together; in the HFL and LFL setting, the item embeddings are shared between the different parties. This section formalizes the problems in VFL, HFL and LFL, then introduces a non-private federated matrix factorization paradigm with convergence guarantees.

3.1 Problem Formulation

We assume there are s parties in the learning process. The party is an abstraction that has different meanings in cross-silo FL and cross-device FL [26] scenarios. When a party has more than one user's data, it represents an organization, and the scenario is cross-silo FL; when each party has only one user's data, it represents a user device in practice, and the scenario becomes cross-device FL.

MF in vertical federated learning (VFL). As described in Figure 1(a), each party in the VFL setting has data from the same set of n users $\mathbb{U}$, corresponding to n rows in the rating matrix $\mathbf{X}$, but owns only a subset of items. Let $\{\mathbb{F}_1, \dots, \mathbb{F}_s\}$ be a partition of the set of all items $[m]$ with $|\mathbb{F}_k| = m_k$, and each $\mathbb{F}_k$ be the subset of items *owned by* party $k \in [s]$. Thus, the rating matrix is partitioned *vertically* into $\mathbf{X} = [\mathbf{X}_{\mathbb{F}_1}, \dots, \mathbf{X}_{\mathbb{F}_s}]$ with $\mathbf{X}_{\mathbb{F}_k} \in (\mathbb{R} \cup \{\perp\})^{n \times m_k}$ as the rating information owned by party k. We assume that the users in local rating matrices are aligned in the way that for all $k \in [s]$, the i^{th} rows $(X_{\mathbb{F}_k})_i$ are corresponds to the same user i. The s parties want to collaboratively learn user embeddings $U = [u_i]_{i \in [n]} \in \mathbb{R}^{n \times p}$, and item embeddings $V_{\mathbb{F}_k} = [v_j]_{j \in \mathbb{F}_k} \in \mathbb{R}^{m_k \times p}$ for the items owned by each of them. In practice, the number of parties involved in the vertical federated learning is limited in most scenarios, i.e. $s \leq 10$.

- *Loss functions in VFL.* For each party k, the local loss function is $\mathcal{L}\left(\mathbf{X}_{\mathbb{F}_k}, U, V_{\mathbb{F}_k}\right) = \frac{1}{|\Omega_k|} \sum_{(i,j) \in \Omega_k} \left(\mathbf{X}_{ij} - \langle u_i, v_j \rangle\right)^2$. The global loss function can be rewritten as: $\mathcal{L}\left(\mathbf{X}, U, V\right) = \sum_k \frac{|\Omega_k|}{|\Omega|} \mathcal{L}\left(\mathbf{X}_{\mathbb{F}_k}, U, V_{\mathbb{F}_k}\right) = \frac{1}{|\Omega|} \sum_k \sum_{(i,j)\in\Omega_k} \left(\mathbf{X}_{ij} - \langle u_i, v_j \rangle\right)^2$.

Our algorithms in this paper are based on stochastic gradient descent (SGD). Based on the loss function, the gradients to the user/item embeddings are the following:

$$\nabla_U \mathcal{L}(\mathbf{X}, U, V) = \sum_k \frac{|\Omega_k|}{|\Omega|} \nabla_U \mathcal{L}(\mathbf{X}_{\mathbb{F}_k}, U, V_{\mathbb{F}_k}), \tag{2}$$

$$\nabla_{V_{\mathbb{F}_k}} \mathcal{L}(\mathbf{X}, U, V) = \frac{|\Omega_k|}{|\Omega|} \nabla_{V_{\mathbb{F}_k}} \mathcal{L}(\mathbf{X}_{\mathbb{F}_k}, U, V_{\mathbb{F}_k}). \tag{3}$$

Equation (2) indicates that the updates on the global user embeddings need to be the weighted average of the updates of local user embedding; Equation (3) suggests that the local updates on the item embeddings, although no need to be aggregated, need to be scaled by the a different ratio $|\Omega_k|/|\Omega|$ for each party k.

When using the SGD, each party can sample a submatrix from the local rating matrix consisting of n' rows and m'_k columns. The observed ratings in this sampled submatrix is denoted as I_k. We assume that all parties have about the same "density" of observed ratings, $\tau = |\Omega|/(nm)$. The normalized stochastic gradients on party k to user embeddings and item embeddings are denoted as:

$$\mathbf{g}_U^{(k)} = \frac{1}{n' m'_k \tau} \sum_{(i,j)\in I_k} \nabla_U \mathcal{L}_{i,j}\left(\mathbf{X}_{\mathbb{F}_k}, U^{(k)}, V_{\mathbb{F}_k}\right) \approx \nabla_U \mathcal{L}(\mathbf{X}_{\mathbb{F}_k}, U, V_{\mathbb{F}_k}),$$

$$\mathbf{g}_{V_{\mathbb{F}_k}}^{(k)} = \frac{1}{n' m'_k \tau} \sum_{(i,j)\in I_k} \nabla_{V_{\mathbb{F}_k}} \mathcal{L}_{i,j}\left(\mathbf{X}_{\mathbb{F}_k}, U^{(k)}, V_{\mathbb{F}_k}\right) \approx \nabla_{V_{\mathbb{F}_k}} \mathcal{L}(\mathbf{X}_{\mathbb{F}_k}, U, V_{\mathbb{F}_k}).$$

MF in horizontal federated learning (HFL). As in Figure 1(b), instead of partitioned items, let $\{\mathbb{U}_1, \ldots, \mathbb{U}_s\}$ be a partition of the set of all users $[n]$, and $\mathbb{U}_k$ is the subset of users belongs to party k. Compared with the vertical setting, data of a user are on a single party. All parties potentially have ratings for all the items in $\mathbb{F}$ with size m. The rating matrix $\mathbf{X}$ is thus partitioned *horizontally* as $\mathbf{X} = [\mathbf{X}_{\mathbb{U}_1}; \ldots; \mathbf{X}_{\mathbb{U}_s}]$ where each party k has the rating information as a submatrix $\mathbf{X}_{\mathbb{U}_k} \in (\mathbb{R} \cup \{\perp\})^{n_k \times m}$, in which we can observe a subset of ratings Ω_k. The goal of the s parties in the HFL setting is to learn item embeddings $V = [v_j]_{j\in\mathbb{F}}$ if size $m \times p$ cooperatively with higher utility by exchanging privacy-preserved information, and each party $k \in [s]$ learns a set of user embeddings $U_{\mathbb{U}_k} = [u_i]_{i\in\mathbb{U}_k}$ of size $n_k \times p$ for the users in their local dataset.

- *Loss functions in HFL.* Similar to the VFL, the local loss function is $\mathcal{L}\left(\mathbf{X}_{\mathbb{U}_k}, U_{\mathbb{U}_k}, V\right) = \frac{1}{|\Omega_k|} \sum_{(i,j)\in\Omega_k} \left(\mathbf{X}_{ij} - \langle u_i, v_j \rangle\right)^2$, and the global loss function is $\mathcal{L}\left(\mathbf{X}, U, V\right) = \sum_k \frac{|\Omega_k|}{|\Omega|} \mathcal{L}\left(\mathbf{X}_{\mathbb{U}_k}, U_{\mathbb{U}_k}, V\right)$. Similarly, we denote the gradients and the normalized stochastic gradients on party k to user embeddings and item embeddings and denote them as $\mathcal{L}_{U_{\mathbb{U}_k}}(\mathbf{X}_{\mathbb{U}_k}, U_{\mathbb{U}_k}, V)$, $\mathcal{L}_V(\mathbf{X}_{\mathbb{U}_k}, U_{\mathbb{U}_k}, V)$, $\mathbf{g}_{U_{\mathbb{U}_k}}^{(k)}$ and $\mathbf{g}_V^{(k)}$.

MF in horizontal federated learning (LFL). When formalized as an optimization problem, the LFL setting is an extreme case of the HFL setting with $s = n$ and $\mathbb{U}_i = \{i\}$. Thus, the loss functions, the gradients and convergence can inherit naturally from HFL.

3.2 Common FL Paradigm for MF Problem

We introduce a paradigm for solving the MF problem in both VFL and HFL settings. It consists of three stages: **[Stage 1]: initialization and local pre-computation; [Stage 2] cooperative learning; [Stage 3] local fine-tuning**. The paradigm is described as vanilla FMF in Figure 3(a). In VFL settings, the *shared embeddings* are the user embeddings, while the *local embeddings* are item embeddings. The roles switch in the HFL setting, where the *shared embeddings* are the item embeddings and the *local embeddings* are the user embeddings. The first and the last stages are done locally by each party, while the second stage requires communication.

The [Stage 2] cooperative learning of FMF consists of Step (c2) and (s2). The user embeddings and item embeddings are updated locally with SGD. The updating step are shown as the following:

$$\text{VFL:} \qquad U^{(k)} \leftarrow U^{(k)} - \gamma_t \mathbf{g}_U^{(k)}, V_{\mathbb{F}_k} \leftarrow V_{\mathbb{F}_k} - \frac{\gamma_t m_k}{m} \mathbf{g}_{V_{\mathbb{F}_k}}^{(k)}; \tag{4}$$

$$\text{HFL:} \qquad U_{\mathbb{U}_k} \leftarrow U_{\mathbb{U}_k} - \frac{\gamma_t n_k}{n} \mathbf{g}_{U_{\mathbb{U}_k}}^{(k)}, V^{(k)} \leftarrow V_{\mathbb{F}_k} - \gamma_t \mathbf{g}_{V^{(k)}}^{(k)}. \tag{5}$$

Here we assume that all parties have about the same rating "density", $\frac{|\Omega_k|}{|\Omega|} \approx \frac{m_k}{m}$ or $\frac{|\Omega_k|}{|\Omega|} \approx \frac{n_k}{n}$. As mentioned in the previous subsection, the global loss function is a weighted average of the local ones. Thus, the weights are applied in (4) and (5) to match the global loss. Besides, the global shared embeddings are updated as the weighted average of the local ones in Step (s2) of FMF, which is either $U = \sum_{k=1}^{s} \frac{m_k}{m} U^{(k)}$ for VFL or $V = \sum_{k=1}^{s} \frac{n_k}{n} V^{(k)}$ for HFL.

Coordination and communication cost of FMF. Communication only happens in the cooperative learning stage. The coordination logic behind (c2) and (s2) is that for each iteration, the coordination server first requests the locally updated embeddings from each party; after receiving the request, each party uploads their embeddings; the server aggregated the updated embeddings from parties and broadcast the weighted average to all the parties. The coordination server of FMF performs synchronization over all involved parties T times. The total communication cost is $O(Tnp)$ for VFL or $O(Tmp)$ for HFL. Notice that there are T' local iterations between each update. Compared with the gradient-aggregated methods (e.g., [36, 48]), FMF saves a factor of T' synchronizations. In practice, it suffices to have T and T' no less than 100 and 10, respectively, for FMF to converge.

Convergence of FMF. To analyze the theoretical convergence ratio of FMF, the first condition is that the sampled gradients are unbiased to the true gradients with bounded variance. We further assume that 1) the true local gradients are bounded, 2) the global loss function is smooth and 3) the initial loss is bounded by a constant D. These assumptions are practical given a dataset and a proper initialization of the embeddings.

Theorem 2. *If the aforementioned assumptions hold, Algorithm* FMF *converges in* $\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\, \|\nabla \mathcal{L}\left(\mathbf{X}, U, V\right)\| \,\right] = O\left(\sqrt{\frac{D}{T}(1+c)}\right)$ *with fixed step size* $\gamma = O\left(\frac{\sqrt{D}}{T'\sqrt{T}}\right)$, *and* $c = \sum_{k=1}^{s} \frac{m_k^2}{m^2}$ *in VFL,* $c = \sum_{k=1}^{s} \frac{n_k^2}{n^2}$ *in HFL and* $c = \frac{1}{n}$ *for LFL.*

4 PRIVATE VFL MATRIX FACTORIZATION

Although the parties in FMF algorithm do not share the local rating matrices directly with others, the shared embeddings still carry users' recoverable private information. The VFL privacy leakage risks and expected protections are formalized as the following.

VFL privacy leakage and protection. In this setting, as shown in Figure 1(a), we assume that users trust the party as long as their data are handled properly later on, so there is no need to protect the privacy of the information exchange between users and parties.

```
Input: $\{X^{(1)}, \ldots, X^{(s)}\}$ owned by $s$ different parties; the total iteration $T$ and
    local update iterations $T'$ .
Output: User embeddings and item embeddings
    [Stage 1]: initialization and local pre-computation
    (s1) Server: Initialize {shared embeddings}, share to all parties;
    (c1) All parties $k \in [s]$: Initialize {local embeddings};
    [Stage 2]: cooperative learning
    while $t \in [T]$ do
        (c2) All parties $k \in [s]$: Update local version of {shared embeddings} and
        {local embeddings} for $T'$ local iterations; upload {shared embeddings};
        (s2) Server: Aggregate and weighted average the {shared embeddings},
        and share the average to all parties.
    end while
    [Stage 3] local fine-tuning
    (c3) All parties $k \in [s]$: Pass.
```

(a) Vanilla FMF

```
Input: $\{X_{\mathbb{F}_1}, \ldots, X_{\mathbb{F}_s}\}$ owned by $s$ different parties; the total iteration $T$ and
    local update iterations $T'$ .
Output: Private user embeddings and private item embeddings
Define: shared embeddings = $U$; local embeddings = $V_{\mathbb{F}_1}, \ldots V_{\mathbb{F}_s}$
Replace (c2) in FMF: Update $U^{(k)}$ and $V_{\mathbb{F}_k}$ for $T'$ local iterations with em-
    bedding clipping and DP noise; upload $U^{(k)}$ to server;
Replace (c3) in FMF: Fine-tune $V_{\mathbb{F}_k}$ locally for $\kappa$ iterations with embedding
    clipping and DP noise;
```

(b) Changes FMF → VFL-SGDMF

```
Input: $\{X_{\mathbb{U}_1}, \ldots, X_{\mathbb{U}_s}\}$ owned by $s$ different parties; the total iteration $T$ and
    local update iterations $T'$ .
Output: Each party has its own version user/item embeddings
Define: shared embeddings = $V$; local embeddings = $U_{\mathbb{U}_1}, \ldots U_{\mathbb{U}_s}$
Replace (c1) in FMF: Initialize and pre-train $U_{\mathbb{U}_k}$;
Replace (c2) in FMF: Update $V^{(k)}$ for $T'$ local iterations with embedding
    clipping and DP noise; upload $V^{(k)}$ to server;
Replace (c3) in FMF: Fine-tune $U_{\mathbb{U}_k}$ and $V^{(k)}$ locally;
```

