DECENTRALIZED DIFFERENTIALLY PRIVATE ALGORITHMS FOR MATRIX AND TENSOR FACTORIZATION

By

HAFIZ IMTIAZ

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Anand D. Sarwate

And approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2020

ABSTRACT OF THE DISSERTATION

Decentralized Differentially Private Algorithms for Matrix and Tensor Factorization

by Hafiz Imtiaz

Dissertation Director:
Anand D. Sarwate

Many applications of machine learning, such as human health research, involve processing private or sensitive information. Privacy concerns may impose significant hurdles to collaboration in scenarios where there are multiple sites holding data and the goal is to estimate properties jointly across all datasets. Conventional differentially private decentralized algorithms can provide strong privacy guarantees. However, the utility/accuracy of the joint estimates may be poor when the datasets at each site are small. In this work, we propose a new framework, Correlation Assisted Private Estimation (CAPE), for designing privacy-preserving decentralized algorithms with much better accuracy guarantees in an honest-but-curious model. We show that CAPE can be employed in a range of decentralized computations common in machine learning problems.

We note that matrix and tensor factorizations are key components of many decentralized processing pipelines that involve joint subspace learning. In this work, we focus on principal component analysis, independent component analysis, canonical correlation analysis and orthogonal tensor decomposition. Conventional decentralized

differentially private factorization schemes suffer from excessive noise, which leads to sub-optimal subspace/feature learning. We demonstrate that the CAPE framework fits perfectly in these problems and can be employed to remedy the excessive noise issue. More specifically, we develop decentralized algorithms for these matrix and tensor factorization problems and show that, under certain conditions, these algorithms can achieve the same utility as a centralized algorithm using all datasets across sites.

Finally, we employ our CAPE framework to propose an algorithm for solving generalized optimization problems in decentralized settings. We provide a tighter characterization of the functional mechanism and propose modifications such that it can be incorporated in the CAPE framework. Our proposed decentralized functional mechanism is specifically suited for privacy-preserving computation of virtually any differentialble and continuous cost function in the decentralized setting.

For all of our proposed algorithms, we present empirical results to demonstrate that our proposed algorithms outperform existing state-of-the-art algorithms and can be competitive with non-private algorithms in many scenarios of interest. Our results indicate that meaningful privacy can be attained without losing much performance by the virtue of better algorithm design.

Acknowledgements

I am immensely grateful to my adviser, Prof. Anand D. Sarwate, for his guidance, support and encouragement throughout my journey as a graduate student at Rutgers University. He continually trained me to approach and solve research problems through perceptive discussions. He is a very knowledgeable and empathetic mentor. In several occasions he went out of his way to help me overcome my doubts and taught me to be persistent in research. I would like to thank him and the Department of Electrical and Computer Engineering for the financial support during my time at Rutgers University.

I am grateful to my dissertation committee members, Prof. Kristin Dana, Prof. Emina Soljanin, Prof. Roy Yates and Prof. Vince Calhoun for their valuable insights and suggestions on my dissertation. Professors Dana, Soljanin and Yates have been providing me with guidance and inspiration from the early stages of my PhD. Prof. Calhoun has been a mentor and collaborator in all of my projects since 2015. Their passion and deep knowledge of research problems has motivated me and helped me grow as a researcher. I am grateful to Prof. Athina Petropulu, whom I deeply admire. She has been very helpful to me and my wife in many occasions. I would also like to thank Dr. Jafar Mohammadi and Mohsen Ghassemi for numerous helpful discussions.

Completing my PhD could not be possible without the support of my parents and my sister. I can not thank them enough for their immense love and support. My parents have made countless sacrifices for our betterment and I owe them everything I have. Finally, I would like to thank my wife Tahsina and our son Tashreeq. My wife is my closest friend for more than fourteen years and has constantly supported me through all of my ups-and-downs. She has sacrificed a lot for me and our son and I am truly grateful to her. Our son is our companion in every adventure, literally and figuratively. I believe he has made me more responsible and a better person.

Dedication

For my parents, my wife and my son.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Privacy-sensitive learning is important in many applications: examples include human health research, business informatics, and location-based services among others. Releasing any function of private data, even summary statistics and other aggregates, can reveal information about the underlying data [109, 133]. Differential privacy (DP) [50] is a cryptographically motivated and mathematically rigorous framework for measuring the risk associated with performing computations on private data. More specifically, it measures the privacy risk in terms of the probability of identifying the presence of individual data points in a dataset from the results of computations performed on that data. Additionally, it can be used to help manage the privacy loss of individual devices. Pilot efforts in this direction have been made by Google [54], Apple [1, 134, 18, 47], Uber [46], and the US Census [100, 3]. For event-based systems, differential privacy has been used on count streams [90].

It is evident that differential privacy has emerged as a de-facto standard for privacy-preserving technologies in research and practice. It is also useful when the private data is distributed over multiple locations (sites) and each site has its own dataset of human subjects [135, 119, 35, 59]. Data holders may be reluctant or unable to directly share "raw" data to an aggregator due to ethical (e.g. privacy) and technical (e.g. bandwidth) reasons. Some noteworthy examples include:

- medical research consortium of healthcare centers/research labs for fMRI analysis
- decentralized speech processing system to learn model parameters for speaker recognition

- multi-party cyber-physical system for performing global state estimation from sensor signals.

From a statistical standpoint, the number of samples held locally in such applications is usually not large enough for meaningful parameter estimation. Decentralized information processing allows data owners to maintain local control of the data while passing messages to assist in a joint computation across many datasets. If these computations are differentially private, data owners and algorithm developers can measure and control privacy risks.

Differentially private algorithms introduce noise to guarantee privacy. Conventional decentralized differentially private algorithms often have poor utility due to excess noise compared to centralized analyses. The excessive noise problem is more prominent when the local dataset size is small-to-moderate. In this work, we propose a Correlation Assisted Private Estimation (CAPE) framework, which is a novel *decentralized and privacy-preserving* protocol that provides utility close to centralized case. We achieve this by inducing (anti) correlated noise in the differentially private messages. CAPE can be applied to a wide range of computations, including computing loss functions that are separable across sites. This class includes large-scale statistical signal processing and machine learning methods formulated as convex optimization problems, such as empirical risk minimization (ERM).

One of the most common problems in machine learning is that of matrix and tensor factorization in decentralized setting. In this work, we particularly focus on the Singular Value Decomposition (SVD), or Principal Component Analysis (PCA), orthogonal tensor decomposition (OTD), independent component analysis (ICA), and canonical correlation analysis (CCA). Despite some limitations, PCA/SVD is one of the most widely-used preprocessing stages in any machine learning algorithm: it projects data onto a lower dimensional subspace spanned by the singular vectors of the sample second-moment matrix. For example, training a classifier is much faster when the data is first projected onto lower dimensions.

Tensor decomposition is a powerful tool for inference algorithms because it can be used to infer complex dependencies (higher order moments) beyond second-moment

methods such as PCA. This is particularly useful in latent variable models [8] such as mixtures of Gaussians and topic modeling. Tensor decomposition can be considered to be higher-order generalizations of the matrix SVD and PCA. Although computing the decomposition of arbitrary tensors is computationally intractable, efficient algorithms exists for finding decomposition of structured tensors, such as those that appear in several latent variable models [8].

ICA is a very popular blind source separation technique in neuroimaging studies. It assumes that the observed signals are mixtures of statistically independent sources and aims to decompose the mixed signals into those sources. It has been widely used to estimate intrinsic connectivity networks from brain imaging data (e.g. functional magnetic resonance imaging (fMRI)). Successful application of ICA on fMRI can be attributed to both sparsity and statistical independence between the underlying sources [29]. Temporal ICA is shown to be able to identify temporally independent components that represent activation of different neurological regions over time [40].

CCA [71] is a tool for characterizing linear relationships between two (or more) multidimensional variables (or "views"). The views are typically different measurements or modalities of the same physical phenomena. It has been used as a pre-processing step for dimensionality reduction in high-dimensional clustering, statistical analysis, medical studies and recently in machine learning, neuro-science and signal processing [45, 93, 23, 76]. The advantage of CCA over PCA or random projections [44, 139, 4, 85] is that CCA can jointly learn projection maps to improve clustering performance for *multi-view learning* [9, 37]. CCA also has applications in blind source separation, such as in fMRI analysis [94, 45, 42].

Finally, the functional mechanism [143] is an approach for addressing privacy-preserving optimization problems. It uses functional approximation [121] and the Laplace mechanism [48] to create differentially private approximations for any continuous and differentiable function. In theory, this can address any generalized optimization problem that appear commonly in machine learning and signal processing. The advantage of functional mechanism is that any off-the-shelf optimizer can be used to minimize the approximate loss function to find a privacy-preserving optimal parameter.

1.2 Major Contributions

The goal of our work is to develop privacy-preserving matrix and tensor factorization algorithms that operate in decentralized settings – similar to those found in research consortia. As conventional decentralized differentially private schemes suffer from too much noise, we propose to reduce the amount of noise of conventional schemes while guaranteeing differential privacy. We summarize our contributions here:

- We propose a novel decentralized differentially private computation protocol CAPE in Chapter 3. We show that CAPE improves upon the conventional decentralized differentially private schemes and achieves the same level of utility as the pooled data scenario in certain regimes. Additionally, we show that CAPE can be employed in a wide range of computations that frequently appear in machine learning problems. We analyze the privacy guarantee of the CAPE protocol in detail. We note that the privacy analysis is non-trivial as we induce correlated noise for preserving privacy and achieving the same utility level as the pooled-data scenario.

- We employ the CAPE scheme to a number of widely-used matrix and tensor factorization algorithms, namely principal component analysis (Chapter 4), orthogonal tensor decomposition (Chapter 5), independent component analysis (Chapter 6) and canonical correlation analysis (Chapter 7). The proposed decentralized PCA (capePCA) and the decentralized joint ICA (capeDJICA) algorithms are improvements over our previous algorithms [78, 80]. To the best of our knowledge, the proposed decentralized differentially private orthogonal tensor decomposition (capeAGN) and canonical correlaton analysis (capeCCA) algorithms are the first decentralized differentially private algorithms for their respective problems. For all of the proposed algorithms, we show that we can achieve the same utility as the pooled-data scenario in certain regimes.

- We propose an improved functional mechanism (FM) that builds on the work of Zhang et al. [143] (Chapter 8). We use a tighter sensitivity analysis and show analytically that it guarantees less noisy function computation for linear and

logistic regression problems at the expense of an approximate differential privacy guarantee. Empirical validation on real and synthetic data validates our approach.

- We modify the FM such that it can be incorporated into decentralized settings. We employ the CAPE scheme to propose the capeFM algorithm and show that capeFM can achieve the same utility as the pooled data scenario in some regimes. To the best of our knowledge, this work is the first decentralized FM.

- We demonstrate the effectiveness of our algorithms with varying privacy and dataset parameters. Our privacy analysis and empirical results on real and synthetic datasets show that the proposed algorithms can achieve much better utility than the existing state of the art algorithms.

1.3 Organization of the Thesis

The rest of this dissertation is organized as follows. In Chapter 2, we review some of the the relevant works and necessary definitions and notations. In Chapter 3, we describe the proposed CAPE scheme in detail along with proofs and utility analyses. We also formalize the scope of the CAPE scheme. In Chapters 4–7, we show how the CAPE scheme can be employed to develop efficient privacy-preserving decentralized matrix and tensor factorization algorithms, namely the capePCA, the capeAGN, the capeDJICA and the capeCCA algorithms. In Chapter 8, we present a generalized decentralized optimization scheme that can be incorporated in many optimization scenarios that appear in machine learning problems. Our capeFM algorithm employs the CAPE scheme and achieves the same utility as the pooled data scenario. In all these chapters, we provide experimental results that demonstrate the effectiveness of the proposed algorithms on real and synthetic data and compare against competing state-of-the-art algorithms. We focus on investigating the privacy-utility trade-off: how the performance varies as a function of the privacy parameters and the number of samples. In each case, we compare the proposed algorithms with existing (if any) and non-private algorithms and a conventional approach (no correlated noise). Finally, we make some concluding remarks in Chapter 9.

Chapter 2

Background and Preliminaries

In this chapter, we first review some of the the relevant works in matrix and tensor factorization and decentralized optimization problems. We then review some necessary definitions, notations and tensor algebra preliminaries.

