DIFFERENTIAL PRIVACY IN DISTRIBUTED SETTINGS

by

Zitao Li

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Computer Science

West Lafayette, Indiana

December 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Ninghui Li, Chair

Department of Computer Science

Dr. Christopher W. Clifton

Department of Computer Science

Dr. Jean Honorio

Department of Computer Science

Dr. Jeremiah Blocki

Department of Computer Science

Approved by:

Dr. Kihong Park

ACKNOWLEDGMENTS

First and foremost, I would like to express my greatest gratitude to my advisor, Professor Ninghui Li, for his continued guidance and support during my Ph.D. study. His insights and vision in research always turn into valuable advice to me; his enthusiasm cheers me up along the study; his great personality as an advisor makes him become a role model for me beyond research. Without his support, I would not be able to finish this dissertation.

I also sincerely thank Professor Christopher W. Clifton and Professor Jean Honorio. They are not only my committee members but also the ones who offered supervision and great opportunities in my early Ph.D. student journey. Besides, I greatly appreciate Professor Jeremiah Blocki for his constructive suggestions as an external member in my preliminary and final exams.

I am grateful to my research internship mentor Dr. Bolin Ding at Alibaba for providing me with a great research experience. Working with him broadens my horizons and inspires me with many interesting research ideas.

Many thanks to Dr. Tianhao Wang. For me, he is not only a senior lab-mate and coauthor, but also a friend in life and a role model in research. I also thank my other coauthors, Milan Lopuhaä-Zwakenberg, Xiaoguang Li, Professor Boris Skoric, Professor Wenhai Sun, and Professor Ce Zhang, for the enjoyable collaborations on the projects. Besides, my appreciation goes to my lab-mates and friends in Purdue, Huangyi Ge, Jiacheng Li, Weicheng Wang, Mengyue Hang, Zixun Yu and others.

Last but not least, my deepest appreciation goes to my family. I sincerely thank my parents for their love and support. And I want to say a heartfelt "thank you" to my wife for her love, support, and company.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Data is considered the "new oil" in the information society and digital economy. While many commercial activities and government decisions are based on data, the public raises more concerns about privacy leakage when their private data are collected and used. In this dissertation, we investigate the privacy risks in settings where the data are distributed across multiple data holders, and there is only an untrusted central server. We provide solutions for several problems under this setting with a security notion called differential privacy (DP). Our solutions can guarantee that there is only limited and controllable privacy leakage from the data holder, while the utility of the final results, such as model prediction accuracy, can be still comparable to the ones of the non-private algorithms.

First, we investigate the problem of estimating the distribution over a numerical domain while satisfying local differential privacy (LDP). Our protocol prevents privacy leakage in the data collection phase, in which an untrusted data aggregator (or a server) wants to learn the distribution of private numerical data among all users. The protocol consists of 1) a new reporting mechanism called the square wave (SW) mechanism, which randomizes the user inputs before sharing them with the aggregator; 2) an Expectation Maximization with Smoothing (EMS) algorithm, which is applied to aggregated histograms from the SW mechanism to estimate the original distributions.

Second, we study the matrix factorization problem in three federated learning settings with an untrusted server, i.e., vertical, horizontal, and local federated learning settings. We propose a generic algorithmic framework for solving the problem in all three settings. We introduce how to adapt the algorithm into differentially private versions to prevent privacy leakage in the training and publishing stages.

Finally, we propose an algorithm for solving the k-means clustering problem in vertical federated learning (VFL). A big challenge in VFL is the lack of a global view of each data point. To overcome this challenge, we propose a lightweight and differentially private set intersection cardinality estimation algorithm based on the Flajolet-Martin (FM) sketch to convey the weight information of the synopsis points. We provide theoretical utility analysis for the cardinality estimation algorithm and further refine it for better empirical performance.

1. INTRODUCTION

Many services and applications are data-driven in the information society. For example, a government may decide annual funds allocation for disadvantaged children based on census data [1], and insurance companies adjust premium rates based on customers' personal financial situations [2]. However, it has been shown that collecting and using anonymized data can still lead to significant private information leakage [3, 4]. Even when the data are not directly accessible, the adversaries can still conduct model inversion attacks [5], membership inference attacks [6, 7], or reconstruction attacks [8, 9] to steal the sensitive private training information from the published machine learning models. As a framework to balance the data utility and privacy, differential privacy (DP) [10] has been accepted as the *de facto* security notion for privacy protection. Differential private techniques have been studied in research and deployed in many products, such as Google Chrome [11], Apple iOS [12], Microsoft Windows [13], and LinkedIn audience engagements API [14].

In this dissertation, we focus on designing differentially private mechanisms in distributed settings, where data are distributed across multiple parties, and a central server is required by computation paradigms but untrusted. We further assume that those data owners share some common interest in cooperation. They want to ensure their collaboration can lead to useful knowledge derived by the central server based on their shared information. There are several data allocation scenarios we consider in this dissertation.

First, one typical setting is the local setting, where users maintain their private data, and an untrusted server wants to learn aggregated information from those private data. It is also known as the cross-device setting in federated learning. Depending on the context, we call it the local or local federated learning (LFL) setting in this dissertation.

The second common distributed setting is the cross-silo federated learning setting. In this setting, some trustworthy data parties maintain parts of the user data but do not have access to the full view of the data. The data parties are also responsible for protecting user privacy and preventing other parties and the central server from inferring private information about their local data. Depending on how the data are distributed, the federated learning setting can be further categorized into horizontal and vertical variants. Horizontal federated

learning (HFL) assumes that all the data parties have data from different sets of users, but all local datasets have the same attributes. On the other hand, vertical federated learning (VFL) assumes that all data parties have data from the same set of users, but their data attributes differ from each other. This dissertation's contributions include solutions for different problems from the local setting to the HFL and VFL settings.

After describing the distributed settings we consider, the following introduces the exact problems and our solutions in each chapter.

In Chapter 3, we study the problem of recovering the distribution over a numerical domain while satisfying local differential privacy (LDP). While one can discretize a numerical domain and then apply the protocols developed for categorical domains, we show that taking advantage of the numerical nature of the domain can result in a better trade-off of privacy and utility. We propose a new reporting mechanism, the square wave (SW) mechanism. It randomizes the user input satisfying LDP but exploits the numerical nature in reporting by having higher probability to outputs close to the true inputs. We also develop an Expectation Maximization with Smoothing (EMS) algorithm, which is applied to aggregated histograms from the SW mechanism to estimate the original distributions. Extensive experiments demonstrate that our proposed approach, SW with EMS, consistently outperforms other methods in a variety of utility metrics.

In Chapter 4, we focus on the problem of matrix factorization in different federated learning settings and provide differentially private solutions for those settings. Matrix factorization (MF) approximates unobserved ratings in a rating matrix, whose rows correspond to users and columns correspond to items to be rated. It has been serving as a fundamental building block in recommendation systems. This chapter conducts an in-depth study on the problem of matrix factorization when a set of parties want to cooperate in training but refuse to share raw user data. We first propose a common 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 VFL, where multiple parties have ratings from the same set of users but on disjoint sets of items. The second one is HFL, where

parties have ratings from different sets of users but on the same set of items. The third setting is 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 sensitivity-bounding technique called *embedding clipping* is proposed and used in all three settings to ensure differential privacy. For the LFL setting, we further combine differential privacy with secure aggregation to protect the communication between user devices and the server to avoid involving large noise with LDP. We perform experiments to demonstrate the effectiveness of our approaches.

In Chapter 5, we investigate another problem, k-means clustering in VFL. When the user data are distributed vertically on multiple data parties, the correlation of the inter-party attributes, one of the most important information for clustering, is hard to maintain with differential privacy guarantees. We propose a practical solution for differentially private vertical federated k-means clustering, where a set of global centers can be derived with a provable differential privacy guarantee. Our algorithm assumes an untrusted central server aggregating differentially private local centers and membership information from local data parties. Our proposed method lets the server build a weighted grid as the synopsis of the global dataset based on the received information. Final centers are generated by running any k-means algorithm on the weighted grid. Our approach for grid weight estimation uses a lightweight and differentially private set intersection cardinality estimation algorithm based on the Flajolet-Martin sketch. To further improve the estimation accuracy in the setting with more than two data parties, we propose a refined version of the weights estimation algorithm and a parameter tuning strategy to reduce the final k-means cost to be close to that in the central private setting. We provide theoretical utility analysis and experimental evaluation results for the cluster centers computed by our algorithm and show that our approach performs better both theoretically and empirically than the two baselines based on existing techniques.

After a DP background introduction (Chapter 2), the content is organized following a common road map in Chapter 3-5. We first formalize the problems with more details and provide additional background knowledge. Then we present our solutions and provide experimental results, followed by a related work discussion.

2. BACKGROUND

In this dissertation, we consider differential privacy (DP) the main privacy notion to protect users' 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 [3, 15]. In this chapter, we introduce the background of DP.

2.1 Differential Privacy

In the central setting, DP assumes that there is a trusted party holding all users' data. The trusted party can run a randomized mechanism $\mathcal{A}$, taking the dataset as input and releasing the output to the public. The mechanism is differentially private if and only if it satisfies the following definition.

Definition 2.1.1 (Differential privacy [16])**.** *A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for any pair of datasets $\mathbf{X}$, $\mathbf{X}'$ that differ in one record and for all possible subset O of possible outputs of algorithm $\mathcal{A}$,*

$$\mathsf{Pr}\left[\mathcal{A}(\mathbf{X}) \in O\right] \leq \mathrm{e}^{\epsilon}\mathsf{Pr}\left[\mathcal{A}(\mathbf{X}') \in O\right] + \delta.$$

Intuitively speaking, DP requires that the outputs of an algorithm are the same with similar probability after we add or remove any one user's record in the dataset. A smaller ϵ means a stronger privacy guarantee, and δ bounds the probability that a significant privacy leak may occur. The (ϵ, δ) together are usually called *privacy budget* or *privacy loss*.

Laplace mechanism. One of the most classic DP mechanisms, the Laplace mechanism, adds Laplace noise to the return of a function f to ensure the result is differentially private. The variance of the noise depends on GS_f, the *global sensitivity* or the L_1 sensitivity of f, defined with a pair of neighboring datasets as, $\mathsf{GS}_f = \max_{\mathbf{X} \simeq \mathbf{X}'} ||f(\mathbf{X}) - f(\mathbf{X}')||_1$. The Laplace mechanism mechanism $\mathcal{A}$ is formalized as $\mathcal{A}_f(\mathbf{X}) = f(\mathbf{X}) + \mathsf{Lap}\left(\frac{\mathsf{GS}_f}{\epsilon}\right)$, where $\mathsf{Lap}\left(\cdot\right)$ denotes

a random variable sampled from the Laplace distribution $\mathsf{Pr}\left[\mathsf{Lap}\left(b\right)=x\right]=\frac{1}{2b}\mathrm{e}^{-|x|/b}$. When f outputs a vector, $\mathcal{A}$ adds independent samples of $\mathsf{Lap}\left(\frac{\mathsf{GS}_f}{\epsilon}\right)$ to each element of the vector.

Local differential privacy (LDP). The privacy notion can be further strengthen by removing the trusted central party. Assume there are n *users* and one untrusted *aggregator*. Each user possesses a value $v \in \mathcal{D}$, and the aggregator wants to collect information from all users. We further assume that those users share some common interest in the aggregated information, and all of them want to ensure the aggregator can obtain useful knowledge from the aggregated information. To protect privacy, each user randomizes the input value v using an algorithm $\Psi(\cdot) : \mathcal{D} \to \tilde{\mathcal{D}}$, where $\tilde{\mathcal{D}}$ is the set of all possible outputs, and sends $\tilde{v} = \Psi(v)$ to the aggregator.

Definition 2.1.2 (ϵ-Local differential privacy)**.** *An algorithm $\Psi(\cdot) : \mathcal{D} \to \tilde{\mathcal{D}}$ satisfies ϵ-local differential privacy (ϵ-LDP), where $\epsilon \geq 0$, if and only if for any input $v_1, v_2 \in \mathcal{D}$, we have*

$$\forall O \subseteq \tilde{\mathcal{D}} : \ \mathsf{Pr}\left[\Psi(v_1) \in O\right] \leq \mathrm{e}^{\epsilon}\, \mathsf{Pr}\left[\Psi(v_2) \in O\right],$$

where $\tilde{\mathcal{D}}$ denotes the set of all possible outputs of Ψ.

Since a user never reveals v to the aggregator and reports only $\tilde{v} = \Psi(v)$, the user's privacy is still protected even if the aggregator is malicious. Thus, it is a strictly strong privacy notion compared with the central DP. However, while it provides stronger privacy protection for users' data, the aggregated and decoded information usually has larger noise than the one produced by the central DP.

2.2 Properties of DP

Three properties of DP are frequently used to build complicated algorithms: *post-processing*, *parallel composition* and *sequential composition*. The first two properties are straightforward and well-studied. However, How to obtain the tightest privacy loss in the sequential composition is still an open problem, and there are many ongoing works. We introduce several sequential computation methods used in this thesis.

Proposition 2.2.1 (Post-processing). *Any data independent operation on an (ϵ, δ)-DP algorithm's result still satisfies (ϵ, δ)-DP.*

Proposition 2.2.2 (Parallel composition [17]). *If all s mechanisms $\mathcal{A}_1, \ldots, \mathcal{A}_s$ are $(\epsilon^{(1)}, \delta^{(1)}), \ldots, (\epsilon^{(s)}, \delta^{(s)})$ differentially private and they are computed on disjoint datasets $\mathbf{X}_1, \ldots, \mathbf{X}_s$, then the overall privacy loss will be $(\max_k\{\epsilon^{(k)}\}, \max_k\{\delta^{(k)}\})$.*

2.2.1 Sequential composition

We consider three sequential composition for privacy loss in this thesis: naive sequential composition, moment accountant and Rényi DP.

Naive sequential composition. Assume that there are two subroutines $\mathcal{A}_1(\cdot)$ and $\mathcal{A}_2(\cdot)$ that can provides $(\epsilon_1, \delta_1), (\epsilon_2, \delta_2)$-DP protection. The naive *sequential composition* states that $\mathcal{A}_1(\mathbf{X}, \mathcal{A}_2(\mathbf{X}))$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP.

Sequential composition with moments accountant. The private dataset may be accessed multiple times. Moments accountant [18] can be used to provide tight analysis for the privacy loss of a sequence of adaptive mechanisms. In general, let $\mathcal{A}(\mathbf{X}, \mathtt{aux})$ be a function that returns outputs from the dataset $\mathbf{X}$ and an auxiliary input $\mathtt{aux}$. The auxiliary input $\mathtt{aux}$ can include all intermediate results from previous steps. The logarithm of the moment generating function (of privacy loss in $\mathcal{A}$) evaluated at the value λ is defined as

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

which is upper bounded by $\alpha_{\mathcal{A}}(\lambda) = \max_{\mathbf{X}, \mathbf{X}', \mathtt{aux}} \alpha_{\mathcal{A}}(\lambda; \mathbf{X}, \mathbf{X}', \mathtt{aux})$. Suppose there is a sequence of such adaptive mechanisms $\mathcal{A}_1, \ldots, \mathcal{A}_T$, where each $\mathcal{A}_t$ 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 $\mathcal{A}_1, \ldots, \mathcal{A}_T$ via composing the logarithms of the moment generating functions, $\alpha_{\mathcal{A}_t}(\lambda)$'s for $t \in [T]$.

Theorem 2.2.1 (Moment accountants (MA) [18]). *Let a mechanism $\mathcal{A}$ be a sequence of adaptive mechanisms $\mathcal{A}_1, \ldots, \mathcal{A}_T$ defined as above. Then we have 1) for any λ, $\alpha_{\mathcal{A}}(\lambda) \leq \sum_{t=1}^{T} \alpha_{\mathcal{A}_t}(\lambda)$; and 2) for any $\epsilon > 0$, the mechanism $\mathcal{A}$ is (ϵ, δ)-differentially privacy for $\delta = \min_\lambda \exp(\alpha_{\mathcal{A}}(\lambda) - \lambda\epsilon)$.*

In the experiments, we calculate the exact privacy loss using the package [19].

Sequential composition with Rényi Differential Privacy. Compare with MA, the notion of Rényi Differential Privacy (RDP) [20] provides a succinct way to track the privacy loss for a composition of multiple mechanisms when there is no subsampling operation. When given a δ, there is a closed form conversion equation turning Rényi DP to DP.

Definition 2.2.1 (Rényi Differential Privacy [20])**.** *A mechanism $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ is said to satisfy (ν, τ)-RDP if the following holds for any two neighboring datasets $\mathbf{X}, \mathbf{X}'$*

$$\frac{1}{\nu - 1} \log \mathbb{E}_{o \sim \mathcal{A}(\mathbf{X})} \left[\left(\frac{\mathsf{Pr}\left[\mathcal{A}(\mathbf{X}) = o\right]}{\mathsf{Pr}\left[\mathcal{A}(\mathbf{X}') = o\right]} \right)^{\nu} \right] \leq \tau.$$

Fact 2.2.1 (RDP Sequential Composition [20])**.** *If $\mathcal{A}_1$ and $\mathcal{A}_2$ are (ν, τ_1)-RDP and (ν, τ_2)-RDP respectively then the mechanism combining the two $g(\mathcal{A}_1(\mathbf{X}), \mathcal{A}_2(\mathbf{X}))$ is $(\nu, \tau_1 + \tau_2)$-RDP.*

Fact 2.2.2 (RDP to (ϵ, δ)-DP [20])**.** *If a mechanism is (ν, τ)-RDP, then it also satisfies $(\tau + \frac{\log 1/\delta}{\nu - 1}, \delta)$-DP.*

3. ESTIMATING NUMERICAL DISTRIBUTION UNDER LOCAL DIFFERENTIAL PRIVACY

(A version of this chapter has been previously published in SIGMOD 2020 [21]. There is also a public version [22] with additional information.)

3.1 Introduction

In local differential privacy (LDP), there are many *users* and one *aggregator*. Each user sends randomized information to the aggregator, who can infer the aggregated information based on users' reports. LDP techniques enable the gathering of statistics while preserving privacy of every user, without relying on trust in a single trusted third party. LDP techniques have been deployed by companies like Apple [12], Google [23], and Microsoft [13].

Most existing work on LDP focuses on the situations collecting categorical attributes. Existing research [23–27] has developed frequency oracle (FO) protocols for categorical domains, where the aggregator can estimate the frequency of any chosen value in the specified domain (fraction of users with that private value). We call these Categorical Frequency Oracle (CFO) protocols.

Many attributes are ordinal or numerical in nature, e.g., income, age, the amount of time viewing a certain page, the amount of communications, the number of times performing a certain actions, etc. A numerical domain consists of values that have a meaningful total order. One natural approach for dealing with ordinal and numerical attributes under LDP is to first apply binning and then use CFO protocols. That is, one treats all values in a range as one categorical value when reporting. This approach faces the challenge of finding the optimal number of bins, which depends on both the privacy parameter and the data distribution. One improvement over this approach is to apply Hierarchical Histogram-based approaches [28–30], which uses multiple granularities at the same time, and exploit the natural consistency relationships between estimations at different granularities. Recently, Kulkarni et al. [31] studied the accuracy of answering range queries using this approach.

We note that the stronger privacy guarantee offered by LDP (as compared with DP) comes with the cost of significantly higher noises. As a result, many estimated frequencies will be negative. Existing approaches (such as [31]) do not correct this, and are sub-optimal. We propose to apply Alternating Direction Method of Multipliers (ADMM) optimization [32] to improve Hierarchical Histograms, utilizing the constraints that all estimations are non-negative and sum up to 1. Experiments show that the improved version of hierarchy histogram, which we call HH-ADMM, has significantly better utility.

The above methods still use CFO protocols in a blackbox fashion, and existing CFO protocols ignore any semantic relationship between different values. An intriguing research question is whether one can design frequency oracle protocols that directly utilize the ordered nature of the domain and produce better estimations. In this chapter, we answer this affirmatively. We propose an approach that combines what we call a Square Wave reporting mechanism with post-processing using Expectation Maximization and Smoothing.

The key intuition under the Square Wave mechanism is that given input v, one should report a value close to v with higher probability than a value farther away from v. More specifically, assuming the input domain of numerical values is $\mathcal{D} = [0, 1]$, the output domain of Square Wave mechanism is $\tilde{\mathcal{D}} = [-b, 1 + b]$, where b is a parameter depending on the privacy parameter ϵ. A user with value $v \in \mathcal{D}$ reports a value $\tilde{v}$ randomly drawn from a distribution with probability density function $\mathbf{M}_v$. For any $\tilde{v} \in [v - b, v + b]$, probability density is $\mathbf{M}_v(\tilde{v}) = p$, and any $\tilde{v} \in [-b, 1+b] \setminus [v-b, v+b]$, probability density is $\mathbf{M}_v(\tilde{v}) = q$, where $\frac{p}{q} = \mathrm{e}^{\epsilon}$. We define and study different wave shapes of General Wave mechanism other than the above Square Wave, and conclude that Square Wave has the best utility. We also study how to determine the key parameter b, the width of the wave. We propose to choose b to maximize the upper bound of mutual information between the input and the output variable, and can compute b when given the privacy parameter ϵ. Experiments demonstrate the effectiveness of this approach.

Conceptually, the aggregator, after observing the reported values, without any prior knowledge of the input distribution, should perform Maximum Likelihood Estimation (MLE) to infer the input distribution, which can be carried out by the Expectation Maximization (EM) algorithm. Through experiments, we have observed that the result of applying EM

is highly sensitive to the parameter controlling terminating condition. This is because the observed distribution is a combination of the true distribution and the effect of random noise. When EM terminates too early, the result does not fit the true distribution well. When EM terminates too late, the result fits both the true distribution and the effect of noises. It is unclear how one can set the parameter so that one fits the distribution, but not the noise, across different datasets and privacy parameters.

To deal with this challenge, we propose to use smoothing together with the EM algorithm. In each iteration, after the E step and the M step, we add an S (smoothing) step, which averages each estimation with its nearest neighbours, by binomial coefficients. The Expectation Maximization with Smoothing approach was developed in the context of positron emission tomography and image reconstruction [33, 34], and was shown to be equivalent to adding a regularization term penalizing the spiky estimation [33]. Intuitively, EMS uses the prior knowledge that the observation is affected by noise and prefer a smoother distribution to a jagged one. In the experiment, we observe that EMS is stable under different settings, and requires no parameter tuning.

To compare different algorithms for reconstructing distributions of numerical attributes, we first use two metrics measuring the distance of reconstructed cumulative distribution from the true one, namely the Wasserstein distance and KolmogorovSmirnov distance (KS distance). In addition, we also consider accuracy for answering range queries, and accuracy of estimations of different statistics from the reconstructed distributions such as mean, variance and quantiles.

The contributions of this chapter are as follows. (1) We define the problem of reconstructing distributions of numerical attributes under LDP (with non-negativity and sum-up-to-1 constraints) and propose multiple metrics for comparing competing algorithms. (2) We introduce HH-ADMM, which improves upon existing hierarchy histogram based methods. (3) We introduce the method of combining Square Wave (SW) reporting with Expectation Maximization and Smoothing (EMS), and showed that Square Wave is preferable to other wave shapes, and introduce techniques to choose the bandwidth parameter b using mutual information. (4) We conduct extensive experimental evaluations, comparing the proposed methods with state-of-the-art methods (e.g., [31]). Results demonstrate that SW with EMS

Table 3.1. Notations.

Symbol	Description
v	Private input
$\tilde{v}$	Randomized output
$\mathcal{D}$	Domain of private input
$\tilde{\mathcal{D}}$	Domain of the randomized output
$\mathbf{x}$	True private input frequencies
$\hat{\mathbf{x}}$	Estimate of private input frequencies (normalized)
$\tilde{\mathbf{v}}$	Randomized output frequencies (normalized)
$\mathbf{P}$	Cumulative distribution function (CDF)
$\mathbf{M}_v$	Probability density function given input v

and HH-ADAM significantly out-perform existing methods. In addition, SW with EMS generally performs the best under a wide range of metrics, and HH-ADMM performs better than SW-EMS on a very spiking distribution under some of the metrics.

Roadmap. In Section 3.2, we review the problem setting and existing LDP protocols. In Section 3.3, we discuss metrics for measuring the quality of the reconstructed distribution. We describe CFO with binning and HH-ADMM in Section 3.4. SW reporting and EMS reconstruction are introduced in Section 3.5. We show our experimental results in Section 3.6. We give an overview of the related work in Section 3.7, and conclude in Section 3.8.

3.2 Problem Formulation and Existing Solutions

Throughout this chapter, we use bold letters to denote vectors. For example, $\mathbf{v} = \langle v_1, \ldots, v_n \rangle$ is all users' values, and $\mathbf{x} = \langle x_1, \ldots, x_d \rangle$ is frequencies of all values (i.e., $x_i = |\{j \mid v_j = i\}|/n$). If the notation is associated with a tilde (e.g., $\tilde{\mathbf{v}}$), it is the value after LDP perturbation; and a hat (e.g., $\hat{\mathbf{x}}$) denotes the value computed by the aggregator. Capital bold letters denote matrices and functions that take more than one input. Table 3.1 gives some of the frequently used symbols.

3.2.1 Categorical Frequency Oracles

A *frequency oracle (*FO*)* protocol enables the estimation of the frequency of any value $v \in \mathcal{D}$ under LDP. Existing protocols are designed for situations where $\mathcal{D}$ is a categorical domain. We call them *categorical frequency oracle* (CFO) protocols in this chapter. The following are two commonly used CFO protocols.

Generalized Randomized Response (GRR). This CFO protocol generalizes the *randomized response* technique [35], and uses $\tilde{\mathcal{D}} = \mathcal{D}$. It uses as input perturbation function $\mathsf{GRR}(\cdot)$, where $\mathsf{GRR}(v)$ outputs the true value v with probability $p = \frac{e^\epsilon}{e^\epsilon + d - 1}$, and any value $v' \neq v$ with probability $q = \frac{1-p}{d-1} = \frac{1}{e^\epsilon + d - 1}$, where $d = |\mathcal{D}|$ is the domain size.

To estimate the frequency of $v \in \mathcal{D}$ (i.e., the ratio of the users who have v as private value to the total number of users), one counts how many times v is reported, and denote the count as $C(v)$, and then computes

$$\tilde{x}_v = \frac{(C(v)/n) - q}{p - q} \ ,$$

where n is the total number of users. In [36], it is shown that this is an unbiased estimate of the true count, and the variance for this estimate is

$$\mathrm{Var}[\tilde{x}_v] = \frac{d - 2 + e^\epsilon}{(e^\epsilon - 1)^2 \cdot n} \ . \tag{3.1}$$

The variance given in (3.1) is linear to d; thus when the domain size d increases, the accuracy of this protocol is low.

Optimized Local Hashing (OLH) [36]. This protocol deals with a large domain size $d = |\mathcal{D}|$ by first using a hash function to map an input value into a smaller domain of size g (typically $g \ll |\mathcal{D}|$), and then applying randomized response to the hashed value (which leads to $p = \frac{e^\epsilon}{e^\epsilon + g - 1}$). In this protocol, both the hashing step and the randomization step result in information loss. The choice of the parameter g is a tradeoff between losing information during the hashing step and losing information during the randomization step. In [36], it is found that the optimal choice of g that leads to minimal variance is $(e^\epsilon + 1)$.

In OLH, one reports $\langle H, \mathsf{GRR}(H(v))\rangle$, where H is randomly chosen from a family of hash functions that hash each value in $\mathcal{D}$ to $\{1 \dots g\}$, and $\mathsf{GRR}(\cdot)$ is the perturbation function for Generalized Randomized Response, while operating on the domain $\{1 \dots g\}$. Let $\langle H^j, y^j\rangle$ be the report from the j'th user. For each value $v \in \mathcal{D}$, to compute its frequency, one first computes $C(v) = |\{j \mid H^j(v) = y^j\}|$, and then transforms $C(v)$ to its unbiased estimate

$$\tilde{x}_v = \frac{(C(v)/n) - (1/g)}{p - 1/g}.$$

The approximate variance of this estimate is

$$\mathrm{Var}[\tilde{x}_v] = \frac{4\mathrm{e}^\epsilon}{(\mathrm{e}^\epsilon - 1)^2 \cdot n}.$$

Compared with (3.1), the factor $d - 2 + \mathrm{e}^\epsilon$ is replaced by $4\mathrm{e}^\epsilon$. This suggests that for smaller $|\mathcal{D}|$ (such that $|\mathcal{D}| - 2 < 3\mathrm{e}^\epsilon$), GRR is better; but for large $|\mathcal{D}|$, OLH is better and has a variance that does not depend on $|\mathcal{D}|$.

3.2.2 Handling Numerical Attributes

Two methods have been proposed for mean estimation under LDP for numerical attributes. Note that using these methods one can estimate the mean, and not the distribution.

Stochastic Rounding (SR) [37]. The main idea of Stochastic Rounding (SR) is that, no matter what is the input value v, each user reports one of two extreme values, with probabilities depending on v. Here we give an equivalent description of the protocol. Following [37], we assume that the input domain is $[-1, 1]$. Given a value $v \in [-1, 1]$, let $p = \frac{\mathrm{e}^\epsilon}{\mathrm{e}^\epsilon + 1}$ and $q = 1 - p = \frac{1}{\mathrm{e}^\epsilon + 1}$, the SR method outputs a random variable v', which takes the value -1 with probability $q + \frac{(p-q)(1-v)}{2}$ and value 1 with probability $q + \frac{(p-q)(1+v)}{2}$. Since

$$\begin{aligned}\mathbb{E}[v'] &= (-1)\left(q + \frac{(p-q)(1-v)}{2}\right) + q + \frac{(p-q)(1+v)}{2} \\ &= (p-q)v\ ,\end{aligned}$$

let $\tilde{v} = \frac{v'}{p-q}$, we have $\mathbb{E}[\tilde{v}] = v$; thus the mean of $\tilde{v}$ provides an unbiased estimate of the mean for the distribution.

Piecewise Mechanism (PM) [38]. In the Piecewise Mechanism, the input domain is $[-1, 1]$, and the output domain is $[-s, s]$, where $s = \frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}$. For each $v \in [-1, 1]$, there is an associated range $[\ell(v), r(v)]$ where $-s \leq \ell(v) < r(v) \leq s$, such that with input v, a value in the range $[\ell(v), r(v)]$ will be reported with higher probability than a value outside the range. More precisely, we have $\ell(v) = \frac{e^{\epsilon/2} \cdot v - 1}{e^{\epsilon/2}-1}$ and $r(v) = \frac{e^{\epsilon/2} \cdot v + 1}{e^{\epsilon/2}-1}$. The width of the range is $r(v) - \ell(v) = \frac{2}{e^{\epsilon/2}-1}$, and the center is $\frac{\ell(v)+r(v)}{2} = \frac{e^{\epsilon/2}}{e^{\epsilon/2}-1} \cdot v$. Specifically, PM works as follows:

$$\Pr[\mathsf{PM}(v) = \tilde{v}] = \frac{e^{\epsilon/2}}{2} \cdot \frac{e^{\epsilon/2} - 1}{e^{\epsilon/2} + 1} \text{ if } \tilde{v} \in [\ell(v), r(v)],$$
$$\Pr[\mathsf{PM}(v) = \tilde{v}] = \frac{1}{2e^{\epsilon/2}} \cdot \frac{e^{\epsilon/2} - 1}{e^{\epsilon/2} + 1} \text{ otherwise.}$$

It is shown that $\tilde{v}$ is unbiased, and has better variance than SR when ϵ is large [39].

3.3 Utility Metrics

When the private values are in a numerical domain, we need utility metrics that are different from those in categorical domains. In particular, the metrics should reflect the ordered nature of the underlying domain.

3.3.1 Metrics based on Distribution Distance

We want a metric to measure the distance between the recovered density distribution and the true distribution. However, since the distribution is over a metric space, we do not want to use point-wise distance metrics such as the L_1 and L_2 distance or the KullbackLeibler (KL) divergence. For a simple example, consider the case where $\mathcal{D} = \{1, 2, 3, 4\}$, the true distribution is $\mathbf{x} = [0.7, 0.1, 0.1, 0.1]$. The two estimations $\hat{\mathbf{x}}_1 = [0.1, 0.7, 0.1, 0.1]$ and $\hat{\mathbf{x}}_2 = [0.1, 0.1, 0.1, 0.7]$ have the same L_1, L_2, and KL distance from $\mathbf{x}$, but the distance between $\hat{\mathbf{x}}_1$ and $\mathbf{x}$ should be smaller than the distance between $\hat{\mathbf{x}}_2$ and $\mathbf{x}$ when we consider the numerical

nature. To capture this requirement, we propose to use two popular distribution distances as metrics.