(c) Changes FMF → HFL-SGDMF

```
Input: $\{X_1, \ldots, X_n\}$ owned by $n$ different users; the total iteration $T$.
Output: Shared item embeddings $V$
Define: shared embeddings = $V$; local embeddings = $U_1, \ldots, U_n$
Replace (c1) in FMF: All parties $i \in [n]$: Initialize and pretrain $U_i$;
Replace (c2) in FMF: (Round 1) Calculate noise update $\tilde{g}_V^{(i,t)}$ and invoke SA$_\omega$-
    report($\tilde{g}_V^{(i,t)}$); (Round 2) If receive $|\mathbb{U}^{(t,1)}|$ from server, upload $\bar{\xi}^{(i,t)} =$
    $-\xi^{(i,t)} + \xi'^{(i,t)}$;
Replace (s2) in FMF: (Round 1) Invoke SA$_\omega$-agg to get $\tilde{g}_V^{(t)} = \sum_{i \in \mathbb{U}^{(t,1)}} \tilde{g}_V^{(i,t)}$;
    if $|\mathbb{U}^{(t,1)}| \leq (1-\omega)n$ then next iteration; Send user $i \in \mathbb{U}^{(t,1)}$ an integer
    $|\mathbb{U}^{(t,1)}|$; (Round 2) Update $V^{(t)} = V^{(t-1)} - \frac{\gamma_t}{|\mathbb{U}^{(t,1)}|}\left(\tilde{g}_V^{(t)} + \sum_{i \in [\mathbb{U}^{(t,2)}]} \bar{\xi}_i\right)$;
Replace (c3) in FMF: Fine-tune local $U_i$;
```

(d) Changes FMF → LFL-SGDMF

Figure 3: Federated matrix factorization algorithms

During the learning process of VFL, the exchanged intermediate results, such as locally updated embeddings, may leak users' private information to other parties and the server. A naive privacy leakage is that when a party never updates a user embedding, the coordination server can easily conclude that the user has no rating with that party. Besides, the learned user/item embeddings are used in public services (e.g., recommendation systems), so privacy protection needs to be provided for both user/item embeddings. Thus, privacy protect is expected for information exchange between parties and coordination server, and user/item embeddings in VFL.

4.1 Private MF in VFL: VFL-SGDMF

Compared with the non-private FMF, a few additional steps are required in [Stage 2] (s2) to change it into a differentially private mechanism, called VFL-SGDMF.

4.1.1 *Bounding sensitivity.*

In order to bound the sensitivity of the loss function (1), [33] sets constraints on the inner product of u_i and v_j. Gradient clipping method in [1] can be for MF problem and [41] projects each element of the gradient to $[-1, 1]$. Unlike the previous works, we propose *embedding clipping*, which imposes hard constraints on the norms of user/item embeddings in order to bound the gradients from a single rating. Extending for per-user privacy, we also adapt the same strategy, *random trimming*, as [33] in addition to bound the user level sensitivity.

Embedding clipping. We notice that the scores in rating matrices are usually in a bounded range $[0, R]$ where R is some finite number (e.g., 5 or 10). This property motivates the non-negative matrix factorization [2, 46], in which the user embeddings and item embeddings are restricted to be non-negative. In this paper, we enforce stronger constraints that the norm of the user embeddings and item embeddings are also bounded. That is,

$$\mathcal{U} : \forall i \in [n], \|u_i\|_2^2 \leq R, u_i \geq 0, \quad \mathcal{V} : \forall j \in [m], \|v_j\|_2^2 \leq R, v_j \geq 0. \quad (6)$$

With constraints in (6) and Cauchy–Schwarz inequality, we have $\langle u_i, v_j \rangle \leq R$. The gradient with respect to u_i from the j-th item is $\nabla_{u_i} \mathcal{L}_{i,j}(U, V) = -2v_j \left(X_{ij} - \langle u_i, v_j \rangle\right)$ if X_{ij} is observed, otherwise is 0. For per-rating privacy, a pair of neighbouring dataset $\mathbf{X}$ and $\mathbf{X}'$ differ at index (i^*, j^*) with $X_{i^*j^*} \neq \perp$ and $X'_{i^*j^*} = \perp$. With u_i and v_j bounded as (6), then the sensitivity of the gradient to a user embedding is $\Delta_{2,u} = \left\|-2v_{j^*}\left(X_{i^*j^*} - \langle u_{i^*}, v_{j^*} \rangle\right) - 0\right\|_2 \leq 2R^{3/2}$. Similarly, we can obtain sensitivity $\Delta_{2,v} = 2R^{3/2}$ for $\nabla_{v_j} \mathcal{L}_{i,j}(U, V)$.

We denote the embedding clipping on user embeddings $\Pi_{\mathcal{U}}(U)$, a projection functions that first turns all negative entries of U into 0, then normalizes each row vector with $\frac{u_i}{\max\{1, \|u_i\|_2/\sqrt{R}\}}$ to satisfy (6); $\Pi_{\mathcal{V}}(V)$ performs the same projection for V to satisfy (6). We provide proof that this two-step operation indeed can project each row vector to the closest point in $\mathcal{U}$ or $\mathcal{V}$ in our full version.

Although the additional restrictions may affect user embeddings and item embeddings' expressiveness, it can provide regularization to prevent overfitting and provide a nice bounded-gradient property. While gradient clipping method clips the gradient from different records, then aggregates the gradient and updates the embeddings, embedding clipping is equivalent to first aggregating the gradients then clip. Both methods introduce bias in the update, but embedding clipping has advantages over the gradient clipping in the MF as shown in our experiments.

Random trimming. To bound the sensitivity in per-user privacy, each party trims their local dataset so that there are at most $\theta^{(k)}$

ratings per user remaining in the local rating matrix of party k, and turns the others to $\perp$.

4.1.2 Amplifying privacy with mini-batches. It has been proven in other literature that sampling can amplify privacy. Notice that per-user privacy has a stronger constraint on sampling to achieve privacy amplification compared with per-rating privacy. Because all ratings from the same user are considered as one record in per-user privacy, the sampling needs to be applied on user-level when per-user privacy is required; sampling on columns has no privacy amplification. That is, n' rows from the local (trimmed) rating matrix are sampled , and the sampling rate is $\eta = \frac{n'}{n}$. To be more consistent and easy to compare, we apply this user-level sampling under both the per-rating and per-user privacy.

4.1.3 Private local updates. To protect both intermediate and final results, both U and V need to be perturbed with random noise and projected back to the feasible domain as defined as (6). The locally updates on user embeddings and item embeddings becomes:

$$U^{(k)} = \Pi_{\mathcal{U}}\left(U^{(k)} - \gamma_t \tilde{\mathbf{g}}_U^{(k)}\right), V_{\mathbb{F}_k} = \Pi_{\mathcal{V}}\left(V_{\mathbb{F}_k} - \frac{\gamma_t m_k}{m} \tilde{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k)}\right), \tag{7}$$

where $\tilde{\mathbf{g}}_U^{(k)} = \mathbf{g}_U^{(k)} + \xi_U^{(k,t,t')}, \tilde{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k)} = \mathbf{g}_{V_{\mathbb{F}_k}}^{(k)} + \xi_{V_{\mathbb{F}_k}}^{(k,t,t')}$. The noise $\xi_U^{(k,t,t')}$ and $\xi_{V_{\mathbb{F}_k}}^{(k,t,t')}$ have the same shape as the gradients, and their elements are sampled independently from zero-mean Gaussian distribution for each local update, with variance depending on privacy definitions and privacy parameters. Details are discussed later.

4.1.4 `VFL-SGDMF` *local fine-tuning.* We notice that compared with the convergence of the non-private and fully centralized matrix factorization algorithm, the convergence of the `VFL-SGDMF` algorithm mainly suffers from two aspects: the DP noise and the non-convexity of the problem. In our experiments, the user embeddings change little in the final iterations, and the loss decreases very slow in the final iterations. Thus, we can add a local fine-tuning **(c3)** as the following: for the last κ iterations, user embeddings are fixed, only item embeddings are updated locally on each party. So each party accesses the local rating matrices $TT' + \kappa$ times in training.

The benefit of fixing the users embedding and training item embeddings locally is that the problem becomes a convex optimization problem, and the sensitivity of each local update is smaller as only item embeddings change in the final iterations. Turning the problem into a convex problem and the smaller noise for each update, we can obtain smaller and more stable losses in training.

4.2 Analysis of Privacy Guarantee

To analysis both per-rating and per-user privacy, quantifying privacy loss of a party is an intermediate step. So we use $(\epsilon^{(k)}, \delta^{(k)})$ to characterize the privacy loss of party k after participating in the VFL computation. Although these quantities are not our final goal, $(\epsilon^{(k)}, \delta^{(k)})$ can be understood as the privacy loss of party k when the other $s-1$ parties collude or controlled by an adversary.

- *Sensitivity of per-rating privacy.* When each party updates local U and $V_{\mathbb{F}_k}$ together in `VFL-SGDMF`, the ℓ_2 sensitivity of the gradient to $(U, V_{\mathbb{F}_k})$ will be $\Delta_{rating} = 2\sqrt{2}R^{3/2}$ if adding/removing one rating; in the final κ iterations where only local item embeddings are updated, the sensitivity of each iteration is $\Delta_{rating} = \Delta_{2,v} = 2R^{3/2}$.
- *Sensitivity for per-user privacy after trimming.* After trimming, for any pair of neighboring database $\mathbf{X}_{\mathbb{F}_k}$ and $\mathbf{X}'_{\mathbb{F}_k}$, we have a user i such that ratings $(\mathbf{X}_{\mathbb{F}_k})_{ij} = \perp$ for all items $j \in [\mathbb{F}_k]$ in $\mathbf{X}_{\mathbb{F}_k}$ while at most $\theta^{(k)}$ ratings $(\mathbf{X}'_{\mathbb{F}_k})_{ij} \neq \perp$. Thus, the per-user ℓ_2-sensitivity is $\Delta_{user} = \theta^{(k)} \Delta_{rating}$ on party k when updating U and V together, and $\Delta_{user} = \theta^{(k)} \Delta_{2,v}$ if only updating item embeddings.

Composition of privacy across iterations. The standard deviation of the Gaussian distribution, from which $\xi_U^{(k,t,t')}$ and $\xi_{V_{\mathbb{F}_k}}^{(k,t,t')}$ are sampled from, is determined by the sensitivity and the privacy budget. It can be written as $\sigma^{(k)} = z\Delta_{rating}$ for per-rating privacy or $\sigma^{(k)} = z\Delta_{user}$ for per-user privacy, where z is called *noise multiplier.* Based on moment accountants [1], the relation between noise multiplier z and privacy budget is as the following:

Proposition 2. *Let the sampling rate be η in each local iteration. After running $TT' + \kappa$ local iterations,* `VFL-SGDMF` *introduces at most $(\epsilon^{(k)}, \delta^{(k)})$ per-rating or per-user privacy loss for party k if the noise multiplier $z^2 = c_1 \frac{\eta^2 (TT'+\kappa) \ln 1/\delta^{(k)}}{(\epsilon^{(k)})^2}$ with some constant c_1.*

End-to-end privacy guarantees. The per-rating and per-user privacy of `VFL-SGDMF` can be bounded as the following.

- *Per-rating privacy composition in VFL.* Notice that the parallel composition property holds in the context of moment accountants as well. In per-rating setting, global privacy loss can be bounded with the following theorem derived based on the parallel composition:

Theorem 3 (Global per-rating privacy). *If the noise multiplier is set as above, then* `VFL-SGDMF` *is* $(\max_{k\in[s]} \epsilon^{(k)}, \max_{k\in[s]} \delta^{(k)})$ *per-rating differentially private globally.*

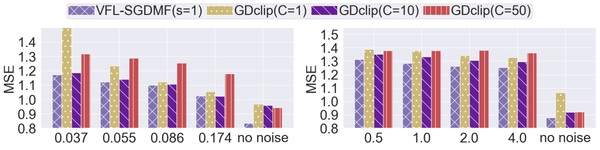
- *Per-user privacy composition in VFL.* Users may have ratings distributed across multiple parties, so the parallel composition property does not hold when composing per-user privacy loss. A naive composition gives $\epsilon = \sum_{k=1}^{s} \epsilon^{(k)}$. However, composing with moment accountants can give a tighter loss:

Theorem 4 (Global per-user privacy). *If for each party k has per-user privacy loss at most $(\epsilon^{(k)}, \delta^{(k)})$ and $\delta = \delta^{(1)} = \ldots = \delta^{(s)}$, then overall per-user privacy loss of*`VFL-SGDMF` *is* $(\sqrt{\sum_{k=1}^{s} (\epsilon^{(k)})^2}, \delta)$.

Notice that we only consider one-time training in this paper. The privacy budget and the hyper-parameters are predefined, so that privacy budget will not run out during the training. However, the real-world DP applications often provide a privacy guarantee within a given time period (i.e. a day) [43]. Following the same spirit, our algorithm can also be executed daily.

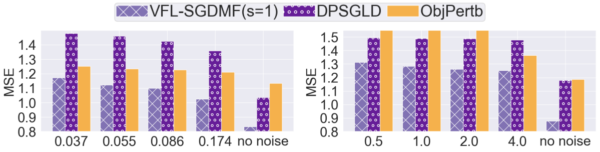
4.3 Empirical Evaluation for VFL

Datasets. We use three datasets in our experiments. The first one is the MovieLens 10M (ML10M) dataset [21], the second one is MovieLens 25M (ML25M) and the other one is the LibimSeTi [7] dataset. MovieLens 10M is a dataset with 10 million ratings on 10, 681 movies by 71, 567 users from the MovieLens website. MovieLens 25M dataset has 25 million ratings on 62,000 movies by 162,000 users. ML10M and ML25M are similar as their ratings are all between 0.5 to 5. LibimSeTi contains more than 17 million anonymous ratings of 168, 791 profiles made by 135, 359 LibimSeTi users, and



(a) Per-rating privacy **(b) Per-user privacy, $\theta = 10$**

Figure 4: Embedding clip v.s. gradient clip.