2.1 Related Works

There is a vast literature [27, 108, 138, 111, 136, 75, 66, 115, 144] on solving optimization problems in decentralized settings, both with and without privacy concerns. In the machine learning context, the most relevant ones to our current work are those using ERM and stochastic gradient descent (SGD) [36, 38, 127, 2, 82, 92, 16, 97, 140]. Additionally, several works studied decentralized differentially private learning for locally trained classifiers [117, 120, 15]. One of the most common approaches for ensuring differential privacy in optimization problems is to employ randomized gradient computations [127, 16, 2]. Another common approach is to employ the output perturbation [38], which adds noise to the output of the optimization problem according to the sensitivity of the optimization variable. Note that, both of these approaches involve computing the sensitivity (of the gradient or the output variable) and then adding noise scaled to the sensitivity [50]. The problem with output perturbation is that the relation between the data and the parameter set is often hard to characterize. This is due to the complex nature of the optimization: the sensitivity is very difficult to compute. However, differentially private gradient descent methods can circumvent this by bounding the gradients at the expense of slowing down the convergence process [2]. Finally, one can employ objective perturbation [38, 115], where we need to perturb the objective function and find the minimizer of the perturbed objective function. However, the objective

function has to satisfy some strict conditions, which are not met in many practical optimization problems [143]. In addition to optimization problems, Smith [125] proposed a general approach for computing summary statistics using the *sample-and-aggregate* framework and both the Laplace and Exponential mechanisms [105]. Jing [83] proposed a unique approach that uses perturbed histograms for releasing a class of M-estimators in a non-interactive way.

Differentially private algorithms provide different guarantees than Secure Multiparty Computation (SMC) based methods (see [64, 123, 52, 84, 132, 21] for thorough comparisons between SMC and differential privacy based methods). Gade and Vaidya [58] applied a combination of SMC and differential privacy for decentralized optimization in which each site adds and subtracts arbitrary functions to confuse the adversary. Bonawitz et al. [22] proposed a communication-efficient method for *federated learning* over a large number of mobile devices. The most recent work in this line is that of Heikkilä et al. [69], who also studied the relationship of additive noise and sample size in a decentralized setting. In their model, S data holders communicate their data to M computation nodes to compute a function. Anandan and Clifton [7] proposed to *reduce* the noise added for differential privacy.

In the conventional decentralized differentially private computation (i.e., where each data owner randomizes the output function), a much higher level of noise is required compared to that of a centralized (or pooled-data) analysis. Some recent works [14, 19] explore the space in between the decentralized and centralized scenarios. Mechanisms have been proposed that are easy to implement with limited accuracy loss with respect to the centralized model. In this respect, Erlingsson et al. [55] recently showed that shuffling provides a privacy amplification, whereas the Encode, Shuffle, Analyze (ESA) model required a trusted shuffler that received messages from data owners and permuted them before they are released [19]. However, these shuffling mechanisms require all parties to follow the protocol exactly. Our work is inspired by the seminal work of Dwork et al. [49] that proposed decentralized noise generation for preserving privacy.

2.1.1 Principal Component Analysis

Several decentralized PCA algorithms [95, 12, 24, 10, 101, 78] have been proposed. Liang et al. [95] proposed a decentralized PCA scheme where it is necessary to send both the left and right singular vectors along with corresponding singular values from each site to the aggregator. Feldman et al. [57] proposed an improvement upon this, where each site sends a $D \times R$ matrix to the aggregator. Balcan et al. [12] proposed a further improved version using fast sparse subspace embedding [39] and randomized SVD [65]. However, none of these algorithms satisfy any privacy guarantee. Prior to this work, we proposed two differentially-private decentralized PCA algorithms. To our knowledge, these are the only existing differentially-private decentralized PCA algorithms. The method by Imtiaz et al. [80] proposed to send data from one site to another in a sequential manner, which is less fault-tolerant. Additionally, the method by Imtiaz and Sarwate [78] suffers from larger noise variance as it employs the conventional decentralized differentially private scheme.

2.1.2 Orthogonal Tensor Decomposition

For a complete introduction to the history of tensor decompositions, see the comprehensive survey of Kolda and Bader [87]. The CANDECOMP/PARAFAC, or CP decomposition [34, 68] and the Tucker decomposition [137] are generalizations of the matrix SVD to multi-way arrays. While finding the decomposition of arbitrary tensors is computationally intractable, specially structured tensors appear in some latent variable models. Such tensors can be decomposed efficiently [8, 87] using a variety of approaches such as generalizations of the power iteration [89]. Exploiting such structures in higher-order moments to estimate the parameters of latent variable models has been studied extensively using the so-called orthogonal tensor decomposition (OTD) [8, 73, 72, 86]. To our knowledge, these decompositions have not been studied in the setting of distributed data.

This work builds upon our earlier work on decentralized differentially private eigenvector calculations [78] and centralized differentially private OTD [79]. To our knowledge, this is the first efficient and fault-tolerant algorithm for OTD in decentralized settings. Wang and Anandkumar [141] recently proposed an algorithm for differentially private tensor decomposition using a noisy version of the tensor power iteration [8, 89]. Their algorithm adds noise at each step of the iteration and the noise variance grows with the predetermined number of iterations. They also make the restrictive assumption that the input to their algorithm is orthogonally decomposable. Our centralized OTD algorithms [79] avoid these assumptions and achieve better empirical performance (although without sample complexity guarantees).

2.1.3 Decentralized Joint Independent Component Analysis

Several studies have shown that brain activation patterns can be used as biomarkers to detect brain disorders such as Attention-deficit/hyperactivity disorder, Schizophrenia and Alzheimer's disease [30, 31]. Functional magnetic resonance imaging (fMRI) is one of the most commonly employed approaches to obtain the brain activation patterns. As mentioned before, we particularly focus on the decentralized joint ICA (djICA) algorithm, which can perform temporal ICA of fMRI data. The goal of temporal ICA is to identify temporally independent components that represent activation of different neurological regions over time [40]. However, it requires more samples than are typically available from a single study. This is because of computational complexity and statistical sample size – the ratio of spatial to temporal dimensions often requires the aggregate temporal dimension to be similar to the voxel dimension [11]. A number of modified ICA algorithms exist for joining various data sets [131] together and performing simultaneous decomposition of data from a number of subjects and modalities [99]. For instance, group spatial ICA (GICA) is a noteworthy one for multi-subject analysis of task- and resting-state fMRI data [5, 32, 31]. It assumes that the spatial map components are similar across subjects. On the other hand, the joint ICA (jICA) [30] algorithm for multi-modal data fusion assumes that the mixing process is similar over a group of subjects. However, group temporal ICA also assumes common

spatial maps but pursues statistical independence of timecourses. Consequently, like jICA, the common spatial maps from temporal ICA describe a common mixing process among subjects. While very interesting, temporal ICA of fMRI is typically not investigated because of the small number of time points in each data set, which leads to unreliable estimates [11]. The djICA approach overcomes that limitation by leveraging information from datasets of multiple sites.

2.1.4 Canonical Correlation Analysis

CCA operates with multiple views of the same physical phenomena and finds the bases for each view such that the correlation matrix between the data projected onto the bases is diagonal and the correlations on the diagonal are maximized [23]. As mentioned before, it has been used in applications involving high-dimensional clustering, statistical analysis, medical studies and recently in machine learning, neuro-science and signal processing [45, 93, 23, 76]. CCA also has applications in blind source separation, such as in fMRI analysis [94, 45, 42].

2.1.5 General Function Computation

Our application to decentralized differentially private function computation in this work builds on the *functional mechanism* [143], which uses functional approximation [121] and the Laplace mechanism [50] to create differentially private approximations for any continuous and differentiable function. Zhang et al.'s [143] approach can be considered as a more generalized variant of objective perturbation [38] because it can be applied to any continuous and differentiable function. However, the approach results in a very noisy objective function and thus, a poor estimate of the optimization variable. Additionally, it does not scale well to decentralized problems. We provide a better analysis of the sensitivity of their approximation and adapt the approach to the decentralized setting.

2.2 Notation and Definitions

We begin by specifying the notation convention used in the manuscript. We then review the necessary definitions that will be used throughout this dissertation. Finally, we present some preliminaries on tensors and tensor decomposition from Anandkumar [8].

Notation. We denote vectors with bold lower case letters (e.g., $\mathbf{x}$), matrices with bold upper case letters (e.g. $\mathbf{X}$), tensors with calligraphic scripts (e.g. $\mathcal{X}$), scalars with regular letters (e.g., M) and indices with lower case letters (e.g., m). Indices typically run from 1 to their upper-case versions (e.g., $m \in \{1, 2, \ldots, M\} \triangleq [M]$). We denote the n-th column of the matrix $\mathbf{X}$ as $\mathbf{x}_n$. We use $\|\cdot\|_2$, $\|\cdot\|_F$ and $\mathrm{tr}(\cdot)$ for the Euclidean (or $\mathcal{L}_2$) norm of a vector or spectral norm of a matrix, the Frobenius norm, and the trace operation, respectively. We denote the inner-product between two arrays as $\langle \cdot, \cdot \rangle$. For example, if $\mathbf{A}$ and $\mathbf{B}$ are two matrices then $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{tr}\left(\mathbf{A}^\top \mathbf{B}\right)$. Finally, the density of the standard Normal random variable is given by

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right).$$

2.2.1 Privacy Definitions

Definition 1 ((ϵ, δ)-Differential Privacy [50])**.** *An algorithm $\mathcal{A}(\mathbb{D})$ taking values in a set $\mathbb{T}$ provides (ϵ, δ)-differential privacy if*

$$\Pr[\mathcal{A}(\mathbb{D}) \in \mathbb{S}] \leq \exp(\epsilon) \Pr[\mathcal{A}(\mathbb{D}') \in \mathbb{S}] + \delta$$

for all measurable $\mathbb{S} \subseteq \mathbb{T}$ and all data sets $\mathbb{D}$ and $\mathbb{D}'$ differing in a single entry (neighboring datasets).

This definition essentially states that the probability of the output of an algorithm is not changed significantly if the corresponding database input is changed by just one entry. Here, ϵ and δ are privacy parameters, where low ϵ and δ ensure more privacy. The parameter δ can be interpreted as the probability that the algorithm fails to provide

privacy risk ϵ. Several mechanisms can be employed to ensure that an algorithm satisfies differential privacy. Additive noise mechanisms such as the Gaussian or Laplace mechanisms [50, 48] and random sampling using the exponential mechanism [105] are among the most common ones. For additive noise mechanisms, the standard deviation of the noise is scaled to the *sensitivity* of the computation.

Definition 2 ($\mathcal{L}_p$-sensitivity [50])**.** *The $\mathcal{L}_p$-sensitivity of a vector-valued function $f(\mathbb{D})$ is defined as*

$$\Delta := \max_{\mathbb{D},\mathbb{D}'} \|f(\mathbb{D}) - f(\mathbb{D}')\|_p,$$

where $\mathbb{D}$ and $\mathbb{D}'$ are neighboring datasets.

We will focus on $p = 1$ and 2 in this work.

Definition 3 (Gaussian Mechanism [48])**.** *Let $f : \mathbb{D} \mapsto \mathbb{R}^D$ be an arbitrary D dimensional function with $\mathcal{L}_2$-sensitivity Δ. The Gaussian Mechanism with parameter τ adds noise scaled to $\mathcal{N}(0, \tau^2)$ to each of the D components of the output and satisfies (ϵ, δ) differential privacy if*

$$\tau \geq \frac{\Delta}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}}. \tag{2.1}$$

Note that, for any given (ϵ, δ) pair, we can calculate a noise variance τ^2 such that addition of a noise term drawn from $\mathcal{N}(0, \tau^2)$ guarantees (ϵ, δ)-differential privacy. Since there are infinitely many (ϵ, δ) pairs that yield the same τ^2, we parameterize our methods using τ^2 [77] in this work.