Wasserstein Distance (aka. Earth Mover Distance). Wasserstein distance measures the cost of moving the probability mass (or density) from distribution to another distribution. In this chapter, we use the one dimensional Wasserstein distance. For a discrete domain, define the cumulative function $\mathbf{P} : [0,1]^d \times \mathcal{D} \mapsto [0,1]$ that takes a distribution $\mathbf{x}$ and a value v, and output $\mathbf{P}(\mathbf{x}, v) = \sum_{i=1}^{v} x_v$. Let $\mathbf{x}$ and $\hat{\mathbf{x}}$ be two distributions. The one dimensional Wasserstein distance is the L_1 difference between their cumulative distributions:

$$W_1(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{v \in \mathcal{D}} |\mathbf{P}(\mathbf{x}, v) - \mathbf{P}(\hat{\mathbf{x}}, v)| \ . \tag{3.2}$$

For continuous domain, $\mathbf{x}$ is the probability density function with support on $[0,1]$, $\mathbf{P}(\mathbf{x}, v) = \int_{t=0}^{v} x(t)dt$. The one dimensional Wasserstein distance is

$$W_1(\mathbf{x}, \hat{\mathbf{x}}) = \int_{v \in \mathcal{D}} |\mathbf{P}(\mathbf{x}, v) - \mathbf{P}(\hat{\mathbf{x}}, v)| \ dv \ .$$

Kolmogorov-Smirnov (KS) Distance. KS distance is the maximum absolute difference at any point between the cumulative functions of two distributions:

$$d_{KS}(\mathbf{x}, \hat{\mathbf{x}}) = \sup_{v \in \mathcal{D}} |\mathbf{P}(\mathbf{x}, v) - \mathbf{P}(\hat{\mathbf{x}}, v)| \ .$$

Both of Wasserstein distance and KS distance can be considered as measures for the errors of answering prefix range queries on numerical domains with constraints that the estimate must be non-negative and sum up to 1. Wasserstein distance is the error of sum of all prefix queries; and the KS distance is the maximum error of prefix queries.

3.3.2 Semantic and Statistical Quantities

Range queries have been used as the main utility metrics for research in this area [28, 31, 40, 41]. Also, we consider the basic statistics from the estimated data distributions and check whether they are accurate.

Range Query. Define the range query function $\mathbf{R}(\mathbf{x}, i, \alpha) = \mathbf{P}(\mathbf{x}, i+\alpha) - \mathbf{P}(\mathbf{x}, i)$, where α specifies the range size. Given the true distribution $\mathbf{x}$ and the estimated distribution $\hat{\mathbf{x}}$, range queries reflect the quality of estimate with randomly sampling i and calculating the following:

$$|\mathbf{R}(\mathbf{x}, i, \alpha) - \mathbf{R}(\hat{\mathbf{x}}, i, \alpha)| \ .$$

Mean. We denote μ as the mean of the true distribution, and $\hat{\mu}$ as the estimated mean. To measure mean accuracy, we use the absolute value of the difference between these two, i.e. $|\mu - \hat{\mu}|$.

Variance. We use σ^2 to denote the variance of the true distribution, and $\hat{\sigma}^2$ for the variance from the reconstructed distribution. To measure variance accuracy, we use the absolute value of the difference between these two, i.e. $|\sigma^2 - \hat{\sigma}^2|$.

Quantiles. Quantiles are cut points dividing the range of a probability distribution into intervals with equal probabilities. Formally, $\mathbf{Q}(\mathbf{x}, \beta) = \arg\max_v \{\mathbf{P}(\mathbf{x}, v) \leq \beta\}$. In the experiment, define $B = \{10\%, 20\%, \ldots, 90\%\}$, we measure the following:

$$\frac{1}{|B|} \sum_{\beta \in B} |\mathbf{Q}(\mathbf{x}, \beta) - \mathbf{Q}(\hat{\mathbf{x}}, \beta)| \ .$$

3.4 Using CFO Protocols for Numerical Domains

In this section, we present two approaches that use CFO protocols to reconstruct distributions over an discrete numerical domain $\mathcal{D} = \{1, 2 \cdots, d\}$. Continuous numerical domains can be buckized into discrete ones.

3.4.1 CFO with Binning

Given a numerical domain, one can make it discrete using binning, and then have each user report which bin the private value is in using a CFO protocol. For a given domain size and privacy parameter ϵ, one chooses either OLH or GRR, based on which one gives lower estimation variance. After obtaining density estimations for all the bins, one computes a density distribution for the domain by assuming uniform distribution within each bin. However, some estimated values may be negative, which does not lead to valid cumulative distribution functions on the domain. In [41], it is shown that a post-processing method called Norm-Sub can be applied to improve estimation. Norm-sub converts negative estimates to 0 and subtracts the same amount to all the positive estimates so that they sum up to 1. If some positive estimates become negative after the subtraction, the process is repeated. This results in an estimation such that each estimation is non-negative and all estimations sum up to 1. It can thus be interpreted as a probability distribution.

Challenge of Choosing Bin Size. When using binning, there are two sources of errors: noise and bias due to grouping values together. More bins lead to greater error due to noises. Fewer bins lead to greater error due to biases. Choosing the bin size is a trading-off of the above two sources of errors, and the effect of each choice depends both on the privacy parameter ϵ, and on property of the distribution. For example, when a distribution is smooth, one would prefer using less bins, as the bias error is small, and when a distribution is spiky, using more bins would perform better. In our experiments, we observe that even if we could choose the optimal bin size empirically for each dataset and ϵ value (which is infeasible to do in practice due to privacy), the result would still be worse than the method to be proposed in Section 3.5. We thus chose not to develop ways to choose bin size based on ϵ, and just report results of this method under several different bin sizes.

3.4.2 Hierarchy-based Methods

Hierarchy-based methods, including Hierarchy Histogram (HH) in [28, 29] and Haar in [30], were first proposed in the centralized setting of DP. In [31], Kulkarni et al. studied

the HH method and the Haar in the context of LDP. In order to adapt Haar method to the local setting, they used Hadamard random response (HRR) as the frequency oracle. HRR is similar to Local Hashing method introduced in the Section 3.2.1, but fixing $g = 2$ and using a Hadamard matrix as the family of hash functions. To make it clear in the context, we call the LDP version of Haar as HaarHRR.

HH in LDP. Given a positive integer β and a discrete, ordered domain with size $d = |\mathcal{D}|$, one can construct a β-ary tree with d leaves corresponding to values in $\mathcal{D}$. There are $(h+1)$ layers in the tree, where $h = \log_\beta d$ (for simplicity, we assume that $\log_\beta d$ is an integer). The $(h+1)$-th layer is the root. A user with value v chooses a layer $\ell \in \{1, \ldots, h\}$ uniformly at random, and then reports ℓ as well as the perturbed value of v's ancestor node at layer ℓ. For each node in the tree, the aggregator can obtain an estimate of its frequency. Assuming that the distribution differences among the h groups are negligible, for each parent-child relation, one expects that the sum of child estimations equals the that of the parent. Constrained inference techniques [28] are applied to ensure this property.

HaarHRR. Similar to HH, one can use a binary tree to estimate distribution with Discrete Haar Transform [31]. Specifically, each leaf represents the frequency of a value. Define the height of a leaf node as 0; and the height of an inner nodes a is denotes as $h(a)$. Each inner node now represents the Haar coefficient $c_a = \frac{C_l^{(a)} - C_r^{(a)}}{2^{h(a)/2}}$, where $C_l^{(a)}$ (or $C_r^{(a)}$) is the sum of all leaves of left (or right) subtree of node a.

In the LDP setting, for a user with value v, the Haar coefficients on each layer has exactly one element equal to -1 or 1, while others are all zeros. Similar to HH, each user chooses a layer $\ell \in \{1, \ldots, h\}$ uniformly at random, then apply Hadamard randomized response (HRR) on layer ℓ which depends on Hadamard matrix $\phi \in \{-1, 1\}^{2^{h-\ell} \times 2^{h-\ell}}$. With HRR reports from users, the aggregator can calculate unbiased estimates for the Haar coefficients on layer ℓ. Due the limit of space, more details can be found in [31].

Difference from the Centralized Setting. When using hierarchy-based method, there are two ways to ensure the privacy constraint. One is to divide the privacy budget, where one builds a single tree for all values. Since each value affects the counts at every level, one splits the privacy budget among the levels. The other is to divide the population among the

layers, where each value contributes to the estimation of a single layer, and one can use the whole privacy budget for each count. When dividing the population, the absolute level of noise is less than the case of dividing privacy budget; however, the total count also decreases, magnifying the impact of noise. In addition, dividing the population introduces sampling errors, as users are divided into different groups, which may have different distribution from the global one.

In the centralized setting, because the amount of added noise is low, it is better to divide the privacy budget, as one avoids sampling errors. In [29], it was found that in the centralized setting, the optimal branching factor for HH is around 16. And this results in better performance than using the Haar method, which can be applied only to a binary hierarchy. In the LDP setting, because the amount of noise is much larger, sampling errors can be mostly ignored, and it is better to divide the population instead of privacy budget. As a result, the optimal branching factor for HH is around 5, making it similar to the Haar method. This was theoretically proved and empirically demonstrated in [31, 40].

3.4.3 HH-ADMM

We note that there are other ways to improve hierarchy-based mechanism in the LDP setting. First, the larger noise in the LDP setting results in negative estimates. We can exploit the prior knowledge that the true counts are non-negative to improve the negative estimates. Second, the total true count is known, as LDP protects privacy of reported values and not the fact that one is reporting. These are not exploited in [31]. We propose to use the Alternating Direction Method of Multipliers (ADMM) algorithm [32] to post-process the hierarchy estimation. The usage of ADMM was proposed in [42] for the centralized setting. Our method applies this to LDP, and has two additional differences from [42]. First, we use L_2 norm in the objective function because the noise by CFO is well approximated by Gaussian noise, and minimizing L_2 norm achieves MLE. In the centralized setting, Laplace noise is used, and L_1 norm is minimized in [42]. Second, we pose an additional constraint

that the estimates sum up to n, which is known in LDP setting. In the setting considered in [42], n is unknown.

The HH-ADMM Algorithm. Given a constant vector $\tilde{\mathbf{x}}$, ADMM is an efficient algorithm that aims to find $\hat{\mathbf{x}}$ that satisfies the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\hat{\mathbf{x}} - \tilde{\mathbf{x}}\|_2^2 \\ \text{subject to} \quad & \mathbf{A}\hat{\mathbf{x}} = 0, \quad \hat{\mathbf{x}} \succcurlyeq 0, \quad \hat{\mathbf{x}}_0 = 1 \end{aligned} \tag{3.3}$$

In the hierarchy histogram case of LDP, $\tilde{\mathbf{x}}$ represents the concatenation of estimates from all the layers, where $\tilde{\mathbf{x}}_0$ is the root. $\hat{\mathbf{x}}$ is the post-processed estimates. The hierarchical constraints state that the estimate of each internal node should be equal to the sum of estimates of its children nodes. This can be represented by an equation $\mathbf{A}\hat{\mathbf{x}} = 0$, where $\mathbf{A}$ has one row for each internal node and one column for each node, and a_{ij} is defined as:

$$a_{ij} = \begin{cases} 1, & \text{if } i = j \\ -1, & \text{node } j \text{ is a child of node } i \\ 0, & \text{otherwise} \end{cases}$$

The optimization problem (3.3) improves the estimation by enforcing the non-negativity ($\hat{\mathbf{x}} \succcurlyeq 0$) and sum-up-to-1 ($\hat{\mathbf{x}}_0 = 1$) compared with [31]. Because of the limit of space, we refer the readers who want to know the detail of derivation to [42] for more information.

3.5 Square Wave and Expectation Maximization with Smoothing

The methods we presented in Section 3.4 use CFO protocols as black-boxes and do not fully exploit the ordered nature of the domains. We propose a new approach that uses a Square Wave reporting mechanism with post-processing conducted using Expectation Maximization with Smoothing (EMS).

3.5.1 General Wave Reporting

We first study a family of randomized reporting mechanisms that we call General Wave mechanisms. The intuition behind this approach is to try to increase the probability that a noisy reported value carries meaningful information about the input. This is also the implicit goal driving the development of CFO protocols beyond GRR. In GRR, one reports a value in $\mathcal{D}$. Intuitively, if the reported value is the true value, then the report is a "useful signal", as it conveys the extract correct information about the true input. If the reported value is not the true value, the report is in some sense noise that needs to be removed. The probability that a useful signal is generated is $p = \frac{e^\epsilon}{e^\epsilon + d - 1}$, where $d = |\mathcal{D}|$ is the size of the domain. When d is large, p is small, and GRR performs poorly. The essence of OLH and other CFO protocols is that one reports a randomly selected set of values, where one's true value has a higher probability of being selected than other values. In some sense, each "useful signal" is less sharp, since it is a set of values, but there is a much higher probability that a useful signal is transmitted.

Exploiting the ordinal nature of the domain, we note that a report that is different from but close to the true value v also carries useful information about the distribution. Therefore, given input v, we can report values closer to v with a higher probability than values that are farther away from v.

Without loss of generality, we assume that $\mathcal{D} = [0, 1]$ consists of floating point numbers between 0 and 1. The random reporting mechanism can be defined by a family of probability density functions (PDF) over the output domain, with one PDF for each input value. We denote the output probability density function for v as $\mathbf{M}_v(\tilde{v}) = \mathsf{Pr}\left[\Psi(v) = \tilde{v}\right]$.

Following the above intuition, we want $\mathbf{M}_v(\tilde{v})$ to satisfy the property that $\mathbf{M}_v(\tilde{v}) = q$ when $|\tilde{v} - v| > b$, and $q \leq \mathbf{M}_v(\tilde{v}) \leq e^\epsilon q$ when $|\tilde{v} - v| \leq b$, where b is a parameter to be chosen. To ensure that for values close to the two ends, the range of near-by values is the same, we enlarge the output domain $\tilde{\mathcal{D}} = [-b, 1 + b]$. We formalize the idea as the following general wave mechanism.

Definition 3.5.1 (General Wave Mechanism (GW))**.** *With input domain $\mathcal{D} = [0, 1]$ and output domain $\tilde{\mathcal{D}} = [-b, 1 + b]$, a randomization mechanism $\Psi : \mathcal{D} \to \tilde{\mathcal{D}}$ is an instance*

of general wave mechanism if for all $v \in \mathcal{D}$, there is a wave function $W : \mathbb{R} \to [q, \mathrm{e}^{\epsilon} q]$ with constants $q > 0$ and $\epsilon > 0$, such that the output probability density function $\mathbf{M}_v(\tilde{v}) = W(\tilde{v} - v)$:

1. *$W(z) = q$ for $|z| > b$;*
2. *$\int_{-b}^{b} W(z)\ dz = 1 - q$.*

Theorem 3.5.1. GW *satisfies ϵ-LDP.*

Proof. For any two possible input value $v_1, v_2 \in \mathcal{D}$ and any set of possible output $T \subseteq \tilde{\mathcal{D}}$ of GW, we have $\frac{\Pr[\mathsf{GW}(v_1) \in T]}{\Pr[\mathsf{GW}(v_2) \in T]} = \frac{\int_{\tilde{v} \in T} \mathbf{M}_{v_1}(\tilde{v}) d\tilde{v}}{\int_{\tilde{v} \in T} \mathbf{M}_{v_2}(\tilde{v}) d\tilde{v}}$. By definition for all $v_1, v_2 \in \mathcal{D}$ and $T \subset \tilde{\mathcal{D}}$ we have $\frac{\Pr[\mathsf{GW}(v_1) \in T]}{\Pr[\mathsf{GW}(v_2) \in T]} \leq \frac{\int_{\tilde{v} \in T} \mathrm{e}^{\epsilon} q\ d\tilde{v}}{\int_{\tilde{v} \in T} q\ d\tilde{v}} = \mathrm{e}^{\epsilon}$. □

3.5.2 The Square Wave mechanism

GW can have different wave shapes. An intriguing question is what shape should be used. Following the same intuition in [26], given different values $v \neq v'$, if $\mathbf{M}_v$ and $\mathbf{M}_{v'}$ are identical, then there is no way to distinguish those different input values. Therefore, the hope is that the farther apart $\mathbf{M}_v$ and $\mathbf{M}_{v'}$ are, the easier it is to tell them apart. We use the difference between two output distributions, Wasserstein (a.k.a., earth-mover) distance as the utility metric. Based on this, we find the Square Wave mechanism, where supports for $[v - b, v + b]$ are the same, is optimal. We also empirically compare GW of other shapes with Square Wave mechanism in Section 3.6.4. The experimental results support our intuition.

Specification of Square Wave Reporting. The Square Wave mechanism SW is defined as:

$$\forall v \in \mathcal{D}, \tilde{v} \in \tilde{\mathcal{D}},\ \mathbf{M}_v(\tilde{v}) = \begin{cases} p, & \text{if } |v - \tilde{v}| \leq b\ , \\ q, & \text{otherwise}\ . \end{cases} \tag{3.4}$$

By maximizing the difference between p and q while satisfying the total probability adds up to 1, the values p, q can be derived as:

$$p = \frac{\mathrm{e}^{\epsilon}}{2b\mathrm{e}^{\epsilon} + 1}\ , \quad q = \frac{1}{2b\mathrm{e}^{\epsilon} + 1}\ .$$

For each input v, the probability mass distribution for the perturbed output looks like a square wave, with the high plateau region centered around v. We thus call it the Square Wave (SW) reporting mechanism.

Theorem 3.5.2. *For any fixed b and ϵ, the* SW *is the* GW *that maximizes the Wasserstein distance between any two output distributions of two different inputs.*

Theorem 3.5.2 can be proved by using the following Lemma 1 and Lemma 2.

Lemma 1. *Given $v_1, v_2 \in \mathcal{D}$ as inputs to general wave mechanism, where $v_2 > v_1$ and let $\Delta = v_2 - v_1 > 0$, the Wasserstein distance between the output distributions of general wave mechanism is $\Delta(1 - (2b+1)q)$.*

Proof. Given two different input values v_1 and v_2 which satisfy $v_2 - v_1 = \Delta > 0$, let $\mathbf{M}_{v_1}$ and $\mathbf{M}_{v_2}$ are the corresponding output distributions. Define a function $\textsc{diff}(z)$ as the following:

$$\textsc{diff}(z) = \begin{cases} 0\ , & \text{if } z \le -b \\ 1-(2b+1)q\ , & \text{if } z \ge b \\ \int_{-b}^{z}(W(z')-q)\ dz'\ , & \text{otherwise.} \end{cases}$$

The cumulative function of SW can be written as

$$\mathbf{P}(\mathbf{M}_v, \tilde{v}) = (b+\tilde{v})q + \mathrm{DIFF}(\tilde{v}-v)$$

Therefore,

$$\begin{aligned}\int_{-b}^{1+b} \mathbf{P}(\mathbf{M}_v, \tilde{v})d\tilde{v} =& \frac{q}{2}(1+2b)^2 + \int_{-b}^{b}\mathrm{DIFF}(z)dz \\ &+ (1-(2b+1)q)(1-v)\ .\end{aligned}$$

Following the definition of Wasserstein distance of one dimensional data with ℓ_1 norm in Section 3.3, and as $\mathbf{P}(\mathbf{M}_{v_1}, \tilde{v}) \geq \mathbf{P}(\mathbf{M}_{v_2}, \tilde{v})$ for all $\tilde{v}$, it follows that

$$\begin{aligned} W_1(\mathbf{M}_{v_1}, \mathbf{M}_{v_2}) &= \int_{\tilde{\mathcal{D}}} |\mathbf{P}(\mathbf{M}_{v_1}, \tilde{v}) - \mathbf{P}(\mathbf{M}_{v_2}, \tilde{v})| d\tilde{v} \\ &= \int_{-b}^{1+b} (\mathbf{P}(\mathbf{M}_{v_1}, \tilde{v}) - \mathbf{P}(\mathbf{M}_{v_2}, \tilde{v}))\, d\tilde{v} \\ &= (1 - (2b+1)q)\Delta \ . \end{aligned}$$

□

Lemma 1 shows that we need to minimize q if we want to maximize the Wasserstein distance between any two output distributions. Thus, we have the following lemma.

Lemma 2. *For any fixed b and ϵ, the minimum q for general wave mechanism is $q = \frac{1}{2b\mathrm{e}^\epsilon + 1}$, which can be achieved if and only if the mechanism is* SW.

Proof. By criteria of the definition of GW, we have

$$\begin{aligned} 1 &= q + \int_{-b}^{b} W(z)dz \leq 1 + (2b)\mathrm{e}^\epsilon q \\ \Rightarrow q &\geq \frac{1}{2b\mathrm{e}^\epsilon + 1} \end{aligned}$$

We have equality iff $\mathbf{M}_v(\tilde{v}) = \mathrm{e}^\epsilon q$ for all $\tilde{v} \in [v-b, v+b]$, which turns out to be SW. □

Comparison with PM Mechanism. Square Wave (SW) reporting is similar to the Piecewise Mechanism (PM) for mean estimation [39] (see Section 3.2.2). PM directly sums up the randomized reports to estimate the mean of distribution, while the outputs of SW are used to reconstruct the whole distribution (the reconstruction method will be described in Subsection 3.5.5). Driven by the different focus, the reporting mechanisms are also different. PM has to be unbiased for mean estimation, so the input values are not always at the center

of high probability region. For example, given input $v = -1$, the high probability range in PM is $[-\frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}, -1]$.

Communication cost. With SW, each report consists of a single floating point number. The communication cost is thus a small constant for each user, similar to protocols such as GRR and OLH.

3.5.3 Choosing b

An important parameter to choose for the Square Wave reporting mechanism is b. In Square Wave reporting, a value that is within b of true input is reported with a probability that is e^{ϵ} times the probability that a "far" value is reported. The optimal choice of b depends on the privacy parameter ϵ. For a larger ϵ, a smaller b is preferred. When ϵ goes to infinity, a value of $b \to 0$ leads to total recovery of input distribution, and any $b > 0$ leads to information loss. Intuitively, the optimal choice of b also depends on the input distribution. For a distribution with probability density concentrated at one point, one would prefer smaller b. For a distribution with more or less evenly distributed probability density, one would prefer a larger b. However, since we do not know the distribution of the private values, we want to choose a b value independent of the distribution, but can perform reasonably well over different distributions.

We choose b to maximize the upper bound of mutual information between the input and output of the Square Wave reporting. We also empirically study the effect of varying b (see Section 3.6.4). The experimental results show that choosing b by this method results in optimal or close to optimal choices of b.

Let V and $\tilde{V}$ be the input and output random variables representing the input and output of SW, respectively. The mutual information between V and $\tilde{V}$ can be represented by the difference between differential entropy and conditional differential entropy of V and $\tilde{V}$:

$$I(V, \tilde{V}) = h(V) - h(V|\tilde{V}) = h(\tilde{V}) - h(\tilde{V}|V) \ .$$

The quantity $I(V, \tilde{V})$ depends on the input distribution, which we want to avoid. Therefore, we consider an upper bound of $I(V, \tilde{V})$, which is achieved when $\tilde{V}$ is uniformly distributed on $\tilde{\mathcal{D}}$. Let U be the random variable that is uniformly distributed in $\tilde{\mathcal{D}}$. Because $h(\tilde{V}) \leq h(U)$, we have:

$$I(V, \tilde{V}) \leq h(U) - h(\tilde{V}|V). \tag{3.5}$$

In (3.5), the first term of RHS is

$$h(U) = \log(2b + 1).$$

The second term of RHS only depends on SW:

$$\begin{aligned} h(\tilde{V}|V) &= -\int_v \mathsf{Pr}\left[V = v\right](2bp \log p + q \log q) \\ &= -(2bp \log p + q \log q) \\ &= -\frac{2b\epsilon e^\epsilon}{2be^\epsilon + 1} + \log(2be^\epsilon + 1)\ . \end{aligned}$$

So the mutual information is determined by a function of b,

$$\log\left(\frac{2b+1}{2be^\epsilon + 1}\right) + \frac{2b\epsilon e^\epsilon}{2be^\epsilon + 1}\ .$$

By making its derivative to 0, we get

$$b = \frac{\epsilon e^\epsilon - e^\epsilon + 1}{2e^\epsilon(e^\epsilon - 1 - \epsilon)}\ .$$

Note that b is a non-increasing function with ϵ. When ϵ goes to ∞, b goes to 0. When ϵ goes to 0, b goes to $1/2$, which leads to an output domain that doubles the size of the input domain, and for each input value, half of the output domain are considered "close" to the input value.

3.5.4 Bucketizing

The aggregator receives perturbed reports from users and needs to reconstruct the distribution on $\mathcal{D}$. Our approach performs this reconstruction on a discretized domain, i.e., histograms over the domain. The bucketization step can be performed either before or after applying the randomization step. We discuss the two approaches below. In experiments, we use the "randomize before bucketize" approach.

"Randomize before bucketize" (R-B). Here each user possesses a floating point number in $\tilde{\mathcal{D}} = [0, 1]$, applies the Square Wave mechanism in Section 3.5.2, and sends the result to the aggregator. The aggregator receives values in $\tilde{\mathcal{D}} = [-b, 1 + b]$, discretizes the reported values into $\tilde{d}$ buckets in $\tilde{\mathcal{D}}$, and constructs a histogram with $\tilde{d}$ bins. Using the method in Section 3.5.5, the aggregator can reconstruct an estimated input histogram of d bins. In experiments, we set $\tilde{d} = d$ for simplicity.

We compare the results of choosing different $\tilde{d}$ in Section 3.6.4, and found that the results are similar so long as $\tilde{d}$ does not deviate far from $\sqrt{N}$.

"Bucketize before randomize" (B-R) or discrete input domain. Alternatively, a user can perform the discretization step first, and then perform randomization. The SW mechanism can be naturally applied in a discrete domain as well. Assume input domain size is $d = |\mathcal{D}|$, discrete SW mechanism has output domain size $\tilde{d} = |\tilde{\mathcal{D}}| = d + 2b$, and randomizes input values as the following:

$$\forall v \in \mathcal{D}, \tilde{v} \in \tilde{\mathcal{D}},\ \Pr[\mathsf{SW}(v) = \tilde{v}] = \begin{cases} p, & \text{if } |v - \tilde{v}| \le b \\ q, & \text{otherwise,} \end{cases}$$

where $p = \frac{e^\epsilon}{(2b+1)e^\epsilon + d - 1}$ and $q = \frac{1}{(2b+1)e^\epsilon + d - 1}$. In this case, one can set $b = \left\lfloor \frac{\epsilon e^\epsilon - e^\epsilon + 1}{2e^\epsilon(e^\epsilon - 1 - \epsilon)} d \right\rfloor$.

The above discrete SW mechanism can also be applied when the input domain is already discrete (e.g., age). We conducted experiments comparing doing R-B versus B-R, and found that they are very similar.

3.5.5 Estimating Distribution from Reports

The aggregator receives perturbed values and faces an estimation problem. Note that post-processing of the output of a mechanism that satisfies differential privacy (the perturbed values from users) does not affect its privacy guarantee [43].

Without relying on any prior knowledge of the actual distribution, the natural approach is to conduct Maximum Likelihood Estimation (MLE). We use a $\tilde{d} \times d$ matrix $\mathbf{M}$ to characterize the randomization process. More specifically, the matrix $\mathbf{M} \in [0,1]^{\tilde{d} \times d}$ denotes the transformation probabilities, where $\mathbf{M}_{j,i}$ represents the probability of output value falling in bucket $\tilde{B}_j$, $j \in [\tilde{d}]$, given input in bucket B_i, $i \in [d]$, (assuming the input data fall uniformly at random within bucket B_i). Each column of $\mathbf{M}$ sums up to 1.

Expectation-Maximization (EM) Algorithm. Given the probability matrix $\mathbf{M}$ as defined above, we can use an Expectation-Maximization (EM) algorithm to reconstruct the distribution. The aggregator receives n randomized values from users, which are denoted as $\tilde{\mathbf{v}} = \{\tilde{v}_1, \ldots, \tilde{v}_n\}$, and finds $\hat{\mathbf{x}}$ that maximizes the log-likelihood $L(\hat{\mathbf{x}}) = \ln \mathsf{Pr}[\tilde{\mathbf{v}}|\hat{\mathbf{x}}]$.

Let n_j be the number of values in $\tilde{B}_j$ is reported. The EM algorithm for post-processing the square wave reporting is shown in Algorithm 1. Note that there are existing works that use EM to post-process results of CFO (e.g., [44, 45]), but our proposed EM algorithm takes aggregated results and is thus more efficient. Because of limitation of space, we omit the derivation of EM algorithm.

Theorem 3.5.3. *The EM algorithm converges to the maximum-likelihood (ML) estimator of the true frequencies* $\mathbf{x}$.

Proof. To prove EM algorithm converges to the maximum likelihood estimator, it is enough to show the log-likelihood function is concave [46]. In the context of our problem

$$L(\mathbf{x}) = \ln \mathsf{Pr}[\tilde{\mathbf{v}}|\mathbf{x}] = \ln \prod_{k=1}^{n} \mathsf{Pr}[\tilde{v}_k|\mathbf{x}]$$
$$= \sum_{k=1}^{n} \ln \left(\sum_{i=1}^{d} \mathbf{x}_i \mathsf{Pr}[\tilde{v}_k | v \in B_i] \right),$$

Algorithm 1 Post-processing EM algorithm

Input: $\mathbf{M}, \tilde{\mathbf{v}}$

Output: $\hat{\mathbf{x}}$

while not converge **do**

E-step: $\forall i \in \{1, ..., d\}$,

$$P_i = \hat{\mathbf{x}}_i \sum_{j \in [\tilde{d}]} n_j \frac{\mathsf{Pr}\left[\tilde{v} \in \tilde{B}_j | v \in B_i, \hat{\mathbf{x}}\right]}{\mathsf{Pr}\left[\tilde{v} \in \tilde{B}_j | \hat{\mathbf{x}}\right]}$$
$$= \hat{\mathbf{x}}_i \sum_{j \in [\tilde{d}]} n_j \frac{\mathbf{M}_{j,i}}{\sum_{k=1}^{d} \mathbf{M}_{j,k} \hat{\mathbf{x}}_k}$$

M-step: $\forall i \in \{1, ..., d\}$,

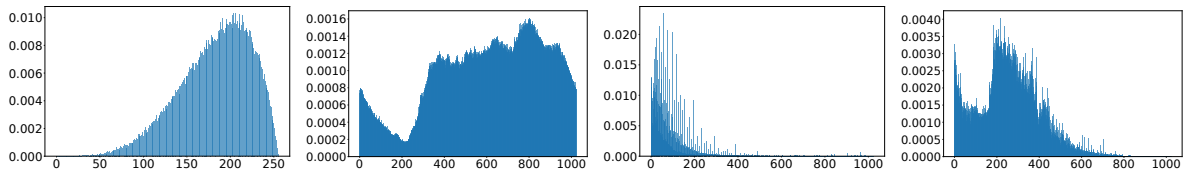
$$\hat{\mathbf{x}}_i = \frac{P_i}{\sum_{k'=1}^{d} P_{k'}}$$

end while

Return $\hat{\mathbf{x}}$

where $\mathsf{Pr}\left[\tilde{v}_k | v \in B_i\right]$ are constants determined by SW method. Thus, $L(\mathbf{x})$ is a concave function. □

Stopping Criteria. Through experiments, we have observed that the result of applying EM is highly sensitive to the parameter controlling terminating condition. If EM terminates too early, the reconstructed distribution is still far from the true one. If EM terminates too late, while the reconstructed distribution does fit the observation better (higher likelihood), it is also getting farther away from the true distribution to fit the noise. One of the most common stopping criteria for EM algorithm is checking whether the relative improvement of log-likelihood is small [44]. Namely, when $\left|L(\hat{\mathbf{x}}^{(t+1)}) - L(\hat{\mathbf{x}}^{(t)})\right| < \tau$ for some small positive number τ, EM algorithm stops. The choice of τ depends on many factors, including the smoothness of distribution and the amount of noise added by the square wave distribution. Empirically, we find that if we set τ proportional to e^{ϵ}, EM algorithm generally performs better than the one using a fixed τ. However, on some datasets that have a smoother distribution, the recovered result still over-fits the noise. Several of our attempts at finding a



(a) Beta(5, 2) (b) Taxi pickup time (c) Income (d) Retirement

Figure 3.1. Normalized frequencies of datasets for experiments.

stopping condition that make EM perform well consistently did not succeed. This motivates us to apply smoothing in EM.

EMS Algorithm. Using prior knowledge in estimation can make results less sensitive to the noise and more accurate than MLE solution. By the nature of numerical domain, adjacent numerical values' frequencies should not vary dramatically. With this observation, we can add a smoothing step after the M-step in the EM algorithm, boosting the accuracy with prior knowledge. We call the EM algorithm with smoothing steps as EMS algorithm. The idea of adding smoothing step into EM algorithm dates back to 1990s [33, 34] in the context of positron emission tomography and image reconstruction. The authors showed that a simple local smoothing method, the weight average with binomial coefficients of a bin value and the values of its nearest neighbours, could improve the estimation dramatically. We adopt this smoothing method. That is, after the M-step, the smoothing step will average each estimate with its adjacent ones with binomial coefficients $(1, 2, 1)$:

$$\hat{\mathbf{x}}_i = \frac{1}{2}\hat{\mathbf{x}}_i + \frac{1}{4}\left(\hat{\mathbf{x}}_{i-1} + \hat{\mathbf{x}}_{i+1}\right).$$

It was proved that adding the smoothing step is equivalent to adding a regularization term penalizing the spiky estimation [33], which can be viewed as applying Bayesian inference with a prior that prefers smoother distribution to jagged ones [47]. In more recent work, the idea of EMS is also applied to spatial data [48] and biophysics data [49].

3.6 Experiments

3.6.1 Experimental Setup

Datasets. We use the following datasets to conduct our experiments. One of them is synthetic, and the other three are real world datasets. All of them consist of numerical values. For CFO based methods, we discretize the values to the same granularity as the output of SW with EMS/EM method. Also, in order to compare with HH and HH-ADMM, which have optimal branching factor close to 4 [31], we choose the granularity (number of buckets in histogram) to be power of 4.