(a) Per-rating privacy **(b) Per-user privacy, $\theta = 10$**

Figure 5: Compare VFL-SGDMF (s=1), DP-SGLD and ObjPertb.

the scores in LibimSeTi are between 0 to 10. We also pre-process the LibimSeTi dataset with the same process as [41].

We split the datasets into s disjoint sub-datasets by movie/profiles, such that the ratings of each movie/profile can appear in only one dataset. Each of the sub-dataset simulates the local dataset for a party. We keep 10% of the ratings for testing in each local dataset and use the remaining for training. To measure the quality of the embeddings, we use the mean squared error (MSE) to measure the closeness between the inferences and the true ones in testing sets.

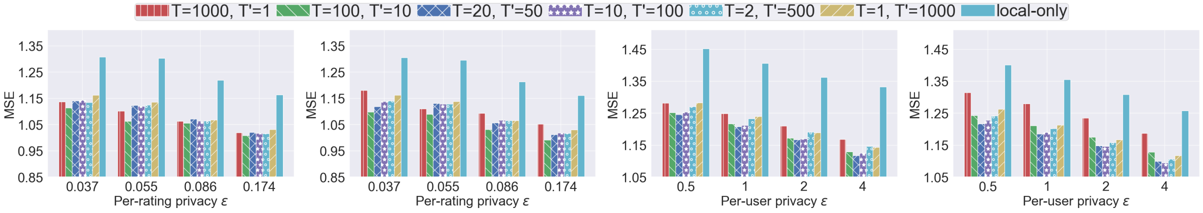
Because the existing DP MF methods are mainly in central setting, so we first compare [1, 24, 33] with VFL-SGDMF and set $s = 1$. To show the benefit of the communication in VFL-SGDMF, we also use *local-only* as a baseline. In the local-only method, each party trains locally with TT' iterations DP-SGD [1] to get local private user embeddings $U^{(k)}$ and private item embeddings $V_{\mathbb{F}_k}$, but they never communicate with each other during the training process.

Hyper-parameters. We fix $p = 20$ for the embeddings in all experiments. Although there are possible trade-offs between T, T', sampling rate and variance of Gaussian noise for a fixed ϵ, we fix $TT' = 1000$ (each party access local dataset 1000 times), sampling rate 0.01 and adjust the variance of Gaussian noise for different ϵ for the experiments in this paper. Also, the best trimming threshold $\theta^{(k)}$ for per-user privacy may vary for different numbers of parties, different datasets and different privacy budgets. For per-user privacy, we fix the trimming threshold $\theta = 10$ when $s = 1$, $\theta^{(k)} = 5$ when $s \in \{2, 5, 10\}$, and set $\theta^{(k)} = 2$ when $s = 18$. Because a too large $\theta^{(k)}$ requires large noise to protect privacy, a too-small $\theta^{(k)}$ abandons too much information. We tune the learning rates for different privacy budgets based on the training loss and pick the one giving the lowest training loss. The learning rates decrease linearly as [1] for the first 80% iterations and stay the same for the final 20% iterations. While we only report the results with the fixed setting mentioned above, there may be multiple optimal combinations. One can also tune the hyper-parameters with part of the privacy budget in a differentially private way as [1, 20].

Experiment results. We evaluate the VFL-SGDMF against gradient clipping and other methods [24, 33] with $s = 1$. We also analyze the effect of data partition and the effect of different parameters, i.e., synchronization frequencies and number of parties.

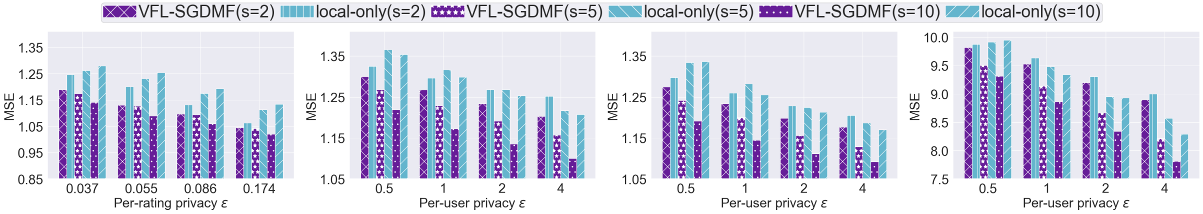
- *Non-private central (s=1) results.* For references, the embeddings trained in the non-private central setting with SGD have MSE of 0.83012, 0.8631, 4.23981 on testing sets of ML10M, ML25M and LibimSeTi datasets, respectively.
- *Comparing embedding clipping with gradient clipping.* To exclude the effect of trimming, we first show the per-rating setting and compare gradient clipping with embedding clipping in the central setting (s=1). We show in Figure 4(a) that if we only do the clipping but do not add noise, we observe that the gradient clipping method can easily overfit the training dataset so that the final results are bad, especially when the grading clipping threshold $C = 50$. With per-rating privacy, embedding clipping has MSEs slightly lower than the gradient clipping method with the best threshold $C = 10$. Gradient clipping has slightly worse performance because gradients have similar magnitudes and cancel out with each other after clipping. We can observe similar comparison result in Figure 4(b) for per-user privacy with trimming $\theta = 10$. The best clipping threshold is again $C = 10$, while embedding clipping still slightly outperforms it in different privacy levels. A lesson from these experiments is that embedding clipping is better than the gradient clipping approach regardless of the additional effort needed to pick the optimal C.
- *Comparing VFL-SGDMF with [24, 33].* In Figure 5, we compare our embedding clipping method with DP-SGLD in [33] and the objective perturbation (ObjPertb) approach in [24] with both per-rating and per-user privacy. We show that VFL-SGDMF can do better in both per-rating and per-user privacy. Compared with the DP-SGLD, our privacy composition is based on moment accountants, which shows a tighter composition of privacy loss so that smaller noise is required for each iteration. Besides, the DP-SGLD approach bounds the sensitivity by bounding the difference between predicted ratings and true ratings, introducing huge computation and communication overhead in the FL setting. The ObjPertb method performs poorly in our experiments, especially for per-user privacy. The main reason could be that the noise added in the objective function biases the gradients and the algorithm never has a chance to correct the errors.
- *Comparing different data partition.* We split the dataset ML10M in two different ways. 1) There are 18 categories of movies in ML10M, so we assume each party owns only one of those categories. When splitting by category, if a movie belongs to multiple categories, we assign it to the category with fewest movies to prevent the case that a category has too few movies. 2) We also randomly assign a movie to one of the 18 parties in a random split setting.

Comparing Figure 6(a) with Figure 6(b) for per-rating privacy, or comparing Figure 6(c) with Figure 6(d) for per-user privacy setting, with our VFL-SGDMF method and set $T = 100$ and $T' = 10$, there is no significant difference for per-rating privacy between those two split settings except for most cases; for per-user privacy, MSEs with random-split are slightly smaller than the split-by-category setting. The local-only method has no significant difference between the two splitting methods and per-rating privacy, but the MSEs are significantly higher with split-by-category than the one of random-split with per-user privacy. The results suggest that with appropriate private synchronization frequencies, VFL-SGDMF can learn the embeddings regardless of how items are distributed.



(a) Split by category, s=18 (b) Split randomly, s=18 (c) Split by category, s=18 (d) Split randomly, s=18

Figure 6: Per-rating and per-user privacy split by category v.s. split randomly on MovieLens 10M dataset.



(a) MovieLens 10M (per-rating) (b) MovieLens 10M (per-user) (c) MovieLens 25M (per-user) (d) LibimSeTi (per-user)

Figure 7: Results of randomly split items into different number of parties s.

- *Comparing different synchronization frequencies.* In Figure 6, we also compare the results of different T and T' but fixing the total local update iteration as $TT' = 1000$. When $T = 1000, T' = 1$, the algorithm becomes similar to the method aggregating private gradient in each iteration; when $T = 1, T' = 1000$, the algorithm becomes similar to model average as it only averages the local user embedding of different parties finally. Comparing different settings of T and T', we find that the setting $T = 100, T' = 10$ and $T = 20, T' = 50$ are usually the two with the best results. All settings of VFL-SGDMF have smaller MSE than the local-only method. The results indicate that frequent synchronization prevents the parties from learning good embeddings. It is because each local update step is noisy, and each party needs a few local steps to make meaningful progress; aggregating the noisy updates improves the quality of embeddings less than aggregating the one with meaningful progress.
- *Comparing the number of parties and the value of cooperation.* We compare our VFL-SGDMF to the *local-only* method with different numbers of parties ($s = 2, 5, 10$) and fix $T = 100, T' = 10$ for the experiments in Figure 7. In all levels of privacy guarantees and different numbers of parties, our VFL-SGDMF has smaller MSEs than the local-only methods. We observe that when the number of parties increases, the MSEs of our VFL-SGDMF algorithm decrease as the number of parties increases. In the per-rating privacy context, as the number of parties increases, the total number of local updates also increases, but the noise added to each update remains the same. Although as the number of parties increases in the per-user privacy, the noise in each local update increases, but the maximum number of ratings from the same user after trimming also increases as we fix $\theta^{(k)} = 5$, and the total number of local updates increase as well. These benefits may outweigh the increase of noise in our algorithm VFL-SGDMF. LibimSeTi MSEs are higher because the range of the LibimSeTi (0 to 10) is larger than the one of MovieLens (0.5 to 5).

5 PRIVATE HFL MATRIX FACTORIZATION

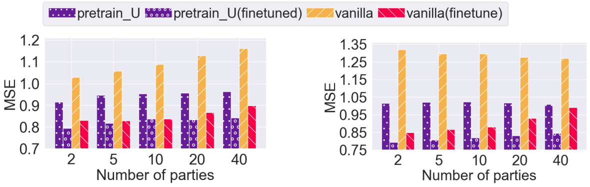
This section considers solving the MF problem in HFL setting. The use cases of HFL are different from the VFL setting, so we summarize different privacy risks and expected protection as following.

HFL privacy leakage and protection. The exchange of information under the HFL setting is described in Figure 1(b). Each party manages the ratings of a subset of users. The privacy risks are different for user/item embeddings in the HFL setting. Because each user has all his/her data stored on only one party and the embeddings are only used internally, the users have higher trust levels on the parties in our specification. Therefore, we assume that the users allow the parties to learn non-private embeddings locally as long as they are unpublished and only for internal usage. The main *privacy leakage risk* is in the communication between the coordinate server and the parties. Thus, our algorithm is designed to protect privacy for the communication between parties and the coordinate server, and limit the sensitive information learned by others.

5.1 Private MF in HFL: HFL-SGDMF

In order to provide the expected protection, we propose HFL-SGDMF with privacy guarantee. We denote $\tilde{g}_V^{(k)}$ as the noisy gradients with DP noise. We observed that when updating both user/item embeddings together and the updates of item embeddings are protected by DP noise, the user embeddings make limited progress in our experiments. It is because when the user embeddings are optimized together with noisy item embeddings, item embeddings updated noisily transmit the noise to user embeddings. We show how to convert FMF into HFL-SGDMF in Figure 3(c), and summarize as below.

- *[Stage 1]: initialization and local pre-computation.* Besides randomly initializing the embeddings in (1c), each party pre-trains their user embeddings locally with embedding clipping but without trimming and noise, and not updating item embeddings.



(a) per-rating privacy $\epsilon = 0.055$

(b) per-user privacy $\epsilon = 1.0$

Figure 8: Compare pre-train-U with vanilla train-both.

- *[Stage 2]: cooperative learning.* Same as the VFL-SGDMF, embedding clipping and random trimming are used to bound the privacy in the privacy context in (2c). Each party updates the local item embedding with differential privacy protection: $V^{(k)} = \Pi_{\mathcal{V}}\left(V^{(k)} - \gamma_t \tilde{g}_V^{(k)}\right)$, where $\tilde{g}_V^{(k)}$ is the privatized gradients of item embeddings from mini-batch with sampling rate $\eta = n'_k / n_k$ and fixed $U_{\mathbb{U}_k}$. The parties in this HFL process synchronize and averaging the *item embeddings* after every T' local iterations in Step (2s).
- *[Stage 3]: local fine-tuning.* After Stage 2, each party further fine-tunes their local version of user/item embeddings to get more accurate local inference results with untrimmed local data.

5.2 Analysis of Privacy Guarantee

In HFL-SGDMF, only the communication in the second stage and $\{V^{(k)} | k \in [s]\}$ are protected by differential privacy. The final local user/item embeddings after fine-tuned are unpublished.

DP sensitivity and necessary noise. Notice that since we only need to protect privacy on the exchange of item embeddings in the HFL stage, the sensitivity is $\Delta_{rating} = \Delta_{2,v}$ for per-rating privacy or $\Delta_{user} = \theta^{(k)} \Delta_{2,v}$ for per-user privacy for each local update.

Per-rating privacy and per-user privacy. The required standard deviation for per-rating or per-user privacy can be written in the same way as $\sigma^{(k)} = z\Delta_{rating}$ for per-rating privacy or $\sigma^{(k)} = z\Delta_{user}$ for per-user privacy with noise multiplier z. The parallel composition can be applied to both the per-rating and per-user privacy. So similar to the privacy analysis in Section 4.2, the following proposition can conclude the privacy loss in the HFL-SGDMF.

Proposition 3 (HFL-SGDMF privacy). HFL-SGDMF *ensures that the shared information from party k satisfies* $(\epsilon^{(k)}, \delta^{(k)})$ *per-rating/ per-user privacy , if the noise in each local update is sampled from Gaussian distribution with zero mean and noise multiplier* $z^2 = c_2 \frac{\eta^2 T T' \ln 1/\delta^{(k)}}{(\epsilon^{(k)})^2}$ *with some constant* c_2. *The overall* HFL-SGDMF *is* $(\max_{k\in[s]}\left\{\epsilon^{(k)}\right\}, \max_{k\in[s]}\left\{\delta^{(k)}\right\})$ *per-rating/per-user private.*

5.3 Empirical Evaluation for HFL

We use the same two datasets as in the VFL setting, MovieLens 10M and LibimSeTi. We horizontally partition these datasets into s subsets randomly to simulate the HFL setting in our experiments.