Definition 4 (Rényi Differential Privacy [107])**.** *A randomized mechanism $\mathcal{A} : \mathbb{D} \mapsto \mathbb{T}$ is (α, ϵ_r)-Rényi differentially private if, for any adjacent $D, D' \in \mathbb{D}$, the following holds: $D_\alpha\left(\mathcal{A}(D)\|\mathcal{A}(D')\right) \leq \epsilon_r$. Here, $D_\alpha\left(P(x)\|Q(x)\right) = \frac{1}{\alpha-1}\log \mathbb{E}_{x\sim Q}\left(\frac{P(x)}{Q(x)}\right)^\alpha$ and $P(x)$ and $Q(x)$ are probability density functions defined on $\mathbb{T}$.*

It has been shown that conventional privacy analysis of multi-shot algorithms tend to exaggerate the total privacy loss [2, 107]. RDP offers a much simpler composition

rule that is shown to be tight [107]. Therefore, we employ RDP to characterize and analyze our multi-shot decentralized joint independent component analysis algorithm, in addition to conventional privacy analysis.

2.2.2 Tensor Preliminaries

Tensors are multi-dimensional arrays, higher dimensional analogs of matrices. An M-way or M-th order tensor is an element of the tensor product of M vector spaces. **Fibers** are higher order analogs of rows and columns. A fiber is defined by fixing every index but one. An M-way tensor $\mathcal{X} \in \mathbb{R}^{D_1 \times \ldots \times D_M}$ is rank-1 if it can be written as the outer product of M vectors:

$$\mathcal{X} = \mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \ldots \otimes \mathbf{x}_M,$$

where $\mathbf{x}_m \in \mathbb{R}^{D_m}$ and $\otimes$ denotes the outer product. **Matricization** (or *unfolding* or *flattening*) is the process of reordering the elements of an M-way tensor into a matrix. The mode-m matricization of $\mathcal{X} \in \mathbb{R}^{D_1 \times \ldots \times D_M}$ is denoted as $\mathbf{X}_{(m)}$ and is found by arranging the mode-m fibers of $\mathcal{X}$ as the columns of $\mathbf{X}_{(m)}$. A **mode-m product** is multiplying a tensor by a matrix in mode-m. Let $\mathcal{X} \in \mathbb{R}^{D_1 \times \ldots \times D_M}$ and $\mathbf{U} \in \mathbb{R}^{J \times D_m}$ then

$$[\mathcal{X} \times_m \mathbf{U}]_{d_1 \ldots d_{m-1}, j, d_{m+1} \ldots d_M} = \sum_{d_m=1}^{D_m} [\mathcal{X}]_{d_1 \ldots d_M} [\mathbf{U}]_{j, d_m}.$$

We can also represent the mode-m flattened tensor as

$$\mathcal{Y} = \mathcal{X} \times_m \mathbf{U} \iff \mathbf{Y}_{(m)} = \mathbf{U}\mathbf{X}_{(m)}.$$

The **vectorization** of the tensor $\mathcal{X}$ is defined as [124, 116]

$$\mathrm{vec}\mathcal{X} = \sum_{d_1=1}^{D_1} \cdots \sum_{d_M=1}^{D_M} [\mathcal{X}]_{d_1, \ldots, d_M} \mathbf{e}_{d_1}^{D_1} \circ \cdots \circ \mathbf{e}_{d_M}^{D_M},$$

where $\circ$ denotes the Kronecker product [87] and $\mathbf{e}^{D_m}$ denotes the D_m-dimensional elementary (or unit basis) vector. We note that $\text{vec}\mathcal{X}$ is a $(\prod_{m=1}^{M} D_m)$-dimensional vector. A tensor is called **symmetric** if the entries do not change under any permutation of the indices. The **rank** of a tensor $\mathcal{X}$ is the smallest number of rank-1 tensors that sums to the original tensor [86]. The **norm** of a tensor $\mathcal{X} \in \mathbb{R}^{D_1 \times \ldots \times D_M}$ [8] is

$$\|\mathcal{X}\| = \sqrt{\sum_{d_1=1}^{D_1} \cdots \sum_{d_M=1}^{D_M} [\mathcal{X}]^2_{d_1,\ldots,d_M}}.$$

This is equivalent to the matrix Frobenius norm. We note that the norm $\|\mathcal{X}\|$ of a tensor $\mathcal{X}$ is equal to the $\mathcal{L}_2$-norm of the vectorized version of the same tensor, $\text{vec}\mathcal{X}$. That is, $\|\mathcal{X}\| = \|\text{vec}\mathcal{X}\|_2$. We also observe that for a vector $\mathbf{x} \in \mathbb{R}^D$, if the $\mathcal{L}_2$-norm $\|\mathbf{x}\|_2 = 1$ then

$$\|\mathbf{x} \otimes \cdots \otimes \mathbf{x}\| = 1 \text{ because}$$
$$[\mathbf{x} \otimes \cdots \otimes \mathbf{x}]_{d_1,\ldots,d_M} = [\mathbf{x}]_{d_1} \cdots [\mathbf{x}]_{d_M}.$$

The **operator norm** of an M-way symmetric tensor $\mathcal{X} \in \mathbb{R}^{D \times \ldots \times D}$ is defined [8] as

$$\|\mathcal{X}\|_{\text{op}} = \sup_{\|\mathbf{x}\|_2=1} \left|\mathcal{X}\left(\mathbf{x}, \mathbf{x}, \ldots, \mathbf{x}\right)\right|.$$

Finally, a tensor $\mathcal{X} \in \mathbb{R}^{D_1 \times \ldots \times D_M}$ can be considered to be a multi-linear map [8] in the following sense: for a set of matrices $\{\mathbf{V}_m \in \mathbb{R}^{D_m \times K_m} : m = 1, 2, \ldots, M\}$, the $(k_1, k_2, \ldots, k_M)$-th entry in the M-way tensor representation of $\mathcal{Z} = \mathcal{X}\left(\mathbf{V}_1, \ldots, \mathbf{V}_M\right) \in \mathbb{R}^{K_1 \times \ldots \times K_M}$ is

$$[\mathcal{Z}]_{k_1 \ldots k_M} = \sum_{d_1 \ldots d_M} [\mathcal{X}]_{d_1 \ldots d_M} [\mathbf{V}]_{d_1,k_1} \cdots [\mathbf{V}]_{d_M,k_M}.$$

Therefore, we have

$$\mathcal{X}\left(\mathbf{V}_1 \ldots \mathbf{V}_M\right) = \mathcal{X} \times_1 \mathbf{V}_1^\top \cdots \times_M \mathbf{V}_M^\top.$$

2.2.3 Orthogonal Tensor Decomposition

As mentioned before, we consider the decomposition of symmetric tensors in this work. Let $\mathcal{X}$ be an M-way D dimensional symmetric tensor. Given real valued vectors $\mathbf{v}_k \in \mathbb{R}^D$, Comon et al. [41] showed that there exists a decomposition of the form

$$\mathcal{X} = \sum_{k=1}^{K} \lambda_k \mathbf{v}_k \otimes \mathbf{v}_k \otimes \cdots \otimes \mathbf{v}_k.$$

Without loss of generality, we can assume that $\|\mathbf{v}_k\|_2 = 1 \ \forall k$. If we can find a matrix $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_K] \in \mathbb{R}^{D \times K}$ with orthogonal columns, then we say that $\mathcal{X}$ has an orthogonal symmetric tensor decomposition [86]. Such tensors are generated in several applications involving latent variable models. We present two of the applications in Section 5.1 from Anandkumar et al. [8]. There are a lot of other scenarios where this model can be applied with little modification. We start with eigenvalue decomposition of symmetric matrices. If $\mathbf{M} \in \mathbb{R}^{D \times D}$ is a symmetric rank-K matrix then we know that the SVD of $\mathbf{M}$ is given by

$$\mathbf{M} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top = \sum_{k=1}^{K} \lambda_k \mathbf{v}_k \mathbf{v}_k^\top = \sum_{k=1}^{K} \lambda_k \mathbf{v}_k \otimes \mathbf{v}_k,$$

where $\mathbf{\Lambda} = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_K\}$ and $\mathbf{v}_k$ is the k-th column of the orthogonal matrix $\mathbf{V}$. Now, as mentioned before, the orthogonal decomposition of a 3-rd order symmetric tensor $\mathcal{X} \in \mathbb{R}^{D \times D \times D}$ is a collection of orthonormal vectors $\{\mathbf{v}_k\}$ together with corresponding positive scalars $\{\lambda_k\}$ such that

$$\mathcal{X} = \sum_{k=1}^{K} \lambda_k \mathbf{v}_k \otimes \mathbf{v}_k \otimes \mathbf{v}_k.$$

A unit vector $\mathbf{u} \in \mathbb{R}^D$ is an **eigenvector** of $\mathcal{X}$ with corresponding **eigenvalue** λ if

$\mathcal{X}(\mathbf{I}, \mathbf{u}, \mathbf{u}) = \lambda \mathbf{u}$, where $\mathbf{I}$ is the $D \times D$ identity matrix [8]. To see this, one can observe

$$\begin{aligned}\mathcal{X}(\mathbf{I}, \mathbf{u}, \mathbf{u}) &= \sum_{k=1}^{K} \lambda_k \left(\mathbf{I}^\top \mathbf{v}_k\right) \otimes \left(\mathbf{u}^\top \mathbf{v}_k\right) \otimes \left(\mathbf{u}^\top \mathbf{v}_k\right) \\ &= \sum_{k=1}^{K} \lambda_k \left(\mathbf{u}^\top \mathbf{v}_k\right)^2 \mathbf{v}_k.\end{aligned}$$

By the orthogonality of the $\mathbf{v}_k$, it is clear that $\mathcal{X}(\mathbf{I}, \mathbf{v}_k, \mathbf{v}_k) = \lambda_k \mathbf{v}_k \ \forall k$. Now, the orthogonal tensor decomposition proposed in [8] is based on the mapping

$$\mathbf{u} \mapsto \frac{\mathcal{X}(\mathbf{I}, \mathbf{u}, \mathbf{u})}{\|\mathcal{X}(\mathbf{I}, \mathbf{u}, \mathbf{u})\|_2}, \tag{2.2}$$

which can be considered as the tensor equivalent of the well-known matrix power method.

2.2.4 Miscellaneous Definitions

Definition 5 (Majorization)**.** *Consider two vectors* $\mathbf{a} \in \mathbb{R}^S$ *and* $\mathbf{b} \in \mathbb{R}^S$ *with non-increasing entries (i.e.,* $a_i \geq a_j$ *and* $b_i \geq b_j$ *for* $i < j$*). Then* $\mathbf{a}$ *is majorized by* $\mathbf{b}$*, denoted* $\mathbf{a} \prec \mathbf{b}$*, if and only if the following holds:*

$$\sum_{s=1}^{S} a_s = \sum_{s=1}^{S} b_s \text{ and } \sum_{s=1}^{J} a_s \leq \sum_{s=1}^{J} b_s \ \forall J \in [S].$$

Consider $\mathbf{n}_{\text{sym}} \triangleq \frac{N}{S}[1, \ldots, 1] \in \mathbb{R}^S$ for some positive N. Then, any vector $\mathbf{n} = [N_1, \ldots, N_S] \in \mathbb{R}^S$ with non-increasing entries and $\sum_{s=1}^{S} |N_s| = N$ majorizes $\mathbf{n}_{\text{sym}}$, or $\mathbf{n}_{\text{sym}} \prec \mathbf{n}$.

Definition 6 (Schur-convex functions)**.** *The function* $K : \mathbb{R}^S \mapsto \mathbb{R}$ *is Schur-convex if for all* $\mathbf{a} \prec \mathbf{b} \in \mathbb{R}^S$ *we have* $K(\mathbf{a}) \leq K(\mathbf{b})$.

Chapter 3

Correlated Noise Scheme

In this chapter, we first demonstrate the problem with conventional decentralized differentially private computations using a simple example. We then propose a Correlation Assisted Private Estimation (CAPE) framework, which is a novel *decentralized and privacy-preserving* protocol that provides utility close to centralized case. We propose a scheme for incorporating the CAPE protocol in asymmetric network/privacy settings. Finally, we present experimental results with synthetic data to demonstrate the effectiveness of CAPE algorithm applied to a decentralized gradient descent problem.

3.1 Decentralized Data Setting

We consider a decentralized data setting with S sites and a central aggregator node (see Figure 3.1). Each site $s \in [S]$ holds N_s samples and the total number of samples across all sites is given by $N = \sum_{s=1}^{S} N_s$. We assume that all parties are "honest but curious". That is, the sites and the aggregator will follow the protocol but a subset may collude to learn another site's data/function output. Additionally, we assume that the data samples in the local sites are disjoint. We use the terms "distributed" and "decentralized" interchangeably in this work.