Synthetic Beta(5, 2) dataset. Originally, the distribution is in the continuous domain $[0, 1]$. One hundred thousand samples are generated. In experiments, we reconstruct the histogram with 256 buckets for all methods.

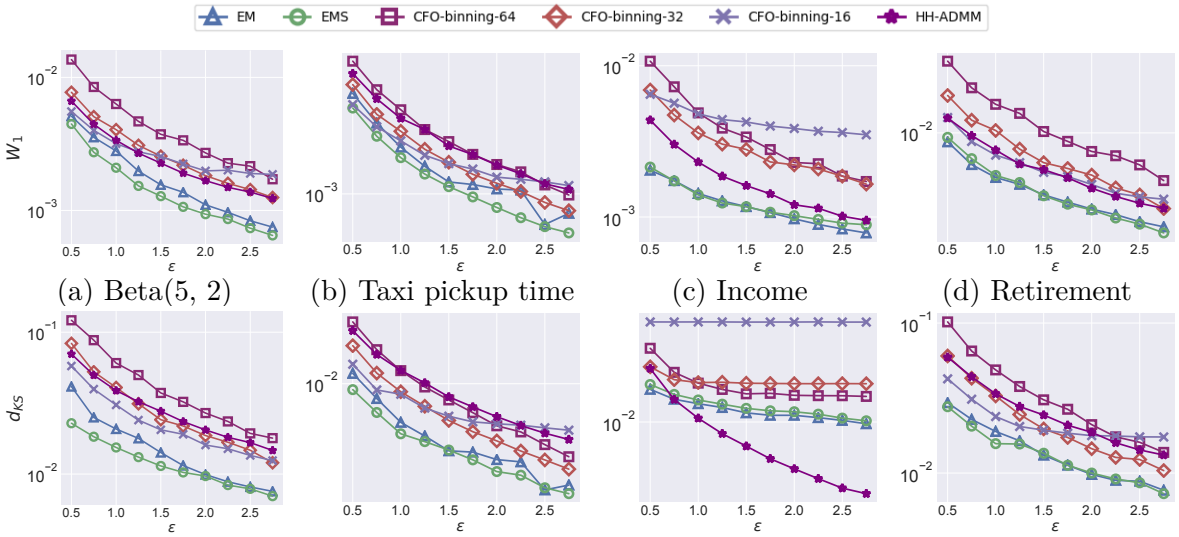
Taxi dataset's attribute pick-up time. Taxi pickup time dataset comes from 2018 January New York Taxi data [50]. Originally, the dataset contains the pickup time in a day (in seconds). We map the values into $[0, 1]$. There are $2,189,968$ samples in the dataset. In experiments, all estimated histograms have 1024 buckets.

Income dataset. We use the income information of the 2017 American Community Survey [51]. The data range is $[0, 1563000)$. We extract the values that are smaller than 524288 (i.e., 2^{19}) and map them into $[0, 1]$. There are $2,308,374$ samples after pre-processing. We choose to set the estimated histograms with 1024 buckets.

Retirement dataset. The San Francisco employee retirement plans data [52] contains integer values from $-28,700$ to $101,000$. We extract values that are non-negative and smaller than $60,000$, and map them into $[0, 1]$. There are $178,012$ samples after post-processing. In experiments, we reconstruct the histogram with 1024 buckets for all methods.

The income dataset is spiky because many people tend to report with precision up to hundreds or thousands (e.g., people are more likely to report \$3000 instead of more precise value like \$3050 or \$2980.)

Competitors. In the experiments, we consider several existing methods, including methods that obtain mean (PM, SR) and Hierarchy-based Methods (HH, HaarHRR). We also



(a) Beta(5, 2) (b) Taxi pickup time (c) Income (d) Retirement

(e) Beta(5,2) (f) Taxi pickup time (g) Income (h) Retirement

Figure 3.2. Results of distribution distances (first row: Wasserstein distance, second row: KS distance), varying ϵ.

Table 3.2. Methods and evaluated metrics.

Methods \ Metrics	Wasserstein and KS distance	Range Query	Mean & Variance	Quantile
SW with EMS/EM (this chapter)	✓	✓	✓	✓
HH-ADMM (this chapter)	✓	✓	✓	✓
CFO binning	✓	✓	✓	✓
HH [31] and HaarHRR [31]		✓		
PM [39] and SR [37]			✓	

consider CFO with binning methods, our proposed method HH-ADMM, and SW with EMS/EM. To the more specific, we summarize the methods and metrics evaluated in Table 3.2.

- Piecewise Mechanism (PM) and Stochastic Rounding (SR) (See Section 3.2.2) are only evaluated for mean and variance. They were designed for mean, and we adapted them to also estimate variance.

- For CFO with binning, we partition $\mathcal{D}$ into c consecutive, non-overlapping chunks. We consider $c = 16, 32, 64$, which are the best performing c values.

- For HH, HaarHRR and HH-ADMM, similar to [31], we use a branching factor of 4. HH and HaarHRR are only evaluated for range queries as they produce estimation results with negative values, which are not valid probability distributions. Other metrics are defined for probability distributions.

- For SW with EM and EMS as post-processing, we set $\tau = 10^{-3}\mathrm{e}^{\epsilon}$ for EM and $\tau = 10^{-3}$ for EMS.

As a brief overview of the experiment results, SW with EMS performs best with different privacy budgets and different metrics. HH-ADMM performs best on the income dataset

under some of the metrics. We also experimentally demonstrate the better utility of SW over other wave shapes in GW and the near-optimal choice of b for SW.

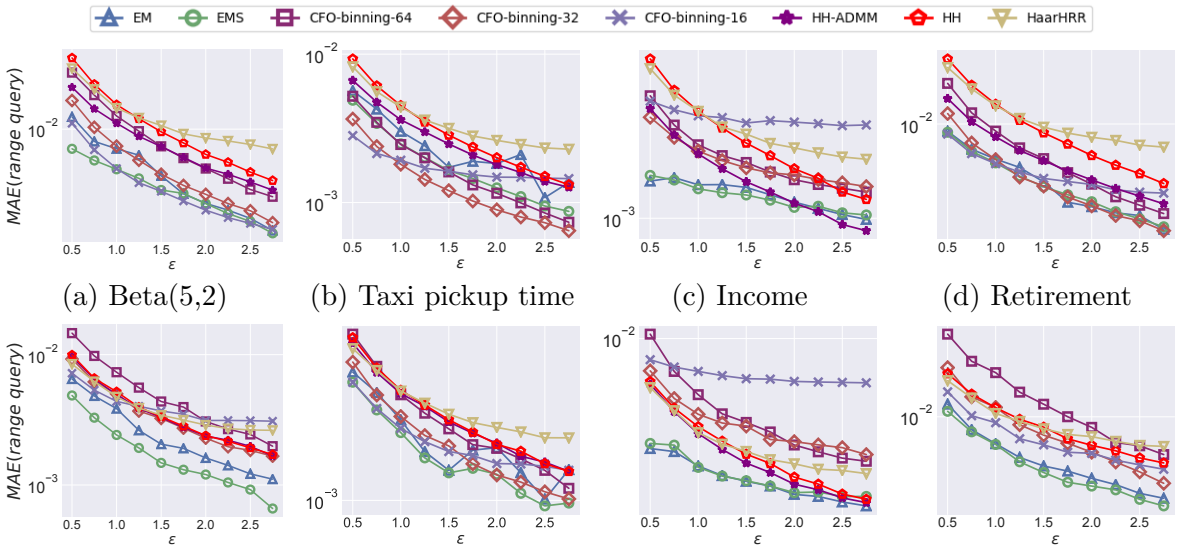
Evaluation methodology. The algorithms are implemented using Python 3.6 and Numpy 1.15; the experiments are conducted on a server with Intel Xeon 4108 and 128GB memory. For each dataset and each method, we repeat the experiment 100 times and take the mean.

3.6.2 Distribution Distance

We first evaluate metrics that capture the quality of the recovered distributions. Note that HH and haarHRR are not included (but HH-ADMM is) because HH or haarHRR does not result in valid distributions.

Wasserstein Distance. Figure 3.2a-3.2d shows the Wasserstein distance W_1 of reconstructed distribution and the true distribution. In most cases, SW with EMS performs best, followed by EM and HH-ADMM. For the CFO-binning methods, when ϵ is small, larger binning sizes (i.e., fewer number of bins) tend to give better performance. The lines for larger binning sizes flatten as ϵ increases, showing that the errors are dominated by biases due to binning. When ϵ becomes larger, CFO-binning with smaller bin sizes (i.e., more bins) becomes better. We observe that even if we could choose the optimal bin size empirically for each dataset and ϵ value, the result would still be worse than SW with EMS.

KS Distance. Figure 3.2e-3.2h show the K-S distance. For Beta, taxi pickup time and retirement datasets, SW with EMS generally performs the best, followed by EM. For the income dataset, HH-ADMM performs better than EM and EMS under this metric, especially under larger ϵ values. This is because the income dataset is more spiky, due to the fact that people tend to report income using round numbers. HH-ADMM is better at preserving some of the spikes in the distribution, whereas SW with EM or EMS will smooth the spikes. Since KS distance measures maximum difference at one point in CDF, HH-ADMM results in lower errors under KS distance, even though it produces higher error under Wasserstein Distance. For similar reason, CFO with larger bin size also perform poorly on the income dataset under KS distance.



(a) Beta(5,2) (b) Taxi pickup time (c) Income (d) Retirement

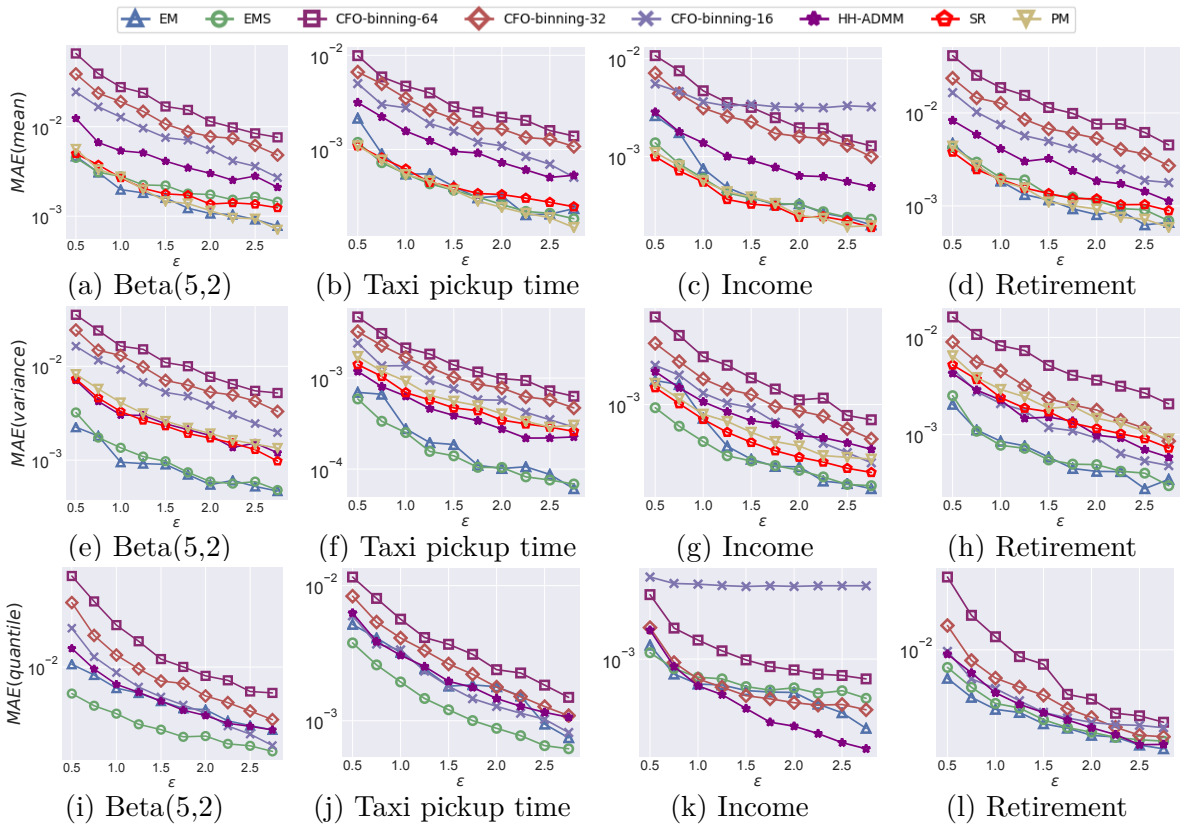
(e) Beta(5,2) (f) Taxi pickup time (g) Income (h) Retirement

Figure 3.3. MAE of random range query with range $\alpha = 0.1$ (first row) and $\alpha = 0.4$ (second row).

3.6.3 Semantic and Statistical Quantities

We compare the results of different methods using the range query and statistic quantities including mean, variance, quantiles. For mean and variance, we also consider the SR and PM, which were designed for mean estimations. All results are measured by Mean Absolute Error (MAE).

Range Query. The queries are randomly generated, but with fixed range sizes. Denote the left and right of the range as i and $i+\alpha$, we randomly generate $i \in [0, 1-\alpha]$ with $\alpha = 0.1$ and 0.4. The results in Figure 3.3 shows that SW with EMS outperforms HH and HaarHRR [31]. In fact, it is the best in most cases, except when $\alpha = 0.1$ in the taxi pickup time dataset and in low privacy region of income dataset. However, SW with EMS has performance similar to CFO-binning-64 when $\alpha = 0.1$ and still outperforms all the hierarchy-based approaches in taxi pickup time dataset. For the income dataset, EM and EMS performs well in high



(a) Beta(5,2) (b) Taxi pickup time (c) Income (d) Retirement

(e) Beta(5,2) (f) Taxi pickup time (g) Income (h) Retirement

(i) Beta(5,2) (j) Taxi pickup time (k) Income (l) Retirement

Figure 3.4. MAE for estimating mean (first row), variance (second row), and quantiles (third row).

privacy range (i.e., $\epsilon \leq 2$), while HH-ADMM performs best in low privacy range, followed by EM and EMS.

Mean Estimation. Results for mean estimation are showed in Figure 3.4a-3.4d. SR performs better than PM when ϵ is small, but worse when ϵ is larger. This is consistent with the analysis in [39]. Note that SR and PM devote all privacy budget to estimate mean. While SW with EMS can estimate the full distribution, it performs comparable to the best of SR and PM for estimating the mean. We also see that HH-ADMM has better performances than all other CFO-binning methods, but is still inferior to SW with EMS.

Variance Estimation. Although SR and PM are proposed for mean estimation, they can be modified to support variance estimation as well. Specifically, we randomly sample 50% of users to estimate mean first. The estimated mean is then broadcast to the remaining users.

Then each user compares his secret value and the received estimated mean, and reports the squared difference (i.e., $(v_i - \tilde{\mu})^2$) to the server, who averages them to obtain variance.

The experimental results are showed in Figure 3.4e- 3.4h. As we can see, the error of SR and PM is larger than EM or EMS in most cases. One reason is that only half of the users are used for variance estimation (the other half is necessary for mean estimation). The relative performance of other methods are similar to previous experiments.

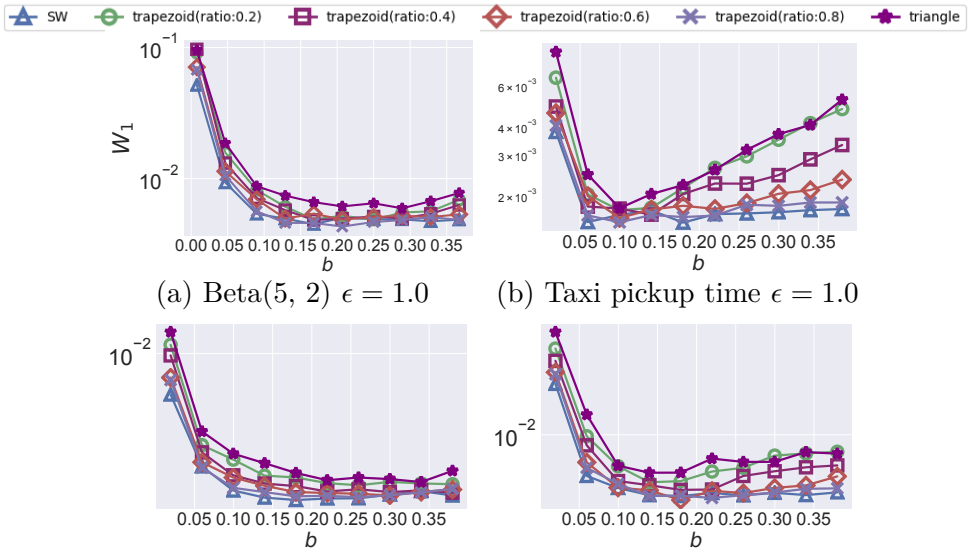
Quantile Estimation. Experimental results are shown in Figure 3.4i-3.4l. Ignoring the spiky income dataset for now, our proposed SW with EMS performs best. Moreover, we observe that SW with EM sometimes performs better but is not stable, because it is sensitive to parameters. HH-ADMM performs worse than SW, but close to the best of CFO with binning. For CFO with binning, because of the trade-off between estimation noise and the bias within the bins, larger bin sizes typically perform better in smaller ϵ ranges, while the smaller bin sizes narrows the gap as ϵ increases.

For the spiky income dataset, even for $\epsilon = 0.5$, larger bin sizes give worse utility (1 to 2 orders of magnitude) than other mechanisms. This also demonstrates that the optimal bin size is data-dependent. HH-ADMM successfully captures the spikiness of the dataset and thus performs the best.

3.6.4 Wave Shapes and Parameters

Here we compare the different shapes of General Wave (GW) with SW, and different parameters of SW.

Different shapes of wave in GW. In Section 3.5.2, we analytically show that SW is preferred because it maximizes the Wasserstein distance between output distributions. We empirically compare SW with other wave forms. We consider 5 other GW mechanism with different shape, including 4 trapezoid shapes and one triangle shape. The upper side to bottom side length ratio of trapezoid wave are $0.2, 0.4, 0.6$ and 0.8. The experimental results in Figure 3.5 show when $\epsilon = 1$, SW gives the best estimated distributions in terms



(a) Beta(5, 2) $\epsilon = 1.0$ (b) Taxi pickup time $\epsilon = 1.0$

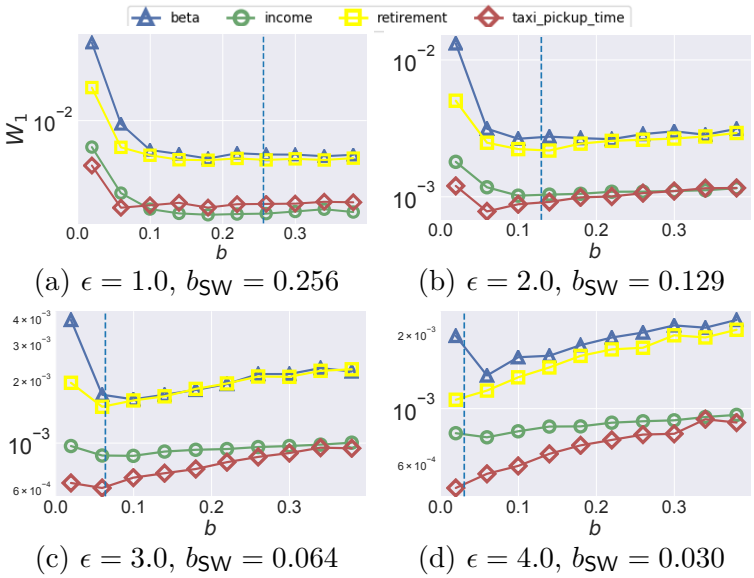
(c) Income $\epsilon = 1.0$ (d) Retirement $\epsilon = 1.0$

Figure 3.5. Comparison of different shapes of wave in GW. Ratios are the upper/lower length ratios for trapezoids.

of Wasserstein distance, no matter how we change b. As the ratio decreases, the recovery accuracy also degrades in general. The results support our intuition in Section 3.5.2.

SW with different b. In Section 3.5.3, we propose to use $b_{\mathsf{SW}} = \frac{\epsilon e^{\epsilon} - e^{\epsilon} + 1}{2e^{\epsilon}(e^{\epsilon} - 1 - \epsilon)}$. Figure 3.6 reports experimental results with different b. Our choice of b_{SW}, which is indicated as the vertical dotted line, is among the ones that provide best utility. We have also evaluated b on other metrics; the results give similar conclusion, and are omitted because of space limitation.

Bucketization granularity. To see what is the optimal bucketization granularity on different datasets, we choose 4 different numbers of buckets (256, 512, 1024 and 2048) then compare the Wasserstein distance between the estimated distributions and the true distributions. For simplicity, we use same number of buckets for both $\tilde{\mathcal{D}}$ and $\mathcal{D}$. The experimental results in Figure 3.7 show different datasets have different optimal bucketization granularity. For Beta(5,2), we have best result when the number of buckets is 256. For the other 3 datasets, dividing $\mathcal{D}$ into 1024 buckets can give us best performance in most cases.



(a) $\epsilon = 1.0$, $b_{\mathsf{SW}} = 0.256$ (b) $\epsilon = 2.0$, $b_{\mathsf{SW}} = 0.129$

(c) $\epsilon = 3.0$, $b_{\mathsf{SW}} = 0.064$ (d) $\epsilon = 4.0$, $b_{\mathsf{SW}} = 0.030$

Figure 3.6. Wasserstein distances between the true data and the estimation produced by EMS algorithm with fixed ϵ values and varying b from 0.01 to 0.38. Dotted vertical lines means the used b_{SW} in Section 3.5.3.

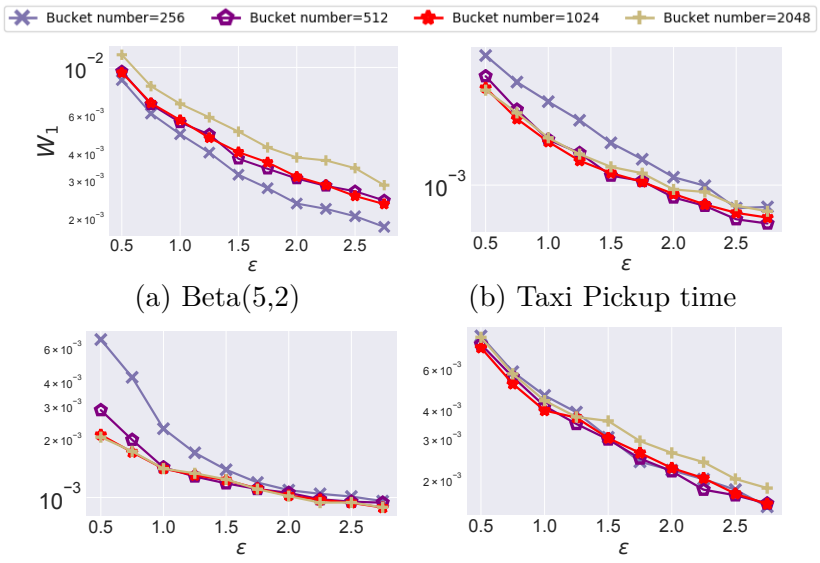
3.7 Related Work

We summarize some existing literature focusing on LDP as the following.

Categorical Frequency Oracle. One basic mechanism in LDP is to estimate frequencies of values. There have been several mechanisms [13, 23–26, 53] proposed for this task. Among them, [25] introduces OLH, which achieves low estimation errors and low communication costs. We develop new frequency oracles for numerical attributes.

Handling Ordinal/Numerical Data. When the data is ordinal, the straightforward approach is to bucketize the data and apply categorical frequency oracles. [54] considers distribution estimation, but with a strictly weaker privacy definition. There are also mechanisms that can handle numerical setting, but focusing on the specific task of mean estimation, i.e. SR [13, 37] and PM [39]. These two approaches have been discussed in Section 3.2 and compared in the experiments.

Post-processing. Given the result of a privacy-preserving algorithm, one can utilize the structural information to post-process it so that the utility can be improved. In the setting



(a) Beta(5,2) (b) Taxi Pickup time

(c) Income (d) Retirement

Figure 3.7. Wasserstein distance between estimated and true distribution with different bucketization granularity.

of centralized DP, Hay et al. [28] propose an efficient hierarchical method to minimize L_2 difference between the original result and the processed result. Besides that, the authors of [42] also consider the non-negativity constraint and propose to use ADMM to obtain result that achieves maximal likelihood. As ADMM is not efficient for high dimensional case, a gradient descent based algorithm is proposed [55].

In the LDP setting, [39] and [31] also consider the hierarchy structure and apply the technique of [28]. We propose to use ADMM instead of [28], which improves utility.

Without using the hierarchical constraint (only consider CFO), Jia et al. [56] propose to utilize external information about the dataset (e.g., assume it follows a power-law distribution), and Wang et al. [41] consider the constraints that the distribution is non-negative and sum up to 1. Bassily [57] and Kairouz et al. [58] study the post-processing for some CFO with MLE. Compared with those existing methods, our work is also a post-processing method but is applied to a new Square Wave reporting method and requires different techniques (such as EMS algorithm).

Shuffling. Recently, shuffle-DP [59–61] is introduced as an intermediate framework between centralized DP and LDP. By assuming there is a trusted third party who shuffles the

reports of a ϵ-LDP protocol before sending them to the aggregator, it is proved in [59] that the output of those shuffled reports will satisfy (ϵ', δ)-DP, for some $\epsilon' = O((1 \wedge \epsilon)e^{\epsilon}\sqrt{\log(1/\delta)/n})$. Our SW mechanism is fully compatible with shuffling, and its privacy amplification effects can be analyzed by the same tools introduced in [59].

3.8 Conclusion

We have studied the problem of reconstructing the distribution of a numerical attribute under LDP. We introduce HH-ADMM as an improvement to existing hierarchy-based methods. Most importantly, we propose the method of combining Square Wave reporting with Expectation Maximization and Smoothing. We show that Square Wave mechanism has the best utility among general wave mechanisms, and introduce techniques to choose the bandwidth parameter b by maximizing an upper bound of mutual information. Extensive experimental evaluations demonstrate that SW with EMS generally performs the best under a wide range of metrics. We expect these protocols and findings to help improving the deployment of LDP protocols to collect and analyse numerical information.

4. FEDERATED MATRIX FACTORIZATION WITH PRIVACY GUARANTEE

(A version of this chapter has been previously published in VLDB 2022 [62].)

4.1 Introduction

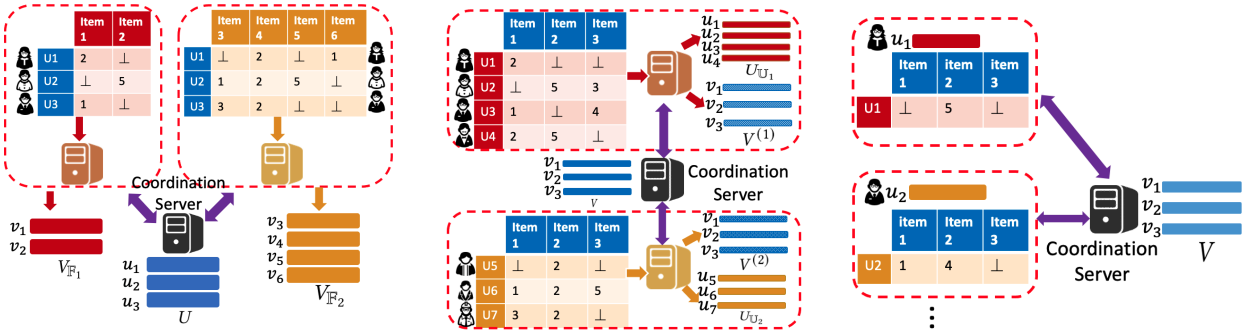
Federated learning (FL) has been applied as a paradigm for multiple parties to collaboratively train a model without directly sharing their data. FL can be categorized according to the data partition as summarized in [63]. Based on whether the data are partitioned by features or users, there are *vertical 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 chapter.

This chapter investigates the *matrix factorization* problem (e.g., [64–67]) 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 [68, 69].

4.1.1 Coordination and Trust Models

For federated matrix factorization, we assume that the coordination pattern is the classic server-client model [63, 70]. 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 [71], 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 [72]. Even if no local



(a) VFL setting (b) HFL setting (c) LFL setting

Figure 4.1. 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.

Table 4.1. Communication cost examples .

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 ($O(\log n + mp)$)

- Our default experiment settings is sync iteration $T = 100$, local iteration $T' = 10$ and embedding dimension $p = 20$.
- LFL-SGDMF relies on secure aggregation, requiring additional $O(\log n)$ for decoding .

data are shared explicitly and the secure multiparty computation techniques [73, 74] are applied in communication between parties, the final models are still vulnerable to inference attacks [75]. To provide provable resistance to such attacks, efforts have been made to protect federate learning with differential privacy (DP) [10].

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 chapter for different FL settings in Figures 4.1a-4.1c.

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 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 [76, 77]); 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 [78], 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 4.3-4.6.

4.1.2 Our Contributions

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

- We introduce an abstracted computation paradigm in Section 4.3 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 guar-

antees 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 [79, 80]. In this chapter, instead of using gradient clipping, we propose *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 4.4, 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 4.4, 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 4.5. 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` [18, 81] synchronizing privatized gradient from all parties in every iteration.
- Extending from the cross-silo HFL setting to cross-device LFL setting in Section 4.6, 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 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.

4.2 Problem Formulation

4.2.1 Matrix Factorization

The input to the *matrix factorization* (MF) [64] 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 $\mathbf{\Omega} = \{(i,j) | \mathbf{X}_{ij} \neq \perp\}$. 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 $\mathcal{L}(\mathbf{X}, U, V)$:

$$\frac{1}{|\mathbf{\Omega}|} \sum_{(i,j) \in \mathbf{\Omega}} \mathcal{L}_{i,j}(\mathbf{X}, U, V) = \frac{1}{|\mathbf{\Omega}|} \sum_{(i,j) \in \mathbf{\Omega}} (\mathbf{X}_{ij} - \langle u_i, v_j \rangle)^2, \tag{4.1}$$

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 4.3.

4.2.2 Differential Privacy in Matrix Factorization

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 [72, 82] have shown that sharing gradients directly can leak user privacy. In this chapter, we employ DP [10] 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 [3, 15].

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 chapter, 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 [83]. 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 chapter 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 [18]. 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 chapter, we propose *embedding clipping* (in Section 4.4) to bound the sensitivity of the gradient.

• *Trimming [76].* 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 chapter. The privacy losses are bounded for the three different settings in Section 4.4 to 4.6.

4.3 Federated Matrix Factorization Computation Paradigm

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.

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 [63] 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 4.1a, 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 $\{\mathbb{F}_1, \ldots, \mathbb{F}_s\}$ 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 $\mathbf{X} = [\mathbf{X}_{\mathbb{F}_1}, \ldots, \mathbf{X}_{\mathbb{F}_s}]$ 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 $\mathcal{L}\left(\mathbf{X}_{\mathbb{F}_k}, U, V_{\mathbb{F}_k}\right) = \frac{1}{|\mathbf{\Omega}_k|} \sum_{(i,j) \in \mathbf{\Omega}_k} \left(\mathbf{X}_{ij} - \langle u_i, v_j \rangle\right)^2$. The global loss function can be rewritten as: $\mathcal{L}\left(\mathbf{X}, U, V\right) = \sum_k \frac{|\mathbf{\Omega}_k|}{|\mathbf{\Omega}|} \mathcal{L}\left(\mathbf{X}_{\mathbb{F}_k}, U, V_{\mathbb{F}_k}\right) = \frac{1}{|\mathbf{\Omega}|} \sum_k \sum_{(i,j) \in \mathbf{\Omega}_k} \left(\mathbf{X}_{ij} - \langle u_i, v_j \rangle\right)^2$.

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

$$\nabla_U \mathcal{L}(\mathbf{X}, U, V) = \sum_k \frac{|\mathbf{\Omega}_k|}{|\mathbf{\Omega}|} \nabla_U \mathcal{L}(\mathbf{X}_{\mathbb{F}_k}, U, V_{\mathbb{F}_k}), \tag{4.2}$$

$$\nabla_{V_{\mathbb{F}_k}} \mathcal{L}(\mathbf{X}, U, V) = \frac{|\mathbf{\Omega}_k|}{|\mathbf{\Omega}|} \nabla_{V_{\mathbb{F}_k}} \mathcal{L}(\mathbf{X}_{\mathbb{F}_k}, U, V_{\mathbb{F}_k}). \tag{4.3}$$

Equation (4.2) indicates that the updates on the global user embeddings need to be the weighted average of the updates of local user embedding; Equation (4.3) suggests that the local updates on the item embeddings, although no need to be aggregated, need to be scaled by the a different ratio $|\mathbf{\Omega}_k|/|\mathbf{\Omega}|$ 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,

$\tau = |\mathbf{\Omega}|/(nm)$. 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 4.1b, 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 $\mathbf{\Omega}_k$. 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 $\mathcal{L}\left(\mathbf{X}_{\mathbb{U}_k}, U_{\mathbb{U}_k}, V\right) = \frac{1}{|\mathbf{\Omega}_k|} \sum_{(i,j)\in\mathbf{\Omega}_k} \left(\mathbf{X}_{ij} - \langle u_i, v_j \rangle\right)^2$, and the global loss function is $\mathcal{L}\left(\mathbf{X}, U, V\right) = \sum_k \frac{|\mathbf{\Omega}_k|}{|\mathbf{\Omega}|} \mathcal{L}\left(\mathbf{X}_{\mathbb{U}_k}, U_{\mathbb{U}_k}, V\right)$. 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.