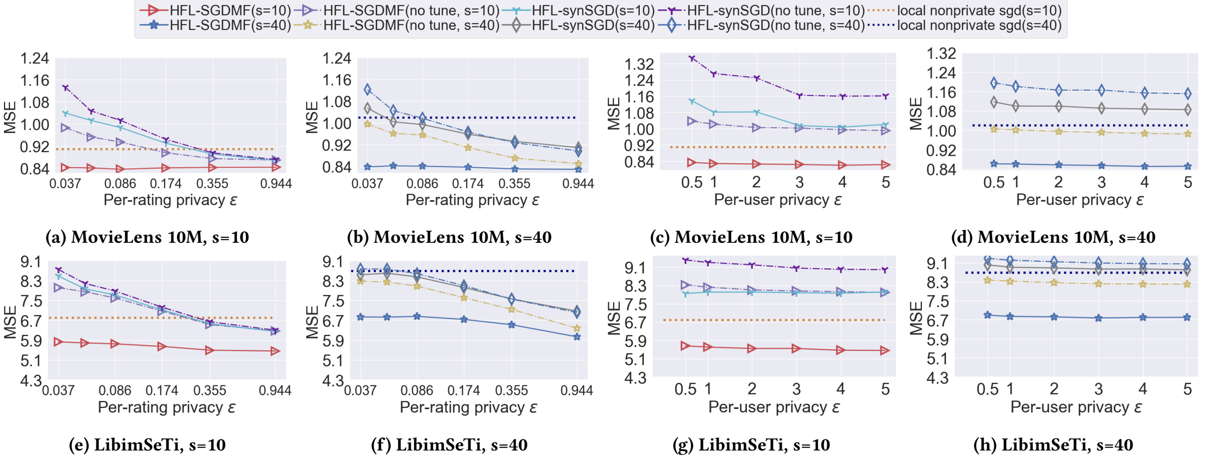
- *HFL Baselines.* For the HFL setting, we compare HFL-SGDMF with two baselines: one is non-private local training with SGD; the other one is the adaptation of DPSGD in HFL setting, called HFL-synSGD, where the noisy gradients $\tilde{g}_V^{(k)}$ are aggregated every iteration.
- *Hyper-parameters.* For HFL-SGDMF, we set $T = 100$ and $T' = 10$ as the previous section, which means each party queries the local dataset 1000 times during the HFL on item embeddings stage. To make it a fair comparison, we set the number synchronization iteration in HFL-synSGD to be 1000, and we let HFL-SGDMF and HFL-synSGD have the same number of iteration in the fine-tuning stage as well. For per-user privacy, we set $\theta^{(k)} = 10$ for all experiments with both HFL-synSGD and HFL-SGDMF and for all datasets. We set $\theta^{(k)}$ larger than the one in the VFL setting because all ratings of a user are stored on one party in the HFL setting. The learning rates in the experiments are tuned in the same way as in Section 4.3.

Empirical results. We show the experiments how the pretrain-U approach outperform the train-both approach, and compare our HFL-SGDMF with the HFL-synSGD for $s = 10$ and 40.

- *Improvement with pre-train U.* In Figure 8, we provide experiments comparing the pre-train-U approach and the vanilla train-both (user/item embeddings together) approach with numbers of parties from 2 to 40, with or without fine-tuning. The results show that pre-train U can significantly improve the prediction quality compared to the vanilla approach training user and item embedding in the same iteration. The main intuition is that if we update the user embedding and the item embedding in the same iteration, the user embeddings are largely affected by the noisy item embedding. The user embedding updates do not make much progress because the item embeddings oscillate with the DP noise, and the progress made on the user embeddings may be canceled out in the next iteration. However, when the user embeddings are pre-trained, the item embeddings are fixed so that every update in the pre-train phase can very likely improve the quality of user embeddings. In the cooperative learning stage, fixing the user embedding can convert the problem into a convex problem, so that item embeddings' privacy preserved learning process is easier.
- *Comparing with gradient-average.* The horizontal dotted lines in Figure 9 are the MSE of training locally and non-privately with SGD. We show that after fine-tuning, the final embeddings given by HFL-SGDMF have lower MSEs than local non-private training in most cases. It means that each party can benefit from the privacy-preserving HFL process with our HFL-SGDMF. When the number of parties becomes 40, the MSEs of HFL-SGDMF before the fine-tuning stage are already lower than the non-private local training ones in both ML10M and LibimSeTi datasets, which means the differentially private item embeddings have better utility than the ones trained non-privately only from local data when $s = 40$. Figure 9 also compares the results of HFL-synSGD. It shows that the HFL-synSGD gives higher MSE in all settings, no matter before or after fine-tuning. Thus, the experiments show that our HFL-SGDMF, in which user embeddings and item embedding are updated separately in the first two stages, can provide better utility compared to the HFL-synSGD, while has smaller communication cost.

6 TO CROSS-DEVICE LEARNING: LFL-SGDMF

In the previous section, we focused on the HFL setting, which can be categorized as the cross-silo setting in FL. The HFL setting can be naturally extended from to cross-device setting with the number of parties the same as number of users, $s = n$. That is, each party in the LFL setting is just a proxy of a user holding one user's ratings.



(a) MovieLens 10M, s=10

(b) MovieLens 10M, s=40

(c) MovieLens 10M, s=10

(d) MovieLens 10M, s=40

(e) LibimSeTi, s=10

(f) LibimSeTi, s=40

(g) LibimSeTi, s=10

(h) LibimSeTi, s=40

Figure 9: Horizontal random split with different number of parties (per-rating: left two columns; per-user: right two columns).

LFL privacy leakage and protection. As analyzed in Section 1.1, the user embeddings are trained and used only locally, so there is no risk of privacy leakage. *LFL privacy leakage* happens in the information exchange with the coordination server about the item embeddings training. A coordination server controls the item embeddings, so all the users share the same item embeddings. Although the coordination server has no access to the user data directly, the updates of item embeddings can reveal users' sensitive information. Thus, we need to ensure the shared updates of item embeddings are protected by privacy in our LFL setting.

- *MF with LDP and its limitation.* Compared with the centralized privacy setting, LDP considers a strictly stronger privacy-preserved model such that any output shared by a user device should be about as likely regardless of the actual user data. With LDP protocols, user data are randomized on local devices before being sent to the data aggregator. However, given the large number of items and the sparsity of the ratings, hiding which items are rated and protecting the exact ratings require large noise. An LDP method for the MF problem was proposed in [41], Private-GD-DR, which uses an LDP mean-estimation protocol and a dimension reduction technique. However, their algorithm can provide limited improvement on embeddings as shown in Section 6.3.
- *Secure aggregation and its limitation.* Secure aggregation [5, 6] was proposed to aggregate numerical data and reveals only the aggregated sum of the user updates to the server. Because user dropout is unavoidable in the LFL setting, the SA protocols in [5, 6] are designed to tolerate at most ω fraction of dropout users during the execution. An individual user's input is protected by composition of masks, but after summing up all reports, the symmetric masks are canceled out if less than ωn users drop out, and only the true sum remains. We denote the user reporting process as SA$_{\omega}$-report, and the server aggregation process as SA$_{\omega}$-agg. However, secure aggregation protocols are vulnerable to membership attacks or re-identification attacks.

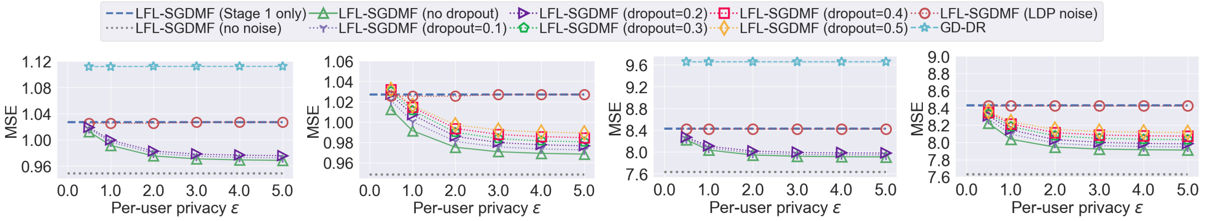
6.1 DP with Secure Aggregation

Different from the HFL setting, user devices in LFL may become unavailable and/or the communication between user devices and the central party may be disconnected from time to time – a user is said to *drop out* if either of the two cases happens. For example, if a user device is in an area where the internet connection is not stable, its update may be lost; or a user device can be busy with local tasks and refuses to be involved in the computation. So we need to ensure that the privacy guarantees of algorithms in the LFL setting hold even when a significant amount of users drop out in one or multiple iterations during the federated training process.

To overcome the limitations of both LDP and secure aggregation, we propose a new algorithm with the central DP guarantee, LFL-SGDMF, which is adapted from the FMF as shown in Figure 3(d). The changes from FMF in [Stage 1] initialization and local pre-computation and [Stage 3] local fine-tuning are similar to HFL-SGDMF, where the user embeddings are pre-trained after initialization, and the item embeddings are fixed. Somewhat surprisingly, only pre-training U can already outperform the reported results in [41] with LDP as shown in our experiments in Section 6.3.

The main changes are in the steps of [Stage 2] cooperative learning. The basic idea here is to aggregate the update of item embeddings through SA but ensure the aggregated gradients satisfy DP, while introducing noise as small as possible. If we consider SA as an oracle, there are two rounds of communication in each iteration.

- *Round 1 coordination pattern.* In the first round of communication, the coordination server initiates the SA$_{\omega}$ and request update from each party. After receiving the requests from the server, each party i samples one rating, $\mathbf{X}_{ij}$ and reports noisy update is $\tilde{\mathbf{g}}_V^{(i,t)} = \nabla_V \mathcal{L}_{i,j}\left(u_i, V^{(t)}\right) + \boldsymbol{\xi}^{(i,t)}$. The elements of noise $\boldsymbol{\xi}^{(i,t)}$ are sampled from Gaussian distribution $(0, \zeta_1 \sigma_v^2)$. Then the party invokes the oracle SA$_{\omega}$- report($\tilde{\mathbf{g}}_V^{(i,t)}$). The server aggregates the reported updates by invoking SA$_{\omega}$-agg. We denote the set of survivors in the first round as $\mathbb{U}^{(t,1)}$. SA$_{\omega}$ is designed with a tolerable dropout rate ω. If less than $(1-\omega)n$ users survive in this iteration, the SA$_{\omega}$ protocol halts, and the server learns nothing in this iteration and execute next iteration. Otherwise, the aggregated gradients decoded from SA$_{\omega}$ can be decomposed as $\tilde{\mathbf{g}}_V^{(t)} = \sum_{i \in \mathbb{U}^{(t,1)}} \nabla_V \mathcal{L}_{i,j}\left(u_i, V^{(t)}\right) + \sum_{i \in \mathbb{U}^{(t,1)}} \xi^{(i,t)}$.



(a) MovieLens 10M, $\omega = 0.3$ **(b) MovieLens 10M, $\omega = 0.6$** **(c) LibimSeTi, $\omega = 0.3$** **(d) LibimSeTi, $\omega = 0.6$**

Figure 10: Results under LFL setting with different dropout rates.

• *Round 2 coordination pattern.* The coordination server first broadcasts $|\mathbb{U}^{(t,1)}|$ to the survivors and requests replacing noise from them. After notified by the coordination server, the survive party i sends $\bar{\xi}^{(i,t)} = -\xi^{(i,t)} + \xi'^{(i,t)}$ to the server, where the elements of $\xi'^{(i,t)}$ are sampled from Gaussian distribution $(0, \zeta_2 \sigma_v^2)$. Dropout can also happen in Round 2, so we denote the set of survivors as $\mathbb{U}^{(t,2)}$. The server aggregates the replacing noise and the aggregated noisy gradients after this two-round exchange are decomposed as:

$$\tilde{g}_V^{(t)} = \sum_{i \in \mathbb{U}_t} \nabla_V \mathcal{L}_{i,j}\left(u_i, V^{(t)}\right) + \sum_{i \in \mathbb{U}^{(t,1)} - \mathbb{U}^{(t,2)}} \xi^{(i,t)} + \sum_{i \in \mathbb{U}^{(t,2)}} \xi'^{(i,t)}$$

With such information, the coordination server updates the item embeddings and broadcasts the updated embedding to all parties.

• *Communication cost.* The first round aggregation in every iteration depends on the SA oracle, which requires $O(\log n + mp)$ for each user, and it dominate the cost for each iteration. So the total communication cost of LFL-SGDMF is $O(T(\log n + mp))$ per user.

6.2 Analysis of Privacy Guarantee

There are three parameters directly affecting the privacy privacy guarantee, σ_v^2, ζ_1 and ζ_2. σ_v^2 is the nob for (ϵ, δ)-DP guarantee when there is no dropout. When SA_ω finishes successfully, the sum is composed of at least $|\mathbb{U}^{(t,1)}| \geq (1-\omega)n$ users and ζ_1 makes sure that the aggregated sum obtained by the server through SA_ω should be at least (ϵ, δ)-DP. Another parameter ζ_2 is used to reduce "unnecessary" noise added in the SA_ω round. By setting σ_v^2, ζ_1 and ζ_2 properly, LFL-SGDMF has the following privacy guarantee.

Theorem 5. *If $\sigma_v^2 = c_3 (\Delta_{2,v})^2 \frac{T_v \ln 1/\delta}{\epsilon^2}$, $\zeta_1 = \frac{1}{(1-\omega)n}$ and $\zeta_2 = \frac{1}{|\mathbb{U}^{(t,1)}|}$ and a constant c_3, the* LFL-SGDMF *can satisfy (ϵ, δ)-DP and tolerate at most ω fraction of user dropping out in each iteration.*

Whenever the serve can decode the aggregated sum with SA_ω, the aggregated sum contains noise with variance $\frac{|\mathbb{U}^{(t,1)}|}{(1-\omega)n}\sigma_v^2 \geq \sigma_v^2$. If there is a set $\mathbb{U}^{(t,2)}$ users survived in the second round, the final noise variance is $\left(\frac{|\mathbb{U}^{(t,2)}|}{|\mathbb{U}^{(t,1)}|} + \frac{|\mathbb{U}^{(t,1)}| - |\mathbb{U}^{(t,2)}|}{(1-\omega)n}\right)\sigma_v^2$. Notice that $\frac{|\mathbb{U}^{(t,2)}|}{|\mathbb{U}^{(t,1)}|} + \frac{|\mathbb{U}^{(t,1)}| - |\mathbb{U}^{(t,2)}|}{(1-\omega)n} \geq 1$, which means the noise is still sufficient to provide (ϵ, δ)-DP on the aggregated sum.