3.2 Conventional Approach to Decentralized DP Computations

We describe the problem with conventional decentralized differentially private computations [77]. Then we propose the CAPE approach to improve the performance. Suppose we want to compute the average of N data samples. Each sample x_n is a scalar with

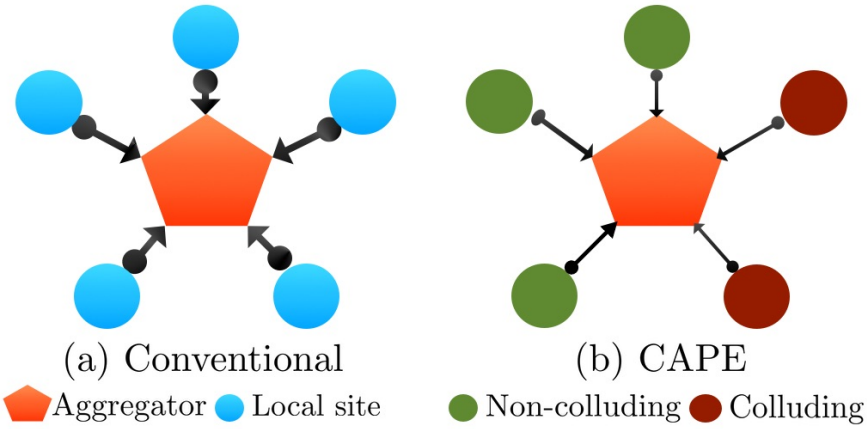

Figure 3.1: The structure of the network: (a) conventional, (b) CAPE

$x_n \in [0, 1]$. We denote the vector of N data samples as

$$\mathbf{x} = [x_1, \ldots, x_{N-1},\ x_N]^\top.$$

We are interested in computing the (ϵ, δ)-differentially private estimate of the mean function:

$$f(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} x_n.$$

To compute the sensitivity [50] of the scalar-valued function $f(\mathbf{x})$, we consider a neighboring data vector $\mathbf{x}' = [x_1, \ldots, x_{N-1},\ x'_N]^\top$. We observe

$$\left| f(\mathbf{x}) - f(\mathbf{x}') \right| = \frac{1}{N} \left| x_N - x'_N \right| \leq \frac{1}{N},$$

which follows from the assumption $x_n \in [0, 1]$. Therefore, to compute the (ϵ, δ)-differentially private estimate of the average $a = f(\mathbf{x})$, we can employ the Gaussian mechanism [50, 48] to release $\hat{a} = a + e$, where $e \sim \mathcal{N}\left(0, \tau^2\right)$ and

$$\tau = \frac{1}{N\epsilon} \sqrt{2 \log \frac{1.25}{\delta}}.$$

Now, in the decentralized setting, each site s holds N_s samples as the entries of $\mathbf{x}_s \in \mathbb{R}^{N_s}$ (see Figure 3.1(a)). We assume $N_s = \frac{N}{S}$ for simplicity. To compute the global average

non-privately, the sites can send $a_s = f(\mathbf{x}_s)$ to the aggregator and the average computed by aggregator ($a_{\text{conv}} = \frac{1}{S}\sum_{s=1}^{S} a_s$) is exactly equal to the average we would get if all the data samples were available in the aggregator node. However, with the privacy concern and considering that the aggregator is honest-but-curious, the sites can employ the conventional decentralized differentially private computation technique. That is, the sites will release (send to the aggregator node) an (ϵ, δ)-differentially private estimate of the function $f(\mathbf{x}_s)$ of their local data $\mathbf{x}_s$. More specifically, each site will generate a noise $e_s \sim \mathcal{N}\left(0, \tau_s^2\right)$ and release/send $\hat{a}_s = f(\mathbf{x}_s) + e_s$ to the aggregator, where

$$\tau_s = \frac{1}{N_s\epsilon}\sqrt{2\log\frac{1.25}{\delta}} = \frac{S}{N\epsilon}\sqrt{2\log\frac{1.25}{\delta}}.$$

The aggregator can then compute the (ϵ, δ) differentially private approximate average as $a_{\text{conv}} = \frac{1}{S}\sum_{s=1}^{S}\hat{a}_s$. We observe

$$a_{\text{conv}} = \frac{1}{S}\sum_{s=1}^{S}\hat{a}_s = \frac{1}{S}\sum_{s=1}^{S} a_s + \frac{1}{S}\sum_{s=1}^{S} e_s.$$

The variance of the estimator a_{conv} is $S \cdot \frac{\tau_s^2}{S^2} = \frac{\tau_s^2}{S} \triangleq \tau_{\text{conv}}^2$. However, if we had all the data samples at the aggregator (pooled-data scenario), we could compute the (ϵ, δ)-differentially private estimate of the average as $a_{\text{pool}} = \frac{1}{N}\sum_{n=1}^{N} x_n + e_{\text{pool}}$, where $e_{\text{pool}} \sim \mathcal{N}\left(0, \tau_{\text{pool}}^2\right)$ and $\tau_{\text{pool}} = \frac{1}{N\epsilon}\sqrt{2\log\frac{1.25}{\delta}} = \frac{\tau_s}{S}$. We observe the ratio

$$\frac{\tau_{\text{pool}}^2}{\tau_{\text{conv}}^2} = \frac{\frac{\tau_s^2}{S^2}}{\frac{\tau_s^2}{S}} = \frac{1}{S}.$$

That is, the decentralized differentially private averaging scheme will always result in a worse performance than the differentially private pooled data case. In the following, we describe a protocol which exploits a correlated noise scheme to achieve the same noise level as the pooled data case.

Algorithm 3.1 Generate zero-sum noise

Require: Local noise variances $\{\tau_s^2\}$; security parameter λ, threshold value t

1: Each site generate $\hat{e}_s \sim \mathcal{N}(0, \tau_s^2)$
2: Aggregator computes $\sum_{s=1}^{S} \hat{e}_s$ according to $\mathsf{SecureAgg}(\lambda, t)$ [22]
3: Aggregator broadcasts $\sum_{s=1}^{S} \hat{e}_s$ to all sites $s \in \{1, \ldots, S\}$
4: Each site computes $e_s = \hat{e}_s - \frac{1}{S} \sum_{s'=1}^{S} \hat{e}_{s'}$
5: **return** e_s

Algorithm 3.2 Correlation Assisted Private Estimation (CAPE)

Require: Data samples $\{\mathbf{x}_s\}$, local noise variances $\{\tau_s^2\}$

1: **for** $s = 1, \ldots, S$ **do** ▷ at each site
2: Generate e_s according to Algorithm 3.1
3: Generate $g_s \sim \mathcal{N}(0, \tau_g^2)$ with $\tau_g^2 = \frac{\tau_s^2}{S}$
4: Compute and send $\hat{a}_s \leftarrow f(\mathbf{x}_s) + e_s + g_s$
5: **end for**
6: Compute $a_{\text{cape}} \leftarrow \frac{1}{S} \sum_{s=1}^{S} \hat{a}_s$ ▷ at the aggregator
7: **return** a_{cape}

3.3 Proposed Scheme: CAPE

3.3.1 Trust/Collusion Model

In our proposed CAPE scheme, we assume that all of the S sites and the central node follow the protocol honestly. However, up to $S_C = \lceil \frac{S}{3} \rceil - 1$ sites can collude with an adversary to learn about some site's data/function output. The central node is also honest-but-curious (and therefore, can collude with an adversary). An adversary can observe the outputs from each site, as well as the output from the aggregator. Additionally, the adversary can know everything about the colluding sites (including their private data). We denote the number of non-colluding sites with S_H such that $S = S_C + S_H$. Without loss of generality, we designate the non-colluding sites with $\{1, \ldots, S_H\}$ (see Figure 3.1(b)).

3.3.2 Correlated Noise

We design the noise generation procedure such that: i) we can ensure (ϵ, δ) differential privacy of the algorithm output from each site and ii) achieve the noise level of the pooled data scenario in the final output from the aggregator. We achieve that by

employing a correlated noise addition scheme. Considering the same decentralized averaging problem as Section 3.2, we intend to release (and send to the aggregator) $\hat{a}_s = f(\mathbf{x}_s) + e_s + g_s$ from each site s, where e_s and g_s are two noise terms. The variances of e_s and g_s are chosen to ensure that the noise $e_s + g_s$ is sufficient to guarantee (ϵ, δ)-differential privacy to $f(\mathbf{x}_s)$. Here, each site generates the noise $g_s \sim \mathcal{N}(0, \tau_g^2)$ locally and the noise $e_s \sim \mathcal{N}(0, \tau_e^2)$ jointly with all other sites such that $\sum_{s=1}^{S} e_s = 0$. We employ the recently proposed secure aggregation protocol (SecureAgg) by Bonawitz et al. [22] to generate e_s that ensures $\sum_{s=1}^{S} e_s = 0$. The SecureAgg protocol utilizes Shamir's t-out-of-n secret sharing [122] and is communication-efficient.

3.3.3 Detailed Description of CAPE Protocol

In our proposed scheme, each site $s \in [S]$ generates a noise term $\hat{e}_s \sim \mathcal{N}(0, \tau_s^2)$ independently. The aggregator computes $\sum_{s=1}^{S} \hat{e}_s$ according to the SecureAgg protocol and broadcasts it to all the sites. Each site then sets

$$e_s = \hat{e}_s - \frac{1}{S} \sum_{s'=1}^{S} \hat{e}_{s'}$$

to achieve $\sum_{s=1}^{S} e_s = 0$. We show the complete noise generation procedure in Algorithm 3.1. Note that, the original SecureAgg protocol is intended for computing sum of D-dimensional vectors in a finite field $\mathbb{Z}_\lambda^D$. However, we need to perform the summation of Gaussian random variables over $\mathbb{R}$ or $\mathbb{R}^D$. To accomplish this, each site can employ a mapping map : $\mathbb{R} \mapsto \mathbb{Z}_\lambda$ that performs a stochastic quantization [126] for large-enough λ. The aggregator can compute the sum in the finite field according to SecureAgg and then invoke a reverse mapping remap : $\mathbb{Z}_\lambda \mapsto \mathbb{R}$ before broadcasting $\sum_{s=1}^{S} \hat{e}_s$ to the sites. Algorithm 3.1 can be readily extended to generate array-valued zero-sum noise terms. We observe that the variance of e_s is given by

$$\tau_e^2 = \mathbb{E}\left[\left(\hat{e}_s - \frac{1}{S}\sum_{s'=1}^{S} \hat{e}_{s'}\right)^2\right] = \left(1 - \frac{1}{S}\right)\tau_s^2. \tag{3.1}$$

Additionally, we choose

$$\tau_g^2 = \frac{\tau_s^2}{S}. \tag{3.2}$$

Each site then generates the noise $g_s \sim \mathcal{N}(0, \tau_g^2)$ independently and sends $\hat{a}_s = f(\mathbf{x}_s) + e_s + g_s$ to the aggregator. Note that neither of the terms e_s and g_s has large enough variance to provide (ϵ, δ)-differential privacy guarantee to $f(\mathbf{x}_s)$. However, we chose the variances of e_s and g_s to ensure that the $e_s + g_s$ is sufficient to ensure a differential privacy guarantee to $f(\mathbf{x}_s)$ at site s. The chosen variance of g_s also ensures that the output from the aggregator would have the same noise variance as the differentially private pooled-data scenario. To see this, observe that we compute the following at the aggregator (Step 6 of Algorithm 3.2):

$$a_{\text{cape}} = \frac{1}{S}\sum_{s=1}^{S} \hat{a}_s = \frac{1}{S}\sum_{s=1}^{S} f(\mathbf{x}_s) + \frac{1}{S}\sum_{s=1}^{S} g_s,$$

where we used $\sum_s e_s = 0$. The variance of the estimator a_{cape} is $\tau^2_{\text{cape}} = S \cdot \frac{\tau_g^2}{S^2} = \tau^2_{\text{pool}}$, which is the exactly the same as if all the data were present at the aggregator. This claim is formalized in Lemma 3.1. We show the complete algorithm in Algorithm 3.2. The privacy of Algorithm 3.2 is given by Theorem 3.1.