4.3.1 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 4.2a. 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:

$$\text{VFL:} \quad U^{(k)} \leftarrow U^{(k)} - \gamma_t \mathbf{g}_U^{(k)}, V_{\mathbb{F}_k} \leftarrow V_{\mathbb{F}_k} - \frac{\gamma_t m_k}{m} \mathbf{g}_{V_{\mathbb{F}_k}}^{(k)}; \tag{4.4}$$

$$\text{HFL:} \quad U_{\mathbb{U}_k} \leftarrow U_{\mathbb{U}_k} - \frac{\gamma_t n_k}{n} \mathbf{g}_{U_{\mathbb{U}_k}}^{(k)}, V^{(k)} \leftarrow V_{\mathbb{F}_k} - \gamma_t \mathbf{g}_{V^{(k)}}^{(k)}. \tag{4.5}$$

Here we assume that all parties have about the same rating "density", $\frac{|\mathbf{\Omega}_k|}{|\mathbf{\Omega}|} \approx \frac{m_k}{m}$ or $\frac{|\mathbf{\Omega}_k|}{|\mathbf{\Omega}|} \approx \frac{n_k}{n}$. As mentioned in the previous subsection, the global loss function is a weighted average of the local ones. Thus, the weights are applied in (4.4) and (4.5) 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., [84, 85]), 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

Input: $\{\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(s)}\}$ 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 precomputation

(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: $\{\mathbf{X}_{\mathbb{F}_1}, \ldots, \mathbf{X}_{\mathbb{F}_s}\}$ 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 embedding clipping and DP noise; upload $U^{(k)}$ to server;

Replace (c3) in `FMF`: Fine-tune $V_{\mathbb{F}_k}$ locally for κ iterations with embedding clipping and DP noise;

(b) Changes `FMF` $\rightarrow$ `VFL-SGDMF`

Input: $\{\mathbf{X}_{\mathbb{U}_1}, \ldots, \mathbf{X}_{\mathbb{U}_s}\}$ 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` $\rightarrow$ `HFL-SGDMF`

Input: $\{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$ 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 $\tilde{\mathbf{g}}_V^{(i,t)}$ and invoke $\texttt{SA}_\omega$-$\texttt{report}(\tilde{\mathbf{g}}_V^{(i,t)})$; (Round 2) If receive $|\mathbb{U}^{(t,1)}|$ from server, upload $\bar{\boldsymbol{\xi}}^{(i,t)} = -\boldsymbol{\xi}^{(i,t)} + \boldsymbol{\xi}'^{(i,t)}$;

Replace (s2) in `FMF`: (Round 1) Invoke $\texttt{SA}_\omega$-`agg` to get $\tilde{\mathbf{g}}_V^{(t)} = \sum_{i \in \mathbb{U}^{(t,1)}} \tilde{\mathbf{g}}_V^{(i,t)}$; **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 $V^{(t)} = V^{(t-1)} - \frac{\gamma_t}{|\mathbb{U}^{(t,1)}|}\left(\tilde{\mathbf{g}}_V^{(t)} + \sum_{i \in [\mathbb{U}^{(t,2)}]} \bar{\boldsymbol{\xi}}_i\right)$;

Replace (c3) in `FMF`: Fine-tune local U_i;

(d) Changes `FMF` $\rightarrow$ `LFL-SGDMF`

Figure 4.2. Federated matrix factorization algorithms

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 4.3.1. *If the aforementioned assumptions hold, Algorithm* **FMF** *converges in* $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla\mathcal{L}\left(\mathbf{X},U,V\right)\right\|\right] = O\left(\sqrt{\frac{D}{T}(1+c)}\right)$ *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.*

FMF convergence proof

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 $\tau=\frac{\sum_{k=1}^{s}|\mathbf{\Omega}_k|}{nm}$. The loss function of our problem is

$$L(W)=\frac{1}{\sum_{k=1}^{s}|\mathbf{\Omega}_k|}\sum_{k=1}^{s}\sum_{(i,j)\in\mathbf{\Omega}_k}(\mathbf{X}_{ij}-\langle u_i,v_j\rangle)^2=\frac{1}{nm\tau}\sum_{k=1}^{s}\sum_{(i,j)\in\mathbf{\Omega}_k}(\mathbf{X}_{ij}-\langle u_i,v_j\rangle)^2 \tag{4.6}$$

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

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.

$$\nabla_U\mathcal{L}_{i,j}\left(U,V_{\mathbb{F}_k}\right)=-2(\mathbf{X}_{ij}-\langle u_i,v_j\rangle)\mathbf{e}_i^n v_j \tag{4.7}$$

$$\nabla_{V_{\mathbb{F}_k}}\mathcal{L}_{i,j}\left(U,V_{\mathbb{F}_k}\right)=-2(\mathbf{X}_{ij}-\langle u_i,v_j\rangle)\mathbf{e}_j^{m_k} u_i \tag{4.8}$$

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 t-th meta iteration as the following:

$$\bar{\mathbf{g}}_U^{(k,t)} = \nabla_U \mathcal{L}\left(\mathbf{X}_{\mathbb{F}_k}, U^{(k,t)}, V_{\mathbb{F}_k}^{(t)}\right) = \frac{1}{nm_k\tau} \sum_{(i,j)\in\mathbf{\Omega}_k} \nabla_U \mathcal{L}_{i,j}\left(U^{(k,t)}, V_{\mathbb{F}_k}^{(t)}\right), \tag{4.9}$$

$$\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} = \nabla_{V_{\mathbb{F}_k}} \mathcal{L}\left(\mathbf{X}_{\mathbb{F}_k}, U^{(k,t)}, V_{\mathbb{F}_k}^{(t)}\right) = \frac{1}{nm_k\tau} \sum_{(i,j)\in\mathbf{\Omega}_k} \nabla_{V_{\mathbb{F}_k}} \mathcal{L}_{i,j}\left(U^{(k,t)}, V_{\mathbb{F}_k}^{(t)}\right) \tag{4.10}$$

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.

$$\mathbf{g}_U^{(k,t,t')} = \frac{1}{n'm'_k\tau} \sum_{(i,j)\in I_k} \nabla_U \mathcal{L}_{i,j}\left(U^{(k,t,t')}, V_{\mathbb{F}_k}^{(t,t')}\right), \tag{4.11}$$

$$\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} = \frac{1}{n'm'_k\tau} \sum_{(i,j)\in I_k} \nabla_{V_{\mathbb{F}_k}} \mathcal{L}_{i,j}\left(U^{(k,t,t')}, V_{\mathbb{F}_k}^{(t,t')}\right) \tag{4.12}$$

The sampled gradient of U and V are unbiased estimate to the true gradient (with $I_k = \mathbf{\Omega}_k$), that is

$$\mathbb{E}_{I_k}\left[\mathbf{g}_U^{(k,t,t')}\right] = \bar{\mathbf{g}}_U^{(k,t,t')}, \quad \mathbb{E}_I\left[\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right] = \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \tag{4.13}$$

$$\bar{\mathbf{g}}_U^{(t)} = \frac{1}{m}\sum_{k=1}^{s} m_k \bar{\mathbf{g}}_U^{(k,t)}, \quad \bar{\mathbf{g}}_V^{(t)} = \begin{bmatrix} \frac{m_1}{m}\bar{\mathbf{g}}_{V_{\mathbb{F}_1}}^{(k,t)} \\ \vdots \\ \frac{m_s}{m}\bar{\mathbf{g}}_{V_{\mathbb{F}_s}}^{(k,t)} \end{bmatrix} \tag{4.14}$$

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:

$$\mathbb{E}_{I_k}\left[\left\|\mathbf{g}_U^{(k,t,t')} - \bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] \leq \sigma^2_{\mathbb{S}_{\mathtt{Sub}},U} \tag{4.15}$$

$$\mathbb{E}_{I_k}\left[\left\|\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} - \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right] \leq \sigma^2_{\mathbb{S}_{\mathtt{Sub}},V} \tag{4.16}$$

The updating rule can be written as the following:

$$U^{(k,t,t')} = U^{(k,t,t'-1)} - \gamma \mathbf{g}_U^{(k,t,t')} \tag{4.17}$$

$$V_{\mathbb{F}_k}^{(t,t')} = V_{\mathbb{F}_k}^{(t,t'-1)} - \frac{\gamma m_k}{m}\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} \tag{4.18}$$

$$U^{(t+1)} = \frac{1}{m}\sum_{k=1}^{s} m_k U^{(k,t,T')} \tag{4.19}$$

$$V^{(t+1)} = \begin{bmatrix} V_{\mathbb{F}_1}^{(t,T')} \\ \vdots \\ V_{\mathbb{F}_s}^{(t,T')} \end{bmatrix} \tag{4.20}$$

$$\tag{4.21}$$

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 $\sigma^2_{\mathbb{S}_{\mathtt{Sub}},U}$ and $\sigma^2_{\mathbb{S}_{\mathtt{Sub}},V}$. Both $\sigma^2_{\mathbb{S}_{\mathtt{Sub}},U}$ and $\sigma^2_{\mathbb{S}_{\mathtt{Sub}},V}$ goes to 0 as $\frac{n'm'_k}{nm_k} \to 1$. 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 4.3.1 in VFL case, we need to use the following two lemmas to bound the updates of U and V for each meta-iteration.

Lemma 3. *For Algorithm* **FMF** *in VFL case, with the same assumptions as Theorem 4.3.1 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*

$$- \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, U^{(t+1)} - U^{(t)} \right\rangle + \frac{\beta}{2}\left\|U^{(t+1)} - U^{(t)}\right\|_F^2\right] \tag{4.22}$$

$$\leq -\gamma T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|^2 + \gamma^2\beta T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\| \sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')}\right\|\right] \tag{4.23}$$

$$+ \frac{\beta\gamma^2 T'^2\sigma^2_{\mathbb{S}_{\text{Sub}},U}}{2} + \frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] \tag{4.24}$$

Lemma 4. *For Algorithm* **FMF** *in VFL case, with the same assumptions as Theorem 4.3.1 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*

$$- \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_V^{(t)}, V^{(t+1)} - V^{(t)} \right\rangle + \frac{\beta}{2}\left\|V^{(t+1)} - V^{(t)}\right\|_F^2\right] \tag{4.25}$$

$$\leq -\gamma T'\left\|\bar{\mathbf{g}}_V^{(t)}\right\|^2 + \sum_{k=1}^{s}\frac{T'\gamma^2 m_k^3\beta}{m^3}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\mathbf{g}_{V_{\mathbb{F}_k}^{(k,t,t')}}\right\|\right] \tag{4.26}$$

$$+ \sum_{k=1}^{s}\frac{\beta\gamma^2 m_k^2 T'^2\sigma^2_{\mathbb{S}_{\text{Sub}},V}}{2m^2} + \sum_{k=1}^{s}\frac{\beta T' m_k^2\gamma^2}{2m^2}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\mathbf{g}_{V_{\mathbb{F}_k}^{(k,t,t')}}\right\|^2\right] \tag{4.27}$$

Proof of Theorem 4.3.1. High level idea follows the proof in [86]. With smoothness condition,

$$\mathcal{L}\left(\mathbf{X}, W^{(t+1)}\right) - \mathcal{L}\left(\mathbf{X}, W^{(t)}\right) \tag{4.28}$$

$$\leq \left\langle \nabla_{W^{(t)}} \mathcal{L}\left(W^{(t)}\right), \begin{bmatrix} U^{(t+1)} - U^{(t)} \\ V^{(t+1)} - V^{(t)} \end{bmatrix} \right\rangle + \frac{\beta}{2} \left\| \begin{bmatrix} U^{(t+1)} - U^{(t)} \\ V^{(t+1)} - V^{(t)} \end{bmatrix} \right\|_F^2 \tag{4.29}$$

$$= \left\langle \bar{\mathbf{g}}_U^{(t)}, U^{(t+1)} - U^{(t)} \right\rangle + \frac{\beta}{2} \left\| U^{(t+1)} - U^{(t)} \right\|_F^2 \tag{4.30}$$

$$+ \left\langle \bar{\mathbf{g}}_V^{(t)}, V^{(t+1)} - V^{(t)} \right\rangle + \frac{\beta}{2} \left\| V^{(t+1)} - V^{(t)} \right\|_F^2 \tag{4.31}$$

By Lemma 3 and 4 :

$$\mathbb{E}\left[\mathcal{L}\left(\mathbf{X}, W^{(t+1)}\right)\right] - \mathcal{L}\left(\mathbf{X}, W^{(t)}\right) \tag{4.32}$$

$$\leq -\gamma T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\|^2 + \gamma^2 \beta T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\| \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} \right\| \right] \tag{4.33}$$

$$+ \frac{\beta \gamma^2 T'^2 \sigma^2_{\mathbb{S}_{\mathtt{Sub}},U}}{2} + \frac{\beta \gamma^2 T'}{2} \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} \right\|^2 \right] \tag{4.34}$$

$$- \gamma T' \left\| \bar{\mathbf{g}}_V^{(t)} \right\|^2 + \sum_{k=1}^{s} \frac{T' \gamma^2 m_k^3 \beta}{m^3} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\| \right] \tag{4.35}$$

$$+ \sum_{k=1}^{s} \frac{\beta \gamma^2 m_k^2 T'^2 \sigma^2_{\mathbb{S}_{\mathtt{Sub}},V}}{2m^2} + \sum_{k=1}^{s} \frac{\beta T' m_k^2 \gamma^2}{2m^2} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] \tag{4.36}$$

$$= -\gamma T' \left\| \bar{\mathbf{g}}_W^{(t)} \right\|^2 + \tag{4.37}$$

$$\gamma^2 \beta T' \left(\left\| \bar{\mathbf{g}}_U^{(t)} \right\| \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} \right\| \right] + \frac{T' \sigma^2_{\mathbb{S}_{\mathtt{Sub}},U}}{2} + \frac{1}{2} \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} \right\|^2 \right] \right. \tag{4.38}$$

$$\left. + \sum_{k=1}^{s} \frac{m_k^3}{m^3} \left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)} \right\| \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\| \right] + \sum_{k=1}^{s} \frac{m_k^2 T' \sigma^2_{\mathbb{S}_{\mathtt{Sub}},V}}{2m^2} + \sum_{k=1}^{s} \frac{m_k^2}{2m^2} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')} \right\|^2 \right] \right) \tag{4.39}$$

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

$$\mathbb{E}\left[\mathcal{L}\left(\mathbf{X}, W^{(t+1)}\right)\right] - \mathcal{L}\left(\mathbf{X}, W^{(t)}\right) \tag{4.40}$$

$$\leq -\gamma T' \left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2 + \gamma^2 \beta T' \left((T'-1)L_U^2 + \frac{T'\sigma^2_{\mathbb{S}_{\text{Sub}},U}}{2} + \frac{T'L_U^2}{2}\right) \tag{4.41}$$

$$+ \gamma^2 \beta T' \left((T'-1)\sum_{k=1}^{s}\frac{m_k^3}{m^3}L_V^2 + \sum_{k=1}^{s}\frac{T'\sigma^2_{\mathbb{S}_{\text{Sub}},V}m_k^2}{2m^2} + \sum_{k=1}^{s}\frac{T'L_V^2 m_k^2}{2m^2}\right) \tag{4.42}$$

$$\leq -\gamma T' \left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2 + \gamma^2 \beta T' \left((T'-1)L_U^2 + \frac{T'\sigma^2_{\mathbb{S}_{\text{Sub}},U}}{2} + \frac{T'L_U^2}{2}\right) \tag{4.43}$$

$$+ \gamma^2 \beta T' \left((T'-1)L_V^2\sum_{k=1}^{s}\frac{m_k^3}{m^3} + \frac{T'\sigma^2_{\mathbb{S}_{\text{Sub}},V}}{2}\sum_{k=1}^{s}\frac{m_k^2}{m^2} + \frac{T'L_V^2}{2}\sum_{k=1}^{s}\frac{m_k^2}{m^2}\right) \tag{4.44}$$

$$\leq -\gamma T' \left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2 + \gamma^2 \beta T' \left((T'-1)L_U^2 + \frac{T'\sigma^2_{\mathbb{S}_{\text{Sub}},U}}{2} + \frac{T'L_U^2}{2}\right) \tag{4.45}$$

$$+ \gamma^2 \beta T' \left((T'-1)L_V^2\sum_{k=1}^{s}\frac{m_k^3}{m^3} + \frac{T'\sigma^2_{\mathbb{S}_{\text{Sub}},V}}{2}\sum_{k=1}^{s}\frac{m_k^2}{m^2} + \frac{T'L_V^2}{2}\sum_{k=1}^{s}\frac{m_k^2}{m^2}\right) \tag{4.46}$$

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

$$\mathbb{E}\left[\mathcal{L}\left(\mathbf{X}, W^{(T)}\right)\right] - \mathcal{L}\left(\mathbf{X}, W^{(1)}\right) \tag{4.47}$$

$$\leq -\gamma T' \sum_{t=1}^{T}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_W^{(t)}\right\|^2\right] + \gamma^2 \beta T' T \left((T'-1)L_U^2 + \frac{T'\sigma^2_{\mathbb{S}_{\text{Sub}},U}}{2} + \frac{T'L_U^2}{2}\right) \tag{4.48}$$

$$+ \gamma^2 \beta T' T \left((T'-1)L_V^2 C_{V,1} + \frac{T'\sigma^2_{\mathbb{S}_{\text{Sub}},V}C_{V,2}}{2} + \frac{T'L_V^2 C_{V,2}}{2}\right), \tag{4.49}$$

$$\tag{4.50}$$

which implies

$$\gamma T' \sum_{t=1}^{T} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_W^{(t)} \right\|^2 \right] \tag{4.51}$$

$$\leq \mathcal{L}\left(\mathbf{X}, W^{(1)}\right) - \mathbb{E}\left[\mathcal{L}\left(\mathbf{X}, W^{(T)}\right) \right] + \gamma^2 \beta T' T \left((T'-1) L_U^2 + \frac{T' \sigma_{\mathbb{S}_{\text{Sub}},U}^2}{2} + \frac{T' L_U^2}{2} \right) \tag{4.52}$$

$$+ \gamma^2 \beta T' T \left((T'-1) L_V^2 C_{V,1} + \frac{T' \sigma_{\mathbb{S}_{\text{Sub}},V}^2 C_{V,2}}{2} + \frac{T' L_V^2 C_{V,2}}{2} \right) \tag{4.53}$$

$$\tag{4.54}$$

Taking the average of LHS:

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_W^{(t)} \right\|^2 \right] \leq \frac{\mathcal{L}\left(\mathbf{X}, W^{(1)}\right) - \mathbb{E}\left[\mathcal{L}\left(\mathbf{X}, W^{(T)}\right) \right]}{\gamma T' T} \tag{4.55}$$

$$+ \gamma \beta \left((T'-1) L_U^2 + \frac{T' \sigma_{\mathbb{S}_{\text{Sub}},U}^2}{2} + \frac{T' L_U^2}{2} \right) \tag{4.56}$$

$$+ \gamma \beta \left((T'-1) L_V^2 C_{V,1} + \frac{T' \sigma_{\mathbb{S}_{\text{Sub}},V}^2 C_{V,2}}{2} + \frac{T' L_V^2 C_{V,2}}{2} \right) \tag{4.57}$$

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 $\gamma = O\left(\sqrt{\frac{D}{\beta T (T')^2 (L_U^2 + L_V^2 C_{V,2} + \sigma_{\mathbb{S}_{\text{Sub}},U}^2 + \sigma_{\mathbb{S}_{\text{Sub}},V}^2 C_{V,2})}} \right)$, we can have

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_W^{(t)} \right\|^2 \right] = O\left(\frac{\sqrt{D\beta (L_U^2 + L_V^2 C_{V,2} + \sigma_{\mathbb{S}_{\text{Sub}},U}^2 + \sigma_{\mathbb{S}_{\text{Sub}},V}^2 C_{V,2})}}{\sqrt{T}} \right) \tag{4.58}$$

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

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_W^{(t)} \right\|^2 \right] = O\left(\frac{\sqrt{D\beta (L_U^2 + L_V^2 C_{V,2} + \sigma_{\mathbb{S}_{\text{Sub}},U}^2 + \sigma_{\mathbb{S}_{\text{Sub}},V}^2 C_{V,2})}}{\sqrt{T}} \right) \tag{4.59}$$

$$= O\left(\sqrt{\frac{D}{T} (1 + \sum_{k=1}^{s} \frac{m_k^2}{m^2})} \right) \tag{4.60}$$

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.

□

Proof of Lemma 3. 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

$$\left\langle \bar{\mathbf{g}}_U^{(t)}, U^{(t+1)} - U^{(t)} \right\rangle + \frac{\beta}{2} \left\| U^{(t+1)} - U^{(t)} \right\|_2^2 \tag{4.61}$$

$$= \frac{1}{m} \sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1} \left\langle \bar{\mathbf{g}}_U^{(t)}, \left(U^{(k,t,t'+1)} - U^{(k,t,t')} \right) \right\rangle + \frac{\beta}{2m^2} \left\| \sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1} \left(U^{(k,t,t'+1)} - U^{(k,t,t')} \right) \right\|^2 \tag{4.62}$$

$$= -\frac{\gamma}{m} \sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1} \left\langle \bar{\mathbf{g}}_U^{(t)}, \mathbf{g}_U^{(k,t,t')} \right\rangle + \frac{\beta\gamma^2}{2m^2} \left\| \sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1} \mathbf{g}_U^{(k,t,t')} \right\|^2 \tag{4.63}$$

By taking the expectation and the unbiased assumption, we have

$$\mathbb{E}\left[\mathbb{E}_{I_k} \left[\mathbf{g}_U^{(k,t,t')} | U^{(k,t,t')} \right] \right] = \mathbb{E}\left[\bar{\mathbf{g}}_U^{(k,t,t')} \right] \tag{4.64}$$

The expectation, (4.61) becomes

$$-\frac{\gamma}{m} \sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, \bar{\mathbf{g}}_U^{(k,t,t')} \right\rangle \right] + \frac{\beta\gamma^2}{2m^2} \mathbb{E}\left[\left\| \sum_{k=1}^{s} m_k \sum_{t'=0}^{T'-1} \mathbf{g}_U^{(k,t,t')} \right\|^2 \right] \tag{4.65}$$

The inner product in first term of the above equation:

$$\sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, \bar{\mathbf{g}}_U^{(k,t,t')} \right\rangle \right] \tag{4.66}$$

$$= \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, \bar{\mathbf{g}}_U^{(k,t,0)} + \bar{\mathbf{g}}_U^{(k,t,t')} - \bar{\mathbf{g}}_U^{(k,t,0)} \right\rangle \right] \tag{4.67}$$

$$\geq \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, \sum_{k=1}^{s} \frac{m_k}{m} \bar{\mathbf{g}}_U^{(k,t,0)} \right\rangle \right] + \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=0}^{T'-1} \mathbb{E}\left[\left\langle \bar{\mathbf{g}}_U^{(t)}, \bar{\mathbf{g}}_U^{(k,t,t')} - \bar{\mathbf{g}}_U^{(k,t,0)} \right\rangle \right] \tag{4.68}$$

$$\geq T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\|^2 - \left\| \bar{\mathbf{g}}_U^{(t)} \right\| \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t=1}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,t')} - \bar{\mathbf{g}}_U^{(k,t,0)} \right\| \right] \tag{4.69}$$

$$\geq T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\|^2 - \beta \left\| \bar{\mathbf{g}}_U^{(t)} \right\| \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t=1}^{T'-1} \mathbb{E}\left[\left\| U^{(k,t,t')} - U^{(k,t,0)} \right\| \right] \tag{4.70}$$

$$= T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\|^2 - \gamma\beta \left\| \bar{\mathbf{g}}_U^{(t)} \right\| \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'-1} \mathbb{E}\left[\left\| \sum_{g=0}^{t'-1} \bar{\mathbf{g}}_U^{(k,t,g)} \right\| \right] \tag{4.71}$$

$$\geq T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\|^2 - \gamma\beta \left\| \bar{\mathbf{g}}_U^{(t)} \right\| \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'-1} \sum_{g=0}^{t'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,g)} \right\| \right] \tag{4.72}$$

$$\geq T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\|^2 - \gamma\beta T' \left\| \bar{\mathbf{g}}_U^{(t)} \right\| \sum_{k=1}^{s} \frac{m_k}{m} \sum_{t'=1}^{T'-1} \mathbb{E}\left[\left\| \bar{\mathbf{g}}_U^{(k,t,g)} \right\| \right] \tag{4.73}$$

$$\tag{4.74}$$

For another term in (4.65), similarly we can have

$$\frac{\beta\gamma^2}{2}\mathbb{E}\left[\left\|\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbf{g}_U^{(k,t,t')}\right\|^2\right] \tag{4.75}$$

$$\leq \frac{\beta\gamma^2}{2}\sum_{k=1}^{s}\frac{m_k}{m}\mathbb{E}\left[\left\|\sum_{t'=0}^{T'-1}\mathbf{g}_U^{(k,t,t')}\right\|^2\right] \text{ (By convexity of L2 norm)} \tag{4.76}$$

$$\leq \frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\mathbb{E}\left[\sum_{t'=0}^{T'-1}\left\|\mathbf{g}_U^{(k,t,t')}\right\|^2\right] \text{ (By Cauchy-Schwartz inequality)} \tag{4.77}$$

$$= \frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\mathbb{E}\left[\sum_{t'=0}^{T'-1}\left\|\left(\mathbf{g}_U^{(k,t,t')}-\bar{\mathbf{g}}_U^{(k,t,t')}+\bar{\mathbf{g}}_U^{(k,t,t')}\right)\right\|^2\right] \tag{4.78}$$

$$= \frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\mathbf{g}_U^{(k,t,t')}-\bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] \tag{4.79}$$

$$+\frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] \text{ (By unbiased assumption)} \tag{4.80}$$

$$\leq \frac{\beta\gamma^2 T'^2\sigma^2_{\mathbb{S}_{\text{Sub}},U}}{2}+\frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] \tag{4.81}$$

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 (4.30) should be

$$-\gamma\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\langle\bar{\mathbf{g}}_U^{(t)},\bar{\mathbf{g}}_U^{(k,t,t')}\right\rangle\right]+\frac{\beta\gamma^2}{2m^2}\mathbb{E}\left[\left\|\sum_{k=1}^{s}m_k\sum_{t'=0}^{T'-1}\mathbf{g}_U^{(k,t,t')}\right\|^2\right] \tag{4.82}$$

$$\leq -\gamma T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|^2+\gamma^2\beta T'\left\|\bar{\mathbf{g}}_U^{(t)}\right\|\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')}\right\|\right] \tag{4.83}$$

$$+\frac{\beta\gamma^2 T'^2\sigma^2_{\mathbb{S}_{\text{Sub}},U}}{2}+\frac{\beta\gamma^2 T'}{2}\sum_{k=1}^{s}\frac{m_k}{m}\sum_{t'=1}^{T'}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_U^{(k,t,t')}\right\|^2\right] \tag{4.84}$$

□

Proof of Lemme 4. We look at the expecation of (4.31) for the update of V,

$$\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_V^{(t)}, V^{(t+1)} - V^{(t)} \right\rangle + \frac{\beta}{2}\left\|V^{(t+1)} - V^{(t)}\right\|^2\right] \tag{4.85}$$

$$= \sum_{k=1}^{s}\left(-\mathbb{E}\left[\left\langle \frac{m_k}{m}\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(t)}, \sum_{t'=0}^{T'-1}\frac{\gamma m_k}{m}\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\rangle\right] + \frac{\beta}{2}\mathbb{E}\left[\left\|\sum_{t'=0}^{T'-1}\frac{\gamma m_k}{m}\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right]\right) \tag{4.86}$$

$$= \sum_{k=1}^{s}\left(-\sum_{t'=0}^{T'-1}\frac{\gamma m_k^2}{m^2}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\rangle\right] + \frac{\beta}{2}\mathbb{E}\left[\left\|\sum_{t'=0}^{T'-1}\frac{\gamma m_k}{m}\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right]\right) \tag{4.87}$$

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

$$\sum_{t'=0}^{T'-1}\frac{\gamma m_k^2}{m^2}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\rangle\right] \tag{4.88}$$

$$= \sum_{t'=0}^{T'-1}\frac{\gamma m_k^2}{m^2}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)} - \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)} + \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\rangle\right] \tag{4.89}$$

$$= \sum_{t'=0}^{T'-1}\frac{\gamma m_k^2}{m^2}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)}\right\rangle\right] + \sum_{t'=0}^{T'-1}\frac{\gamma m_k^2}{m^2}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} - \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)}\right\rangle\right] \tag{4.90}$$

$$= \frac{\gamma T' m_k^2}{m^2}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|^2 + \sum_{t'=0}^{T'-1}\frac{\gamma m_k^2}{m^2}\mathbb{E}\left[\left\langle \bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}, \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} - \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)}\right\rangle\right] \tag{4.91}$$

$$\geq \frac{\gamma T' m_k^2}{m^2}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|^2 - \frac{\gamma m_k^2}{m^2}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')} - \mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,0)}\right\|\right] \tag{4.92}$$

$$\geq \frac{\gamma T' m_k^2}{m^2}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|^2 - \frac{\gamma m_k^2\beta}{m^2}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|V^{(k,t,t')} - V^{(k,t,0)}\right\|\right] \tag{4.93}$$

$$= \frac{\gamma T' m_k^2}{m^2}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|^2 - \frac{\gamma^2 m_k^3\beta}{m^3}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\sum_{g=0}^{t'-1}\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,g)}\right\|\right] \tag{4.94}$$

$$\geq \frac{\gamma T' m_k^2}{m^2}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|^2 - \frac{\gamma^2 m_k^3\beta}{m^3}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|\sum_{t'=0}^{T'-1}\sum_{g=0}^{t'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,g)}\right\|\right] \tag{4.95}$$

$$\geq \frac{\gamma T' m_k^2}{m^2}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|^2 - \frac{T'\gamma^2 m_k^3\beta}{m^3}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|\right] \tag{4.96}$$

$$\tag{4.97}$$

The second term can be bounded as

$$\frac{\beta}{2}\mathbb{E}\left[\left\|\sum_{t'=0}^{T'-1}\frac{\gamma m_k}{m}\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right] \tag{4.98}$$

$$=\frac{\beta\gamma^2 m_k^2}{2m^2}\mathbb{E}\left[\left\|\sum_{t'=0}^{T'-1}\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right] \tag{4.99}$$

$$\leq\frac{\beta\gamma^2 m_k^2 T'}{2m^2}\mathbb{E}\left[\sum_{t'=0}^{T'-1}\left\|\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right] \tag{4.100}$$

$$=\frac{\beta\gamma^2 m_k^2 T'}{2m^2}\mathbb{E}\left[\sum_{t'=0}^{T'-1}\left\|\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}-\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}+\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right] \tag{4.101}$$

$$=\frac{\beta\gamma^2 m_k^2 T'}{2m^2}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}-\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2+\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right] \tag{4.102}$$

$$\leq\frac{\beta\gamma^2 m_k^2 T'^2\sigma_{\mathbb{S}_{\mathtt{Sub}},V}^2}{2m^2}+\frac{\beta T' m_k^2\gamma^2}{2m^2}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right] \tag{4.103}$$

Thus, we have

$$-\mathbb{E}\left[\left\langle\bar{\mathbf{g}}_V^{(t)},V^{(t+1)}-V^{(t)}\right\rangle+\frac{\beta}{2}\left\|V^{(t+1)}-V^{(t)}\right\|_2^2\right] \tag{4.104}$$

$$\leq-\gamma T'\sum_{k=1}^{s}\frac{m_k^2}{m^2}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|^2+\sum_{k=1}^{s}\frac{T'\gamma^2 m_k^3\beta}{m^3}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\mathbf{g}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|\right] \tag{4.105}$$

$$+\sum_{k=1}^{s}\frac{\beta\gamma^2 m_k^2 T'^2\sigma_{\mathbb{S}_{\mathtt{Sub}},V}^2}{2m^2}+\sum_{k=1}^{s}\frac{\beta T' m_k^2\gamma^2}{2m^2}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right] \tag{4.106}$$

$$=-\gamma T'\left\|\bar{\mathbf{g}}_V^{(k,t)}\right\|^2+\sum_{k=1}^{s}\frac{T'\gamma^2 m_k^3\beta}{m^3}\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t)}\right\|\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|\right] \tag{4.107}$$

$$+\sum_{k=1}^{s}\frac{\beta\gamma^2 m_k^2 T'^2\sigma_{\mathbb{S}_{\mathtt{Sub}},V}^2}{2m^2}+\sum_{k=1}^{s}\frac{\beta T' m_k^2\gamma^2}{2m^2}\sum_{t'=0}^{T'-1}\mathbb{E}\left[\left\|\bar{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k,t,t')}\right\|^2\right] \tag{4.108}$$

$$\tag{4.109}$$

□

4.4 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 4.1a, 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. 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.

4.4.1 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`.

Bounding sensitivity.