6.3 Empirical Evaluation for LFL

Our experiments in the LFL setting still use the two datasets: MovieLens 10M and LibimSeTi. We faithfully implement the GD-DR method in [41] for comparison. To make it a fair comparison, we amplify the privacy budget of LDP to the central DP with the best-known result [16]. We also use the stage 1 of LFL-SGDMF as a baseline to show the MSEs without learning item embeddings. Because communication between user devices and the central server is expensive in the LFL setting in LFL, we set $T = 10$ for both LFL-SGDMF and GD-DR, which is the same as [41].

• *Comparing with LDP methods.* In Figure 10, we first compare our LFL-SGDMF with the LDP algorithm, GD-DR, from [41]. The main observation is that LFL-SGDMF can outperform the GD-DR method even with the local training part (Stage 1) only. It means pre-training the local user embeddings can significantly improve the quality of predictions. We also conduct experiments applying the LDP level noise with LFL-SGDMF. The figures show that with such large noise, cooperation can not help improve the result. Although our proposed method, LFL-SGDMF, has a higher communication cost than the GD-DR, it shows significant improvement over the GD-DR because of the adaptive noise with central DP guarantee.

• *Comparing different dropout rates.* Figure 10 also shows the empirical evaluation of the two datasets. We set the tolerable dropout rate ω to be either 0.3 or 0.6. We vary the actual dropout rate from 0 to 0.2 when $\omega = 0.3$ and we also show additional results of dropout rate 0.3, 0.4, 0.5 when $\omega = 0.6$. From the result, as the dropout rate increases, the MSE also increases a little.

When comparing the results with different ω, our method with different ω gives almost the same MSEs for different epsilons when there is no dropout. When there are dropout users, the MSEs of $\omega = 0.6$ are higher than those of $\omega = 0.3$. This is because with the same dropout rate, a similar number of users drop out in the second rounds, the higher ω means larger noise remains.

7 CONCLUSION

This paper comprehensively investigates and provides solutions for the MF problem under three different FL settings, namely, VFL, HFL, and LFL, with provable privacy guarantees, based on the common algorithmic framework FMF. We demonstrate the utility of our proposed methods with extensive experiments. Challenging future tasks can be generalizing the algorithms to solve the problem with continuous updating ratings with privacy guarantees, and providing solutions to handle the new coming users and items.

ACKNOWLEDGMENTS

This work is supported in part by the United States National Science Foundation under Grant No. 1931443.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 308–318. https://doi.org/10.1145/2976749.2978318

[2] Charu C Aggarwal et al. 2016. *Recommender systems*. Vol. 1. Springer, New York, USA.

[3] Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. 2019. Differential privacy has disparate impact on model accuracy. *Advances in Neural Information Processing Systems* 32 (2019), 15479–15488.

[4] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, IEEE, New York, NY, USA, 1–6. https://doi.org/10.1109/MLSP.2016.7738886

[5] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1253–1269. https://doi.org/10.1145/3372297.3417885

[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1175–1191. https://doi.org/10.1145/3133956.3133982

[7] Lukas Brozovsky and Vaclav Petricek. 2007. Recommender System for Online Dating Service.

[8] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2021. Secure Federated Matrix Factorization. *IEEE Intelligent Systems* 36, 5 (2021), 11–20. https://doi.org/10.1109/MIS.2020.3014880

[9] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. 2020. VAFL: a Method of Vertical Asynchronous Federated Learning. arXiv:2007.06081 [cs.LG]

[10] Xiangyi Chen, Steven Z. Wu, and Mingyi Hong. 2020. Understanding Gradient Clipping in Private SGD: A Geometric Perspective. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., Red Hook, NY, USA, 13773–13782. https://proceedings.neurips.cc/paper/2020/file/9ecff5455677b38d19f49ce658ef0608-Paper.pdf

[11] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 3574–3583.

[12] Irit Dinur and Kobbi Nissim. 2003. Revealing Information While Preserving Privacy. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (San Diego, California) *(PODS '03)*. Association for Computing Machinery, New York, NY, USA, 202–210. https://doi.org/10.1145/773153.773173

[13] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, Shai Halevi and Tal Rabin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 265–284.

[14] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. 2017. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application* 4 (2017), 61–84.

[15] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14)*. Association for Computing Machinery, New York, NY, USA, 1054–1067. https://doi.org/10.1145/2660267.2660348

[16] Vitaly Feldman, Audra McMillan, and Kunal Talwar. 2021. Hiding Among the Clones: A Simple and Nearly Optimal Analysis of Privacy Amplification by Shuffling. arXiv:2012.12803 [cs.LG]

[17] Adrian Flanagan, Were Oyomno, Alexander Grigorievskiy, Kuan E. Tan, Suleiman A. Khan, and Muhammad Ammad-Ud-Din. 2021. Federated Multi-view Matrix Factorization for Personalized Recommendations. In *Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, Cham, 324–347. https://doi.org/10.1007/978-3-030-67661-2_20

[18] Rainer Gemulla, Erik Nijkamp, Peter J. Haas, and Yannis Sismanis. 2011. Large-Scale Matrix Factorization with Distributed Stochastic Gradient Descent. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Diego, California, USA) *(KDD '11)*. Association for Computing Machinery, New York, NY, USA, 69–77. https://doi.org/10.1145/2020408.2020426

[19] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. 2021. Shuffled Model of Differential Privacy in Federated Learning. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Arindam Banerjee and Kenji Fukumizu (Eds.), Vol. 130. PMLR, USA, 2521–2529. https://proceedings.mlr.press/v130/girgis21a.html

[20] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. 2010. *Differentially private combinatorial optimization*. SIAM, Philadelphia, PA, USA, 1106–1125. https://doi.org/10.1137/1.9781611973075.90 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611973075.90

[21] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (dec 2015), 19 pages. https://doi.org/10.1145/2827872

[22] Yaochen Hu, Peng Liu, Linglong Kong, and Di Niu. 2019. Learning Privately over Distributed Features: An ADMM Sharing Approach. arXiv:1907.07735 [cs.LG]

[23] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. 2019. FDML: A Collaborative Machine Learning Framework for Distributed Features. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 2232–2240. https://doi.org/10.1145/3292500.3330765

[24] Jingyu Hua, Chang Xia, and Sheng Zhong. 2015. Differentially private matrix factorization. In *Proceedings of the 24th International Conference on Artificial Intelligence* (Buenos Aires, Argentina) *(IJCAI'15)*. AAAI Press, USA, 1763–1770.

[25] Prateek Jain, Om Dipakbhai Thakkar, and Abhradeep Thakurta. 2018. Differentially Private Matrix Completion Revisited. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, USA, 2215–2224. https://proceedings.mlr.press/v80/jain18b.html

[26] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and Open Problems in Federated Learning. arXiv:1912.04977 [cs.LG]

[27] Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang. 2017. Federated tensor factorization for computational phenotyping. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, New York, NY, USA, 887–895.

[28] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. arXiv:1610.02527 [cs.LG]

[29] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2017. Federated Learning: Strategies for Improving Communication Efficiency. arXiv:1610.05492 [cs.LG]

[30] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[31] Ninghui Li, Wahbeh Qardaji, and Dong Su. 2012. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. Association for Computing Machinery, New York, NY, USA, 32–33.

[32] Yang Liu, Yingting Liu, Zhijie Liu, Yuxuan Liang, Chuishi Meng, Junbo Zhang, and Yu Zheng. 2020. Federated forest. *IEEE Transactions on Big Data* 01 (2020), 1–1. https://doi.org/10.1109/TBDATA.2020.2992755

[33] Ziqi Liu, Yu-Xiang Wang, and Alexander Smola. 2015. Fast Differentially Private Matrix Factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems* (Vienna, Austria) *(RecSys '15)*. Association for Computing Machinery, New York, NY, USA, 171–178. https://doi.org/10.1145/2792838.2800191

[34] Jing Ma, Qiuchen Zhang, Jian Lou, Joyce C. Ho, Li Xiong, and Xiaoqian Jiang. 2019. Privacy-Preserving Tensor Factorization for Collaborative Health Data Analysis. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1291–1300. https://doi.org/10.1145/3357384.3357878

[35] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Aarti Singh and Jerry Zhu (Eds.), Vol. 54. PMLR, USA, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

[36] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *International Conference on Learning Representations*. OpenReview.net. https://openreview.net/forum?id=BJ0hF1Z0b

[37] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. Association for Computing Machinery, New York, NY, USA, 19–30.

[38] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, New York, NY, USA, 691–706. https://doi.org/10.1109/SP.2019.00029

[39] Arvind Narayanan and Vitaly Shmatikov. 2006. How to break anonymity of the netflix prize dataset.

[40] Yilin Shen and Hongxia Jin. 2014. Privacy-preserving personalized recommendation: An instance-based approach via differential privacy. In *2014 IEEE International Conference on Data Mining*. IEEE, New York, NY, USA, 540–549.

[41] Hyejin Shin, Sungwook Kim, Junbum Shin, and Xiaokui Xiao. 2018. Privacy enhanced matrix factorization for recommendation with local differential privacy. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1770–1782.

[42] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, New York, NY, USA, 245–248. https://doi.org/10.1109/GlobalSIP.2013.6736861

[43] Apple Differential Privacy Team. 2021. *Differential Privacy Overview - Apple*. Apple. inc. https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf

[44] TensorFlow-Privacy. 2021. TensorFlow Privacy. https://github.com/tensorflow/privacy. Accessed: 2021-12-01.

[45] Yu-Xiong Wang and Yu-Jin Zhang. 2012. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering* 25, 6 (2012), 1336–1353.

[46] Yu-Xiong Wang and Yu-Jin Zhang. 2012. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering* 25, 6 (2012), 1336–1353.

[47] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.

[48] Nan Wu, Farhad Farokhi, David Smith, and Mohamed Ali Kaafar. 2020. The value of collaboration in convex machine learning with differential privacy. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, New York, NY, USA, 304–317. https://doi.org/10.1109/SP40000.2020.00025

[49] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. 2020. Privacy Preserving Vertical Federated Learning for Tree-based Models. *Proceedings of the VLDB Endowment* 13, 11 (2020), 2090–2103.

[50] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

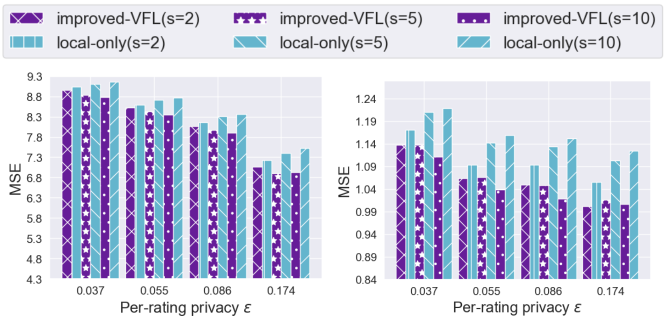
[51] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. 2006. Learning from incomplete ratings using non-negative matrix factorization. *Proceedings of the 2006 SIAM international conference on data mining* 1 (2006), 549–553.

[52] Handong Zhao, Zhengming Ding, and Yun Fu. 2017. Multi-view clustering via deep matrix factorization. *Proceedings of the AAAI Conference on Artificial Intelligence* 31, 1 (2017), –. https://ojs.aaai.org/index.php/AAAI/article/view/10867

[53] Fan Zhou and Guojing Cong. 2018. On the convergence properties of a K-step averaging stochastic gradient descent algorithm for nonconvex optimization. *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (2018), 3219–3227.

[54] Ligeng Zhu and Song Han. 2020. Deep leakage from gradients. *Federated Learning* 12500 (2020), 17–31.

A ADDITIONAL EXPERIMENTS



(a) LibimSeTi, per-rating privacy **(b) MovieLens 25M**

Figure 11: Additional results of randomly split items into different number of parties s.

B PROOF OF FMF CONVERGENCE

In this appendix, we provide the proof of FMF in the VFL setting. Notice that the proof can be extended to the HFL easily, exchange the user embeddings and item embeddings in the corresponding steps.

B.1 FMF convergence preliminaries

We focus on the convergence of $U \in \mathbb{R}^{n\times p}, V = \begin{bmatrix} V_{\mathbb{F}_1} \\ \vdots \\ V_{\mathbb{F}_s} \end{bmatrix} \in \mathbb{R}^{m\times p}$ and $m = \sum_{k=1}^{s} m_k$. To simplify the notation, we use $W = (U, V)$ in some place of the proof. Assume we know the global rating density $\tau = \frac{\sum_{k=1}^{s} |\Omega_k|}{nm}$. The loss function of our problem is

$$L(W) = \frac{1}{\sum_{k=1}^{s} |\Omega_k|} \sum_{k=1}^{s} \sum_{(i,j)\in\Omega_k} \left(\mathbf{X}_{ij} - \langle u_i, v_j \rangle\right)^2 = \frac{1}{nm\tau} \sum_{k=1}^{s} \sum_{(i,j)\in\Omega_k} \left(\mathbf{X}_{ij} - \langle u_i, v_j \rangle\right)^2 \tag{8}$$

$$\text{s.t.} \forall i \in [n], \quad \|u_i\|_2^2 \le a', \quad u_i \ge 0$$
$$\forall k \in [s], j \in [m], \quad \|v_j\|_2^2 \le b', \quad v_j \ge 0$$

We denote $\nabla_U \mathcal{L}_{i,j}\left(U, V_{\mathbb{F}_k}\right)$ (or, $\nabla_{V_{\mathbb{F}_k}} \mathcal{L}_{i,j}\left(U, V_{\mathbb{F}_k}\right)$) as the gradient for U (or, V) based on data on index (i, j) of party k in the computation.

$$\nabla_U \mathcal{L}_{i,j}\left(U, V_{\mathbb{F}_k}\right) = -2(\mathbf{X}_{ij} - \langle u_i, v_j \rangle)\mathbf{e}_i^n v_j \tag{9}$$

$$\nabla_{V_{\mathbb{F}_k}} \mathcal{L}_{i,j}\left(U, V_{\mathbb{F}_k}\right) = -2(\mathbf{X}_{ij} - \langle u_i, v_j \rangle)\mathbf{e}_j^{m_k} u_i \tag{10}$$

where $\mathbf{e}_i^n$ is the unit vector of size $n \times 1$ with 1 on i^{th} row and zeros elsewhere, similarly $\mathbf{e}_i^{m_k}$ is the unit vector of size $m_k \times 1$ with 1 on j^{th} row and zeros elsewhere.