Theorem 3.1 (Privacy of CAPE Algorithm (Algorithm 3.2))**.** *Consider Algorithm 3.2 in the decentralized data setting of Section 3.1 with $N_s = \frac{N}{S}$ and $\tau_s^2 = \tau^2$ for all sites $s \in [S]$. Suppose that at most $S_C = \lceil \frac{S}{3} \rceil - 1$ sites can collude after execution. Then Algorithm 3.2 guarantees (ϵ, δ)-differential privacy for each site, where (ϵ, δ) satisfy the relation $\delta = 2\frac{\sigma_z}{\epsilon - \mu_z}\phi\left(\frac{\epsilon - \mu_z}{\sigma_z}\right)$, $\phi(\cdot)$ is the density for standard Normal random variable and (μ_z, σ_z) are given by*

$$\mu_z = \frac{S^3}{2\tau^2 N^2(1+S)}\left(\frac{S - S_C + 2}{S - S_C} + \frac{\frac{9}{S-S_C}S_C^2}{S(1+S) - 3S_C^2}\right), \tag{3.3}$$

$$\sigma_z^2 = \frac{S^3}{\tau^2 N^2(1+S)}\left(\frac{S - S_C + 2}{S - S_C} + \frac{\frac{9}{S-S_C}S_C^2}{S(1+S) - 3S_C^2}\right). \tag{3.4}$$

Remark 1. *Theorem 3.1 is stated for the symmetric setting: $N_s = \frac{N}{S}$ and $\tau_s^2 = \tau^2$ $\forall s \in$*

$[S]$. *As with many algorithms using the approximate differential privacy, the guarantee holds for a range of* (ϵ, δ) *pairs subject to a tradeoff constraint between* ϵ *and* δ*, as in the simple case in* (2.1).

Proof. We identify the S_H non-colluding sites with $s \in \{1, \ldots, S_H\} \triangleq \mathbb{S}_H$ and the S_C colluding sites with $s \in \{S_H + 1, \ldots, S\} \triangleq \mathbb{S}_C$. The adversary can observe the outputs from each site (including the aggregator). Additionally, the colluding sites can share their private data and the noise terms, $\hat{e}_s$ and g_s for $s \in \mathbb{S}_C$, with the adversary. For simplicity, we assume that all sites have equal number of samples (i.e., $N_s = \frac{N}{S}$) and $\tau_s^2 = \tau^2$. We present a scheme in Section 3.3.7 that incorporates unequal sample size/privacy requirements at sites.

To infer the private data of the sites $s \in \mathbb{S}_H$, the adversary can observe $\hat{\mathbf{a}} = [\hat{a}_1, \ldots, \hat{a}_{S_H}]^\top \in \mathbb{R}^{S_H}$ and $\hat{e} = \sum_{s \in \mathbb{S}_H} \hat{e}_s$. Note that the adversary can learn the partial sum $\hat{e}$ because they can get the sum $\sum_s \hat{e}_s$ from the aggregator and the noise terms $\{\hat{e}_{S_H+1}, \ldots, \hat{e}_S\}$ from the colluding sites. Therefore, the vector $\mathbf{y} = \left[\hat{\mathbf{a}}^\top, \hat{e}\right]^\top \in \mathbb{R}^{S_H+1}$ is what the adversary can observe to make inference about the non-colluding sites. To prove differential privacy guarantee, we must show that

$$\left| \log \frac{g(\mathbf{y}|\mathbf{a})}{g(\mathbf{y}|\mathbf{a}')} \right| \leq \epsilon$$

holds with probability (over the randomness of the mechanism) at least $1 - \delta$. Here, $\mathbf{a} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{S_H})]^\top$ and $g(\cdot|\mathbf{a})$ and $g(\cdot|\mathbf{a}')$ are the probability density functions of $\mathbf{y}$ under $\mathbf{a}$ and $\mathbf{a}'$, respectively. The vectors $\mathbf{a}$ and $\mathbf{a}'$ differ in only one coordinate (neighboring). Without loss of generality, we assume that $\mathbf{a}$ and $\mathbf{a}'$ differ in the first coordinate. We note that the maximum difference is $\frac{1}{N_s}$ as the sensitivity of the function $f(\mathbf{x_s})$ is $\frac{1}{N_s}$. Recall that we release $\hat{a}_s = f(\mathbf{x}_s) + e_s + g_s$ from each site. We observe

$$\mathbb{E}(\hat{a}_s) = f(\mathbf{x}_s),\ \mathrm{Var}(\hat{a}_s) = \tau^2,\ \forall s \in [S]$$
$$\mathbb{E}(\hat{a}_{s_1}\hat{a}_{s_2}) = f(\mathbf{x}_{s_1})f(\mathbf{x}_{s_2}) - \frac{\tau^2}{S},\ \forall s_1 \neq s_2 \in [S].$$

That is, the random variable $\hat{\mathbf{a}}$ is $\mathcal{N}(\mathbf{a}, \Sigma_{\hat{\mathbf{a}}})$, where

$$\Sigma_{\hat{\mathbf{a}}} = (1 + \frac{1}{S})\tau^2\mathbf{I} - \mathbf{1}\mathbf{1}^\top\frac{\tau^2}{S} \in \mathbb{R}^{S_H \times S_H},$$

and $\mathbf{1}$ is a vector of all ones. Without loss of generality, we can assume [48] that $\mathbf{a} = \mathbf{0}$ and $\mathbf{a}' = \mathbf{a} - \mathbf{v}$, where $\mathbf{v} = \left[\frac{1}{N_s}, 0, \ldots, 0\right]^\top$. Additionally, the random variable $\hat{e}$ is $\mathcal{N}(0, \tau_{\hat{e}}^2)$, where $\tau_{\hat{e}}^2 = S_H\tau^2$. Therefore, $g(\mathbf{y}|\mathbf{a})$ is the density of $\mathcal{N}(\mathbf{0}, \Sigma)$, where

$$\Sigma = \begin{bmatrix} \Sigma_{\hat{\mathbf{a}}} & \Sigma_{\hat{\mathbf{a}}\hat{e}} \\ \Sigma_{\hat{\mathbf{a}}\hat{e}}^\top & \tau_{\hat{e}}^2 \end{bmatrix} \in \mathbb{R}^{(S_H+1)\times(S_H+1)}.$$

With some simple algebra, we can find the expression for $\Sigma_{\hat{\mathbf{a}}\hat{e}}$ as:

$$\Sigma_{\hat{\mathbf{a}}\hat{e}} = \left(1 - \frac{S_H}{S}\right)\tau^2\mathbf{1} \in \mathbb{R}^{S_H}.$$

If we denote $\tilde{\mathbf{v}} = \left[\mathbf{v}^\top, 0\right]^\top \in \mathbb{R}^{S_H+1}$ then we observe

$$\begin{aligned}\left|\log\frac{g(\mathbf{y}|\mathbf{a})}{g(\mathbf{y}|\mathbf{a}')}\right| &= \left|-\frac{1}{2}\left(\mathbf{y}^\top\Sigma^{-1}\mathbf{y} - (\mathbf{y}+\tilde{\mathbf{v}})^\top\Sigma^{-1}(\mathbf{y}+\tilde{\mathbf{v}})\right)\right| \\ &= \left|\frac{1}{2}\left(2\mathbf{y}^\top\Sigma^{-1}\tilde{\mathbf{v}} + \tilde{\mathbf{v}}^\top\Sigma^{-1}\tilde{\mathbf{v}}\right)\right| \\ &= \left|\mathbf{y}^\top\Sigma^{-1}\tilde{\mathbf{v}} + \frac{1}{2}\tilde{\mathbf{v}}^\top\Sigma^{-1}\tilde{\mathbf{v}}\right| = |z|,\end{aligned}$$

where $z = \mathbf{y}^\top\Sigma^{-1}\tilde{\mathbf{v}} + \frac{1}{2}\tilde{\mathbf{v}}^\top\Sigma^{-1}\tilde{\mathbf{v}}$. Using the matrix inversion lemma for block matrices [70, Section 0.7.3] and some algebra, we have

$$\Sigma^{-1} = \begin{bmatrix} \Sigma_{\hat{\mathbf{a}}}^{-1} + \frac{1}{K}\Sigma_{\hat{\mathbf{a}}}^{-1}\Sigma_{\hat{\mathbf{a}}\hat{e}}\Sigma_{\hat{\mathbf{a}}\hat{e}}^\top\Sigma_{\hat{\mathbf{a}}}^{-1} & -\frac{1}{K}\Sigma_{\hat{\mathbf{a}}}^{-1}\Sigma_{\hat{\mathbf{a}}\hat{e}} \\ -\frac{1}{K}\Sigma_{\hat{\mathbf{a}}\hat{e}}^\top\Sigma_{\hat{\mathbf{a}}}^{-1} & \frac{1}{K} \end{bmatrix},$$

where $\Sigma_{\hat{\mathbf{a}}}^{-1} = \frac{S}{(1+S)\tau^2}\left(\mathbf{I} + \frac{2}{S_H}\mathbf{1}\mathbf{1}^\top\right)$ and $K = \tau_{\hat{e}}^2 - \Sigma_{\hat{\mathbf{a}}\hat{e}}^\top\Sigma_{\hat{\mathbf{a}}}^{-1}\Sigma_{\hat{\mathbf{a}}\hat{e}}$. Note that z is a Gaussian random variable $\mathcal{N}(\mu_z, \sigma_z^2)$ with parameters $\mu_z = \frac{1}{2}\tilde{\mathbf{v}}^\top\Sigma^{-1}\tilde{\mathbf{v}}$ and $\sigma_z^2 = \tilde{\mathbf{v}}^\top\Sigma^{-1}\tilde{\mathbf{v}}$ given

by

$$\mu_z = \frac{S^3}{2\tau^2 N^2(1+S)}\left(\frac{S-S_C+2}{S-S_C} + \frac{\frac{9}{S-S_C}S_C^2}{S(1+S)-3S_C^2}\right),$$
$$\sigma_z^2 = \frac{S^3}{\tau^2 N^2(1+S)}\left(\frac{S-S_C+2}{S-S_C} + \frac{\frac{9}{S-S_C}S_C^2}{S(1+S)-3S_C^2}\right).$$

Now, we observe

$$\begin{aligned}\Pr\left[\left|\log\frac{g(\mathbf{y}|\mathbf{a})}{g(\mathbf{y}|\mathbf{a}')}\right| \le \epsilon\right] &= \Pr\left[|z| \le \epsilon\right] \\ &= 1 - 2\Pr\left[z > \epsilon\right] \\ &= 1 - 2Q\left(\frac{\epsilon-\mu_z}{\sigma_z}\right) \\ &> 1 - 2\frac{\sigma_z}{\epsilon-\mu_z}\phi\left(\frac{\epsilon-\mu_z}{\sigma_z}\right),\end{aligned}$$

where $Q(\cdot)$ is the Q-function [102] and $\phi(\cdot)$ is the density for standard Normal random variable. The last inequality follows from the bound $Q(x) < \frac{\phi(x)}{x}$ [102]. Therefore, the proposed CAPE ensures (ϵ, δ)-differential privacy with $\delta = 2\frac{\sigma_z}{\epsilon-\mu_z}\phi\left(\frac{\epsilon-\mu_z}{\sigma_z}\right)$ for each site, assuming that the number of colluding sites is at-most $\lceil \frac{S}{3} \rceil - 1$. As the local datasets are disjoint and differential privacy is invariant under post processing, the release of a_{cape} also satisfies (ϵ, δ)-differential privacy. □

Remark 2. *We use the* SecureAgg *protocol [22] to generate the zero-sum noise terms by mapping floating point numbers to a finite field. Such mappings are shown to be vulnerable to certain attacks [106]. However, the floating point implementation issues are out of scope for this work. We refer the reader to the work of Balcer and Vadhan [13] for possible remedies. We believe a very interesting direction of future work would be to address the issue in our decentralized data setting.*

3.3.4 Utility Analysis

Our goal is to ensure (ϵ, δ)-differential privacy for each site and achieve $\tau^2_{\text{cape}} = \tau^2_{\text{pool}}$ at the aggregator (see Lemma 3.1). The CAPE protocol guarantees (ϵ, δ)-differential