In order to bound the sensitivity of the loss function (4.1), [76] sets constraints on the inner product of u_i and v_j. Gradient clipping method in [18] can be for MF problem and [78] 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 [76] 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 [87, 88], in which the user embeddings and item embeddings are restricted to be non-negative. In this work, we enforce stronger constraints that the norm of the user embeddings and item embeddings are also bounded. That is,

$$\mathcal{U} : \forall i \in [n], \|u_i\|_2^2 \leq R, u_i \geq 0, \quad \mathcal{V} : \forall j \in [m], \|v_j\|_2^2 \leq R, v_j \geq 0. \tag{4.110}$$

With constraints in (4.110) and CauchySchwarz inequality, we have $\langle u_i, v_j \rangle \leq R$. The gradient with respect to u_i from the j-th item is $\nabla_{u_i} \mathcal{L}_{i,j}(U, V) = -2v_j (\mathbf{X}_{ij} - \langle u_i, v_j \rangle)$ if $\mathbf{X}_{ij}$ 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 $\mathbf{X}_{i^*j^*} \neq \perp$ and $\mathbf{X}'_{i^*j^*} = \perp$. With u_i and v_j bounded as (4.110), then the sensitivity of the gradient to a user embedding is $\Delta_{2,u} = \|-2v_{j^*} (\mathbf{X}_{i^*j^*} - \langle u_{i^*}, v_{j^*} \rangle) - 0\|_2 \leq 2R^{3/2}$. 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 $\mathbf{\Pi}_{\mathcal{U}}(U)$, 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 (4.110); $\mathbf{\Pi}_{\mathcal{V}}(V)$ performs the same projection for V to satisfy (4.110). 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.

Embedding clipping proof

Lemma 5. *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 $\|u_i^*\|_2^2 \leq R$, 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.

□

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 $\perp$.

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.

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 (4.110). The locally updates on user embeddings and item embeddings becomes:

$$U^{(k)} = \mathbf{\Pi}_{\mathcal{U}}\left(U^{(k)} - \gamma_t \tilde{\mathbf{g}}_U^{(k)}\right), V_{\mathbb{F}_k} = \mathbf{\Pi}_{\mathcal{V}}\left(V_{\mathbb{F}_k} - \frac{\gamma_t m_k}{m}\tilde{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k)}\right), \tag{4.111}$$

where $\tilde{\mathbf{g}}_U^{(k)} = \mathbf{g}_U^{(k)} + \boldsymbol{\xi}_U^{(k,t,t')}, \tilde{\mathbf{g}}_{V_{\mathbb{F}_k}}^{(k)} = \mathbf{g}_{V_{\mathbb{F}_k}}^{(k)} + \boldsymbol{\xi}_{V_{\mathbb{F}_k}}^{(k,t,t')}$. The noise $\boldsymbol{\xi}_U^{(k,t,t')}$ and $\boldsymbol{\xi}_{V_{\mathbb{F}_k}}^{(k,t,t')}$ 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.

`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.

4.4.2 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 $\boldsymbol{\xi}_U^{(k,t,t')}$ and $\boldsymbol{\xi}_{V_{\mathbb{F}_k}}^{(k,t,t')}$ 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 [18], the relation between noise multiplier z and privacy budget is as the following:

Proposition 4.4.1. *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 4.4.1 (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 4.4.2 (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 work. 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) [89]. Following the same spirit, our algorithm can also be executed daily.

Privacy proofs

Proof of Theorem 4.4.1. 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, $\mathbf{\Omega}_k \cup \{(i^*, j^*)\} = \mathbf{\Omega}'_k$. 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 $\mathbf{M}^{(k)}_{\mathbb{S}_{\text{Sub}}}(\cdot)$ be the function of sampling and computing the differentially private gradients to all parameters of party k, $W = (U, V_{\mathbb{F}_k})$. With (4.111), it is easy to see that if $\mathbf{M}^{(k)}_{\mathbb{S}_{\text{Sub}}}(\mathbf{X}_{\mathbb{F}_k}) \sim \mu_0$, then $\mathbf{M}^{(k)}_{\mathbb{S}_{\text{Sub}}}(\mathbf{X}'_{\mathbb{F}_k}) \sim (1-\eta)\mu_0 + \eta\mu_1$ for some distribution μ_0 and μ_1. Following the proof of Lemma 3 in [18], the log of moment generating function $\alpha_{\mathbf{M}^{(k)},t,t'}(\lambda) \leq \frac{\eta^2\lambda(\lambda+1)}{(1-\eta)\sigma^2} + O\left(\eta^2\lambda^2/\sigma^3\right)$. Then follow the proof of Theorem 1 in [18], 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 $\mathbf{M}$ has s subroutines, $\mathbf{M}^{(1)}, \ldots, \mathbf{M}^{(s)}$, where $\mathbf{M}^{(k)}$ 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 $\alpha_{\mathbf{M}}(\lambda) = \sum_{t=1}^{T}\sum_{t'=1}^{T'} \alpha_{\mathbf{M},t,t'}(\lambda)$.

Same as traditional DP language, moment accountants also has parallel composition property (Proposition 4.4.2). With Proposition 4.4.2, we can have $\alpha_{\mathbf{M}}(\lambda) = \sum_{t=1}^{T}\sum_{t'=1}^{T'} \max_{k\in[s]}\{\alpha_{\mathbf{M}^{(k)},t,t'}(\lambda)\}$. If $\alpha_{\mathbf{M}^{(k)},t,t'}(\lambda)$ are the same for all t and t', then $\alpha_{\mathbf{M}}(\lambda) = \max_{k\in[s]}\{\sum_{t=1}^{T}\sum_{t'=1}^{T'} \alpha_{\mathbf{M}^{(k)},t,t'}(\lambda)\}$.

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

Proposition 4.4.2 (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* $\mathbf{M}$ *has* s *subroutines,* $\mathbf{M}^{(1)}, \ldots, \mathbf{M}^{(s)}$*, where* $\mathbf{M}^{(k)}$ *can only access* $\mathbf{X}_k$*, then*

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

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

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

Then

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

□

Proof of Theorem 4.4.2. 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 $\mathbf{M}'(\mathbf{X})$ which queries

the large rating matrix sTT' times. Each iteration $t_g \in [sTT']$, $\mathbf{M}'$ only samples from $\mathbf{X}_k$ where $k = t_g \bmod TT'$, 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 2.2.1, the moment is bounded by

$$\alpha_{\mathbf{M},\mathsf{total}}(\lambda) = \sum_{t=1}^{T}\sum_{k=1}^{s}\sum_{t'=1}^{T'} \alpha_{\mathbf{M},k,t,t'}(\lambda) \leq \sum_{k=1}^{s} \frac{TT'\eta^2\lambda^2}{z^2}.$$

By the tail bound in Theorem 2.2.1, to ensure `VFL-SGDMF` is (ϵ, δ)-per user DP, it is sufficient to show

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

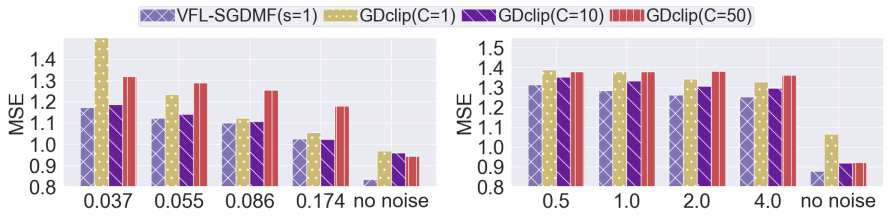
That is equivalent to find a smallest ϵ such that $\epsilon^2 \geq \sum_{k=1}^{s} \frac{TT'\eta^2\log(1/\delta)}{z^2}$. For each party k, if it already satisfies $(\epsilon^{(k)}, \delta^{(k)})$-DP for party level, it means

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

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

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

□



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

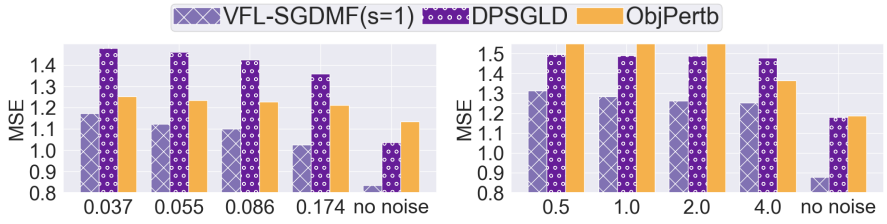
Figure 4.3. Embedding clip v.s. gradient clip.

4.4.3 Empirical Evaluation for VFL

Datasets. We use three datasets in our experiments. The first one is the MovieLens 10M (ML10M) dataset [90], the second one is MovieLens 25M (ML25M) and the other one is the LibimSeTi [91] 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 the scores in LibimSeTi are between 0 to 10. We also pre-process the LibimSeTi dataset with the same process as [78].

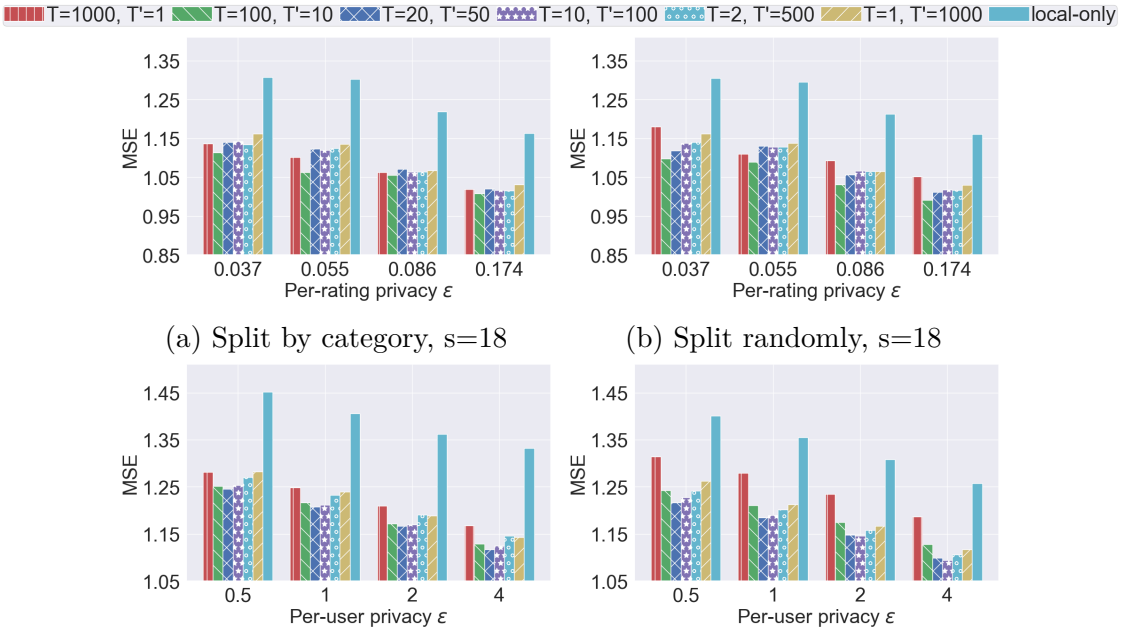
We split the datasets into s disjoint sub-datasets by movie/profiles, such that the ratings of each movie/profile can appear in only one dataset. Each of the sub-dataset simulates the local dataset for a party. We keep 10% of the ratings for testing in each local dataset and use the remaining for training. 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 [18, 76, 77] 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 [18] 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.



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

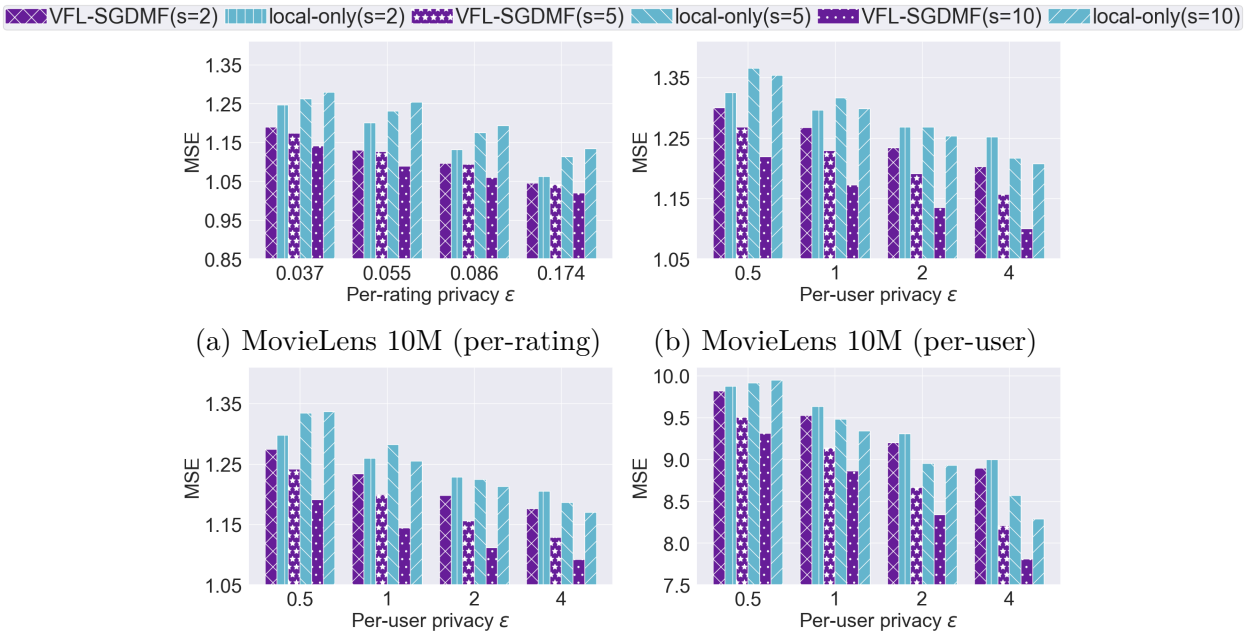
Figure 4.4. Compare `VFL-SGDMF` (s=1), DP-SGLD and ObjPertb.



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

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

Figure 4.5. 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 4.6. Results of randomly split items into different number of parties s.

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 work. 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 [18] 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 [18, 92].

Experiment results. We evaluate the `VFL-SGDMF` against gradient clipping and other methods [76, 77] 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 4.3a 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 4.3b 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 [76, 77].* In Figure 4.4, we compare our embedding clipping method with DP-SGLD in [76] and the objective perturbation (ObjPertb) approach in [77] 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 4.5a with Figure 4.5b for per-rating privacy, or comparing Figure 4.5c with Figure 4.5d 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.

• *Comparing different synchronization frequencies.* In Figure 4.5, 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 4.6. 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).

4.5 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 4.1b. 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.

4.5.1 Private MF in HFL: HFL-SGDMF

In order to provide the expected protection, we propose HFL-SGDMF with privacy guarantee. We denote $\tilde{\mathbf{g}}_V^{(k)}$ 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 4.2c, 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.

• *[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: $V^{(k)} = \mathbf{\Pi}_{\mathcal{V}}\left(V^{(k)} - \gamma_t \tilde{\mathbf{g}}_V^{(k)}\right)$, where $\tilde{\mathbf{g}}_V^{(k)}$ 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).

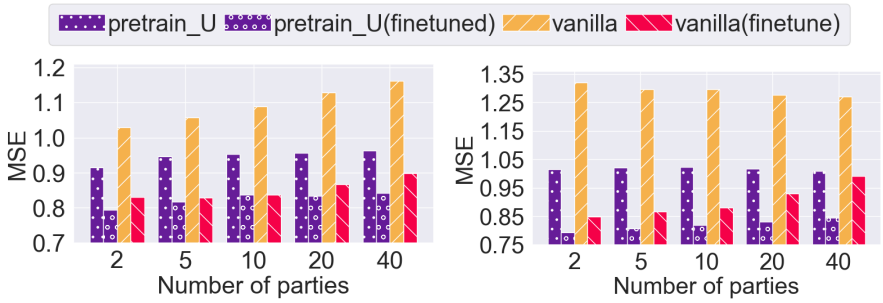
• *[Stage 3]: local fine-tuning.* After Stage 2, each party further fine-tunes their local version of user/item embeddings to get more accurate local inference results with untrimmed local data.

4.5.2 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-tuning are unpublished.

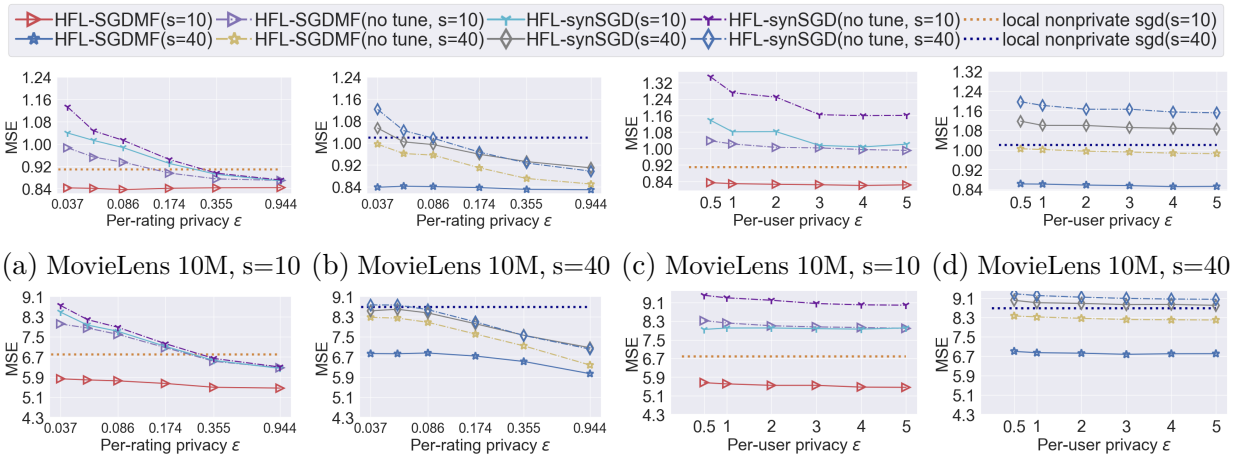
DP sensitivity and necessary noise. Notice that since we only need to protect privacy on the exchange of item embeddings in the HFL stage, the sensitivity is $\Delta_{rating} = \Delta_{2,v}$ for per-rating privacy or $\Delta_{user} = \theta^{(k)}\Delta_{2,v}$ for per-user privacy for each local update.

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 4.4.2, the following proposition can conclude the privacy loss in the `HFL-SGDMF`.



(a) per-rating privacy $\epsilon = 0.055$ (b) per-user privacy $\epsilon = 1.0$

Figure 4.7. Compare pre-train-U with vanilla train-both.



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

Proposition 4.5.1 (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 $z^2 = c_2 \frac{\eta^2 TT' \ln 1/\delta^{(k)}}{(\epsilon^{(k)})^2}$ with some constant c_2. The overall* ***HFL-SGDMF*** *is $(\max_{k\in[s]} \{\epsilon^{(k)}\}, \max_{k\in[s]} \{\delta^{(k)}\})$ per-rating/per-user private.*

4.5.3 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 $\tilde{\mathbf{g}}_V^{(k)}$ 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 4.4.3.

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 4.7, 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 4.8 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 4.8 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.

4.6 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.

LFL privacy leakage and protection. As analyzed in Section 4.1.1, 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 [78], 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 4.6.3.

• *Secure aggregation and its limitation.* Secure aggregation [73, 93] 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 [73, 93] 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 $\texttt{SA}_\omega\texttt{-report}$, and the server aggregation process as $\texttt{SA}_\omega\texttt{-agg}$. However, secure aggregation protocols are vulnerable to membership attacks or re-identification attacks.

4.6.1 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 4.2d. 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 [78] with LDP as shown in our experiments in Section 4.6.3.

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 $\texttt{SA}_\omega$ 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 $\texttt{SA}_\omega$- $\texttt{report}(\tilde{\mathbf{g}}_V^{(i,t)})$. The server aggregates the reported updates by invoking $\texttt{SA}_\omega$-`agg`. We denote the set of survivors in the first round as $\mathbb{U}^{(t,1)}$. $\texttt{SA}_\omega$ is designed with a tolerable dropout rate ω. If less than $(1-\omega)n$ users survive in this iteration, the $\texttt{SA}_\omega$ protocol halts, and the server learns nothing in this iteration and execute next iteration. Otherwise, the aggregated gradients decoded from $\texttt{SA}_\omega$ can be decomposed as $\tilde{\mathbf{g}}_V^{(t)} = \sum_{i\in\mathbb{U}^{(t,1)}} \nabla_V \mathcal{L}_{i,j}\left(u_i, V^{(t)}\right) + \sum_{i\in\mathbb{U}^{(t,1)}} \boldsymbol{\xi}^{(i,t)}$.

• *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 $\bar{\boldsymbol{\xi}}^{(i,t)} = -\boldsymbol{\xi}^{(i,t)} + \boldsymbol{\xi}'^{(i,t)}$ to the server, where the elements of $\boldsymbol{\xi}'^{(i,t)}$ 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:

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

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.

4.6.2 Analysis of Privacy Guarantee

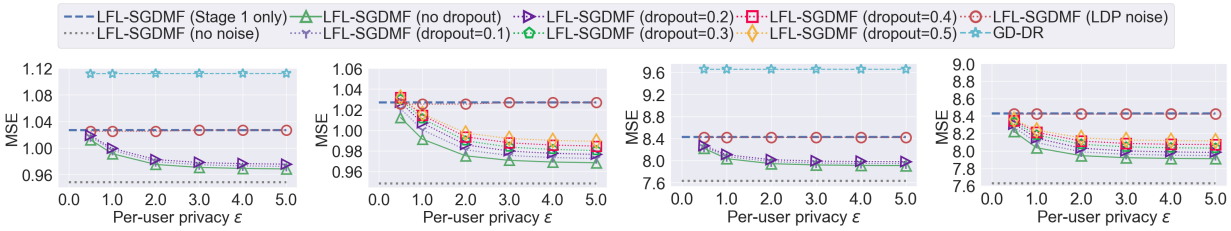
There are three parameters directly affecting the privacy privacy guarantee, σ_v^2, ζ_1 and ζ_2. σ_v^2 is the nob for (ϵ, δ)-DP guarantee when there is no dropout. When $\texttt{SA}_\omega$ finishes successfully, the sum is composed of at least $|\mathbb{U}^{(t,1)}| \geq (1-\omega)n$ users and ζ_1 makes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

Theorem 4.6.1. *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 $\texttt{SA}_\omega$, 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.

4.6.3 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 [78] for comparison. To make it a fair comparison, we amplify the privacy budget of LDP to the central DP with the best-known result [94]. 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



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

Figure 4.9. Results under LFL setting with different dropout rates.

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 [78].

• *Comparing with LDP methods.* In Figure 4.9, we first compare our `LFL-SGDMF` with the LDP algorithm, GD-DR, from [78]. 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 4.9 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.

4.7 Related Work

The majority of the work enforcing DP in FL is under the HFL setting [63, 70], and they can be categorized into two classes. One class of works, such as `DP-FedSGD` [84, 85], are extensions based on the `DPSGD` [18, 81], in which a server aggregates the (privatized) gradients from each party and updates models centrally. Another class of techniques, like `DP-FedAvg` [95, 96], 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 chapter has the following differences in the HFL setting compared with those existing works. 1) Different from [84, 95], we assume that there is no fully trusted coordinator, and all the information shared by the parties must be differentially private. 2) Compared with [85, 96], MF is a non-convex optimization problem with unbounded norm of gradient. 3) Different from the DPSGD [18], 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 4.4. There are also results under non-private HFL setting [97, 98].

Existing work about other problems in the VFL setting includes learning tree models with secure multiparty computation techniques [74, 99] and training composed models [100]. A recent work [101] studies the generalized linear model with distributed features under the ADMM framework. Another paper [102] discussing asynchronous supervised learning with VFL assumes the labels are public accessible. According to our knowledge, we are the first 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, 94, 103, 104]. There are cryptographic methods in the LFL setting called secure aggregation [73, 93] in parallel. The authors of [78] 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 [76, 77, 83, 105]. Also, the MF problem was studied with homomorphic encryption [106]. Without

privacy protection, federated matrix factorization was studied as a multi-view learning problem in [107]. Some other work focuses on the tensor factorization in HFL, like [108] with non-private aggregation and [109] solving the problem with EASGD and output perturbation for DP.

4.8 Conclusion

This chapter 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.

5. DIFFERENTIALLY PRIVATE VERTICAL FEDERATED CLUSTERING

(A public version of this chapter has been previously published [110].)

5.1 Introduction

This chapter focuses on an important federated learning setting, the *vertical federated learning* (VFL) setting. Its difference from the *horizontal federated learning* (HFL) setting is that all parties have data from the *same set of users, but their data attributes are different from each other*. VFL has been an interesting topic in the research area since the early 2000s [74, 111–114]. The papers are usually motivated by medical or financial use cases, where the users' private data are not allowed to be shared between data parties. More recently, VFL has been adapted by some fintech companies for more real-world services. For example, WeBank demonstrates how they do risk-control for car insurance with VFL techniques [2]. In the use case, WeBank, which has data including users' personal financial situations, cooperates with car rental companies, which have the users' car-rental-related data, to train a risk control model giving the most reasonable insurance premium rate. In generally, cooperation among data parties with different information about the same set of users can help them discover more correlations and potentially increase the utility with a more comprehensive model.

How to perform VFL while not leaking private information has been an interesting topic in the security and privacy community. Many existing VFL works are based on secure multiparty computation (SMC), including algorithms learning classification tree models [74, 99, 112], regression models [115] and clustering models [111]. However, the SMC-based methods' final results cannot provide provable resistance to membership attacks or reconstruction attacks, and they usually have high computation and communication overheads. Other literature employs DP as the security notion to provide resistance to the membership and reconstruction attacks. Privacy-preserving data mining on the statistics from vertically distributed databases can be dated back to [116], when the idea of DP had not yet been formalized. More recently, researchers have developed VFL algorithms with DP guarantee

for matrix factorization [117], regression [118], and boosting model [102]. We provide the first solution for the DP VFL clustering problem in this chapter.

Notice that the VFL setting is more challenging than the HFL setting because of the following factors. 1) The correlations between the distributed attributes are harder to detect and require carefully distilled information sharing even in non-private VFL. In contrast, a data party in HFL can at least have some understanding of the correlations among all attributes based on its local dataset. The correlation detection becomes even harder in the privacy-preserving VFL setting. Because learning the correlation requires "connecting" a user's records in different local datasets, which either leaks user privacy or ruins the utility without carefully designed protocols. 2) The VFL setting that a user's information is spread into different parties also means privacy loss accumulates when a user's record is accessed by each data party locally. On the contrary, in HFL, a user's record only appears in one data party's dataset, and we only need to argue the privacy guarantee for each individual data party (with a property of DP called parallel composition, more details later). Our solution proposed in this work solves the challenges while providing DP as the privacy guarantee for the clustering problem.

Our contributions. This work proposes a novel solution for the differentially private vertical federated k-means. We assume there are multiple data parties and an untrusted central server. All the information shared by each data party in the training process as well as the final result satisfy DP. The key idea is to have the data parties generate differentially private "data synopsis" that describes local k-means results (based on a party's partial view). The server then runs a central k-means on the Cartesian product of all partial centers considering the weights of them, where the weight for each joint center is calculated based on our novel algorithm to estimate the cardinality of intersection among user indices belonging to each partial center. Our main contributions are summarized as follows:

- We propose the first (according to our knowledge) differentially private VFL k-means algorithm with an untrusted central server. We innovatively introduce the Flajolet-Martin sketch into the VFL setting and build our brand new intersection cardinality estimation algorithm (Algorithm 4 and Algorithm 5) based on it. We show a theoretical utility guarantee

and empirical evaluations for the final k centers derived by the central server. Our proposed method has a small computation overhead, formal DP guarantees on the shared information and the final results, and reasonable accuracy.

- To improve the accuracy in the scenario with more than two data parties, we further improve the algorithm for the central server to estimate the intersection cardinalities (Algorithm 6). The improved algorithm outperforms the basic one and all other baselines when the number of data parties is greater than 2. We also introduce a rule to choose the local clustering granularity automatically.

- Our experiments show that our proposed methods can outperform the other baseline methods and even approach the non-private VFL k-means algorithm when sufficient users are in the dataset. We also conduct ablation studies to empirically demonstrate the impact and effectiveness of each component of our algorithm.

Roadmap. The rest of the chapter is organized as the following: we revisit the necessary background information and an overview of the VFL clustering problem for this work in Section 5.2; we illustrate our solution and provide more details of the key components in Section 5.3 and 5.4; experimental results are shown in Section 5.5; Section 5.6 discusses the related work from different perspectives, followed by a conclusion in Section 5.7.

5.2 Problem Formulation and Overview of Approach

In this section, we define the problem of differentially private k-means under vertical federated leaning (VFL), provide an overview of our proposed approach, and discuss the first phase solution. The problem of VFL k-means (without DP) has been studied before by Ding et al. [119]. We thus describe the approach in [119], the new challenges when we need to satisfy DP, and our framework.

5.2.1 k-means Clustering

Clustering algorithms have a long history in computer science research, and k-means [120] is one of the most well-known clustering problems. With a parameter k and a dataset X,

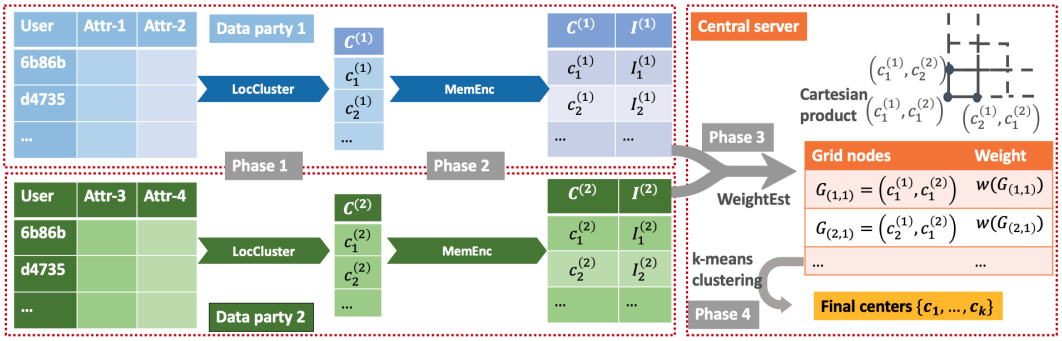


Figure 5.1. Vertical federated k-means clustering with two data parties.

the goal of the problem is to output a set of k centers that can minimize the distance of data points to their nearest centers. The problem can be formalized as to minimize the cost of k-means for a set of k centers $\mathbf{C}$ and an input dataset $\mathbf{X} \in \mathbb{R}^{n \times m}$, which is defined as

$$\texttt{cost}_{\mathbf{X}}(\mathbf{C}) := \sum_{x \in \mathbf{X}} (\min_{c \in \mathbf{C}} \|x - c\|_2^2).$$

The problem can be extended to a weighted point set, where each data point $x \in \mathbf{X}$ also has a weight $w(x)$ associated with it. It can be considered as a general version for the scenario where there are $w(x)$ copies of the same data point x in $\mathbf{X}$. In this case, the k-means cost is defined as $\texttt{cost}_{\mathbf{X}}(\mathbf{C}) := \sum_{x \in \mathbf{X}} w(x) \cdot (\min_{c \in \mathbf{C}} \|x - c\|^2)$.

Theoretically, there is always a set of optimal k centers and the cost is denoted as $\texttt{OPT}_{k,\mathbf{X}} = \min_{|\mathbf{C}|=k} \texttt{cost}_{\mathbf{X}}(\mathbf{C})$. However, finding the optimal set of centers for a dataset is NP-hard [121]. Thus, researchers have developed efficient but heuristic algorithms that do not guarantee optimality. For example, the most commonly used algorithm, Lloyds algorithm [122] has time complexity $O(nmk)$.

Given $\mathbf{X}$ and k, a set of k centroids is called a (β, λ)-approximate solution if $\texttt{cost}_{\mathbf{X}}(\mathbf{C}) \leq \beta \cdot \texttt{OPT}_{\mathbf{X}}^k + \lambda$. In the non-private setting, the best known proven approximation is $\beta \approx 6.357, \lambda = 0$ [123].

5.2.2 VFL Clustering Problem Formulation

We formalize the VFL k-means clustering as the following.

Vertical federated learning (VFL). Federated learning (FL) [124] focuses on learning tasks among multiple data parties without directly sharing their local data. Vertical FL (VFL) assumes that each data party's data are from different features of the same set of users. Consider a dataset $\mathbf{X}$ where each row corresponds to a user, and each column corresponds to a feature. The setting of VFL is that $\mathbf{X}$ is vertically split into $\mathbf{X} = [\mathbf{X}^{(1)}| \dots, |\mathbf{X}^{(S)}]$, so that each data party $\ell \in [S]$ has a local dataset $\mathbf{X}^{(\ell)}$ with some features. We assume each user is labeled with a unique `id` (e.g., MAC address) and is consistent across all the data parties.

Security model. Our model assumes that a central server orchestrates the process and obtains the results. For each party, any information shared is protected with the DP guarantee. The central server is untrusted because it is curious to learn extra private information during the computation process. We assume that the central server does not collude with any data party, which is the same as the previous work [125, 126]. Because all the information received by the central party is differentially private, then the final deliverable (i.e., the k centers) generated by the central server can be considered a post-processing output in DP.