All parties will synchronize U for T times. We take each synchronization as the last step of a meta iteration. In each meta iteration $t \in [T]$, each party updates U for T' times. In each local SGD iteration (t, t'), each party k samples a sub-matrix $\hat{\mathbf{X}}_{\mathbb{F}_k}^{(t,t')}$ of n'_k rows and m'_k columns, and observed ratings with indices I_k Assume that the sampling rate is $n'_k = q_n n_k$, $m'_k = q_m m_k$ and $\eta = q_n q_m$.

We denote normalized gradient for U base on the all local data of party k for the tth meta iteration as the following:

$$\bar{\mathbf{g}}_U^{(k,t)} = \nabla_U \mathcal{L}\left(\mathbf{X}_{\mathbb{F}_k}, U^{(k,t)}, V_{\mathbb{F}_k}^{(t)}\right) = \frac{1}{nm_k\tau} \sum_{(i,j)\in\Omega_k} \nabla_U \mathcal{L}_{i,j}\left(U^{(k,t)}, V_{\mathbb{F}_k}^{(t)}\right), \tag{11}$$

$$\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} = \nabla_{V_{\mathbb{F}_k}} \mathcal{L}\left(\mathbf{X}_{\mathbb{F}_k}, U^{(k,t)}, V_{\mathbb{F}_k}^{(t)}\right) = \frac{1}{nm_k\tau} \sum_{(i,j)\in\Omega_k} \nabla_{V_{\mathbb{F}_k}} \mathcal{L}_{i,j}\left(U^{(k,t)}, V_{\mathbb{F}_k}^{(t)}\right) \tag{12}$$

For the normalized gradients of party k in (t, t') iteration, we denoted them as $\bar{\mathbf{g}}_U^{(k,t,t')}$ and $\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}$. Globally, we denote $\bar{\mathbf{g}}_U$ (or, $\bar{\mathbf{g}}_V$) as the gradient to U (or, V) to average gradients (over all parties' data).

$$\bar{\mathbf{g}}_U^{(t)} = \frac{1}{nm\tau} \sum_{k=1}^{s} \sum_{(i,j)\in\Omega_k} \nabla_U \mathcal{L}_{i,j}\left(U^{(t)}, V_{\mathbb{F}_k}^{(t)}\right)$$

$$\bar{\mathbf{g}}_V^{(t)} = \frac{1}{nm\tau} \begin{bmatrix} \sum_{(i,j)\in\Omega_1} \nabla_{V_{\mathbb{F}_1}} \mathcal{L}_{i,j}\left(U^{(t)}, V_{\mathbb{F}_1}^{(t)}\right) \\ \vdots \\ \sum_{(i,j)\in\Omega_s} \nabla_{V_{\mathbb{F}_s}} \mathcal{L}_{i,j}\left(U^{(t)}, V_{\mathbb{F}_s}^{(t)}\right) \end{bmatrix}$$

We denote $\mathbf{g}_U^{(k,t,t')}$ (or, $\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}$) as the normalized gradient for U base on the subset of data in I_k of party k for (t, t') iteration.

$$\mathbf{g}_U^{(k,t,t')} = \frac{1}{n'm'_k\tau} \sum_{(i,j)\in I_k} \nabla_U \mathcal{L}_{i,j}\left(U^{(k,t,t')}, V_{\mathbb{F}_k}^{(t,t')}\right), \quad \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} = \frac{1}{n'm'_k\tau} \sum_{(i,j)\in I_k} \nabla_{V_{\mathbb{F}_k}} \mathcal{L}_{i,j}\left(U^{(k,t,t')}, V_{\mathbb{F}_k}^{(t,t')}\right) \tag{13}$$

The sampled gradient of U and V are unbiased estimate to the true gradient (with $I_k = \Omega_k$), that is

$$\mathbb{E}_{I_k}\left[\mathbf{g}_U^{(k,t,t')}\right] = \bar{\mathbf{g}}_U^{(k,t,t')}, \quad \mathbb{E}_I\left[\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right] = \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \tag{14}$$

$$\bar{\mathbf{g}}_U^{(t)} = \frac{1}{m} \sum_{k=1}^{s} m_k \bar{\mathbf{g}}_U^{(k,t)}, \quad \bar{\mathbf{g}}_V^{(t)} = \begin{bmatrix} \frac{m_1}{m} \bar{\mathbf{g}}_{V_{\mathbb{F}_1}}^{(k,t)} \\ \vdots \\ \frac{m_s}{m} \bar{\mathbf{g}}_{V_{\mathbb{F}_s}}^{(k,t)} \end{bmatrix} \tag{15}$$

Assume that the density of the observed ratings is the same for all sampled sub-matrix, then each party samples approximately $\eta n m_k \tau$ observed rating for each batch. The sampling process introduced bounded variance which will decrease as the size of I increases:

$$\mathbb{E}_{I_k}\left[\left\|\mathbf{g}_U^{(k,t,t')} - \bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] \leq \sigma^2_{\mathbb{S}_{\mathsf{Sub}},U} \tag{16}$$

$$\mathbb{E}_{I_k}\left[\left\|\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} - \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right] \leq \sigma^2_{\mathbb{S}_{\mathsf{Sub}},V} \tag{17}$$

The updating rule can be written as the following:

$$U^{(k,t,t')} = U^{(k,t,t'-1)} - \gamma \mathbf{g}_U^{(k,t,t')} \tag{18}$$

$$V_{\mathbb{F}_k}^{(t,t')} = V_{\mathbb{F}_k}^{(t,t'-1)} - \frac{\gamma m_k}{m} \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} \tag{19}$$

$$U^{(t+1)} = \frac{1}{m} \sum_{k=1}^{s} m_k U^{(k,t,T')} \tag{20}$$

$$V^{(t+1)} = \begin{bmatrix} V_{\mathbb{F}_1}^{(t,T')} \\ \vdots \\ V_{\mathbb{F}_s}^{(t,T')} \end{bmatrix} \tag{21}$$

$$\tag{22}$$

We assume that both $\mathbf{g}_U^{(k,t)}$ and $\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t)}$ are unbiased to $\bar{\mathbf{g}}_U^{(k,t)}$ and $\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}$; the sampling variance is bounded by $\sigma^2_{\mathbb{S}_{\mathsf{Sub}},U}$ and $\sigma^2_{\mathbb{S}_{\mathsf{Sub}},V}$. Both $\sigma^2_{\mathbb{S}_{\mathsf{Sub}},U}$ and $\sigma^2_{\mathbb{S}_{\mathsf{Sub}},V}$ goes to 0 as $\frac{n'm'_k}{nm_k} \longrightarrow 1$. To show the convergence, we further make the following assumptions: 1) local gradients are bounded $\forall k \in [s]$, $\left\|\bar{\mathbf{g}}_U^{(k,t)}\right\| \leq L_U$, $\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\| \leq L_V$; 2) the global loss function is β smooth; and 3) the initial loss is bounded as $\mathcal{L}\left(\mathbf{X}, U^{(0)}, V^{(0)}\right) - \mathcal{L}\left(\mathbf{X}, U^*, V^*\right) \leq D$. These assumptions usually hold in the MF problem, as the ratings are bounded.

To prove Theorem 2 in VFL case, we need to use the following two lemmas to bound the updates of U and V for each meta-iteration.

Lemma 1. *For Algorithm* FMF *in VFL case, with the same assumptions as Theorem 2 and assume the learning rate γ is unchanged within each meta-iteration, we have the following bound for the change of U after each meta-iteration*

$$-\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, U^{(t+1)} - U^{(t)} \right\rangle + \frac{\beta}{2}\left\| U^{(t+1)} - U^{(t)} \right\|_F^2 \right] \tag{23}$$

$$\leq -\gamma T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\|^2 + \gamma^2 \beta T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\| \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} \right\| \right] \tag{24}$$

$$+ \frac{\beta\gamma^2 T'^2 \sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2}{2} + \frac{\beta\gamma^2 T'}{2} \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} \right\|^2 \right] \tag{25}$$

Lemma 2. *For Algorithm* FMF *in VFL case, with the same assumptions as Theorem 2 and assume the learning rate γ is unchanged within each meta-iteration, we have the following bound for the change of V after each meta-iteration*

$$-\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_V^{(t)}, V^{(t+1)} - V^{(t)} \right\rangle + \frac{\beta}{2}\left\| V^{(t+1)} - V^{(t)} \right\|_F^2 \right] \tag{26}$$

$$\leq -\gamma T' \left\| \bar{\mathbf{g}}_V^{(t)} \right\|^2 + \sum_{k=1}^{s} \frac{T'\gamma^2 m_k^3 \beta}{m^3} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \mathbf{g}_{V_{\mathbb{F}_k}^{(k,t,t')}} \right\| \right] \tag{27}$$

$$+ \sum_{k=1}^{s} \frac{\beta\gamma^2 m_k^2 T'^2 \sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2}{2m^2} + \sum_{k=1}^{s} \frac{\beta T' m_k^2 \gamma^2}{2m^2} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \mathbf{g}_{V_{\mathbb{F}_k}^{(k,t,t')}} \right\|^2 \right] \tag{28}$$

B.2 Proof of Theorem 2 in VFL

Proof of Theorem 2. High level idea follows the proof in [53]. With smoothness condition,

$$\mathcal{L}\left(\mathbf{X}, W^{(t+1)}\right) - \mathcal{L}\left(\mathbf{X}, W^{(t)}\right) \tag{29}$$

$$\leq \left\langle \nabla_{W^{(t)}} \mathcal{L}\left(W^{(t)}\right), \begin{bmatrix} U^{(t+1)} - U^{(t)} \\ V^{(t+1)} - V^{(t)} \end{bmatrix} \right\rangle + \frac{\beta}{2} \left\| \begin{bmatrix} U^{(t+1)} - U^{(t)} \\ V^{(t+1)} - V^{(t)} \end{bmatrix} \right\|_F^2 \tag{30}$$

$$= \left\langle \bar{\mathbf{g}}_U^{(t)}, U^{(t+1)} - U^{(t)} \right\rangle + \frac{\beta}{2}\left\| U^{(t+1)} - U^{(t)} \right\|_F^2 \tag{31}$$

$$+ \left\langle \bar{\mathbf{g}}_V^{(t)}, V^{(t+1)} - V^{(t)} \right\rangle + \frac{\beta}{2}\left\| V^{(t+1)} - V^{(t)} \right\|_F^2 \tag{32}$$

By Lemma 1 and 2 :

$$\mathbb{E}\left[\mathcal{L}\left(\mathbf{X}, W^{(t+1)}\right) \right] - \mathcal{L}\left(\mathbf{X}, W^{(t)}\right) \tag{33}$$

$$\leq -\gamma T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\|^2 + \gamma^2 \beta T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\| \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} \right\| \right] \tag{34}$$

$$+ \frac{\beta\gamma^2 T'^2 \sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2}{2} + \frac{\beta\gamma^2 T'}{2} \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} \right\|^2 \right] \tag{35}$$

$$-\gamma T' \left\| \bar{\mathbf{g}}_V^{(t)} \right\|^2 + \sum_{k=1}^{s} \frac{T'\gamma^2 m_k^3 \beta}{m^3} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\| \right] \tag{36}$$

$$+ \sum_{k=1}^{s} \frac{\beta\gamma^2 m_k^2 T'^2 \sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2}{2m^2} + \sum_{k=1}^{s} \frac{\beta T' m_k^2 \gamma^2}{2m^2} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] \tag{37}$$

$$= -\gamma T' \left\| \bar{\mathbf{g}}_W^{(t)} \right\|^2 + \tag{38}$$

$$\gamma^2 \beta T' \left(\left\| \bar{\mathbf{g}}_U^{(t)} \right\| \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} \right\| \right] + \frac{T' \sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2}{2} + \frac{1}{2} \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} \right\|^2 \right] \right. \tag{39}$$

$$\left. + \sum_{k=1}^{s} \frac{m_k^3}{m^3} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\| \right] + \sum_{k=1}^{s} \frac{m_k^2 T' \sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2}{2m^2} + \sum_{k=1}^{s} \frac{m_k^2}{2m^2} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] \right) \tag{40}$$

With the assumption that $\|\bar{\mathbf{g}}_U\| \leq L_U$, $\|\bar{\mathbf{g}}_V\| \leq L_V$. We can have

$$\mathbb{E}\left[\mathcal{L}\left(\mathbf{X}, W^{(t+1)}\right)\right] - \mathcal{L}\left(\mathbf{X}, W^{(t)}\right) \tag{41}$$

$$\leq -\gamma T' \left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2 + \gamma^2 \beta T' \left((T'-1)L_U^2 + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2}{2} + \frac{T'L_U^2}{2}\right) \tag{42}$$

$$+ \gamma^2 \beta T' \left((T'-1)\sum_{k=1}^{s} \frac{m_k^3}{m^3} L_V^2 + \sum_{k=1}^{s} \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2 m_k^2}{2m^2} + \sum_{k=1}^{s} \frac{T'L_V^2 m_k^2}{2m^2}\right) \tag{43}$$

$$\leq -\gamma T' \left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2 + \gamma^2 \beta T' \left((T'-1)L_U^2 + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2}{2} + \frac{T'L_U^2}{2}\right) \tag{44}$$

$$+ \gamma^2 \beta T' \left((T'-1)L_V^2 \sum_{k=1}^{s} \frac{m_k^3}{m^3} + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2}{2} \sum_{k=1}^{s} \frac{m_k^2}{m^2} + \frac{T'L_V^2}{2} \sum_{k=1}^{s} \frac{m_k^2}{m^2}\right) \tag{45}$$

$$\leq -\gamma T' \left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2 + \gamma^2 \beta T' \left((T'-1)L_U^2 + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2}{2} + \frac{T'L_U^2}{2}\right) + \gamma^2 \beta T' \left((T'-1)L_V^2 \sum_{k=1}^{s} \frac{m_k^3}{m^3} + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2}{2} \sum_{k=1}^{s} \frac{m_k^2}{m^2} + \frac{T'L_V^2}{2} \sum_{k=1}^{s} \frac{m_k^2}{m^2}\right) \tag{46}$$