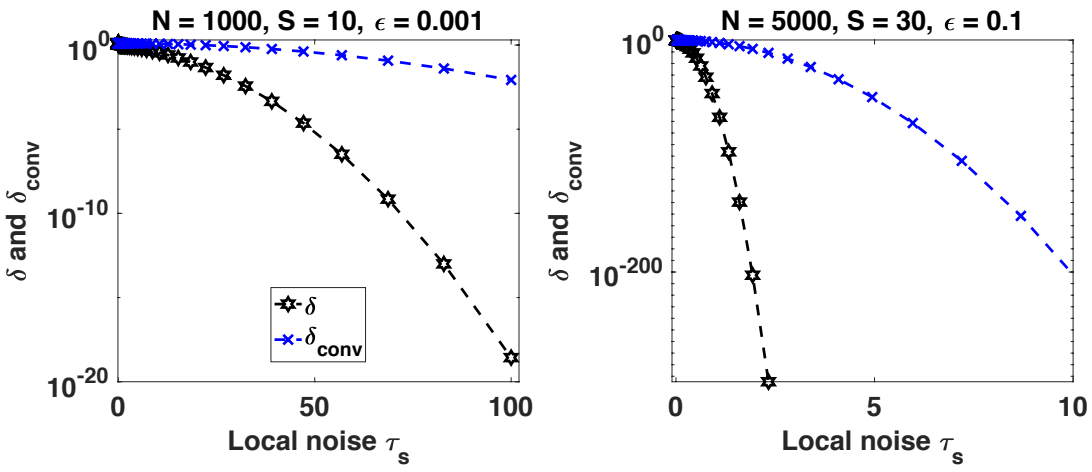

Figure 3.2: Variation of δ and δ_{conv} with τ_s for different values of S and ϵ

privacy with $\delta = 2\frac{\sigma_z}{\epsilon-\mu_z}\phi\left(\frac{\epsilon-\mu_z}{\sigma_z}\right)$. We claim that this δ guarantee is much better than the δ guarantee in the conventional decentralized differentially private scheme. We empirically validate this claim by comparing δ with δ_{conv} in Figure 3.2. Here, δ_{conv} is the smallest δ guarantee we can afford in the conventional decentralized differentially private scheme to achieve the same noise variance as the pooled-data scenario for a given ϵ. We plot δ and δ_{conv} against different τ_s values for $S_C = \left\lceil \frac{S}{3} \right\rceil - 1$ and different combinations of ϵ and S. We observe from the figure that δ is always smaller than δ_{conv}.

Additionally, we empirically compare δ and δ_{conv} for weaker collusion assumptions (i.e., fewer colluding sites) in Figure 3.3. To that end, we vary the fraction $\frac{S_C}{S}$ and plot the resulting δ and δ_{conv} for different combinations of ϵ, S and τ_s. Again, we observe that δ is always smaller than δ_{conv}. That is, we are ensuring a much better privacy guarantee by employing the CAPE scheme over the conventional approach for achieving the same noise level at the aggregator output (and therefore the same utility) as the pooled data scenario.

Lemma 3.1. *Consider the symmetric setting: $N_s = \frac{N}{S}$ and $\tau_s^2 = \tau^2$ for all sites $s \in [S]$. Let the variances of the noise terms e_s and g_s (Step 4 of Algorithm 3.2) be $\tau_e^2 = \left(1 - \frac{1}{S}\right)\tau^2$ and $\tau_g^2 = \frac{\tau^2}{S}$, respectively. If we denote the variance of the additive noise (for preserving privacy) in the pooled data scenario by τ_{pool}^2 and the variance of the estimator a_{cape} (Step 6 of Algorithm 3.2) by τ_{cape}^2 then Algorithm 3.2 achieves the same expected error as the pooled-data scenario (i.e., $\tau_{\text{pool}}^2 = \tau_{\text{cape}}^2$).*

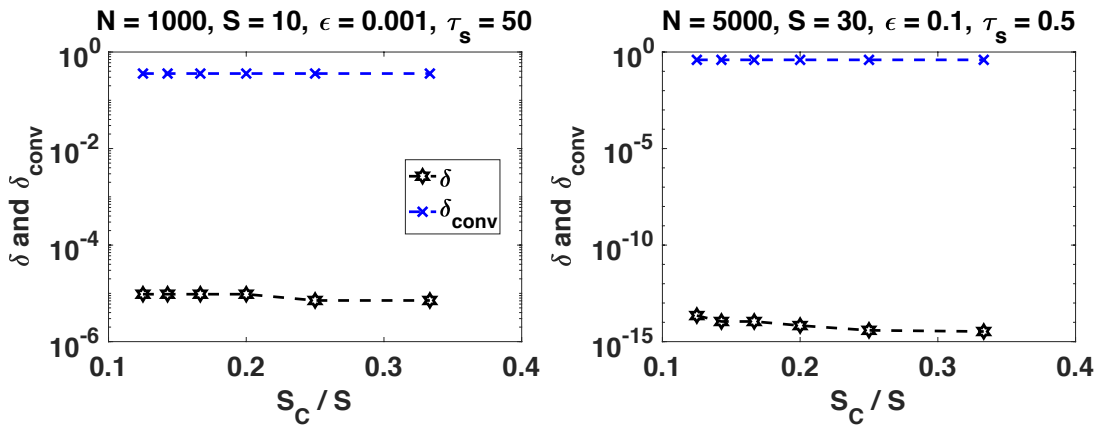

Figure 3.3: Variation of δ and δ_{conv} with $\frac{S_C}{S}$, for different values of S and ϵ

Proof. We prove the lemma according to [77]. Recall that in the pooled data scenario, the sensitivity of the function $f(\mathbf{x})$ is $\frac{1}{N}$, where $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_S]$. Therefore, to approximate $f(\mathbf{x})$ satisfying (ϵ, δ) differential privacy, we need to have additive Gaussian noise standard deviation at least $\tau_{\text{pool}} = \frac{1}{N\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$. Next, consider the decentralized data setting with local noise standard deviation given by

$$\tau_s = \frac{1}{N_s\epsilon}\sqrt{2\log\frac{1.25}{\delta}} = \frac{S}{N\epsilon}\sqrt{2\log\frac{1.25}{\delta}} = \tau$$

We observe $\tau_{\text{pool}} = \frac{\tau_s}{S} \implies \tau^2_{\text{pool}} = \frac{\tau^2}{S^2}$. We will now show that the CAPE algorithm will yield the same noise variance of the estimator at the aggregator. Recall that at the aggregator we compute $a_{\text{cape}} = \frac{1}{S}\sum_{s=1}^{S}\hat{a}_s = \frac{1}{N}\sum_{n=1}^{N}x_n + \frac{1}{S}\sum_{s=1}^{S}g_s$. The variance of the estimator a_{cape} is:

$$\tau^2_{\text{cape}} \triangleq S\cdot\frac{\tau_g^2}{S^2} = \frac{\tau_g^2}{S} = \frac{\tau^2}{S^2},$$

which is exactly the same as the pooled data scenario. Therefore, the CAPE algorithm allows us to achieve the same additive noise variance as the pooled data scenario, while satisfying (ϵ, δ) differential privacy at the sites and for the final output from the aggregator, where (ϵ, δ) satisfy the relation $\delta = 2\frac{\sigma_z}{\epsilon-\mu_z}\phi\left(\frac{\epsilon-\mu_z}{\sigma_z}\right)$. □

Proposition 3.1. *(Performance gain) If the local noise variances are $\{\tau_s^2\}$ for $s \in [S]$ then the* CAPE *algorithm achieves a gain of $G = \frac{\tau^2_{\text{conv}}}{\tau^2_{\text{cape}}} = S$, where τ^2_{conv} and τ^2_{cape} are the*

noise variances of the final estimate at the aggregator in the conventional decentralized differentially private scheme and the CAPE *scheme, respectively.*

Proof. The local noise variances are $\{\tau_s^2\}$ for $s \in [S]$. In the conventional decentralized differentially private scheme, we compute the following at the aggregator:

$$a_{\text{conv}} = \frac{1}{S}\sum_{s=1}^{S} a_s + \frac{1}{S}\sum_{s=1}^{S} e_s.$$

The variance of the estimator is:

$$\tau_{\text{conv}}^2 = \sum_{s=1}^{s} \frac{\tau_s^2}{S^2} = \frac{1}{S^2}\sum_{s=1}^{s} \tau_s^2.$$

In the CAPE scheme, we compute the following quantity at the aggregator:

$$a_{\text{cape}} = \frac{1}{S}\sum_{s=1}^{S} a_s + \frac{1}{S}\sum_{s=1}^{S} e_s + \frac{1}{S}\sum_{s=1}^{S} g_s.$$

The variance of the estimator is:

$$\tau_{\text{cape}}^2 = \sum_{s=1}^{s} \frac{\tau_g^2}{S^2} = \frac{1}{S^3}\sum_{s=1}^{s} \tau_s^2.$$

Therefore, the gain of the CAPE scheme over conventional decentralized differentially private approach is

$$G = \frac{\tau_{\text{conv}}^2}{\tau_{\text{cape}}^2} = S,$$

which completes the proof. □

Note that, even in the case of site drop-out, we achieve $\sum_s e_s = 0$, as long as the number of active sites is above some threshold (see Bonawitz et al. [22] for details). Therefore, the performance gain of CAPE remains the same irrespective of the number of dropped-out sites.

Table 3.1: Comparison of communication overhead

Algorithm	**Site**	**Aggregator**
CAPE	$O(S+D)$	$O(S^2+SD)$
Heikkilä et al. [69]	$\Theta(DM)$	$\Theta(SDM)$
Bonawitz et al. [22]	$O(S+D)$	$O(S^2+SD)$

3.3.5 Communication Overhead

The conventional D-dimensional averaging needs only one message from each site, thus SD or $\Theta(SD)$ is the communication complexity. Our CAPE scheme employs the SecureAgg protocol to compute the zero-sum noise. The SecureAgg protocol [22] entails an $O(S+D)$ overhead for each site and $O(S^2+SD)$ for the server/aggregator. The rest of our scheme requires $\Theta(D)$ and $\Theta(SD)$ communication overheads for the sites and the aggregator, respectively. On the other hand, the scheme proposed in [69] has a communication cost proportional to $(S+1)DM$ or $\Theta(SDM)$, where M is the number of compute nodes. Goryczka et al. [64] compared several secret sharing, homomorphic encryption and perturbation-based secure sum aggregation and showed their communication complexities. Except for the secret sharing approach (which requires $O(S^2)$ overhead), the other approaches are $O(S)$ in communication complexity. A comparison of communication overhead for different algorithms are shown in Table 3.1.

3.3.6 Scope of CAPE

CAPE is motivated by scientific research collaborations that are common in medicine and biology. Privacy regulations prevent sites from sharing the local raw data. Additionally, the data is often high dimensional (e.g., in neuroimaging) and sites have small sample sizes. Joint learning across datasets can yield discoveries that are impossible to obtain from a single site. CAPE can benefit functions f with sensitivities satisfying some conditions (see Proposition 3.2). In addition to the averaging function, many functions of interest have sensitivities that satisfy such conditions. Examples include

the empirical average loss functions used in machine learning and deep neural networks. Additionally, we can use the Stone-Weierstrass theorem [121] to approximate a loss function f and apply CAPE, as we show in Chapter 8. Furthermore, we can use the nomographic representation of functions to approximate a desired function in a decentralized manner [88, 129, 28, 128] (for applications in communications [110, 98, 61, 62]), while keeping the data differentially private. More common applications include gradient based optimization algorithms, k-means clustering and estimating probability distributions.

Proposition 3.2. *Consider a decentralized setting with $S > 1$ sites in which site $s \in [S]$ has a dataset $\mathbb{D}_s$ of N_s samples and $\sum_{s=1}^{S} N_s = N$. Suppose the sites are computing a function $f(\mathbb{D})$ with $\mathcal{L}_p$ sensitivity $\Delta(N)$ employing the* CAPE *scheme. Denote $\mathbf{n} = [N_1,\ N_2,\ \ldots, N_S]$ and observe the ratio $H(\mathbf{n}) = \frac{\tau^2_{\text{cape}}}{\tau^2_{\text{pool}}} = \frac{\sum_{s=1}^{S} \Delta^2(N_s)}{S^3\Delta^2(N)}$. Then the* CAPE *protocol achieves $H(\mathbf{n}) = 1$, if*

- *for convex $\Delta(N)$ we have: $\Delta\left(\frac{N}{S}\right) = S\Delta(N)$*
- *for general $\Delta(N)$ we have: $S^3\Delta^2(N) = \sum_{s=1}^{S} \Delta^2(N_s)$.*

Proof. We start with reviewing a lemma [103, Proposition C.2] necessary for the proof.