Goal. All the S parties want to cooperatively generate differentially private k centers $\mathbf{C}$ in the full domain (with all attributes) that can approximately minimize the k-means loss $\texttt{cost}_{\mathbf{X}}(\mathbf{C})$. The challenge is that each party only has its local view (a few attributes), where the final centers are evaluated with a full view (all attributes).

5.2.3 A Non-Private Baseline

Ding et al. [119] consider VFL k-means, and aim to avoid the data communication cost of sending all data to a server. The challenge here is that each party has only the local view. A natural approach is thus first to construct a global approximation of the data points and then perform k-means clustering on the approximation. In the approach taken in [119], each data party first finds local cluster centers, then reports to the server these local cluster

centers together with which local cluster each data point belongs to. The central server can assemble the local centers and local clustering memberships to create a set of weighted pseudo data points of the full dataset. We provide more details below.

Each party ℓ performs clustering to find k' local cluster centers $\mathbf{C}^{(\ell)} = \left\{c_1^{(\ell)}, \ldots, c_{k'}^{(\ell)}\right\}$; and then sends the k' centers together with membership information $\mathbf{I}^{(\ell)} = \left\{\mathcal{M}_1^{(\ell)}, \ldots, \mathcal{M}_{k'}^{(\ell)}\right\}$ to the central server, where $\mathcal{M}_a^{(\ell)} = \left\{\texttt{id} \mid a = \arg\min \left\|x_{\texttt{id}}^{(\ell)} - c_a^{(\ell)}\right\|_2^2\right\}$. The server constructs $(k')^S$ pseudo data points as a grid from the Cartesian product of the local centers received from S parties, i.e., $\mathbf{G} = \{(c_{a_1}^{(1)}, \ldots, c_{a_S}^{(S)}) \mid \forall (a_1, \ldots, a_S) \in [k']^S\}$; and assigns the cardinality of the intersection of the corresponding clusters as weights to them such that $w(\mathbf{G}_{(a_1,\ldots,a_S)}) = \left|\mathcal{M}_{a_1}^{(1)} \cap \ldots \cap \mathcal{M}_{a_S}^{(S)}\right|$.

This algorithm with only one round of communication can perform well because the grid built by the central server actually maintains most of the necessary information about the local datasets: each local center $c_{a_\ell}^{(\ell)}$ is the exact average of the user data in $\mathcal{M}_{a_\ell}^{(\ell)}$; moreover, data points in the intersection $\mathcal{M}_{a_1}^{(1)} \cap \ldots \cap \mathcal{M}_{a_S}^{(S)}$ are expected to distribute around the pseudo point $(c_{a_1}^{(1)}, \ldots, c_{a_S}^{(S)})$. If the intersection has a small cardinality or even is an empty set, we can know that the pseudo point can be ignored. The weighted grid is similar to a useful data synopsis in the k-means cost analysis, called *coreset* [127], which approximates the original dataset information. As long as the weighted grid nodes are representative enough for a subset of points, the central server can find final centers without accessing the distributed datasets.

5.2.4 Challenge in the Privacy-preserving Setting

The approach described in Section 5.2.3 does not consider the privacy leakage problem, as the local cluster centers and membership information sent to the server contain sensitive information. To protect users' private information, all the information sent to the central server, including (1) local cluster centers $\mathbf{C}^{(\ell)}$ and (2) local membership information, should be differentially private.

For the clustering centers, there already exists comprehensive research of the k-means algorithm in the central DP setting [128–135]. Thus we can choose a method that works well.

Sending local membership information while satisfying DP is, however, very challenging. To the best of our knowledge, there is no effective DP algorithm for sharing membership information, especially in the scenario with more than two parties. Fortunately, the reason that we need to share the membership information is to estimate the weights of each pseudo data point. Thus, we do not need to share precise membership information, and just need a private way to *estimate the cardinality of the intersection among multiple parties*. The main technical contribution of this work is a solution to this problem, which we will described in Section 5.3. Our proposed approach leverages DP FM sketch and is extended to support the intersection operation among parties.

5.2.5 The Overall Framework

Figure 5.1 is a visualized workflow of Algorithm 2 with two data parties (note that our algorithm/analysis work with the general case of multiple parties). The formal description is given in Algorithm 2, which consists of four phases:

Phase 1: Each party clusters local data and generates differentially private local centers (sub-procedure LocCluster).

Phase 2: Each party encodes the differentially private "membership information" of each local cluster with the private centers and user data points (sub-procedure MemEnc).

Phase 3: The central server first randomly queries a party for an estimate of the total number of users with the Laplace mechanism and a small privacy budget ϵ_0[1]. Then the central server receives the private local clustering centers and local membership information of the local clusters. It builds a weighted grid, where grid nodes are the Cartesian product of different parties' local centers, and have the estimate of

[1]↑We set $\epsilon_0 = 0.02\epsilon$ unless we specify in the following text.

intersection cardinality of the corresponding clusters as their weights (Line 3(c) and sub-procedure WeightEstimate).

Phase 4: The central server runs a known central k-means algorithm on the weighted grid to generate the final k centers.

Algorithm 2 Private Vertical Federated Clustering

Input: Local datasets $\{\mathbf{X}^{(\ell)} \in \mathbb{R}^{n \times m_\ell} \mid \ell \in [S]\}$, total privacy budget (ϵ, δ) is divided as $\epsilon_0 = (1-b)\epsilon$, $\epsilon_1 = \frac{b\epsilon}{2S}$, $\epsilon_2 = \frac{b\epsilon}{2S}, \delta_2 = \frac{\delta}{S}$ for each data party, k and k' for clustering, and auxiliary membership encoding parameters aux.

Output: A set of k centers $\{c_1, \ldots, c_k\} \in \mathbb{R}^{k \times m}$

1: Each data party $\ell \in [S]$:
   (a): $\{c_1^{(\ell)}, \ldots, c_{k'}^{(\ell)}\} \leftarrow \mathsf{LocCluster}(\mathbf{X}^{(\ell)}, \epsilon_1, k')$
2: Each data party $\ell \in [S]$:
   (a): $\mathbf{I}^{(\ell)} \leftarrow \mathsf{MemEnc}(\mathbf{X}^{(\ell)}, \{c_1^{(\ell)}, \ldots, c_{k'}^{(\ell)}\}, \epsilon_2, \delta_2, \mathsf{aux})$
   (b): sends $\mathbf{C}^{(\ell)} = \{c_1^{(\ell)}, \ldots, c_{k'}^{(\ell)}\}$ and $\mathbf{I}^{(\ell)}$ to server
3: Central server:
   (a): uses ϵ_0 to estimate the total number of user $\hat{n}$
   (b): receives $\{(\mathbf{C}^{(\ell)}, \mathbf{I}^{(\ell)}) | \ell \in [S]\}$ from all parties
   (c): computes grid by Cartesian product $\mathbf{G} \leftarrow \mathbf{C}^{(1)} \times \ldots \times \mathbf{C}^{(S)}$
   (d): computes $w(\mathbf{G}) \leftarrow \mathsf{WeightEst}(\hat{n}, \epsilon_2, \delta_2, \{\mathbf{I}^{(\ell)} | \ell \in [S]\})$
4: Central server:
   (a): computes and outputs $\{c_1, \ldots, c_k\} \leftarrow k\text{-means}(G, w(G), k)$

In what follows, Section 5.2.6 describes our approach for Phase 1, and Section 5.3 describes our approach for Phase 2 and 3.

5.2.6 Private Local Clustering

We review some existing solutions to generate private centers in the central setting and then explain our adaptation to our VFL setting.

DPLloyd. A straight-forward differentially private central k-means is the *DPLloyd* [131, 136]. In each iteration, the assignment step is the same as the non-private Lloyd algorithm, where each data point is assigned to the closest center produced from the previous iteration. The updating step ensures DP by 1) using the Laplace mechanism with sensitivity 1 to get the noisy count of data points assigned to the center, 2) using the Laplace mechanism with

sensitivity r (it requires that all attributes are bounded in $[-r, r]$) and a split privacy budget for each dimension to get the noisy sums of the data points assigned to the same center. The centers are updated as the averages of all data points in the same cluster with the noisy count and noisy sum. Every iteration consumes privacy budget for computing noisy sums and noisy counts.

DPLSF. There are two recently proposed algorithms for differentially private k-means with theoretical performance guarantees, one for the central setting [137] and one for the local setting [138]. Both algorithms are built on a theoretical concept called efficiently decodable net. But how to implement the efficiently decodable net in practice is still unclear. Therefore, the authors also propose a DP k-means algorithm based on *locality sensitive hashing (LSH) forest* to approximate the effect of the efficiently decodable net. The central DP implementation is open-sourced [139]. The high-level idea is to partition the data points based on their LSH outputs, generate differentially private means and counts for these partitions, and finally run a (non-private) k-means algorithm on the means with counts as weights. We call this method DPLSF. We choose DPLSF as the instantiation of LocCluster in this work because it is shown to outperform other existing methods in experiments [139].

Adapting DPLSF to VFL setting. The implementation in [139] requires a known L_2 norm upper bound for the data points because of the usage of the Gaussian mechanism. However, assuming the L_2 norm upper bound for each user's data may be unreasonable in the VFL setting because a user's data are spread in different data parties' datasets. Thus, we enforce the domain of all attributes to be $[-1, 1]$ by letting the local party project the original domain to this range before running the algorithm. Consequentially, we calculate the noisy sum of the data points in the partition by splitting the privacy budget to all dimensions and applying the Laplace mechanism on each dimension with sensitivity 1.

5.3 Private Membership Encoding and Weight Estimate

As mentioned in Section 5.2.4, directly sharing the membership information $\{\mathcal{M}_1^{(\ell)}, \ldots, \mathcal{M}_{k'}^{(\ell)}\}$ with the central server leaks user's private information. We introduce our private instantiations of MemEnc and WeightEst together in this section because how

the central server can estimate the weights with WeightEst depends on how data parties encode the membership information with MemEnc. With our instantiations, the data parties generate differentially private membership information and share it with the central server, and the central server estimates the weights of the grid nodes, which are estimates of the cardinalities of the intersections $\left|\mathcal{M}_{a_1}^{(1)} \bigcap \ldots \bigcap \mathcal{M}_{a_S}^{(S)}\right|$, for all $(a_1, \ldots, a_S) \in [k']^S$.

5.3.1 Baselines

Baseline 1: Estimate weights assuming independence among attributes. In this approach, we assume that the distributions of attributes from one party are independent of those from all other parties. Under this assumption, we can compute the intersection cardinality using $\left|\mathcal{M}_{a_1}^{(1)} \bigcap \ldots \bigcap \mathcal{M}_{a_S}^{(S)}\right| \approx \hat{n} \prod_{\ell \in [S]} \frac{|\mathcal{M}_{a_\ell}^{(\ell)}|}{\hat{n}}$.

Following this idea, the private MemEnc only needs to generate a histogram of $\left[\left|\mathcal{M}_1^{(\ell)}\right|, \ldots, \left|\mathcal{M}_{k'}^{(\ell)}\right|\right]$ with Laplace mechanism and privacy budget ϵ_2. Denote the randomized histogram vector as $\tilde{\mathbf{f}}^{(\ell)}$. The central server's sub-procedure WeightEst is $w(\mathbf{G}_{(a_1,\ldots,a_S)}) = \hat{n} \prod_{\ell \in [S]} \frac{\tilde{\mathbf{f}}_{a_\ell}^{(\ell)}}{\hat{n}}$. We call this baseline as `IND-LAP` because it makes the independence assumption and uses the Laplace mechanism.

However, when the assumption of inter-party attributes independence fails, this cardinality estimation can be far from the ground truth and make the final centers far from optimal. The key reason is that the correlation information between the attributes across different data parties is completely ignored. Thus, maintaining the inter-party attribute correlations is the main focus of improving the utility in general scenarios.

Baseline 2: Estimate weights based on local differential privacy protocols. Another choice for aggregating the cardinality information is to let the parties report each user's membership information separately, instead of aggregating the membership information first and then reporting. When such reporting satisfies local differential privacy (LDP) for each user, it also satisfies DP for the whole local dataset. In this work, we apply either the optimized local hashing (OLH) or the general random response (GRR) protocol in [140] (which is used depends on the privacy parameter ϵ_2 and the domain size), and name this approach

as LDP-AGG. We set $\epsilon_0 = 0$ because LDP-AGG does not need to estimate the number of users.

Local memberships of a user in S different data parties can be seen as an S-dimension record. Thus, it is equivalent to randomizing each "dimension" of a user record independently with LDP protocols. After receiving all the local memberships of a user, the server first computes the probability vector of this user in all possible intersections $\mathcal{M}_{a_1}^{(1)} \cap \ldots \cap \mathcal{M}_{a_S}^{(S)}$. Then the server sums the probability vectors of all users to get the desired weights. The correlation of each user's attributes is preserved because the server first aggregates all local memberships of each user.

However, the reported memberships are very noisy. Based on the known LDP protocol error analysis [40, Proposition 10], the variance of an estimated cardinality with LDP-AGG is in the order $O\left(\frac{n}{\epsilon_2^{2S}}\right)$ for each intersection. The noise can easily overwhelm the true counts when ϵ_2 is small or S is large.

5.3.2 Prerequisite: DP FM Sketch

As is shown, neither baseline is satisfactory. An effective approach for private MemEnc and WeightEst should accurately maintain most of the correlation information between the inter-party attributes. We propose a new approach based on the Flajolet-Martin (FM) sketch because it can satisfy DP with a little additional overhead and support the *set union operation*.

Cardinality estimation sketches When handling a large amount of data, sketch is a popular succinct data structure that can store some basic information of the data with very low space and time complexity. One of the most well-known sketches is the Flajolet-Martin (FM) sketch [141], which is designed to estimate the cardinality (i.e., the number of distinct elements) of a (multi)set $\mathcal{M}$. In FM sketch, all the elements in $\mathcal{M}$ are hashed with $H_\zeta(\cdot)$. The estimate of the cardinality is $(1+\gamma)^\alpha$, where $\alpha = \max\{H_\zeta(x)|x \in \mathcal{M}\}$ and γ is the parameter for an ideal geometric-value hash function H. Typically, multiple (e.g., 1000) hash functions (H with different hash keys ζ) are used and we take the harmonic/geometric average as the final α. One appealing advantage of FM Sketch is that it is mergeable. With the same hash

key, sketches from different (multi)sets can be merged by taking the maximum, and we can derive the estimate of the cardinality of the union of those (multi)sets. With this property, we can estimate the cardinalities of the union/intersection of the set in the federated setting without leaking private information.

FM sketch achieves DP. Recently, some research results show that a family of hash-based, order-invariant sketches, including FM sketch, satisfy DP as long as the cardinality is large enough and the hash keys are unknown to the adversary [125, 126, 142, 143]. The DP version of the FM sketch algorithm is described as Algorithm 3. We follow the description of the FM sketch in Smith et al. [125], where the FM sketch is implemented using an ideal geometric-value hash function $H : \mathcal{X} \times \mathbb{Z} \to \mathbb{N}_+$ with parameter $\frac{\gamma}{1+\gamma}$. That is, given any finite set of distinct inputs $x_1, \ldots, x_\ell \in \mathcal{X}$, with a hash key $\zeta \sim \text{Uniform}(\mathbb{Z})$, the hashed values $H_\zeta(x_1), \ldots, H_\zeta(x_\ell)$ follow i.i.d. $\text{Geometric}\left(\frac{\gamma}{1+\gamma}\right)$ distribution.

Algorithm 3 DP FM Sketch Generation `DPFM` [125]

Input: a (multi)set $\mathcal{M}$, privacy parameter ϵ', and Geometric distribution parameter γ, an ideal random hash function $H_\zeta(x) \sim \texttt{Geometric}(\frac{\gamma}{1+\gamma})$ when $\zeta \sim \texttt{Uniform}(\mathbb{Z})$

Output: Sketch α for cardinality of $\mathcal{M}$

1: $n_p = \lceil \frac{1}{e^{\epsilon'}-1} \rceil$, $\alpha_{\min} = \lceil \log_{1+\gamma} \frac{1}{1-e^{-\epsilon'}} \rceil$
2: $\alpha_p = \max\{Y_1, \ldots, Y_{n_p}\}$ where $Y_i \sim \texttt{Geometric}(\frac{\gamma}{1+\gamma})$
3: $\alpha_{real} = \max\{H_\zeta(\texttt{id}) | \texttt{id} \in \mathcal{M}\}$
4: Return $\alpha = \max\{\alpha_p, \alpha_{real}, \alpha_{\min}\}$

Algorithm 3 generates a DP FM sketch. The intuition of the non-private FM sketch is that when elements in $\mathcal{M}$ are encoded as a set of geometric random variables and $\alpha_{real} = \max\{H_\zeta(\texttt{id}) | \texttt{id} \in \mathcal{M}\}$, we can expect $(1+\gamma)^{\alpha_{real}} \in \left[\frac{|\mathcal{M}|}{1+\gamma}, (1+\gamma) \cdot |\mathcal{M}|\right]$ with reasonable probability. Compared with the non-private version, the DP FM sketch needs two additional steps to ensure privacy: adding phantom elements, and lower-bounding the output by $\alpha_{\min}$. The phantom elements are used to ensure that the cardinality estimated by the final output is at least n_p; the $\alpha_{\min}$, which is at the $e^{-\epsilon'}-$quantile of the Geometric distribution, is used to ensure a probability that none of the items affects the output. It has been shown that the harmonic/geometric mean of M runs of Algorithm 3 satisfies DP:

Lemma 6 (Privacy guarantee of `DPFM` [125])**.** *Given an ideal geometric-value hash function H, Algorithm 3 is ϵ'-DP. Besides, repeating it M times with different hash keys and $\epsilon' = \frac{\epsilon}{4\sqrt{M\log(1/\delta)}}$ satisfies (ϵ, δ)-DP as long as $\epsilon \leq 2\log(1/\delta)$.*

5.3.3 Sketch-based MemEnc and WeightEst

Unlike the DP FM sketch [125] which aims to estimate the cardinality of a single set, our task is to encode the partitions, namely multiple local clusters consisting of user ids assigned to different local centers. Thus, we extend the FM sketch to encode the partition memberships, called Differentially Private Flajolet-Martin Partition Sketch (DPFMPS). We call an FM sketch vector for $\left\{\mathcal{M}_1^{(\ell)}, \ldots, \mathcal{M}_{k'}^{(\ell)}\right\}$ as a set of FM sketches. The sketch generation function, DPFMPS-Gen (Algorithm 4), is an instantiation of MemEnc. The data parties need to share a set of auxiliary parameters, $\texttt{aux} = \{\gamma, M, \boldsymbol{\zeta} = \{\zeta_1, \ldots, \zeta_M\}\}$, where γ is the Geometric distribution parameter, M is the number of sets of sketches and each $\zeta_i \sim \texttt{Uniform}(\mathbb{Z})$ is the hash key to an ideal geometric-value hash function for the i-th set of sketch. Notice that all data parties need to share the same set of hash keys. Algorithm 4 generates M sets of FM partition sketches and has the following privacy guarantee.

Theorem 5.3.1. *Given an ideal geometric-value hash function H, DPFMPS-Gen (Algorithm 4) generating M sets of partition sketches satisfies (ϵ_2, δ_2)-differential privacy.*

The above theorem is based on the following lemma about the privacy guarantee for each set of sketch, namely $\boldsymbol{A}_i$ in Algorithm 4.

Lemma 7. *Given an ideal geometric-value hash function H, each set of the partition sketch (i.e. a row $\boldsymbol{A}_i$) is ϵ'-DP.*

With Lemma 7, the privacy guarantee claimed in Theorem 5.3.1 can be derived with the sequential composition of RDP [20] and converted back to (ϵ, δ)-DP following the same proof as in [125].

Estimate intersection cardinality. One key observation of the local partition is that an `id` can be clustered to one and only one $\mathcal{M}_a^{(\ell)}$ by each data party ℓ. Thus, the intersection

Algorithm 4 DPFMPS-Gen

Input: A set of k' centers $\mathbf{C}^{(\ell)}$, dataset $\mathbf{X}^{(\ell)}$, privacy parameter ϵ_2 and δ_2, Geometric distribution paramter γ, number of sketches M and the corresponding hash keys $\boldsymbol{\zeta} = \{\zeta_1, \ldots, \zeta_M\}$

Output: M sets of sketches $\boldsymbol{A}$

$\epsilon' = \frac{\epsilon_2}{4\sqrt{M \log(1/\delta_2)}}$

Generate $\left\{\mathcal{M}_1^{(\ell)}, \ldots, \mathcal{M}_{k'}^{(\ell)}\right\}$ where $\mathtt{id} \in \mathcal{M}_j^{(\ell)}$ based on $\mathbf{C}^{(\ell)}$

$\boldsymbol{A} \leftarrow \mathbf{0}^{M \times k'}$

for $i \in [M], a \in [k']$ **do**

    $\boldsymbol{A}_{i,a} = \mathtt{DPFM}(\mathcal{M}_a^{(\ell)}, \epsilon', \gamma, \zeta_i)$

end for

Return $\boldsymbol{A}$

Algorithm 5 DPFMPS-BasicEst

Input: Estimate of the total number of users $\hat{n}$, Privacy parameter ϵ_2 and δ_2, the FM partition sketches $\{\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(s)}\}$

Output: Estimate of the weights $w(\mathbf{G})$

$\mathbf{U} \leftarrow \mathbf{0}^{M \times (k')^s}$, $\mathbf{w}' \leftarrow \mathbf{0}^{(k')^s}$

$\epsilon' = \frac{\epsilon_2}{4\sqrt{M \log(1/\delta_2)}}$, $n_p = \lceil \frac{1}{\mathrm{e}^{\epsilon'} - 1} \rceil$

for $i \in [M]$, $(a_1, \ldots, a_s) \in [k']^s$ **do**

    $\mathbf{U}_{i,(a_1,\ldots,a_s)} = \max\{\boldsymbol{A}_{i,a}^{(\ell)} \mid \ell \in [s], a \neq a_\ell\}$

end for

for $(a_1, \ldots, a_s) \in [k']^s$ **do**

    $\boldsymbol{\alpha}_{(a_1,\ldots,a_s)} = \mathtt{HarmonicMean}\left(U_{:,(a_1,\ldots,a_s)}\right)$

    $\mathbf{w}'_{(a_1,\ldots,a_s)} = (1+\gamma)^{\boldsymbol{\alpha}_{(a_1,\ldots,a_s)}} - s(k'-1)n_p$

    $w(\mathbf{G}_{(a_1,\ldots,a_s)}) = \hat{n} - \mathbf{w}'_{(a_1,\ldots,a_s)}$

end for

Ensure $\sum_{(a_1,\ldots,a_s)} w(\mathbf{G}_{(a_1,\ldots,a_s)}) = \hat{n}$ and $w(\mathbf{G})_{(a_1,\ldots,a_s)} \geq 0$

Return $w(\mathbf{G})$

problem can be transformed into the union problem by an extension of the inclusion-exclusion principle:

$$\bigcap_{\ell=1}^{S} \mathcal{M}_{a_\ell}^{(\ell)} = \overline{\bigcup_{\ell=1}^{S} \overline{\mathcal{M}_{a_\ell}^{(\ell)}}} = \overline{\bigcup_{\ell=1}^{S} \left(\bigcup_{a \neq a_\ell} \mathcal{M}_{a}^{(\ell)} \right)}. \tag{5.1}$$

Algorithm 5 gives the detailed procedure. There are $(k')^S$ possible intersections that need to be estimated, and there are M sets of FM sketches. In Line 2, $\mathbf{U}$ is initialized to store the FM sketches for the cardinalities of the complementary set of intersections; $\mathbf{w}'$ is an intermediate vector representing the cardinalities of union of complements, i.e. $\left|\bigcup_{\ell=1}^{s} \overline{\mathcal{M}_{a_\ell}^{(\ell)}}\right|$.

To estimate the cardinalities of the intersection with Equation (5.1), we first need to calculate the sketch of their complementary set, i.e., $\overline{\mathcal{M}_{a_\ell}^{(\ell)}}$. The corresponding FM sketch can be obtained by taking the max of all other elements in the same sketch set, $\max\left\{A_{i,a}^{(\ell)} | a \neq a_\ell\right\}$, according to the union property of FM sketch. Next, we need to derive the sketch for the union of the complementary partition of all data parties, i.e. $\bigcup_{\ell=1}^{s} \overline{\mathcal{M}_{a_\ell}^{(\ell)}}$. This union's FM sketches can be obtain by $\max\left\{\max\left\{A_{i,a}^{(\ell)} | a \neq a_\ell\right\} | \ell \in [s]\right\}$. Merging the two max operations give us the operation in Line 4.

We use the Harmonic mean over the M FM sketches to estimate the cardinality of $\bigcup_{\ell=1}^{s} \overline{\mathcal{M}_{a_\ell}^{(\ell)}}$ in Line 7, because it is shown [125] that the harmonic mean estimate is more stable and accurate. As this sketch is obtained after $s(k'-1)$ union operations, totally $s(k'-1)n_p$ phantom elements are taken into account. Therefore, we need to subtract $s(k'-1)n_p$ elements from the final estimation in Line 8. Finally, the intersection cardinality can be calculated by subtracting the cardinalities of $\bigcup_{\ell=1}^{s} \overline{\mathcal{M}_{a_\ell}^{(\ell)}}$ from the total number of users $\hat{n}$. Before finally returning the weights, we need to make sure the output is valid by enforcing non-negativity and total-sum equal to $\hat{n}$ on the weights (Line 11).

5.3.4 Privacy, Utility and Communication Cost

Privacy cost. According to the privacy spitting strategy in the Algorithm 2, Theorem 5.3.1 and the sequential composition of DP, the following privacy statement can directly derived.

Theorem 5.3.2. *Algorithm 2 is (ϵ, δ)-DP with DPFMPS-Gen and DPFMPS-BasicEst as the implementation of MemEnc and WeightEst.*

Error analysis of the weights. The cardinality estimate with Algorithm 3 approximates the real cardinality within a factor of $1 \pm \gamma$ and an additive error of $O\left(\sqrt{\ln(1/\delta)}/\epsilon\right)$ [125]. The additive error is because of the phantom elements, and $\alpha_{\min}$ also slightly increases the expectation of the sketch. We provide a refined result to show that the utility of the private FM sketches generated from Algorithm 3 is similar to the non-private FM sketch when the real cardinality is large enough, in order to use the more advanced results from non-private FM sketch research.

Lemma 8. *Let $\hat{\alpha} = \max\{\alpha_{real}, \alpha_p\}$ and α be the return of Algorithm 3. If H is an ideal geometric-value hash function and ϵ' is fixed, we have $\mathbb{E}[\alpha]/\mathbb{E}[\hat{\alpha}] \to 1$ and $\mathrm{Var}[\alpha]/\mathrm{Var}[\hat{\alpha}] \to 1$ when $|\mathcal{M}|$ is sufficiently large.*

With Lemma 8, we can claim that the lower bound $\alpha_{\min}$ does not affect the mean and variance of the sketch $\hat{\alpha}$ too much. So we can treat the α returned by Algorithm 3 approximately the same as the vanilla non-private FM sketch, and we can use the standard deviation results of the non-private FM sketch to analyze the error of `DPFM` in a more fine-grained way.

The standard deviation of a non-private FM sketch estimation can be represented as $\rho N/\sqrt{M}$, where ρ is a constant, N is the cardinality and M is the repetition [141, 144, 145]. Our algorithm has $s(k'-1)$ union operations to derive an element of $\mathbf{U}$, and each set has n_p phantom elements. So the cardinality estimated by $\mathbf{U}_{:,(a_1,\ldots,a_S)}$ becomes $N_{(a_1,\ldots,a_S)} = \left|\bigcup_{\ell\in[S]} \cup_{j\neq a_\ell} \mathcal{M}_j^{(\ell)}\right| + s(k'-1)n_p = \hat{n} - w^*(\mathbf{G}_{a_1,\ldots,a_S}) + s(k'-1)n_p$, where $w^*(\mathbf{G}_{a_1,\ldots,a_S})$ represent the true intersection cardinality $|\mathcal{M}_{a_1}^{(1)} \cap \ldots \cap \mathcal{M}_{a_S}^{(S)}|$. Based on the property of the non-private FM sketch and the value of M and n_p, we can state the following theorem.

Theorem 5.3.3. *Given a constant ρ such that the non-private FM sketch's standard deviation is $\rho N/\sqrt{M}$ where N is the cardinality, and M is number of repetitions. With n_p and ϵ' set as in Algorithm 5, each intersection cardinality estimate generated has standard deviation*

$$\sigma_{(a_1,\ldots,a_s)} = \frac{\rho(n - w^*(\mathbf{G}_{(a_1,\ldots,a_s)}))}{\sqrt{M}} + \frac{4\rho s(k'-1)\sqrt{\log(1/\delta)}}{\epsilon_2}$$

The result is directly derived after plugging in the value of $N_{(a_1,\ldots,a_S)}$ and n_p, and use the approximation $e^y \approx y + 1$ when y is a small positive number.

Final utility cost. When we set $k' = k$ as indicated in the non-private setting [119], we can show that Algorithm 2 is a (β, λ)-approximation algorithm with assumption of accessibility to some guaranteed central (private) k-means algorithm.

Theorem 5.3.4. *Assume the data parties have access to a differentially private $(\beta_{priv}, \lambda_{priv})$-approximate k-means algorithm, and the central server has access to a (non-private) $(\beta_o, 0)$-approximate k-means algorithm. Algorithm 2 with DPFMPS-Gen and DPFMPS-BasicEst is a (β, λ)-approximation algorithm with probability $1 - \omega$, where*

$$\beta = 2\beta_{priv} + 4\beta_0 + 4\beta_0\beta_{priv},$$
$$\lambda = 2(\beta_0 + 1)S\lambda_{priv} + O\left(\frac{\beta_0 m^2 k^{1.5S}}{\sqrt{\omega}}\left(\frac{n}{\sqrt{M}} + \frac{S(k-1)\sqrt{\log(1/\delta)}}{\epsilon_2}\right)\right)$$

The multiplicative error β is composed of β_{priv} from the LocCluster and β_0 from the final non-private clustering on central server. The first term of the additive error λ can be understood as the cumulative error of S local DP k-means algorithm results[2], and the second term comes from the cardinality estimation error. The second term can dominate the error when the number of centers k or the number of data parties S is not small. The exponential factor $1.5S$ comes from Chebyshevs inequality with the union bound.

Communication cost. There is only one round of communication between the data parties and the central server. Each data party sends local centers together with M sets

[2]↑The best theoretical results of DP k-means in central setting [137] states that $\lambda_{priv} = O_{\beta,\eta}\left(\frac{km+k^{O_\eta(1)}}{\epsilon_1}\text{ polylog}n\right)$ with a small positive constant η.

of sketches to the server, with communication cost $O\left((m^{(\ell)}+M)k'\right)$. The central server receives all these messages from all local parties with space in $O\left(mk'+SMk'\right)$.

5.3.5 Privacy proofs

Proof of Lemma 7. To prove the privacy guarantee of Algorithm 4, we denote a set of identities as $\mathcal{M} = \{\mathtt{id}_1, \ldots, \mathtt{id}_n\}$, where $\mathtt{id}$ is not necessarily a integer, but any input that is hash-able. For clustering, $\mathcal{M}$ is divided into $\{\mathcal{M}_1^{(\ell)}, \ldots, \mathcal{M}_{k'}^{(\ell)}\}$ according to the differentially private $\mathbf{C}^{(\ell)}$ from the previous phase: For any $\mathtt{id}$, the partition index is $\arg\min_{j\in[k']} \left\| x_{\mathtt{id}}^{(\ell)} - c_j^{(\ell)} \right\|_2^2$. Now given a pair of neighboring datasets $\mathcal{M}$ and $\mathcal{M}'$, where there is an $\mathtt{id}$ such that $\mathtt{id}' \notin \mathcal{M}$ but $\mathtt{id}_{i'} \in \mathcal{M}'$, their clustering memberships, $\{\mathcal{M}_1, \ldots, \mathcal{M}_{k'}\}$ and $\{\mathcal{M}'_1, \ldots, \mathcal{M}'_{k'}\}$, differing at exactly one partition $j \in [k']$, such that $\mathtt{id} \in \mathcal{M}'_j, \mathtt{id} \notin \mathcal{M}_j$; for all other subsets $\forall j' \neq j, \mathcal{M}_{j'} = \mathcal{M}'_{j'}$. Thus, the following proof can be considered as following the parallel composition property. □

5.3.6 Utility proofs

To simplify the description, we denote $\phi_{\mathbf{C}^{(\ell)}}(\cdot)$ as a partition function mapping a user record in the local dataset of party ℓ to the closest center's index in $\mathbf{C}^{(\ell)} = \{c_1^{(\ell)}, \ldots, c_{k'}^{(\ell)}\}$ generated in the previous phase, i.e., $\phi_{\mathbf{C}^{(\ell)}}\left(x_{\mathtt{id}}^{(\ell)}\right) = \arg\min_{a\in[k']} \left\| x_{\mathtt{id}}^{(\ell)} - c_a^{(\ell)} \right\|_2^2$. With the partition function, each data party partitions the users into disjoint sets $\{\mathcal{M}_1^{(\ell)}, \ldots, \mathcal{M}_{k'}^{(\ell)}\}$ such that user $\mathtt{id} \in \mathcal{M}_a^{(\ell)}$ if and only if $a = \phi_{\mathbf{C}^{(\ell)}}\left(x_{\mathtt{id}}^{(\ell)}\right)$.