Let $\sum_{k=1}^{s} \frac{m_k^3}{m^3} = C_{V,1}$, $\sum_{k=1}^{s} \frac{m_k^2}{m^2} = C_{V,2}$. By definition, $\frac{1}{s^2} \leq C_{V,1} \leq 1$ and $\frac{1}{s} \leq C_{V,2} \leq 1$. So summing all the different from $t = 1$ to T, and have constant learning rate γ, we have

$$\mathbb{E}\left[\mathcal{L}\left(\mathbf{X}, W^{(T)}\right)\right] - \mathcal{L}\left(\mathbf{X}, W^{(1)}\right) \tag{47}$$

$$\leq -\gamma T' \sum_{t=1}^{T} \mathbb{E}\left[\left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2\right] + \gamma^2 \beta T' T \left((T'-1)L_U^2 + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2}{2} + \frac{T'L_U^2}{2}\right) \tag{48}$$

$$+ \gamma^2 \beta T' T \left((T'-1)L_V^2 C_{V,1} + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2 C_{V,2}}{2} + \frac{T'L_V^2 C_{V,2}}{2}\right), \tag{49}$$

$$\tag{50}$$

which implies

$$\gamma T' \sum_{t=1}^{T} \mathbb{E}\left[\left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2\right] \tag{51}$$

$$\leq \mathcal{L}\left(\mathbf{X}, W^{(1)}\right) - \mathbb{E}\left[\mathcal{L}\left(\mathbf{X}, W^{(T)}\right)\right] + \gamma^2 \beta T' T \left((T'-1)L_U^2 + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2}{2} + \frac{T'L_U^2}{2}\right) \tag{52}$$

$$+ \gamma^2 \beta T' T \left((T'-1)L_V^2 C_{V,1} + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2 C_{V,2}}{2} + \frac{T'L_V^2 C_{V,2}}{2}\right) \tag{53}$$

$$\tag{54}$$

Taking the average of LHS:

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2\right] \leq \frac{\mathcal{L}\left(\mathbf{X}, W^{(1)}\right) - \mathbb{E}\left[\mathcal{L}\left(\mathbf{X}, W^{(T)}\right)\right]}{\gamma T' T} + \gamma\beta \left((T'-1)L_U^2 + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2}{2} + \frac{T'L_U^2}{2}\right) \tag{55}$$

$$+ \gamma\beta \left((T'-1)L_V^2 C_{V,1} + \frac{T'\sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2 C_{V,2}}{2} + \frac{T'L_V^2 C_{V,2}}{2}\right) \tag{56}$$

Assume $\mathcal{L}\left(\mathbf{X}, W^{(1)}\right) - \mathcal{L}\left(\mathbf{X}, W^*\right) \leq D$. It is obvious that $C_{V,1} \leq C_{V,2}$. We set $\gamma = O\left(\sqrt{\frac{D}{\beta T (T')^2 (L_U^2 + L_V^2 C_{V,2} + \sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2 + \sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2 C_{V,2})}}\right)$, we can have

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2\right] = O\left(\frac{\sqrt{D\beta(L_U^2 + L_V^2 C_{V,2} + \sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2 + \sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2 C_{V,2})}}{\sqrt{T}}\right) \tag{57}$$

Because $L_U \approx L_V$ and $\sigma_{\mathbb{S}_{\mathsf{Sub}},U} \approx \sigma_{\mathbb{S}_{\mathsf{Sub}},V}$ are constants determined by the dataset, we can further simplify the result as

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2\right] = O\left(\frac{\sqrt{D\beta(L_U^2 + L_V^2 C_{V,2} + \sigma_{\mathbb{S}_{\mathsf{Sub}},U}^2 + \sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2 C_{V,2})}}{\sqrt{T}}\right) = O\left(\sqrt{\frac{D}{T}(1 + \sum_{k=1}^{s}\frac{m_k^2}{m^2})}\right) \tag{58}$$

The proof for HFL setting is very similar, by switching the user embeddings with item embedding in the derivations. The result will also be similar, the only different will be the last term.

□

B.3 Proof of Lemma 1

Proof of Lemma 1. If the inner SGD steps have constant step size, i.e. $\gamma_{t'} = \gamma$, for $t' = 1, \ldots, T'$. Also assume that all party use the same sampling rates, then

$$\left\langle \bar{\mathbf{g}}_U^{(t)}, U^{(t+1)} - U^{(t)} \right\rangle + \frac{\beta}{2}\left\|U^{(t+1)} - U^{(t)}\right\|_2^2 \tag{59}$$

$$= \frac{1}{m}\sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1}\left\langle \bar{\mathbf{g}}_U^{(t)}, \left(U^{(k,t,t'+1)} - U^{(k,t,t')}\right)\right\rangle + \frac{\beta}{2m^2}\left\|\sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1}\left(U^{(k,t,t'+1)} - U^{(k,t,t')}\right)\right\|^2 \tag{60}$$

$$= -\frac{\gamma}{m}\sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1}\left\langle \bar{\mathbf{g}}_U^{(t)}, \mathbf{g}_U^{(k,t,t')}\right\rangle + \frac{\beta\gamma^2}{2m^2}\left\|\sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1}\mathbf{g}_U^{(k,t,t')}\right\|^2 \tag{61}$$

By taking the expectation and the unbiased assumption, we have

$$\mathbb{E}\left[\mathbb{E}_{I_k}\left[\mathbf{g}_U^{(k,t,t')} | U^{(k,t,t')}\right]\right] = \mathbb{E}\left[\bar{\mathbf{g}}_U^{(k,t,t')}\right] \tag{62}$$

The expectation, (59) becomes

$$-\frac{\gamma}{m}\sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, \bar{\mathbf{g}}_U^{(k,t,t')}\right\rangle\right] + \frac{\beta\gamma^2}{2m^2}\mathbb{E}\left[\left\|\sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1}\mathbf{g}_U^{(k,t,t')}\right\|^2\right] \tag{63}$$

The inner product in first term of the above equation:

$$\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, \bar{\mathbf{g}}_U^{(k,t,t')}\right\rangle\right] \tag{64}$$

$$= \sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, \bar{\mathbf{g}}_U^{(k,t,0)} + \bar{\mathbf{g}}_U^{(k,t,t')} - \bar{\mathbf{g}}_U^{(k,t,0)}\right\rangle\right] \tag{65}$$

$$\geq \sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, \sum_{k=1}^{s}\frac{m_k}{m}\bar{\mathbf{g}}_U^{(k,t,0)}\right\rangle\right] + \sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, \bar{\mathbf{g}}_U^{(k,t,t')} - \bar{\mathbf{g}}_U^{(k,t,0)}\right\rangle\right] \tag{66}$$

$$\geq T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|^2 - \left\|\bar{\mathbf{g}}_U^{(t)}\right\|\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t=1}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')} - \bar{\mathbf{g}}_U^{(k,t,0)}\right\|\right] \tag{67}$$

$$\geq T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|^2 - \beta\left\|\bar{\mathbf{g}}_U^{(t)}\right\|\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t=1}^{T'-1}\mathbb{E}\left[\left\|U^{(k,t,t')} - U^{(k,t,0)}\right\|\right] \tag{68}$$

$$= T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|^2 - \gamma\beta\left\|\bar{\mathbf{g}}_U^{(t)}\right\|\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'-1}\mathbb{E}\left[\left\|\sum_{g=0}^{t'-1}\bar{\mathbf{g}}_U^{(k,t,g)}\right\|\right] \tag{69}$$

$$\geq T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|^2 - \gamma\beta\left\|\bar{\mathbf{g}}_U^{(t)}\right\|\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'-1}\sum_{g=0}^{t'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,g)}\right\|\right] \tag{70}$$

$$\geq T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|^2 - \gamma\beta T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,g)}\right\|\right] \tag{71}$$

$$\tag{72}$$

For another term in (63), similarly we can have

$$\frac{\beta\gamma^2}{2}\mathbb{E}\left[\left\|\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbf{g}_U^{(k,t,t')}\right\|^2\right] \tag{73}$$

$$\leq \frac{\beta\gamma^2}{2}\sum_{k=1}^{s}\frac{m_k}{m}\mathbb{E}\left[\left\|\sum_{t'=0}^{T'-1}\mathbf{g}_U^{(k,t,t')}\right\|^2\right] \text{(By convexity of L2 norm)} \tag{74}$$

$$\leq \frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\mathbb{E}\left[\sum_{t'=0}^{T'-1}\left\|\mathbf{g}_U^{(k,t,t')}\right\|^2\right] \text{(By Cauchy-Schwartz inequality)} \tag{75}$$

$$= \frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\mathbb{E}\left[\sum_{t'=0}^{T'-1}\left\|\left(\mathbf{g}_U^{(k,t,t')} - \bar{\mathbf{g}}_U^{(k,t,t')} + \bar{\mathbf{g}}_U^{(k,t,t')}\right)\right\|^2\right] \tag{76}$$

$$= \frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\mathbf{g}_U^{(k,t,t')} - \bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] + \frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] \text{(By unbiased assumption)} \tag{77}$$

$$\leq \frac{\beta\gamma^2 T'^2\sigma^2_{\mathbb{S}_{\mathsf{Sub}},U}}{2} + \frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] \tag{78}$$

The first inequality comes from the convexity of the square of ℓ_2-norm. We use Cauchy-Schwartz for the the second inequality. The last equality is because of the unbiased assumption. The last inequality is due to the bounded variance. Finally, expectation of (31) should be

$$-\gamma\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\langle\bar{\mathbf{g}}_U^{(t)}, \bar{\mathbf{g}}_U^{(k,t,t')}\right\rangle\right] + \frac{\beta\gamma^2}{2m^2}\mathbb{E}\left[\left\|\sum_{k=1}^{s}m_k\sum_{t'=0}^{T'-1}\mathbf{g}_U^{(k,t,t')}\right\|^2\right] \tag{79}$$

$$\leq -\gamma T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|^2 + \gamma^2\beta T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')}\right\|\right] \tag{80}$$

$$+\frac{\beta\gamma^2 T'^2\sigma^2_{\mathbb{S}_{\mathsf{Sub}},U}}{2} + \frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] \tag{81}$$

□

B.4 Proof of Lemma 2

Proof of Lemme 2. We look at the expecation of (32) for the update of V,

$$\mathbb{E}\left[\left\langle\bar{\mathbf{g}}_V^{(t)}, V^{(t+1)} - V^{(t)}\right\rangle + \frac{\beta}{2}\left\|V^{(t+1)} - V^{(t)}\right\|^2\right] \tag{82}$$

$$= \sum_{k=1}^{s}\left(-\mathbb{E}\left[\left\langle\frac{m_k}{m}\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(t)}, \sum_{t'=0}^{T'-1}\frac{\gamma m_k}{m}\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\rangle\right] + \frac{\beta}{2}\mathbb{E}\left[\left\|\sum_{t'=0}^{T'-1}\frac{\gamma m_k}{m}\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right]\right) \tag{83}$$

$$= \sum_{k=1}^{s}\left(-\sum_{t'=0}^{T'-1}\frac{\gamma m_k^2}{m^2}\mathbb{E}\left[\left\langle\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\rangle\right] + \frac{\beta}{2}\mathbb{E}\left[\left\|\sum_{t'=0}^{T'-1}\frac{\gamma m_k}{m}\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right]\right) \tag{84}$$

Similar as the previous part, for each k, the first term can be reduced to

$$
\begin{align}
&\sum_{t'=0}^{T'-1} \frac{\gamma m_k^2}{m^2} \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\rangle \right] \tag{85}\\
=& \sum_{t'=0}^{T'-1} \frac{\gamma m_k^2}{m^2} \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)} - \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)} + \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\rangle \right] \tag{86}\\
=& \sum_{t'=0}^{T'-1} \frac{\gamma m_k^2}{m^2} \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)} \right\rangle \right] + \sum_{t'=0}^{T'-1} \frac{\gamma m_k^2}{m^2} \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} - \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)} \right\rangle \right] \tag{87}\\
=& \frac{\gamma T' m_k^2}{m^2} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\|^2 + \sum_{t'=0}^{T'-1} \frac{\gamma m_k^2}{m^2} \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} - \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)} \right\rangle \right] \tag{88}\\
\geq& \frac{\gamma T' m_k^2}{m^2} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\|^2 - \frac{\gamma m_k^2}{m^2} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} - \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)} \right\| \right] \tag{89}\\
\geq& \frac{\gamma T' m_k^2}{m^2} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\|^2 - \frac{\gamma m_k^2 \beta}{m^2} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| V^{(k,t,t')} - V^{(k,t,0)} \right\| \right] \tag{90}\\
=& \frac{\gamma T' m_k^2}{m^2} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\|^2 - \frac{\gamma^2 m_k^3 \beta}{m^3} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \sum_{g=0}^{t'-1} \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,g)} \right\| \right] \tag{91}\\
\geq& \frac{\gamma T' m_k^2}{m^2} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\|^2 - \frac{\gamma^2 m_k^3 \beta}{m^3} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \sum_{g=0}^{t'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,g)} \right\| \right] \tag{92}\\
\geq& \frac{\gamma T' m_k^2}{m^2} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\|^2 - \frac{T' \gamma^2 m_k^3 \beta}{m^3} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\| \right] \tag{93}\\
& \tag{94}
\end{align}
$$

The second term can be bounded as

$$
\begin{align}
&\frac{\beta}{2} \mathbb{E}\left[\left\| \sum_{t'=0}^{T'-1} \frac{\gamma m_k}{m} \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] \tag{95}\\
=& \frac{\beta \gamma^2 m_k^2}{2m^2} \mathbb{E}\left[\left\| \sum_{t'=0}^{T'-1} \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] \tag{96}\\
\leq& \frac{\beta \gamma^2 m_k^2 T'}{2m^2} \mathbb{E}\left[\sum_{t'=0}^{T'-1} \left\| \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] \tag{97}\\
=& \frac{\beta \gamma^2 m_k^2 T'}{2m^2} \mathbb{E}\left[\sum_{t'=0}^{T'-1} \left\| \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} - \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} + \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] \tag{98}\\
=& \frac{\beta \gamma^2 m_k^2 T'}{2m^2} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} - \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 + \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] \tag{99}\\
\leq& \frac{\beta \gamma^2 m_k^2 T'^2 \sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2}{2m^2} + \frac{\beta T' m_k^2 \gamma^2}{2m^2} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] \tag{100}
\end{align}
$$

Thus, we have

$$
\begin{aligned}
&-\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_V^{(t)}, V^{(t+1)} - V^{(t)} \right\rangle + \frac{\beta}{2}\left\| V^{(t+1)} - V^{(t)} \right\|_2^2 \right] && (101)\\
&\leq -\gamma T' \sum_{k=1}^{s} \frac{m_k^2}{m^2} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\|^2 + \sum_{k=1}^{s} \frac{T'\gamma^2 m_k^3 \beta}{m^3} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\| \right] && (102)\\
&\quad + \sum_{k=1}^{s} \frac{\beta\gamma^2 m_k^2 T'^2 \sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2}{2m^2} + \sum_{k=1}^{s} \frac{\beta T' m_k^2 \gamma^2}{2m^2} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] && (103)\\
&= -\gamma T' \left\| \bar{\mathbf{g}}_V^{(k,t)} \right\|^2 + \sum_{k=1}^{s} \frac{T'\gamma^2 m_k^3 \beta}{m^3} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\| \right] && (104)\\
&\quad + \sum_{k=1}^{s} \frac{\beta\gamma^2 m_k^2 T'^2 \sigma_{\mathbb{S}_{\mathsf{Sub}},V}^2}{2m^2} + \sum_{k=1}^{s} \frac{\beta T' m_k^2 \gamma^2}{2m^2} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] && (105)\\
& && (106)
\end{aligned}
$$

□

C PRIVACY PROOFS

C.1 Proof of Theorem 3

Proof of Theorem 3. For $\mathbf{X}_{\mathbb{F}_k}$, let $\mathbf{X}'_{\mathbb{F}_k}$ as its neighbour dataset by changing an element at index (i^*, j^*) from $\perp$ to an observed rating. In another word, $\Omega_k \cup \{(i^*, j^*)\} = \Omega'_k$. With the same $U, V_{\mathbb{F}_k}$, only the gradients to u_{i^*} and v_{j^*} can be affected by the difference of neighbour dataset. For every local iteration, each party k in VFL-SGDMF samples a sub-matrix. The cell (i^*, j^*) is sampled by probability $\eta = \frac{n'm'_k}{nm_k}$.