Lemma 3.2. *If K is symmetric and convex, then K is Schur-convex. The converse does not hold.*

Now, as the sites are computing the function f with $\mathcal{L}_p$ sensitivity $\Delta(N)$, the local noise standard deviation for preserving privacy is proportional to $\Delta(N_s)$ by Gaussian mechanism [50]. It can be written as: $\tau_s = \Delta(N_s)C$, where C is a constant for a given (ϵ, δ) pair. Similarly, the noise standard deviation in the pooled data scenario can be written as: $\tau_{\text{pool}} = \Delta(N)C$. Now, the final noise variance at the aggregator for CAPE protocol is:

$$\tau^2_{\text{cape}} = \sum_{s=1}^{S} \frac{\tau_g^2}{S^2} = \frac{1}{S^3} \sum_{s=1}^{S} \Delta^2(N_s)C^2.$$

Now, we observe the ratio:

$$H(\mathbf{n}) = \frac{\tau^2_{\text{cape}}}{\tau^2_{\text{pool}}} = \frac{\sum_{s=1}^{S} \Delta^2(N_s)}{S^3 \Delta^2(N)}.$$

As we want to achieve the same noise variance as the pooled-data scenario, we need

$$S^3 \Delta^2(N) = \sum_{s=1}^{S} \Delta^2(N_s),$$

which proves the case for general sensitivity function $\Delta(N)$. Now, if $\Delta^2(N)$ is convex then the by Lemma 3.2 the function

$$K(\mathbf{n}) = \sum_{s=1}^{S} \Delta^2(N_s)$$

is Schur-convex. Thus, the minimum of $K(\mathbf{n})$ is obtained when $\mathbf{n} = \mathbf{n}_{\text{sym}}$ (using Definitions 5 and 6). We observe:

$$K_{\min}(\mathbf{n}) = \sum_{s=1}^{S} \Delta^2 \left(\frac{N}{S} \right) = S \cdot \Delta^2 \left(\frac{N}{S} \right).$$

Therefore, when $\Delta(N)$ is convex, we achieve $H(\mathbf{n}) = 1$ if $\Delta(\frac{N}{S}) = S\Delta(N)$. □

3.3.7 Unequal Sample Sizes at Sites

Remark 3. *Note that the* CAPE *algorithm achieves the same noise variance as the pooled-data scenario (i.e., $\tau^2_{\text{cape}} = \tau^2_{\text{pool}}$) in the symmetric setting: $N_s = \frac{N}{S}$ and $\tau_s^2 = \tau^2$ for all sites $s \in [S]$. In general, the ratio $H(\mathbf{n}) = \frac{\tau^2_{\text{cape}}}{\tau^2_{\text{pool}}}$, where $\mathbf{n} \triangleq [N_1,\ N_2,\ \ldots, N_S]$, is a function of the sample sizes in the sites. For our decentralized averaging problem of Section 3.2, we observe:*

$$H(\mathbf{n}) = \frac{N^2}{S^3} \sum_{s=1}^{S} \frac{1}{N_s^2}.$$

As $H(\mathbf{n})$ is a Schur-convex function, it can be shown using majorization theory [103] that

$$1 \leq H(\mathbf{n}) \leq \frac{N^2}{S^3}\left(\frac{1}{(N-S+1)^2}+S-1\right),$$

where the minimum is achieved for the symmetric setting. That is, CAPE *achieves the smallest noise variance at the aggregator in the symmetric setting.*

We now propose a generalization of the CAPE scheme that can be applied to scenarios where different sites have different privacy requirements and/or sample sizes. Additionally, sites may want the aggregator to use different weights for different sites (possibly according to the quality of the output from a site). A scheme for doing so is shown in [77]. In this work, we describe a more refined approach for the robust CAPE scheme proposed here.

Let us assume that site s requires τ_s local noise variance for ensuring (ϵ_s, δ_s)-differential privacy for its output, where

$$\tau_s = \frac{1}{N_s \epsilon_s}\sqrt{2\log\frac{1.25}{\delta_s}},$$

according to the Gaussian mechanism [50]. As before, we intend to parameterize our algorithm using $\{\tau_s\}$ to abstract away $\{(\epsilon_s, \delta_s, N_s)\}$. To initiate the CAPE protocol, each site will generate $\hat{e}_s \sim \mathcal{N}(0, \sigma_s^2)$ and $g_s \sim \mathcal{N}(0, \tau_{gs}^2)$. The aggregator intends to compute a weighted average of each site's data/output with weights selected according to some quality measure. For example, if the aggregator knows that a particular site is suffering from more noisy observations than other sites, it can choose to give the output from that site less weight while combining the site results. Let us denote the weights by $\{\mu_s\}$ such that $\sum_{s=1}^{S}\mu_s = 1$ and $\mu_s \geq 0$. First, the aggregator computes $\sum_{s=1}^{S}\mu_s\hat{e}_s$ using the SecureAgg protocol and broadcasts it to all sites. Each site then sets

$$e_s = \hat{e}_s - \frac{1}{\mu_s S}\sum_{i=1}^{S}\mu_i\hat{e}_i,$$

to achieve $\sum_{s=1}^{S} \mu_s e_s = 0$ and releases

$$\hat{a}_s = a_s + e_s + g_s.$$

Now, the aggregator computes

$$\begin{aligned} a_{\text{cape}} &= \sum_{s=1}^{S} \mu_s \hat{a}_s \\ &= \sum_{s=1}^{S} \mu_s a_s + \sum_{s=1}^{S} \mu_s e_s + \sum_{s=1}^{S} \mu_s g_s \\ &= \sum_{s=1}^{S} \mu_s a_s + \sum_{s=1}^{S} \mu_s g_s, \end{aligned}$$

where we used $\sum_{s=1}^{S} \mu_s e_s = 0$. Now, to achieve the same utility as the pooled data scenario (i.e. $\tau^2_{\text{pool}} = \tau^2_{\text{cape}}$), we need to ensure

$$\text{Var}\left[\sum_{s=1}^{S} \mu_s g_s\right] = \tau^2_{\text{pool}} \implies \sum_{s=1}^{S} \mu_s^2 \tau_{gs}^2 = \tau^2_{\text{pool}}.$$

Additionally, for guaranteeing privacy to the local sites, we need

$$\tau_{es}^2 + \tau_{gs}^2 \geq \tau_s^2,$$

where τ_{es}^2 is the variance of e_s and is a function of σ_s^2. With these constraints, we can formulate a feasibility problem to solve for the unknown noise variances $\{\sigma_s^2, \tau_{gs}^2\}$ as

$$\begin{aligned} \text{minimize} \quad & 0 \\ \text{subject to} \quad & \tau_{es}^2 + \tau_{gs}^2 \geq \tau_s^2, \\ & \sum_{s=1}^{S} \mu_s^2 \tau_{gs}^2 = \tau^2_{\text{pool}}, \end{aligned}$$

for all $s \in [S]$, where $\{\mu_s\}$, τ_{pool} and $\{\tau_s\}$ are known to the aggregator. For this problem, multiple solutions are possible. We present one solution here.

Solution. We observe that the variance τ_{es}^2 of the zero-mean random variable $e_s =$

$\hat{e}_s - \frac{1}{\mu_s S}\sum_{i=1}^{S}\mu_i\hat{e}_i$ can be computed as

$$\begin{aligned}
\tau_{es}^2 &= \mathrm{Var}\left[\hat{e}_s - \frac{1}{\mu_s S}\sum_{i=1}^{S}\mu_i\hat{e}_i\right] \\
&= \mathbb{E}\left[\hat{e}_s^2 + \frac{1}{\mu_s^2 S^2}\left(\sum_{i=1}^{S}\mu_i\hat{e}_i\right)^2 - \frac{2}{\mu_s S}\hat{e}_s\sum_{i=1}^{S}\mu_i\hat{e}_i\right] \\
&= \sigma_s^2 + \frac{1}{\mu_s^2 S^2}\sum_{i=1}^{S}\mu_i^2\sigma_i^2 - \frac{2}{\mu_s S}\cdot\mu_s\sigma_s^2 \\
&= \left(1-\frac{2}{S}\right)\sigma_s^2 + \frac{1}{\mu_s^2 S^2}\sum_{i=1}^{S}\mu_i^2\sigma_i^2.
\end{aligned}$$

Note that we need $\sum_{s=1}^{S}\mu_s^2\tau_{gs}^2 = \tau_{\text{pool}}^2$. One solution is to set

$$\tau_{gs}^2 = \frac{1}{\mu_s^2 S}\tau_{\text{pool}}^2. \tag{3.5}$$

Using the constraint and the expression for τ_{gs}^2, we have

$$\begin{aligned}
&\tau_{es}^2 + \tau_{gs}^2 = \tau_s^2 \\
\implies &\left(1-\frac{2}{S}\right)\sigma_s^2 + \frac{1}{\mu_s^2 S^2}\sum_{i=1}^{S}\mu_i^2\sigma_i^2 + \frac{1}{\mu_s^2 S}\tau_{\text{pool}}^2 = \tau_s^2 \\
\implies &\left(1-\frac{2}{S}\right)\sigma_s^2 + \frac{1}{\mu_s^2 S^2}\mu_s^2\sigma_s^2 + \frac{1}{\mu_s^2 S^2}\sum_{i\neq s}\mu_i^2\sigma_i^2 = \tau_s^2 - \frac{1}{\mu_s^2 S}\tau_{\text{pool}}^2 \\
\implies &\left(1-\frac{1}{S}\right)^2\sigma_s^2 + \frac{1}{\mu_s^2 S^2}\sum_{i\neq s}\mu_i^2\sigma_i^2 = \tau_s^2 - \frac{1}{\mu_s^2 S}\tau_{\text{pool}}^2.
\end{aligned}$$

We can write this in matrix form and solve for $\{\sigma_s^2\}$ as

$$\begin{bmatrix}
\left(1-\frac{1}{S}\right)^2 & \frac{\mu_2^2}{\mu_1^2 S^2} & \frac{\mu_3^2}{\mu_1^2 S^2} & \cdots & \frac{\mu_S^2}{\mu_1^2 S^2} \\
\frac{\mu_1^2}{\mu_2^2 S^2} & \left(1-\frac{1}{S}\right)^2 & \frac{\mu_3^2}{\mu_2^2 S^2} & \cdots & \frac{\mu_S^2}{\mu_2^2 S^2} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{\mu_1^2}{\mu_S^2 S^2} & \frac{\mu_2^2}{\mu_S^2 S^2} & \frac{\mu_3^2}{\mu_S^2 S^2} & \cdots & \left(1-\frac{1}{S}\right)^2
\end{bmatrix}
\begin{bmatrix}\sigma_1^2 \\ \sigma_2^2 \\ \vdots \\ \sigma_S^2\end{bmatrix}
=
\begin{bmatrix}\tau_1^2 - \frac{1}{\mu_1^2 S}\tau_{\text{pool}}^2 \\ \tau_2^2 - \frac{1}{\mu_2^2 S}\tau_{\text{pool}}^2 \\ \vdots \\ \tau_S^2 - \frac{1}{\mu_S^2 S}\tau_{\text{pool}}^2\end{bmatrix}$$

Privacy Analysis in Asymmetric Setting. We present an analysis of privacy for the aforementioned scheme in asymmetric setting. Recall that the adversary can observe $\hat{\mathbf{a}} = [\hat{a}_1, \ldots, \hat{a}_{S_H}]^\top \in \mathbb{R}^{S_H}$ and $\hat{e} = \sum_{s \in \mathbb{S}_H} \hat{e}_s$. In other words, the vector $\mathbf{y} = \left[\hat{\mathbf{a}}^\top, \hat{e}\right]^\top \in \mathbb{R}^{S_H+1}$ is what the adversary can observe to make inference about the non-colluding sites. To prove differential privacy guarantee, we must show that

$$\left| \log \frac{g(\mathbf{y}|\mathbf{a})}{g(\mathbf{y}|\mathbf{a}')} \right| \leq \epsilon$$

holds with probability (over the randomness of the mechanism) at least $1 - \delta$. Here, $\mathbf{a} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{S_H})]^\top$ and $g(\cdot|\mathbf{a})$ and $g(\cdot|\mathbf{a}')$ are the probability density functions of $\mathbf{y}$ under $\mathbf{a}$ and $\mathbf{a}'$, respectively. The vectors $\mathbf{a}$ and $\mathbf{a}'$ differ in only one coordinate (neighboring). Without loss of generality, we assume that $\mathbf{a}$ and $\mathbf{a}'$ differ in the first coordinate. We note that the maximum difference is $\frac{1}{N_s}$ as the sensitivity of the function $f(\mathbf{x_s})$ is $\frac{1}{N_s}$. Recall that we release $\hat{a}_s = f(\mathbf{x}_s) + e_s + g_s$ from each site. We observe

$$\mathbb{E}(\hat{a}_s) = f(\mathbf{x}_s),\ \mathrm{var}(\hat{a}_s) = \tau_s^2,\ \forall s \in [S]$$
$$\mathbb{E}(\hat{a}_{s_1}\hat{a}_{s_2}) = f(\mathbf{x}_{s_1})f(\mathbf{x}_{s_2}) - \frac{\mu_{s_1}\sigma_{s_1}^2}{\mu_{s_2}S} - \frac{\mu_{s_2}\sigma_{s_2}^2}{\mu_{s_1}S} + \frac{1}{\mu_{s_1}\mu_{s_2}S^2}\sum_{i=1}^{S}\mu_i^2\sigma_i^2,\ \forall s_1 \neq s_2 \in [S].$$