Proof of Lemma 8. Consider the output of the geometric-valued hash function H, $Z = \max\{Y_1, \ldots, Y_{\tilde{n}}\}$, where $Y_i \sim \mathtt{Geometric}(\frac{\gamma}{1+\gamma})$. The cumulative mass function of Z is

$$\mathsf{Pr}\left[Z \leq z\right] = \left(1 - \left(1 - \frac{\gamma}{1+\gamma}\right)^z\right)^{\tilde{n}} = \left(1 - \frac{1}{(1+\gamma)^z}\right)^{\tilde{n}}.$$

The probability $\mathsf{Pr}[Z \le \alpha_{\min}] = \frac{1}{e^{\epsilon' \tilde{n}}}$. Based on this result, the ratio of expectation can be written as

$$\begin{aligned}\frac{\mathbb{E}[\alpha]}{\mathbb{E}[\hat{\alpha}]} =& 1 - \frac{\sum_{z \le \alpha_{\min}} \mathsf{Pr}[Z = z](\alpha_{\min} - z)}{\mathbb{E}[\hat{\alpha}]} \\ \ge& 1 - \frac{\sum_{z \le \alpha_{\min}} \mathsf{Pr}[Z = z]\alpha_{\min}}{\mathbb{E}[\hat{\alpha}]} \ge 1 - \frac{\log_{1+\gamma}(1/(1-e^{-\epsilon'}))}{e^{\epsilon' \tilde{n}}\mathbb{E}[\hat{\alpha}]}\end{aligned}$$

Let $\tilde{n} = |\mathcal{M}| + n_p$. When $|\mathcal{M}|$ is large enough, the denominator dominants and the value will go to 0 very quickly. Similar analysis can also be applied to the variance. □

Proof of Theorem 5.3.4. For simplicity, we denote the virtual global dataset as $\mathbf{X} = [\mathbf{X}^{(1)}| \dots |\mathbf{X}^{(S)}]$, the optimal global centers as $\mathbf{C}^* = \{c_1^*, \dots, c_k^*\}$, and the membership of grid g as $\mathcal{M}_g$ We also denote $\phi^*(\cdot)$ as the optimal partition function mapping a data point to a closest center index. We first can bound the $\mathtt{cost}_{\mathbf{X}}$ as the following:

$$\begin{aligned}\mathtt{cost}_{\mathbf{X}}(\mathbf{C}) &= \sum_{i \in [n]} \left\| x_i - c_{\phi*(x_i)} \right\|_2^2 \\ &\le \sum_{g \in \mathbf{G}} \sum_{i \in \mathcal{M}_g} \left\| x_i - g + g - c_{\phi*(g)} \right\|_2^2 \\ &\le 2 \sum_{g \in \mathbf{G}} \sum_{i \in \mathcal{M}_g} \|x_i - g\|_2^2 + \left\| g - c_{\phi*(g)} \right\|_2^2\end{aligned}$$

We first analysis the first term. Notice that every local data party runs a $(\beta_{priv}, \lambda_{priv})$-approximate algorithm, and g consists of S local centers $[c_{a_1}^{(1)} \mid \dots \mid c_{a_S}^{(S)}]$

$$\begin{aligned}\sum_{g \in \mathbf{G}} \sum_{i \in \mathcal{M}_g} \|x_i - g\|_2^2 &= \sum_{\ell \in [S]} \sum_{i \in [n]} (x_i^{(\ell)} - c_{a_\ell}^{(\ell)})^2 \\ &\le \sum_{\ell \in [S]} (\beta_{priv} \mathtt{OPT}_{k, \mathbf{X}^{(\ell)}}^{(\ell)} + \lambda_{priv}) \\ &= \beta_{priv} \sum_{\ell \in [S]} \mathtt{OPT}_{k, \mathbf{X}^{(\ell)}}^{(\ell)} + S\lambda_{priv} \\ &\le \beta_{priv} \mathtt{OPT}_{k, \mathbf{X}} + S\lambda_{priv}\end{aligned}$$

Notice that $w(g)$ is only an estimation of $|\mathcal{M}_g|$ in our algorithm. So we denote $\kappa = \sum_{g\in\mathbf{G}} |w(g) - |\mathcal{M}_g||$. The second term can be relaxed as the following:

$$\begin{aligned}\sum_{g\in\mathbf{G}}\sum_{i\in\mathcal{M}_g} \left\|g - c_{\phi*(g)}\right\|_2^2 &= \sum_{g\in\mathbf{G}} |\mathcal{M}_g| \left\|g - c_{\phi*(g)}\right\|_2^2 \\ &\leq \sum_{g\in\mathbf{G}} |w(g)| \left\|g - c_{\phi*(g)}\right\|_2^2 + 4m^2\kappa\end{aligned}$$

As the central server finally runs an $(\beta_0, 0)$-approximate k-means algorithm, we also denote $\hat{\mathbf{C}} = \{\hat{c}_1, \ldots, \hat{c}_k\}$ as the set of optimal k centers for k-means given $(\mathbf{G}, w(\mathbf{G})$ as the dataset. Thus, it follows that

$$\begin{aligned}&\sum_{g\in\mathbf{G}} |w(g)| \left\|g - c_{\phi*(g)}\right\|_2^2 \\ \leq\ & \beta_0 \sum_{g\in\mathbf{G}} |w(g)| \left\|g - \hat{c}_{\phi*(g)}\right\|_2^2 \\ \leq\ & \beta_0 \sum_{g\in\mathbf{G}} |\mathcal{M}_g| \left\|g - \hat{c}_{\phi*(g)}\right\|_2^2 + 4\beta_0 m^2\kappa \\ \leq\ & \beta_0 \sum_{g\in\mathbf{G}}\sum_{i\in\mathcal{M}_g} \left\|g - x_i + x_i - c^*_{\phi*(x_i)}\right\|_2^2 + 4\beta_0 m^2\kappa \\ \leq\ & 2\beta_0 \sum_{g\in\mathbf{G}}\sum_{i\in\mathcal{M}_g} \|g - x_i\|_2^2 + 2\beta_0 \sum_{g\in\mathbf{G}}\sum_{i\in\mathcal{M}_g} \left\|x_i - c^*_{\phi*(x_i)}\right\|_2^2 + 4\beta_0 m^2\kappa\end{aligned}$$

We have bounded the first term above and can reuse the result here. The second term is the exactly $2\beta_0 \mathtt{OPT}_{k,\mathbf{X}}$ by definition. So when we put every thing together, we have

$$\begin{aligned}\mathtt{cost}_{\mathbf{X}}(\mathbf{C}) \leq\ & (2\beta_{priv} + 4\beta_0 + 4\beta_0\beta_{priv})\, \mathtt{OPT}_{k,\mathbf{X}} \\ & + 2(\beta_0 + 1)S\lambda_{priv} \\ & + 8(\beta_0 + 1)m^2\kappa\end{aligned}$$

To bound the t, we can use Chebyshev's inequality with the union bound of all grid nodes and the results of Lemma 5.3.3, the standard $\sigma = O\left(\frac{n}{\sqrt{M}} + \frac{S(k-1)\sqrt{\log(1/\delta)}}{\epsilon_2}\right)$. Because

there are totally k^S intersection cardinality estimates in the grid, the probability of the odd scenario for each intersection cardinality estimate we want to bound is $\frac{\omega}{k^S}$ because of the union bound. According to the Chebyshevs inequality, for one intersection estimation, $\mathsf{Pr}\left[|w(g) - |\mathcal{M}_g|| \geq \sigma\sqrt{\frac{k^S}{\omega}}\right] \leq \frac{\omega}{k^S}$. Plug in the standard deviation result and consider the there are totally k^S intersection, we can have the result shown in our theorem with $k^{1.5S}$. Thus, with probability at least $1-\omega$,

$$\kappa \leq \frac{k^{1.5S}}{\sqrt{\omega}}\sigma = O\left(\frac{nk^{1.5S}}{\sqrt{\omega M}} + \frac{S(k-1)k^{1.5S}\sqrt{\log(1/\delta)}}{\epsilon_2\sqrt{\omega}}\right)$$

Plugin the result so that

$$\begin{aligned}\texttt{cost}_{\mathbf{X}}(\mathbf{C}) \leq\ & (2\beta_{priv} + 4\beta_0 + 4\beta_0\beta_{priv})\,\texttt{OPT}_{k,\mathbf{X}} \\ & + 2(\beta_0+1)S\lambda_{priv} \\ & + O\left((\beta_0+1)m^2\left(\frac{nk^{1.5S}}{\sqrt{\omega M}} + \frac{S(k-1)k^{1.5S}\sqrt{\log(1/\delta)}}{\epsilon_2\sqrt{\omega}}\right)\right)\end{aligned}$$

□

5.4 Improving Utility of the Algorithm

We showed that DPFMPS-BasicEst has advantages over the baseline methods theoretically in the previous section. However, we empirically find that there is still large room to improve to the basic estimation. We focus on two different aspects of the algorithms and propose our improved versions with higher utility.

5.4.1 Improving Post-processing Estimation Algorithm for More than Two Data Parties

As the number of data parties increases, the accuracy of the weight estimation will decrease dramatically. From the error analysis of the cardinality estimation (Theorem 5.3.3), one can see that the second term in standard deviation $\sigma_{(a_1,\ldots,a_S)}$ increases proportionally with the number of data parties S. Furthermore, the total possible intersection combinations

grow exponentially as $(k')^S$, which decreases the expected number of data points in the intersection, i.e., the expected $w^*(\mathbf{G}_{(a_1,\dots,a_S)})$. The combination of the two factors means that the relative error of the intersection estimation will rapidly increase as the number of parties grows.

Two observations give us hope to lessen the negative effect. The first observation is that estimation of two-party intersection cardinalities is relatively accurate. The second observation is based on the distributive property of set intersection:

$$\left|\mathcal{M}^{(\ell_1)}_{a'_{\ell_1}} \bigcap \mathcal{M}^{(\ell_2)}_{a'_{\ell_2}}\right| = \sum_{\substack{(a_1,\dots,a_S)\in[k']^S \\ a_{\ell_1}=a'_{\ell_1}, a_{\ell_2}=a'_{\ell_2}}} \left|\bigcap \mathcal{M}^{(\ell)}_{a_\ell}\right|, \tag{5.2}$$

which give the connection between the all-party intersection cardinality (i.e., $w(\mathbf{G})$) and the two-party version (i.e., $w(\mathbf{G}^{(\ell_1,\ell_2)})$). Thus, we propose to deal with this challenge by 1) *computing only all pair-wise intersection cardinalities*, and 2) using these two-party intersection cardinalities with Equation (5.2) as constraints, and iteratively update the $w(\mathbf{G})$ to fulfill all this constraints.

The improved estimation is described in Algorithm 6 DPFMPS-2PEst. The central server first estimates the single-party cardinality of all local clusters with only $\boldsymbol{A}^{(\ell)}$ (Line 3). Namely, $w^{(\ell)}_a$ is an estimate for $|\mathcal{M}^{(\ell)}_a|$. Then the server initializes the full grid weights $w(\mathbf{G})$ with only the single-party cardinality information assuming no correlation between the inter-party attributes (Line 5). Next, the server estimates all two-party weights with DPFMPS-BasicEst as a sub-procedure (Line 7). To be more detailed, the grid weights $w(\mathbf{G}^{(\ell_1,\ell_2)})$ returned by DPFMPS-BasicEst with $\boldsymbol{A}^{(\ell_1)}$ and $\boldsymbol{A}^{(\ell_2)}$ are the estimates for $\left\{\left|\mathcal{M}^{(\ell_1)}_{a_{\ell_1}} \bigcap \mathcal{M}^{(\ell_2)}_{a_{\ell_2}}\right| \mid a_{\ell_1}, a_{\ell_2} \in [k']\right\}$. With all pairs of $w(\mathbf{G}^{(\ell_1,\ell_2)})$, the server iteratively updates the grid weights $w(\mathbf{G})$ to make them consistent with all the two-party intersection cardinalities $w(\mathbf{G}^{(\ell_1,\ell_2)})$. In each iteration, the server first randomly selects a pair of data parties (Line 10), groups the current full grid weights $w(\mathbf{G})$ by the cluster indices of the

chosen two parties, and sums the weights in the same group to generate a two-party grid weights $\tilde{w}(\mathbf{G}^{(\ell_1,\ell_2)})$ according to Equation (5.2):

$$\tilde{w}(\mathbf{G}^{(\ell_1,\ell_2)}_{(a'_{\ell_1},a'_{\ell_2})}) = \sum_{\substack{(a_1,\ldots,a_S)\in[k']^S \\ a_{\ell_1}=a'_{\ell_1},a_{\ell_2}=a'_{\ell_2}}} w(\mathbf{G}_{(a_1,\ldots,a_S)})$$

We use the differences between $w(\mathbf{G}^{(\ell_1,\ell_2)})$ and $\tilde{w}(\mathbf{G}^{(\ell_1,\ell_2)})$ to update the weight evenly with a step size η_t (Line 15). After sufficient update iterations, the full-party grid weight $w(\mathbf{G})$ is expected to approximately satisfy the constraints (Equation (5.2)) with all two-party weights $w(\mathbf{G}^{(\ell_1,\ell_2)})$.

Because both DPFMPS-2PEst and DPFMPS-BasicEst are post-processing components in DP definition, the privacy guarantee in Theorem 5.3.2 still holds when DPFMPS-2PEst is used.

Algorithm 6 DPFMPS-2PEst

Input: Estimated total number of users $\hat{n}$, privacy parameter ϵ_2 and δ_2, the FM partition sketches $\{\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(s)}\}$

Output: Estimate of the weights $w(\mathbf{G})$

1: $\epsilon' = \frac{\epsilon_2}{4\sqrt{M\log(1/\delta_2)}}$, $n_p = \lceil \frac{1}{e^{\epsilon'}-1} \rceil$
2: **for** $\ell \in [S], a \in [k']$ **do**
3: $\quad w_a^{(\ell)} \leftarrow (1+\gamma)^{\texttt{HarmonicMean}\left(\boldsymbol{A}^{(\ell)}_{:,a}\right)} - n_p.$
4: **end for**
5: $\forall\{a_1,\ldots,a_S\} \in [k']^S, w(\mathbf{G}_{(a_1,\ldots,a_S)}) \leftarrow \hat{n} \times \prod_{\ell\in[S]} \frac{w^{(\ell)}_{a_\ell}}{\hat{n}}$
6: **for** $(\ell_1,\ell_2) \in [S]$ and $\ell_1 \neq \ell_2$ **do**
7: $\quad w(\mathbf{G}^{(\ell_1,\ell_2)}) \leftarrow$ DPFMPS-BasicEst$(\hat{n},\epsilon_2,\delta_2,\{\boldsymbol{A}^{(\ell_1)},\boldsymbol{A}^{(\ell_2)}\})$
8: **end for**
9: **for** $t \in [T]$ **do**
10: $\quad$ Randomly select a pair $(\ell_1,\ell_2) \in [S]\times[S]$
11: $\quad \tilde{w}(\mathbf{G}^{(\ell_1,\ell_2)}) \leftarrow \texttt{Proj}(w(\mathbf{G}),\ell_1,\ell_2)$
12: $\quad \Delta^{(\ell_1,\ell_2)} \leftarrow \tilde{w}(\mathbf{G}^{(\ell_1,\ell_2)}) - w(\mathbf{G}^{(\ell_1,\ell_2)})$
13: $\quad$ **for** $(a'_{\ell_1},a'_{\ell_2}) \in [k']^2$ **do**
14: $\quad\quad \forall(a_1,\ldots,a_S) \in [k']^S, a_{\ell_1} = a'_{\ell_1}, a_{\ell_2} = a'_{\ell_2}$
15: $\quad\quad w(\mathbf{G}_{(a_1,\ldots,a_S)}) \leftarrow w(\mathbf{G}_{(a_1,\ldots,a_S)}) - \frac{\eta_t}{(k')^{S-2}}\Delta^{(\ell_1,\ell_2)}_{(a'_{\ell_1},a'_{\ell_2})}$
16: $\quad$ **end for**
17: **end for**
18: Return $w(\mathbf{G})$

5.4.2 Auto-adjusted k'

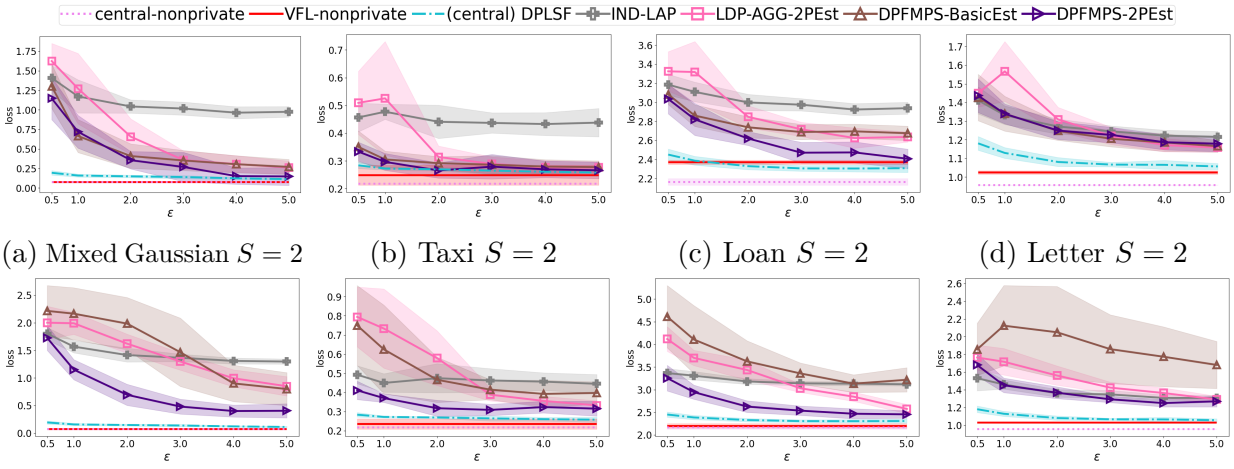
The utility result of Theorem 5.3.4 is given by assuming the number of local and central clusters is the same, i.e., $k' = k$. However, we can see a trade-off on the value of k'. In the non-private setting, the larger the k' is, the more local dataset information is preserved for the final central clustering. On the other hand, the larger the k' is, the more fine-grained the grid becomes, and the fewer records (or smaller $w^*(\mathbf{G}_{a_1,\ldots,a_S})$) are expected to be assigned to the grid node. According to Theorem 5.3.3, a larger k' can make the error of $w(\mathbf{G}_{a_1,\ldots,a_S})$ larger and increase the final cost.

Giving a closed form solution for the local k' to minimize the final loss is difficult. However, as we can see from the error bound of Theorem 5.3.4, the grid quality reflected by the second term of γ_{priv} plays an important role. We propose an empirical rule to set $k' = \max\{k_0, k^{1/S}\}$ for DPFMPS-2PEst to prevent the true cardinalities from being overwhelmed by the noise, where k_0 is the smallest integer that satisfies $2\sigma_{(a_1,\ldots,a_s)} \geq \frac{\hat{n}}{k_0^2}$. Because $w^*(\mathbf{G}_{(a_1,\ldots,a_S)})$ in Theorem 5.3.3 is unknown, we approximate it as $\frac{\hat{n}}{k_0^2}$; we set $s = 2$ in the standard deviation because DPFMPS-2PEst only decodes pairs of intersection cardinalities; and we set $\rho = 0.649$ according to [144]. We experimentally demonstrate that such a choice of k' can be good choices for the datasets in our experiments in Section 5.5.2.

5.5 Experiments

Datasets. In this work, we use four different datasets to empirically verify the advantages of our proposed methods for private vertical federated clustering. To simulate the vertical federated learning, we split the datasets by attributes into S partitions evenly, and all attributes are normalize to $[-1, 1]$ before clustering.

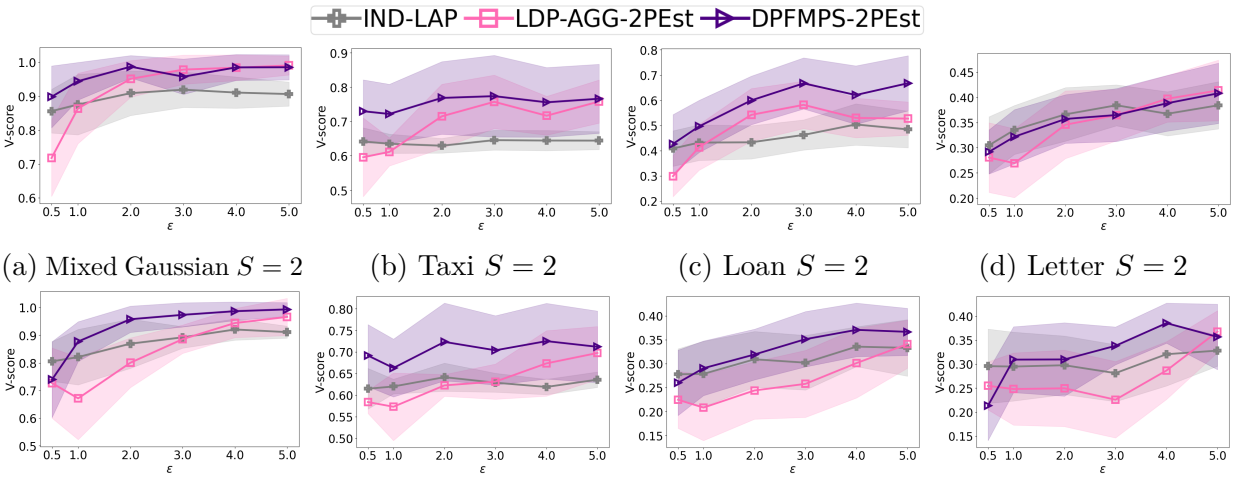
Synthetic mixed Gaussian dataset with k centers. To echo the implicit assumption of k-means problem, we first synthesize a mixed Gaussian dataset with $m = 8$ for evaluation. We generated this data in a similar way as [138] but enforced each dimension's range to be $[-1, 1]$ instead of in a L_2 ball. We first randomly sample $k = 5$ centers in the domain, then



(a) Mixed Gaussian $S = 2$ (b) Taxi $S = 2$ (c) Loan $S = 2$ (d) Letter $S = 2$

(e) Mixed Gaussian $S = 4$ (f) Taxi $S = 4$ (g) Loan $S = 4$ (h) Letter $S = 4$

Figure 5.2. k-means loss with final k centers.



(a) Mixed Gaussian $S = 2$ (b) Taxi $S = 2$ (c) Loan $S = 2$ (d) Letter $S = 2$

(e) Mixed Gaussian $S = 4$ (f) Taxi $S = 4$ (g) Loan $S = 4$ (h) Letter $S = 4$

Figure 5.3. V-measure scores.

randomly sample $n = 20,000$ data points from the Gaussian distribution with the means at those k centers.

New York Taxi dataset [146]. This taxi dataset contains 8 attributes and 100,000 records of taxi trips information, including pick-up/drop-off times and locations. We preprocess the pick-up/drop-off time to make them a number indicating the time in a week, ranging from 0 to $60 \times 60 \times 24 \times 7$ before normalizing them.

Loan dataset [147]. The original Loan dataset has 120 attributes. We extract the first 60k records and 16 numerical attributes of the applicant's credit information and apartment information. We clipped these attributes' values and made them upper-bounded their original 95% quantile to eliminate the outliers.

Letter dataset [148]. This dataset consists of information about black-and-white rectangular pixels displayed as the letters in the English alphabet. There are 16 attributes and 20,000 records in the datasets.

Parameter settings. There are different methods to decide the best k value for the real-world datasets. The Silhouette method [149] is one of the most commonly used. Silhouette measures how closely data points are matched to data within its cluster and how loosely they are matched to data of the neighboring cluster. The datasets we use in this work all have relatively high Silhouette scores when k is smaller. Thus, we fix $k = 5$ for experiments in this work to eliminate the effect of k unless we explicitly mention it.

According to the experimental results in [125], a larger number of repeated FM sketches (hyper-parameter M in our work) leads to a smaller relative error of cardinality estimation. Thus, we fix $M = 4096$ in different experiments, making the communication cost reasonable in the cross-silo FL setting and achieving stable accuracy. We vary ϵ values for different levels of privacy protection strengths, and we fix $\delta = 1/n$ for the experiments.

Evaluation Metric. We evaluate the quality of the final output by two metrics. The first is the sum of central k-means loss with all the data points, i.e., $\sum_{x \in \mathbf{X}} (\min_{c \in \mathbf{C}} \|x - c\|^2)$. The second is the V-measure [150] in the scikit-learn package [151], which measures how "close" the clustering results of VFL algorithms are when compared to the ground truth classes (for the synthetic dataset) or central non-private k-means result (for the real word

datasets). The V-measure is the harmonic mean of two conditional entropy scores measuring the homogeneity and completeness. It is a scores between 0 and 1, and the closer to 1 the better. Because of the space limitation, we refer readers to [150] for detail formulas.

Compared methods. The adapted DPLSF is used as instantiation of LocCluster in all private VFL experiments. So we name the end-to-end private VFL clustering method based on their MemEnc and WeightEst instantiations. We compare the following methods in our experiments:

- our proposed method DPFMPS-BasicEst and DPFMPS-2PEst;
- two baseline methods IND-LAP and LDP-AGG-2PEst (LDP-AGG-2PEst is an improved version of LDP-AGG using the same technique in Section 5.3.1; because directly applying the LDP-AGG gives us a very large loss when $S > 2$);
- the central private k-means method (central) DPLSF from [139];
- the central non-private k-means from the scikit-learn package [151]
- a nonprivate VFL implementation VFL non-private following [119].

5.5.1 End-to-end Comparison

We first demonstrate the end-to-end performance of the algorithm and show the advantages of our proposed algorithms.

k-means loss results. Figure 5.2 shows the loss of the final centers produced by different algorithms. As we can see, the losses of non-private vertical federated k-means are higher than the that of the central version in most cases. Notice that we set $k' = k = 5$ for the non-private VFL algorithm, so there are 25 or 625 grid nodes when $S = 2$ or $S = 4$ accordingly. It means some information is lost when we building the grid, and more fine-grained grid can improve the utility in the non-private setting if we compares the figures in the first rows with the ones in the second row.

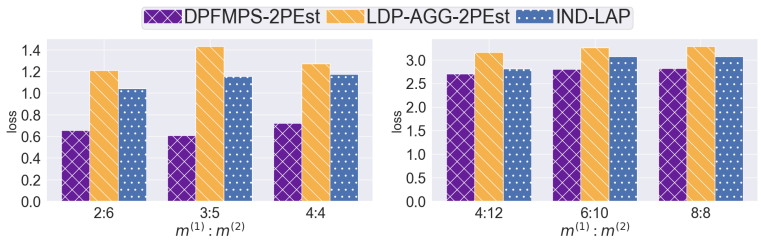
Comparing our proposed method, DPFMPS-2PEst, with the two baseline methods and DPFMPS-BasicEst, we can see that our proposed method can produce final centers with consistently lower losses for most settings. When there is only two parties ($S = 2$), DPFMPS-

BasicEst has the same performance as DPFMPS-2PEst because they are the same; but DPFMPS-2PEst largely improve over DPFMPS-BasicEst when $S = 4$. The exception is the results of the IND-LAP results of the Letter dataset (Figure 5.2d and 5.2h), where IND-LAP approach has slightly better performance when ϵ is small. That is because the attributes of the Letter dataset are relatively independent of each other does not follow the underlying assumption of k-means algorithm. As for the LDP baseline, LDP-AGG-2PEst, we can see that it's performances is close to our proposed method only with large privacy budget but is inferior when ϵ is small, which echoes with the theoretical results in Sections 5.3. When the privacy budget is large enough, the LDP protocol has little randomness to perturb the local membership of a user. In contrast, the FM sketch has its inherent randomness, even with large privacy budgets.

Moreover, comparing the private central with the private VFL algorithms, we can see that the private central VFL method almost always has a higher cost than the central one. The gap between the two exists because of two different reasons. The first reason is that compared with the central setting, only the local centers, and the sketches are shared with the central server, and part of the information is lost. The second reason is that when we use sketches to estimate the cardinalities of the intersections, part of the privacy budget is spent on the estimation.

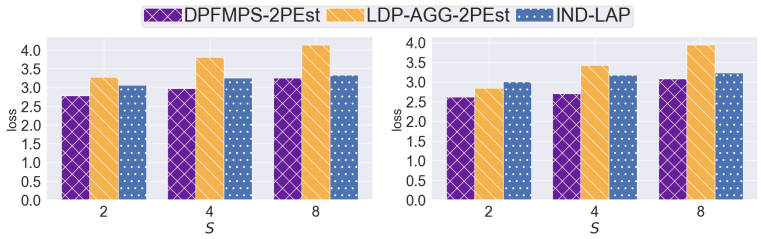
V-measure scores. When generating the synthetic mixed Gaussian dataset, we assign the same label for the data points drawn from the same center. However, the real-world datasets have no label, or the number of labels is different from our experiment setting, so we apply the central non-private k-means algorithm to generate the labels for the dataset. Thus, the closer the score is to 1, the more similar the clustering are to the real labels (for Mixed Gaussian) or the central non-private k-means clustering (Taxi, Loan and Letter).

Figure 5.3 presents the result. The results align with the k-means loss results. In most settings, we can observe that DPFMPS-2PEst outperforms the other two VFL private baseline methods in Figure 5.3. For the Letter dataset, we can see that all three methods have lower scores than the other two. We believe it is because the data in Letter have very independent attributes and bring advantage to the IND-LAP. However, DPFMPS-2PEst still outperforms



(a) Mixed Guassian $\epsilon = 1$ (b) Loan $\epsilon = 1$

Figure 5.4. Effect of unevenly split dataset.



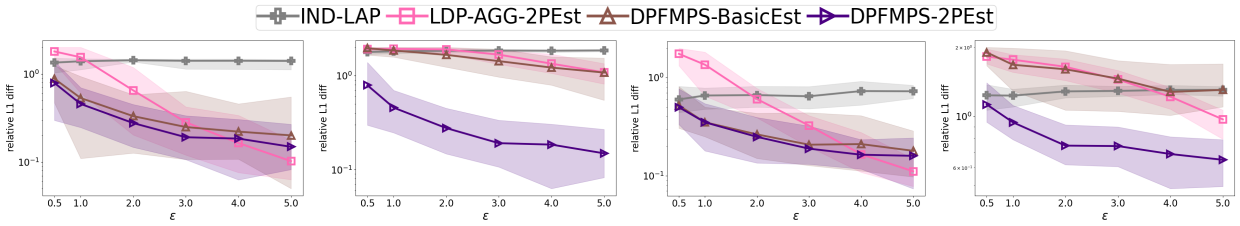
(a) Loan $\epsilon = 1$ (b) Loan $\epsilon = 2$

Figure 5.5. Effect of different S on Loan dataset ($m = 16$).

other methods with $S = 4$ with most privacy budgets, and it also has performance close to best when $S = 2$.

Effect of uneven number of attributes. The previous results show how DPFMPS-2PEst outperforms other baselines method when the attributes are evenly split. We also explore how the methods perform when the each party has different number of attributes. Denote $m^{(i)}$ as the number of attributes in the i-th party's dataset. We split the mixed Gaussian dataset so that $m^{(1)} : m^{(2)}$, is $3 : 5$ or $2 : 6$, and split the Loan dataset to $6 : 10$ and $4 : 12$. The results shown in Figure 5.4 indicate that our DPFMPS-2PEst work consistently well in the sense that the losses of different split ratios do not change much. Besides, DPFMPS-2PEst losses are smaller than the other two baselines in the figure.

Effect of number of parties S. We also compare how the number of parties S can affects the final clustering results in Figure 5.5. The loss generally increases as S increases. This is mainly because the WeightEst component introduces a larger error because of the random noise for privacy. However, DPFMPS-2PEst always provides the lowest loss among the three.



(a) Mixed Gaussian $S = 2$ (b) Mixed Gaussian $S = 4$ (c) Loan $S = 2$ (d) Loan $S = 4$

Figure 5.6. relative error of estimate the intersection cardinalities.

5.5.2 Ablation Study of Components

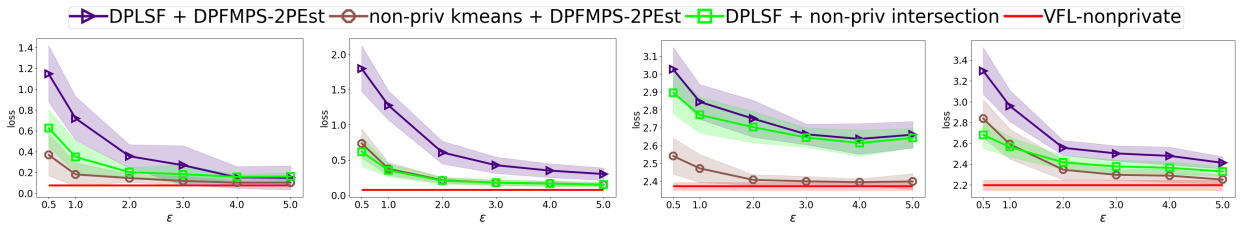
To further demonstrate the impact of enforcing privacy on different components in the vertical clustering algorithm, we perform ablation studies.