Let $\mathsf{M}_{\mathbb{S}_{\mathsf{Sub}}}^{(k)}(\cdot)$ be the function of sampling and computing the differentially private gradients to all parameters of party k, $W = (U, V_{\mathbb{F}_k})$. With (7), it is easy to see that if $\mathsf{M}_{\mathbb{S}_{\mathsf{Sub}}}^{(k)}(\mathbf{X}_{\mathbb{F}_k}) \sim \mu_0$, then $\mathsf{M}_{\mathbb{S}_{\mathsf{Sub}}}^{(k)}(\mathbf{X}'_{\mathbb{F}_k}) \sim (1-\eta)\mu_0 + \eta\mu_1$ for some distribution μ_0 and μ_1. Following the proof of Lemma 3 in [1], the log of moment generating function $\alpha_{\mathsf{M}^{(k)},t,t'}(\lambda) \leq \frac{\eta^2\lambda(\lambda+1)}{(1-\eta)\sigma^2} + O\left(\eta^2\lambda^2/\sigma^3\right)$. Then follow the proof of Theorem 1 in [1], it can be shown that when $\sigma^{(k)} = \frac{(\Delta_2)^2\eta^2 TT' \ln 1/\delta^{(k)}}{\left(\epsilon^{(k)}\right)^2}$, the algorithm is $(\epsilon^{(k)}, \delta^{(k)})$ per-rating DP for party k.

When we consider the global per-rating privacy loss of all s parties, VFL-SGDMF is equivalent to a centralized setting where the dataset $\mathbf{X}$ is partition into s disjoint sub-datasets by items, $\mathbf{X}_{\mathbb{F}_1}, \ldots, \mathbf{X}_{\mathbb{F}_k}$. An mechanism M has s subroutines, $\mathsf{M}^{(1)}, \ldots, \mathsf{M}^{(s)}$, where $\mathsf{M}^{(k)}$ only accesses $\mathbf{X}_{\mathbb{F}_k}$. After every T' iterations, part of the outputs (i.e., local versions of U) of all subroutines will be averaged. Since the averaging is a post-processing step, it does not introduce any additional privacy loss. The local computation and global averaging happen T times. The composition of the moments is $\alpha_{\mathsf{M}}(\lambda) = \sum_{t=1}^{T}\sum_{t'=1}^{T'} \alpha_{\mathsf{M},t,t'}(\lambda)$.

Same as traditional DP language, moment accountants also has parallel composition property (Proposition 4). With Proposition 4, $\alpha_{\mathsf{M}}(\lambda) = \sum_{t=1}^{T}\sum_{t'=1}^{T'} \max_{k\in[s]}\{\alpha_{\mathsf{M}^{(k)},t,t'}(\lambda)\}$. If $\alpha_{\mathsf{M}^{(k)},t,t'}(\lambda)$ are the same for all t and t', then $\alpha_{\mathsf{M}}(\lambda) = \max_{k\in[s]}\{\sum_{t=1}^{T}\sum_{t'=1}^{T'} \alpha_{\mathsf{M}^{(k)},t,t'}(\lambda)\}$.

So VFL-SGDMF is also $(\max_{k\in[s]}\left\{\epsilon^{(k)}\right\}, \max_{k\in[s]}\left\{\delta^{(k)}\right\})$ per-rating differentially privacy globally. □

Proposition 4 (Parallel composition of moment accountants). *If the dataset* $\mathbf{X}$ *is partitioned into* s *parts,* $\mathbf{X}_1, \ldots, \mathbf{X}_s$*, where all the partitions are disjoint, and* M *has* s *subroutines,* $\mathsf{M}^{(1)}, \ldots, \mathsf{M}^{(s)}$*, where* $\mathsf{M}^{(k)}$ *can only access* $\mathbf{X}_k$*, then*

$$\alpha_{\mathsf{M}^{(1:s)}}(\lambda) = \max_{k\in[s]}\{\alpha_{\mathsf{M}^{(k)}}(\lambda)\}$$

Proof. A pair of dataset is neighbouring data if on only if there is one rating is different, which also means there is only one pair of partitions $\mathbf{X}_{k^*}$ and $\mathbf{X}'_{k^*}$ are different. Then

$$\begin{aligned}
& c(o^{(1:s)}; \mathsf{M}^{(1:s)}, \mathsf{aux}^{(1:s)}, \mathbf{X}, \mathbf{X}') \\
&= \log \prod_{k=1}^{s} \frac{\Pr\left[\mathsf{M}^{(k)}(\mathbf{X}_{k^*}) = o^{(k)} | \mathsf{aux}^{(k)}\right]}{\Pr\left[\mathsf{M}^{(k)}(\mathbf{X}'_{k^*}) = o^{(k)} | \mathsf{aux}^{(k)}\right]} \\
&= \sum_{k=1}^{s} \log \frac{\Pr\left[\mathsf{M}^{(k)}(\mathbf{X}_{k^*}) = o^{(k)} | \mathsf{aux}^{(k)}\right]}{\Pr\left[\mathsf{M}^{(k)}(\mathbf{X}'_{k^*}) = o^{(k)} | \mathsf{aux}^{(k)}\right]} \\
&= \log \frac{\Pr\left[\mathsf{M}^{(k^*)}(\mathbf{X}_{k^*}) = o^{(k^*)} | \mathsf{aux}^{(k^*)}\right]}{\Pr\left[\mathsf{M}^{(k^*)}(\mathbf{X}'_{k^*}) = o^{(k^*)} | \mathsf{aux}^{(k^*)}\right]} \\
&= c(o^{(k^*)}; \mathsf{M}^{(k^*)}, \mathsf{aux}^{(k^*)}, \mathbf{X}_{k^*}, \mathbf{X}'_{k^*})
\end{aligned}$$

Then

$$\begin{aligned}
& \alpha_{\mathsf{M}}(\lambda; \mathbf{X}, \mathbf{X}', \mathsf{aux}^{(1:s)}) \\
&= \log \mathbb{E}_{o'^{(1:s)} \sim \mathsf{M}^{(1:s)}(\mathbf{X})} \left[\exp\left(\lambda c(o^{(1:s)}; \mathsf{M}^{(1:s)}, \mathsf{aux}^{(1:s)}, \mathbf{X}, \mathbf{X}')\right) \right] \\
&= \log \mathbb{E}_{o'^{(1:s)} \sim \mathsf{M}^{(1:s)}(\mathbf{X})} \left[\exp\left(\lambda c(o^{(k^*)}; \mathsf{M}^{(k^*)}, \mathsf{aux}^{(k^*)}, \mathbf{X}_{k^*}, \mathbf{X}'_{k^*})\right) \right] \\
&= \alpha_{\mathsf{M}^{(k^*)}}(\lambda, o^{(k^*)}, \mathbf{X}_{k^*}, \mathbf{X}'_{k^*}) \\
&\leq \max_{k} \{\alpha_{\mathsf{M}^{(k)}}(\lambda, o^{(k)}, \mathbf{X}_k, \mathbf{X}'_k)\}
\end{aligned}$$

□

C.2 Proof of Theorem 4

Proof of Theorem 4. The per-user privacy loss from from a global view can be analyzed by the composition of moments. If we take the union of all parties' rating matrices as a whole, $\mathbf{X} = \cup_{k \in [s]} \mathbf{X}_k$, then `VFL-SGDMF` is equivalent to a centralized mechanism $\mathsf{M}'(\mathbf{X})$ which queries the large rating matrix sTT' times. Each iteration $t_g \in [sTT']$, M' only samples from $\mathbf{X}_k$ where $k = t_g \mod TT'$, then updates the local versions of $U^{(k)}$ and $V_{\mathbb{F}_k}$. The sensitivity of the computation based on $\mathbf{X}_k$ is bounded by $\theta^{(k)}\Delta_2$.

So by composability Theorem 1, the moment is bounded by $\alpha_{\mathsf{M},\text{total}}(\lambda) = \sum_{t=1}^{T} \sum_{k=1}^{s} \sum_{t'=1}^{T'} \alpha_{\mathsf{M},k,t,t'}(\lambda) \leq \sum_{k=1}^{s} \frac{TT'\eta^2\lambda^2}{z^2}$. By the tail bound in Theorem 1, to ensure `VFL-SGDMF` is (ϵ, δ)-per user DP, it is sufficient to show

$$\sum_{k=1}^{s} \frac{TT'\eta^2\lambda^2}{z^2} \leq \frac{\lambda}{2}\epsilon$$
$$\exp(-\lambda\epsilon/2) \leq \delta$$

That is equivalent to find a smallest ϵ such that $\epsilon^2 \geq \sum_{k=1}^{s} \frac{TT'\eta^2 \log(1/\delta)}{z^2}$. For each party k, if it already satisfies $(\epsilon^{(k)}, \delta^{(k)})$-DP for party level, it means

$$\frac{TT'\eta^2\lambda^2}{z^2} \leq \frac{\lambda}{2}\epsilon^{(k)}$$
$$\exp(-\lambda\epsilon^{(k)}/2) \leq \delta^{(k)}$$
$$\Rightarrow (\epsilon^{(k)})^2 \geq \frac{TT'\eta^2 \log(1/\delta^{(k)})}{z^2}$$

Thus, if we set $\delta = \delta^{(1)} = \ldots = \delta^{(s)}$, the ϵ is sufficiently large when

$$\epsilon^2 \geq \sum_{k=1}^{s} (\epsilon^{(k)})^2 \frac{\log(1/\delta)}{\log(1/\delta^{(k)})} = \sum_{k=1}^{s} (\epsilon^{(k)})^2$$

□

D PROOF RELATED TO EMBEDDING CLIPPING

Lemma 3. *The embedding clipping operation that first turn all negative entries of U into 0, then normalize each row vector with $\frac{u_i}{\max\{1, \|u_i\|_2/\sqrt{R}\}}$ projects the embedding to the closest vector in $\mathcal{U}$ in terms of ℓ_2 distance.*

Proof. Let u_i be a row vector, the vector u^* is the output of the projection operation given u as input. We try to proof u_i^* is the solution for the following optimization problem.

$$\min \|x - u_i\|_2^2$$
$$s.t. \quad x \geq 0 \text{ and } \|x\|_2^2 \leq R$$

The projection will gives results such that $\forall j \in [p], u_{ij}^* = 0$ if $u_{ij} < 0$; otherwise $u_{ij}^* = \frac{u_{ij}}{\max\{1, \sqrt{\sum_{j,u_{ij}>0} u_{ij}^2}/\sqrt{R}\}}$. We will show that u_i^* satisfies the KKT conditions.

- By definition, $\forall j \in [p], u_{ij}^* \geq 0$ and $\left\|u_i^*\right\|_2^2 \leq R$, so u_i^* satisfies *primal feasibility*.
- In order to satisfy the *dual feasibility* and *complementary slackness*, we need to show that $\mu_j u_{ij}^* = 0$ and $\mu_0(\sum_j (u_{ij}^*)^2 - R) = 0$, where μ_j are the dual variables. By the projection operation $\mu_j = 0$ when $u_{ij} > 0$.
- To satisfy *stationary condition*, we have $\sum_j 2(u_{ij}^* - u_{ij}) - \mu_j + 2\mu_0 u_{ij}^* = 0$.

$$\sum_{j \in [p]} 2(u_{ij}^* - u_{ij}) - \mu_j + 2\mu_0 u_{ij}^* = 0$$

$$\xRightarrow{\text{with dual feasibility}} \left(\sum_{j:u_{ij}<0} -2u_{ij} - \mu_j \right) + \left(\sum_{j:u_{ij}\geq 0} 2(u_{ij}^* - u_{ij}) + 2\mu_0 u_{ij}^* \right) = 0$$

  It's easy to see, for the case $\sqrt{\sum_{j,u_{ij}>0} u_{ij}^2}/\sqrt{R} \leq 1$, we can set $\mu_j = -2u_{ij}$ for $j : u_{ij} < 0$ and $\mu_0 = 0$ so that stationary condition and all the conditions holds.
  For the case $\sqrt{\sum_{j,u_{ij}>0} u_{ij}^2}/\sqrt{R} > 1$, we can set $\mu_j = -2u_{ij}$ for $j : u_{ij} < 0$ and $\mu_0 = \frac{-\sum_{j:u_{ij}\geq 0}(u_{ij}^* - u_{ij})}{\sum_{j:u_{ij}\geq 0} u_{ij}} = 1 - \frac{\sqrt{R}}{\sqrt{\sum_{j,u_{ij}>0} u_{ij}^2}}$ so that stationary condition and all the conditions also holds.
- Based on the previous analysis, the project is generating the embedding that is closest to the original embedding in terms of ℓ_2 norm.

□