Intersection cardinality accuracy comparison. To break-down the error, the first interesting metric is the relative accuracy of the intersection cardinality estimation. The relative error is defined as

$$\frac{1}{n}\sum_{(a_1,\ldots,a_S)} \left\| w(\mathbf{G}_{(a_1,\ldots,a_S)}) - w^*(\mathbf{G}_{(a_1,\ldots,a_S)}) \right\|_1$$

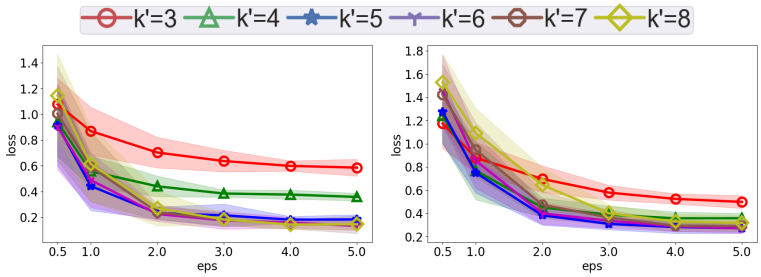
We fix $k = k' = 5$ for fair comparison of all methods and the results are shown as Figure 5.6. Because the evaluation is based on the intermediate results of the end-to-end private algorithm, so only half of the privacy budgets are spend on the intersection estimation. We can see that our proposed method DPFMPS-2PEst can outperform other methods in most experiment settings. The DPFMPS-BasicEst performs similar as DPFMPS-2PEst in the $S = 2$ setting as expected, DPFMPS-2PEst can significantly improve the accuracy when there are more parties ($S = 4$). The LDP baseline LDP-AGG-2PEst still have relative error larger than the DPFMPS-2PEst, even with the 2-way iterative updating. Moreover, as we can see from the figure, the error of the 1-way approach `IND-LAP` is dominated by loss of dependency information between the attributes and barely going down as we increase the privacy budget.

Distinguishing the impact of private LocCluster from private intersection cardinality estimation. In Figure 5.7, we compare the impact of enforcing privacy on



(a) Mixed Gaussian $S = 2$ (b) Mixed Gaussian $S = 4$ (c) Loan $S = 2$ (d) Loan $S = 4$

Figure 5.7. Comparing the impact of (enforcing privacy on) different components.



(a) Mixed Gaussian, $S = 2$ (b) Mixed Gaussian, $S = 4$

Figure 5.8. Different local k' with different privacy budget.

Table 5.1. Automatically chosen k'.

n \ ϵ	$S=2$ 0.5	1	2	3	4	$S=4$ 0.5	1	2	3	4
20000	4	5	5	6	6	4	4	5	5	5
60000	5	6	6	7	7	4	5	6	6	6
100000	6	6	7	7	7	5	6	6	7	7

either the local clustering component or the intersection estimation component. For "`non-priv kmeans + DPFMPS-2PEst`" experiments, each data party uses non-private central k-means to generate local centers and spends ϵ_2 on the intersection estimation algorithm. For "`DPLSF + non-priv intersection`" experiments, each data party spends ϵ_1 privacy budget on DPLSF to generate local centers, and the intersection cardinalities are computed exactly without enforcing any privacy.

As shown Figure 5.7, enforcing the LocCluster private but using the non-private intersection cardinality estimation gives the cost closer to the end-to-end private ones when $S = 2$. However, when $S = 4$, making either component non-private while enforcing the other private has a similar effect on the final loss. These results show that when S is small, the private LocCluster (related to the first term of additive error λ of Theorem 5.3.4) is the component that introduces the majority error. However, they also support our analysis in Section 5.3.4 that the intersection component will introduce a larger error when the number of parties S increases.

Impact of different local k'. As we discuss in Section 5.4.2, in order to trade-off between the error introduce in the membership intersection cardinality estimation with the information loss of local clustering, we can vary the number of local clusters, namely k'. In Figure 5.8, we compare the impact of different numbers of local clusters on the final cost. To give a summary, if ϵ is large, larger k' can benefit the final result by keeping more local information; if S is large, then it would be better to choose a smaller k' to limit the exponentially grown number of grid nodes. We show the automatic decided k' in Table 5.1, which is close to empirical optimal k'.

5.6 Related work

To our knowledge, this is the first work that targeting at solving the clustering problem under vertical federated learning with end-to-end differential privacy guarantee. We summarize the related work from the following three axes.

Vertical federated learning. VFL has been studied in the recent years, sometimes under the name of vertical distributed learning. The most relevant paper is [119], which

proposes a solution for the clustering problem but does not consider the privacy leakage issue. Existing work about other problems in the VFL setting includes learning tree models with secure multiparty computation techniques [74, 99] and training composed models [100]. Some recent papers [101, 152] apply the ADMM framework in the VFL setting. Another paper [102] discussing asynchronous supervised learning with VFL assumes the labels are public accessible.

Data sketches with privacy and private set intersection cardinality estimation. It is originally claimed in [153] that cardinality estimators, including FM sketch as its variants, do not preserve privacy, and any accurate sketch leaks the membership of a user in a set. However, their result assumes that the adversary knows the hash key, which is not a very common DP security setting. Because DP security is based on the assumption that the adversary does not know the random seed; otherwise the adversary can re-generate the random result (i.e., Laplace noise) by itself and recover the true result. Recently, several papers [125, 126, 142, 143] reveal that hash-based, order-invariant sketches satisfy differential privacy as long as the cardinality set is large enough. An earlier paper [154] uses linear sketch and perturbs the sketch with random response, but it has larger relative error [125] and assumes no duplicate.

While the private set intersection cardinality (PSI-CA) problem is an traditional cryptographic problem and has been studies in series of literature [155–160], there is few paper discuss how to solve it under DP. To solve the PSI-CA problem with DP guarantee, the existing solutions also rely on some kinds of data sketch. For example, [161] proposes a solution to support set union or intersection with an untrusted third party based on Bloom filters. However, their solution uses bit-flipping, which introduces large randomness and unable to scale to the setting with more than 2 parties. Other similar work includes using cryptographic tools to encrypt the sketches as [162]. Some other cryptography oriented research [163, 164] use DP definition as a relaxation of the traditional security definition, and develop secure private set intersection protocol resisting malicious adversary. However, their

protocol is only designed for two-party case and can not be easily extended to support more parties.

Differential private k-means. In the central DP setting, some earlier papers includes [128–135] contribute to the theoretical bound for the k-means cost. A practical implementation [165] with cost bounded is based on privately selecting a candidate center set and gradually swapping in better centers into the final k centers. Another open-source implementation of the DP k-means implementation is [139] which is based on locality sensitive hashing. Our algorithm is based on [139] because the authors shows it outperforms [165] on some datasets.

5.7 Conclusion

In this work, we propose the first differentially private solutions for the vertical federated clustering problem. We demonstrate that our solution can outperform other baselines while having strong privacy protection and little communication and computation overhead. The DP intersection cardinality estimation method proposed in this work can be potentially extended and customized to other VFL problems, which can be one of the future research directions.

6. SUMMARY

In this dissertation, we investigate and propose solutions for several problems in distributed settings with differential privacy guarantees.

Firstly, we focus on task estimating numerical value distribution while satisfying local differential privacy. We improve the existing hierarchy histogram data structure by enforcing valid distribution on the estimates and solving it with an efficient ADMM algorithm. We further propose a tailored randomization algorithm for the LDP numerical frequency estimation, the square wave algorithm. We couple the randomization algorithm with a post-processing algorithm based on the EM algorithm with a smoothing step to enable the aggregator to derive the numerical distribution with high accuracy from the noisy reports. Our experimental results show that our proposed methods outperform the existing solution based on categorical frequency oracle with different metrics.

Secondly, we investigate the matrix factorization on sparse rating matrices in all three most popular federated learning settings: VFL, HFL and LFL. We propose a common computation paradigm for the matrix factorization problem in these settings and show the convergence rate with non-private SGD. Based on this, we show how to adapt the common computation paradigm with additional operations, such as embedding clipping, so that it can satisfy different privacy requirements in the three settings.

Finally, our contribution falls in providing a practical solution for the k-means clustering problem in the vertical federated learning setting while satisfying DP. Our main contributions to this problem are the intersection cardinality encoding and estimation algorithms based on FM sketch. We provide theoretical and empirical evidence to show that our methods outperform other baselines.

Base on the explorations in this dissertation, we believe that there are still many interesting open problems in the distributed setting with requirements on privacy protection. For example, generating synthetic data can be a challenging but meaningful problem in the VFL setting; solving the matrix factorization problem with continuous updating ratings is an attractive task for the industry.

REFERENCES

[1] D. Pujol, R. McKenna, S. Kuppam, M. Hay, A. Machanavajjhala, and G. Miklau, "Fair decision making using privacy-protected data," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020, pp. 189–199.

[2] WeBank, *Webank use case*, https://www.fedai.org/cases/a-case-of-traffic-violations-insurance-using-federated-learning/, WeBank, 2022. [Online]. Available: https://www.fedai.org/cases/a-case-of-traffic-violations-insurance-using-federated-learning/.

[3] C. Dwork, A. Smith, T. Steinke, and J. Ullman, "Exposed! a survey of attacks on private data," *Annual Review of Statistics and Its Application*, vol. 4, pp. 61–84, 2017.

[4] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, IEEE, 2008, pp. 111–125.

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

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

[7] J. Li, N. Li, and B. Ribeiro, "Membership inference attacks and defenses in classification models," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 2021, pp. 5–16.

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

[9] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 253–261.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*, S. Halevi and T. Rabin, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284.

[11] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14, Scottsdale, Arizona, USA: Association for Computing Machinery, 2014, pp. 1054–1067, ISBN: 9781450329576. DOI: 10.1145/2660267.2660348. [Online]. Available: https://doi.org/10.1145/2660267.2660348.

[12] A. D. P. Team, *Differential privacy overview*, 2022.

[13] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *NIPS*, 2017.

[14] R. Rogers *et al.*, "Linkedin's audience engagements api: A privacy preserving data analytics system at scale," *arXiv preprint arXiv:2002.05839*, 2020.

[15] N. Li, W. Qardaji, and D. Su, "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*, New York, NY, USA: Association for Computing Machinery, 2012, pp. 32–33.

[16] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*, 2006, pp. 265–284.

[17] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, New York, NY, USA: Association for Computing Machinery, 2009, pp. 19–30.

[18] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, Vienna, Austria: Association for Computing Machinery, 2016, pp. 308–318, ISBN: 9781450341394. DOI: 10.1145/2976749.2978318. [Online]. Available: https://doi.org/10.1145/2976749.2978318.

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

[20] I. Mironov, "Rényi differential privacy," *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 263–275, 2017.

[21] Z. Li, T. Wang, M. Lopuhaä-Zwakenberg, N. Li, and B. koric, "Estimating numerical distributions under local differential privacy," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 621–635.

[22] Z. Li, T. Wang, M. Lopuhaä-Zwakenberg, B. Skoric, and N. Li, "Estimating numerical distributions under local differential privacy," *arXiv preprint arXiv:1912.01051*, 2019.

[23] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: randomized aggregatable privacy-preserving ordinal response," in *CCS*, 2014.

[24] R. Bassily and A. D. Smith, "Local, private, efficient protocols for succinct histograms," in *STOC*, 2015.

[25] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *USENIX Security*, 2017.

[26] J. Acharya, Z. Sun, and H. Zhang, "Hadamard response: Estimating distributions privately, efficiently, and with little communication," *arXiv preprint arXiv:1802.04705*, 2018.

[27] M. Ye and A. Barg, "Optimal schemes for discrete distribution estimation under locally differential privacy," *IEEE Transactions on Information Theory*, 2018.

[28] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially private histograms through consistency," *PVLDB*, vol. 3, no. 1, 2010.

[29] W. H. Qardaji, W. Yang, and N. Li, "Understanding hierarchical methods for differentially private histograms," *PVLDB*, vol. 6, no. 14, 2013.

[30] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," in *ICDE*, 2010.

[31] T. Kulkarni, G. Cormode, and D. Srivastava, "Answering range queries under local differential privacy," *PVLDB*, 2019.

[32] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trendső in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[33] D. Nychka, "Some properties of adding a smoothing step to the em algorithm," *Statistics & probability letters*, vol. 9, no. 2, pp. 187–193, 1990.

[34] B. Silverman, M. Jones, J. Wilson, and D. Nychka, "A smoothed em approach to indirect estimation problems, with particular reference to stereology and emission tomography," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 52, no. 2, pp. 271–303, 1990.

[35] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, 1965.

[36] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *SP*, 2018.

[37] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Minimax optimal procedures for locally private estimation," *Journal of the American Statistical Association*, vol. 113, no. 521, pp. 182–201, 2018.

[38] N. Wang *et al.*, "Collecting and analyzing multidimensional data with local differential privacy," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, IEEE, 2019, pp. 638–649.

[39] N. Wang *et al.*, "Privtrie: Effective frequent term discovery under local differential privacy," in *ICDE*, 2018.

[40] T. Wang *et al.*, "Answering multi-dimensional analytical queries under local differential privacy," in *Proceedings of the 2019 International Conference on Management of Data*, ACM, 2019, pp. 159–176.

[41] T. Wang, Z. Li, N. Li, M. Lopuhaä-Zwakenberg, and B. Skoric, "Locally differentially private frequency estimation with consistency," in *NDSS*, 2020.

[42] J. Lee, Y. Wang, and D. Kifer, "Maximum likelihood postprocessing for differential privacy under consistency constraints," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2015, pp. 635–644.

[43] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, 2014, ISSN: 1551-305X. DOI: 10.1561/0400000042. [Online]. Available: http://dx.doi.org/10.1561/0400000042.

[44] G. C. Fanti, V. Pihur, and Ú. Erlingsson, "Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries," *PoPETs*, vol. 2016, no. 3, 2016.

[45] X. Ren *et al.*, "Lopub: High-dimensional crowdsourced data publication with local differential privacy," *IEEE Trans. Information Forensics and Security*, vol. 13, no. 9, 2018.

[46] J. A. Bilmes *et al.*, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *International Computer Science Institute*, vol. 4, no. 510, p. 126, 1998.

[47] D. Ormoneit and V. Tresp, "Averaging, maximum penalized likelihood and bayesian estimation for improving gaussian mixture probability density estimates," *IEEE Transactions on Neural Networks*, vol. 9, no. 4, pp. 639–650, 1998.

[48] C.-P. S. Fan, J. Stafford, and P. E. Brown, "Local-em and the ems algorithm," *Journal of Computational and Graphical Statistics*, vol. 20, no. 3, pp. 750–766, 2011.

[49] Q. J. Huys and L. Paninski, "Smoothing of, and parameter estimation from, noisy biophysical recordings," *PLoS computational biology*, vol. 5, no. 5, e1000379, 2009.

[50] N. Taxi and L. Commission, *Tlc trip record data*, https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, 2018.

[51] S. Ruggles *et al.*, *Integrated public use microdata series: Version 9.0 [database]*, 2019.

[52] S. F. C. Office, *Sf employee compensation*. [Online]. Available: %5Curl%7Bhttps://www.kaggle.com/san-francisco/sf-employee-compensation#employee-compensation.csv%7D.

[53] R. Bassily, K. Nissim, U. Stemmer, and A. G. Thakurta, "Practical locally private heavy hitters," in *NIPS*, 2017.

[54] S. Wang, Y. Nie, P. Wang, H. Xu, W. Yang, and L. Huang, "Local private ordinal data distribution estimation," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, IEEE, 2017, pp. 1–9.

[55] R. McKenna, D. Sheldon, and G. Miklau, "Graphical-model based estimation and inference for differential privacy," *ICML*, 2019.

[56] J. Jia and N. Z. Gong, "Calibrate: Frequency estimation and heavy hitter identification with local differential privacy via incorporating prior knowledge," *INFOCOM*, 2019.

[57] R. Bassily, "Linear queries estimation with local differential privacy," in *AISTATS*, 2019.

[58] P. Kairouz, K. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in *ICML*, 2016.

[59] B. Balle, J. Bell, A. Gascón, and K. Nissim, "The privacy blanket of the shuffle model," in *Annual International Cryptology Conference*, Springer, 2019, pp. 638–667.

[60] A. Cheu, A. D. Smith, J. Ullman, D. Zeber, and M. Zhilyaev, "Distributed differential privacy via shuffling," in *EUROCRYPT*, 2019, pp. 375–403.

[61] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," *arXiv preprint arXiv:1811.12469*, 2018.

[62] Z. Li, B. Ding, C. Zhang, N. Li, and J. Zhou, "Federated matrix factorization with privacy guarantee," *Proceedings of the VLDB Endowment*, vol. 15, no. 4, pp. 900–913, 2021.

[63] P. Kairouz *et al.*, *Advances and open problems in federated learning*, 2021. arXiv: 1912.04977 [cs.LG].

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

[65] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11, San Diego, California, USA: Association for Computing Machinery, 2011, pp. 69–77, ISBN: 9781450308137. DOI: 10.1145/2020408.2020426. [Online]. Available: https://doi.org/10.1145/2020408.2020426.

[66] S. Zhang, W. Wang, J. Ford, and F. Makedon, "Learning from incomplete ratings using non-negative matrix factorization," *Proceedings of the 2006 SIAM international conference on data mining*, vol. 1, pp. 549–553, 2006.

[67] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2012.

[68] O. Barkan and N. Koenigstein, "Item2vec: Neural item embedding for collaborative filtering," in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, New York, NY, USA: IEEE, 2016, pp. 1–6. DOI: 10.1109/MLSP.2016.7738886.

[69] H. Zhao, Z. Ding, and Y. Fu, "Multi-view clustering via deep matrix factorization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10867.

[70] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[71] A. Narayanan and V. Shmatikov, *How to break anonymity of the netflix prize dataset*, 2006.

[72] L. Zhu and S. Han, "Deep leakage from gradients," *Federated Learning*, vol. 12500, pp. 17–31, 2020.

[73] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 1175–1191, ISBN: 9781450349468. DOI: 10.1145/3133956.3133982. [Online]. Available: https://doi.org/10.1145/3133956.3133982.

[74] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *Proceedings of the VLDB Endowment*, vol. 13, no. 11, pp. 2090–2103, 2020.

[75] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '03, San Diego, California: Association for Computing Machinery, 2003, pp. 202–210, ISBN: 1581136706. DOI: 10.1145/773153.773173. [Online]. Available: https://doi.org/10.1145/773153.773173.

[76] Z. Liu, Y.-X. Wang, and A. Smola, "Fast differentially private matrix factorization," in *Proceedings of the 9th ACM Conference on Recommender Systems*, ser. RecSys '15, Vienna, Austria: Association for Computing Machinery, 2015, pp. 171–178, ISBN: 9781450336925. DOI: 10.1145/2792838.2800191. [Online]. Available: https://doi.org/10.1145/2792838.2800191.

[77] J. Hua, C. Xia, and S. Zhong, "Differentially private matrix factorization," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15, Buenos Aires, Argentina: AAAI Press, 2015, pp. 1763–1770.

[78] H. Shin, S. Kim, J. Shin, and X. Xiao, "Privacy enhanced matrix factorization for recommendation with local differential privacy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1770–1782, 2018.

[79] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov, "Differential privacy has disparate impact on model accuracy," *Advances in Neural Information Processing Systems*, vol. 32, pp. 15 479–15 488, 2019.

[80] X. Chen, S. Z. Wu, and M. Hong, "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, Red Hook, NY, USA: Curran Associates, Inc., 2020, pp. 13 773–13 782. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/9ecff5455677b38d19f49ce658ef0608-Paper.pdf.

[81] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *2013 IEEE Global Conference on Signal and Information Processing*, New York, NY, USA: IEEE, 2013, pp. 245–248. DOI: 10.1109/GlobalSIP.2013.6736861.

[82] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, New York, NY, USA: IEEE, 2019, pp. 691–706. DOI: 10.1109/SP.2019.00029.

[83] P. Jain, O. D. Thakkar, and A. Thakurta, "Differentially private matrix completion revisited," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, USA: PMLR, Jul. 2018, pp. 2215–2224. [Online]. Available: https://proceedings.mlr.press/v80/jain18b.html.

[84] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *International Conference on Learning Representations*, OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=BJ0hF1Z0b.

[85] N. Wu, F. Farokhi, D. Smith, and M. A. Kaafar, "The value of collaboration in convex machine learning with differential privacy," in *2020 IEEE Symposium on Security and Privacy (SP)*, New York, NY, USA: IEEE, 2020, pp. 304–317. DOI: 10.1109/SP40000.2020.00025.

[86] F. Zhou and G. Cong, "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*, pp. 3219–3227, 2018.

[87] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2012.

[88] C. C. Aggarwal *et al.*, *Recommender systems*. New York, USA: Springer, 2016, vol. 1.

[89] A. D. P. Team. "Differential privacy overview - apple," Apple. inc. (2021), [Online]. Available: https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf.

[90] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, Dec. 2015, ISSN: 2160-6455. DOI: 10.1145/2827872. [Online]. Available: https://doi.org/10.1145/2827872.

[91] L. Brozovsky and V. Petricek, *Recommender system for online dating service*, 2007.

[92] A. Gupta, K. Ligett, F. McSherry, A. Roth, and K. Talwar, "Differentially private combinatorial optimization," in *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Philadelphia, PA, USA: SIAM, 2010, pp. 1106–1125. DOI: 10.1137/1.9781611973075.90. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611973075.90. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611973075.90.

[93] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly)logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20, Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1253–1269, ISBN: 9781450370899. DOI: 10.1145/3372297.3417885. [Online]. Available: https://doi.org/10.1145/3372297.3417885.

[94] V. Feldman, A. McMillan, and K. Talwar, *Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling*, 2021. arXiv: 2012.12803 [cs.LG].

[95] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, A. Singh and J. Zhu, Eds., ser. Proceedings of Machine Learning Research, vol. 54, USA: PMLR, Apr. 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html.

[96] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[97] J. Konený, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, *Federated learning: Strategies for improving communication efficiency*, 2017. arXiv: 1610.05492 [cs.LG].

[98] J. Konený, H. B. McMahan, D. Ramage, and P. Richtárik, *Federated optimization: Distributed machine learning for on-device intelligence*, 2016. arXiv: 1610.02527 [cs.LG].

[99] Y. Liu *et al.*, "Federated forest," *IEEE Transactions on Big Data*, no. 01, pp. 1–1, 2020. DOI: 10.1109/TBDATA.2020.2992755.

[100] Y. Hu, D. Niu, J. Yang, and S. Zhou, "Fdml: A collaborative machine learning framework for distributed features," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 2232–2240, ISBN: 9781450362016. DOI: 10.1145/3292500.3330765. [Online]. Available: https://doi.org/10.1145/3292500.3330765.

[101] Y. Hu, P. Liu, L. Kong, and D. Niu, *Learning privately over distributed features: An admm sharing approach*, 2019. arXiv: 1907.07735 [cs.LG].

[102] T. Chen, X. Jin, Y. Sun, and W. Yin, *Vafl: A method of vertical asynchronous federated learning*, 2020. arXiv: 2007.06081 [cs.LG].

[103] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 3574–3583, ISBN: 9781510860964.

[104] A. Girgis, D. Data, S. Diggavi, P. Kairouz, and A. Theertha Suresh, "Shuffled model of differential privacy in federated learning," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, A. Banerjee and K. Fukumizu, Eds., ser. Proceedings of Machine Learning Research, vol. 130, USA: PMLR, Apr. 2021, pp. 2521–2529. [Online]. Available: https://proceedings.mlr.press/v130/girgis21a.html.

[105] Y. Shen and H. Jin, "Privacy-preserving personalized recommendation: An instance-based approach via differential privacy," in *2014 IEEE International Conference on Data Mining*, New York, NY, USA: IEEE, 2014, pp. 540–549.

[106] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *IEEE Intelligent Systems*, vol. 36, no. 5, pp. 11–20, 2021. DOI: 10.1109/MIS.2020.3014880.

[107] A. Flanagan, W. Oyomno, A. Grigorievskiy, K. E. Tan, S. A. Khan, and M. Ammad-Ud-Din, "Federated multi-view matrix factorization for personalized recommendations," in *Machine Learning and Knowledge Discovery in Databases*, Cham: Springer International Publishing, 2021, pp. 324–347, ISBN: 9783030676612. DOI: 10.1007/978-3-030-67661-2_20. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-67661-2_20.

[108] Y. Kim, J. Sun, H. Yu, and X. Jiang, "Federated tensor factorization for computational phenotyping," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: Association for Computing Machinery, 2017, pp. 887–895.

[109] J. Ma, Q. Zhang, J. Lou, J. C. Ho, L. Xiong, and X. Jiang, "Privacy-preserving tensor factorization for collaborative health data analysis," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ser. CIKM '19, Beijing, China: Association for Computing Machinery, 2019, pp. 1291–1300, ISBN: 9781450369763. DOI: 10.1145/3357384.3357878. [Online]. Available: https://doi.org/10.1145/3357384.3357878.

[110] Z. Li, T. Wang, and N. Li, "Differentially private vertical federated clustering," *arXiv preprint arXiv:2208.01700*, 2022.

[111] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 206–215.

[112] J. Vaidya and C. Clifton, "Privacy-preserving decision trees over vertically partitioned data," in *IFIP Annual Conference on Data and Applications Security and Privacy*, Springer, 2005, pp. 139–152.

[113] H. Yunhong, F. Liang, and H. Guoping, "Privacy-preserving svm classification on vertically partitioned data without secure multi-party computation," in *2009 fifth international conference on natural computation*, IEEE, vol. 1, 2009, pp. 543–546.

[114] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[115] B. Gu, Z. Dang, X. Li, and H. Huang, "Federated doubly stochastic kernel learning for vertically partitioned data," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2483–2493.

[116] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *CRYPTO*, 2004, pp. 528–544.

[117] Z. Li, B. Ding, C. Zhang, N. Li, and J. Zhou, "Federated matrix factorization with privacy guarantee," *Proc. VLDB Endow.*, vol. 15, no. 4, pp. 900–913, Dec. 2021, ISSN: 2150-8097. DOI: 10.14778/3503585.3503598. [Online]. Available: https://doi.org/10.14778/3503585.3503598.

[118] C. Wang, J. Liang, M. Huang, B. Bai, K. Bai, and H. Li, "Hybrid differentially private federated learning on vertically partitioned data," *arXiv preprint arXiv:2009.02763*, 2020.

[119] H. Ding, Y. Liu, L. Huang, and J. Li, "K-means clustering with distributed dimensions," in *International Conference on Machine Learning*, PMLR, 2016, pp. 1339–1348.

[120] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.

[121] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "Np-hardness of euclidean sum-of-squares clustering," *Machine learning*, vol. 75, no. 2, pp. 245–248, 2009.

[122] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[123] S. Ahmadian, A. Norouzi-Fard, O. Svensson, and J. Ward, "Better guarantees for k-means and euclidean k-median by primal-dual algorithms," *SIAM Journal on Computing*, vol. 49, no. 4, FOCS17–97, 2019.

[124] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trendső in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[125] A. Smith, S. Song, and A. Thakurta, "The flajolet-martin sketch itself preserves differential privacy: Private counting with minimal space," *Advances in Neural Information Processing Systems 33 pre-proceedings (NeurIPS 2020)*, 2020.

[126] C. Hu *et al.*, "How to make private distributed cardinality estimation practical, and get differential privacy for free," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 965–982.

[127] S. Har-Peled and S. Mazumdar, "On coresets for k-means and k-median clustering," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 2004, pp. 291–300.

[128] U. Stemmer and H. Kaplan, "Differentially private k-means with constant multiplicative error," in *NeurIPS*, 2018.

[129] K. Nissim and U. Stemmer, "Clustering algorithms for the centralized and local models," in *Algorithmic Learning Theory*, PMLR, 2018, pp. 619–653.

[130] Z. Huang and J. Liu, "Optimal differentially private algorithms for k-means clustering," in *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2018, pp. 395–408.

[131] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: The SuLQ framework," in *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2005, pp. 128–138.

[132] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth sensitivity and sampling in private data analysis," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, 2007, pp. 75–84.

[133] D. Feldman, A. Fiat, H. Kaplan, and K. Nissim, "Private coresets," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 361–370.

[134] Y. Wang, Y.-X. Wang, and A. Singh, "Differentially private subspace clustering," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[135] K. Nissim, U. Stemmer, and S. Vadhan, "Locating a small cluster privately," in *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2016, pp. 413–427.

[136] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin, "Differentially private k-means clustering," in *Proceedings of the sixth ACM conference on data and application security and privacy*, 2016, pp. 26–37.

[137] B. Ghazi, R. Kumar, and P. Manurangsi, "Differentially private clustering: Tight approximation ratios," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[138] A. Chang, B. Ghazi, R. Kumar, and P. Manurangsi, "Locally private k-means in one round," *arXiv preprint arXiv:2104.09734*, 2021.

[139] Google, *Differentially private k-means clustering (experimental)*, https://github.com/google/differential-privacy/tree/main/learning/clustering, Google LLC, 2022. [Online]. Available: https://github.com/google/differential-privacy/tree/main/learning/clustering.

[140] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017.*, 2017, pp. 729–745. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tianhao.

[141] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *Journal of computer and system sciences*, vol. 31, no. 2, pp. 182–209, 1985.

[142] S. G. Choi, D. Dachman-soled, M. Kulkarni, and A. Yerukhimovich, "Differentially-private multi-party sketching for large-scale statistics," *Proceedings on Privacy Enhancing Technologies*, vol. 3, pp. 153–174, 2020.

[143] C. Dickens, J. Thaler, and D. Ting, *(nearly) all cardinality estimators are differentially private*, 2022. DOI: 10.48550/ARXIV.2203.15400. [Online]. Available: https://arxiv.org/abs/2203.15400.

[144] K. J. Lang, "Back to the future: An even more nearly optimal cardinality estimation algorithm," *arXiv preprint arXiv:1708.06839*, 2017.

[145] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm," in *Discrete Mathematics and Theoretical Computer Science*, Discrete Mathematics and Theoretical Computer Science, 2007, pp. 137–156.

[146] *New york city taxi trip*, https://www.kaggle.com/c/nyc-taxi-trip-duration, NYC.gov, 2016. [Online]. Available: https://www.kaggle.com/c/nyc-taxi-trip-duration.

[147] *Home credit default risk*, https://www.kaggle.com/competitions/home-credit-default-risk/overview, Home Credit, 2018. [Online]. Available: https://www.kaggle.com/competitions/home-credit-default-risk/overview.

[148] D. J. Slate, *Letter recognition data set*, https://archive.ics.uci.edu/ml/datasets/letter+recognition, Odesta Corporation, 1991. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/letter+recognition.

[149] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[150] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007, pp. 410–420.

[151] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[152] C. Xie, P.-Y. Chen, C. Zhang, and B. Li, "Improving privacy-preserving vertical federated learning by efficient communication with admm," *arXiv preprint arXiv:2207.10226*, 2022.

[153] D. Desfontaines, A. Lochbihler, and D. Basin, "Cardinality estimators do not preserve privacy," *Proceedings on Privacy Enhancing Technologies*, vol. 2, pp. 26–46, 2019.

[154] R. Pagh and N. M. Stausholm, "Efficient differentially private F0 linear sketching," in *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, ser. LIPIcs, vol. 186, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 18:1–18:19. DOI: 10.4230/LIPIcs.ICDT.2021.18. [Online]. Available: https://doi.org/10.4230/LIPIcs.ICDT.2021.18.

[155] E. D. Cristofaro, P. Gasti, and G. Tsudik, "Fast and private computation of cardinality of set intersection and union," in *International Conference on Cryptology and Network Security*, Springer, 2012, pp. 218–231.

[156] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *International conference on the theory and applications of cryptographic techniques*, Springer, 2004, pp. 1–19.

[157] L. Kissner and D. Song, "Privacy-preserving set operations," in *Annual International Cryptology Conference*, Springer, 2005, pp. 241–257.

[158] C. Hazay and Y. Lindell, "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries," in *Theory of Cryptography Conference*, Springer, 2008, pp. 155–175.

[159] S. Jarecki and X. Liu, "Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection," in *Theory of Cryptography Conference*, Springer, 2009, pp. 577–594.

[160] C. Hazay and K. Nissim, "Efficient set operations in the presence of malicious adversaries," in *International Workshop on Public Key Cryptography*, Springer, 2010, pp. 312–331.

[161] R. Stanojevic, M. Nabeel, and T. Yu, "Distributed cardinality estimation of set operations with differential privacy," in *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, IEEE, 2017, pp. 37–48.

[162] B. Kreuter, C. W. Wright, E. S. Skvortsov, R. Mirisola, and Y. Wang, "Privacy-preserving secure cardinality and frequency estimation," 2020.

[163] A. Groce, P. Rindal, and M. Rosulek, "Cheaper private set intersection via differentially private leakage," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, 2019.

[164] B. Kacsmar *et al.*, "Differentially private two-party set operations," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2020, pp. 390–404.

[165] M.-F. Balcan, T. Dick, Y. Liang, W. Mou, and H. Zhang, "Differentially private clustering in high-dimensional euclidean spaces," in *International Conference on Machine Learning*, PMLR, 2017, pp. 322–331.

VITA

Zitao Li was born in Guangzhou, China. He was admitted to a 2+2 collage program held jointly by the Sun Yat-sen University (SYSU) and the Chinese University of Hong Kong (CUHK). He graduated with a Bachelor of Science in computer science in 2016 from CUHK and entered Purdue University in the fall of the same year. Zitao's research was in the area of differential privacy for data processing, under the supervision of Dr. Ninghui Li.