Without loss of generality, we can assume [48] that $\mathbf{a} = \mathbf{0}$ and $\mathbf{a}' = \mathbf{a} - \mathbf{v}$, where $\mathbf{v} = \left[\frac{1}{N_s}, 0, \ldots, 0\right]^\top$. That is, the random variable $\hat{\mathbf{a}}$ is $\mathcal{N}(\mathbf{0}, \Sigma_{\hat{\mathbf{a}}})$, where

$$\Sigma_{\hat{\mathbf{a}}} = \begin{bmatrix} \tau_1^2 & \cdots & -\frac{1}{S}\left(\frac{\mu_1\sigma_1^2}{\mu_S} + \frac{\mu_S\sigma_S^2}{\mu_1}\right) + \frac{\sum_{i=1}^{S}\mu_i^2\sigma_i^2}{\mu_1\mu_S S^2} \\ -\frac{1}{S}\left(\frac{\mu_1\sigma_1^2}{\mu_2} + \frac{\mu_2\sigma_2^2}{\mu_1}\right) + \frac{\sum_{i=1}^{S}\mu_i^2\sigma_i^2}{\mu_1\mu_2 S^2} & \cdots & -\frac{1}{S}\left(\frac{\mu_2\sigma_2^2}{\mu_S} + \frac{\mu_S\sigma_S^2}{\mu_2}\right) + \frac{\sum_{i=1}^{S}\mu_i^2\sigma_i^2}{\mu_2\mu_S S^2} \\ \vdots & \ddots & \vdots \\ -\frac{1}{S}\left(\frac{\mu_1\sigma_1^2}{\mu_S} + \frac{\mu_S\sigma_S^2}{\mu_1}\right) + \frac{\sum_{i=1}^{S}\mu_i^2\sigma_i^2}{\mu_1\mu_S S^2} & \cdots & \tau_{S_H}^2 \end{bmatrix}.$$

Additionally, the random variable $\hat{e}$ is $\mathcal{N}(0, \tau_{\hat{e}}^2)$, where $\tau_{\hat{e}}^2 = \sum_{s=1}^{S_H}\sigma_s^2$. Therefore, $g(\mathbf{y}|\mathbf{a})$

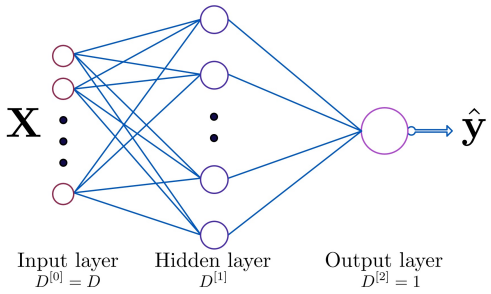

Figure 3.4: Structure of our neural network.

is the density of $\mathcal{N}(\mathbf{0}, \Sigma)$, where

$$\Sigma = \begin{bmatrix} \Sigma_{\hat{\mathbf{a}}} & \Sigma_{\hat{\mathbf{a}}\hat{e}} \\ \Sigma_{\hat{\mathbf{a}}\hat{e}}^{\top} & \tau_{\hat{e}}^2 \end{bmatrix} \in \mathbb{R}^{(S_H+1)\times(S_H+1)}.$$

With some simple algebra, we can find the expression for $\Sigma_{\hat{\mathbf{a}}\hat{e}}$ as:

$$\Sigma_{\hat{\mathbf{a}}\hat{e}} = \begin{bmatrix} \sigma_1^2 - \frac{1}{\mu_1 S}\sum_{i=1}^{S_H} \mu_i^2\sigma_i^2 \\ \sigma_2^2 - \frac{1}{\mu_2 S}\sum_{i=1}^{S_H} \mu_i^2\sigma_i^2 \\ \vdots \\ \sigma_{S_H}^2 - \frac{1}{\mu_{S_H} S}\sum_{i=1}^{S_H} \mu_i^2\sigma_i^2 \end{bmatrix}.$$

The rest of the proof proceeds as the proof of Theorem 3.1. Note that, due to the complex nature of the expression of Σ, we do not have a closed form solution for μ_z and σ_z. However, we can numerically evaluate their values and consequently, the resulting δ guarantee.

3.4 Experimental Results

We empirically show the effectiveness of the proposed CAPE algorithm with applications in a neural network based classifier in decentralized settings. Our CAPE algorithm

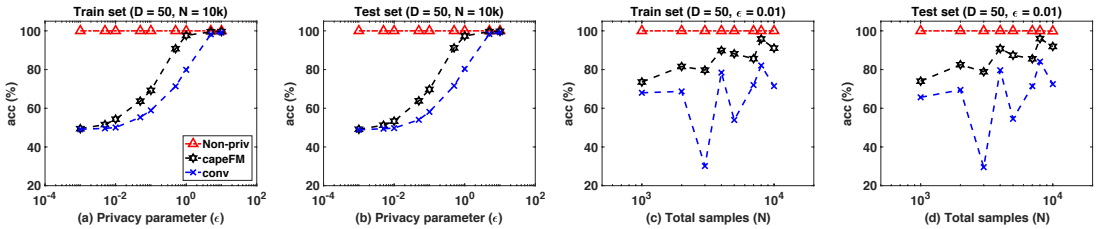

Figure 3.5: Variation of performance of NN based classifier on synthetic data: (a)–(b) with ϵ per iteration; (c)–(d) with N. Fixed parameter: $S = 4$.

can improve a decentralized differentially private computation if the target function has sensitivity satisfying the conditions of Proposition 3.2. Note that, the proposed capeFM algorithm (Chapter 8) is well-suited for virtually any decentralized optimization problem, as we can compute the differentially private approximate $\hat{f}_D(\mathbf{w})$ of the loss function and then use any off-the-shelf optimizer. More specifically, if the loss function can be represented as (8.3), we can employ the capeFM algorithm for a much better utility than existing schemes (see Chapter 8 for details). However, finding such a representation may be challenging for neural networks. In such applications, our CAPE scheme is still applicable, as long as the computation has sensitivity satisfying the conditions of Proposition 3.2. We, therefore, demonstrate the effectiveness of the CAPE algorithm for a neural network based classifier that employs a decentralized differentially private gradient descent [2]. We vary ϵ and total sample size and compare against the non-privacy-preserving pooled-data approach ($\mathsf{Non-priv.}$) and the conventional approach (conv) of decentralized differentially private gradient descent. For the neural network based classifier, we consider the symmetric setting (i.e., $N_s = \frac{N}{S}$ and $\tau_s = \tau$) and show the average performance over 10 independent runs.

We evaluate the performance of the proposed CAPE scheme on a neural network based classifier to classify between the two classes on a synthetic dataset. The samples from the two classes are random Gaussian vectors with unit variance and the means are separated by 1. We have the same decentralized setup as mentioned before: data samples are distributed across S sites. Let $\mathbf{X}_s \in \mathbb{R}^{D \times N_s}$ be the sample matrix at site s with N_s samples in D dimensions. Let $\mathbf{y}_s \in \{0,1\}^{N_s}$ contain the labels for each sample. The global sample matrix and labels vector are given as $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_S] \in \mathbb{R}^{D \times N}$

and $\mathbf{y} = \left[\mathbf{y}_1^\top, \ldots, \mathbf{y}_S^\top\right]^\top \in \mathbb{R}^N$, where $N = \sum_{s=1}^S N_s$. We consider an $L = 2$ layer neural network and $D = 50$, $N = 10k$, $S = 4$. The number of units in layer $l \in \{0, 1, \ldots, L\}$ is denoted by $D^{[l]}$ with $D^{[0]} = D$. We use Rectified Linear Unit (ReLU) activation in the hidden layer and sigmoid activation in the output layer. Our goal is to find the following parameters of the neural network: $\mathbf{W}^{[1]} \in \mathbb{R}^{D^{[1]} \times D^{[0]}}$, $\mathbf{b}^{[1]} \in \mathbb{R}^{D^{[1]}}$, $\mathbf{W}^{[2]} \in \mathbb{R}^{D^{[2]} \times D^{[1]}}$ and $\mathbf{b}^{[2]} \in \mathbb{R}^{D^{[2]}}$, such that $\texttt{round}\left(\hat{\mathbf{y}}\right)$ gives the labels of the samples with minimum deviation from $\mathbf{y}$, where $\hat{\mathbf{y}} = \texttt{sigmoid}\left(\mathbf{W}^{[2]}\mathbf{A}^{[1]} + \mathbf{b}^{[2]}\right)$ and $\mathbf{A}^{[1]} = \texttt{ReLU}\left(\mathbf{W}^{[1]}\mathbf{X} + \mathbf{b}^{[1]}\right)$. We use decentralized gradient descent to minimize the empirical average cross-entropy loss. The cross-entropy loss for the n-th sample $\mathbf{x}_n$ is:

$$l(\hat{y}_n, y_n) = -y_n \log \hat{y}_n - (1 - y_n) \log(1 - \hat{y}_n). \tag{3.6}$$

Each site evaluates the gradients on the local data and sends privacy-preserving approximates of the gradients to the aggregator according to a $\mathsf{Non-priv.}$, CAPE or conv schemes. The aggregator then combines the gradient contributions and updates the parameters. For CAPE, we set $\tau_s = \frac{\Delta}{\epsilon}\sqrt{2\log\frac{1.25}{0.01}}$ for all experiments, where Δ is the $\mathcal{L}_2$ sensitivity of the corresponding gradient. To measure the utility of the estimated parameters, we use the percent accuracy $\text{acc} = \frac{1}{N_{test}} \sum_{n=1}^{N_{test}} \mathcal{I}\left(\texttt{round}(\hat{\mathbf{y}}_{test,n}) = \mathbf{y}_{test,n}\right) \times 100$, where $\mathcal{I}(\cdot)$ is the indicator function and $\texttt{round}\left(\hat{\mathbf{y}}_{test}\right) \in \mathbb{R}^{N_{test}}$ contains the labels of the test-set samples.

Performance Variation with ϵ and N. We consider the privacy-utility tradeoff first. In Figure 3.5(a)–(b), we show the variation of acc with ϵ per iteration on the train and test sets for the synthetic dataset, while keeping N and S fixed. We observe that both of the privacy preserving algorithms: CAPE and conv, perform better as we increase ϵ. The proposed CAPE algorithm performs better than conv and reaches the performance of $\mathsf{Non-priv.}$ even for moderate ϵ values. Next, we consider the variation of acc with total sample size N, while keeping ϵ per iteration and S fixed. In Figure 3.5(c)–(d), we show the variation of acc on the train and test sets. As in the case of varying ϵ, we observe that increasing N (hence increasing N_s) improves acc for the privacy-preserving algorithms. We recall that the variance of the noise added for

privacy is inversely proportional to N^2 – therefore, availability of more samples results in better accuracy for the same privacy level. Again, we observe that the proposed CAPE algorithm outperforms the conv algorithm. However, when N is quite small, the performance gap between CAPE and conv is less pronounced. In practice, we observe that the gradient descent converges much faster under CAPE than the conv, matching the performance of $\mathsf{Non-priv}$. This can be explained by recalling the fact that the CAPE algorithm offers the benefit of much less additive noise at the aggregator. As the parameters are updated at the aggregator, the gradient descent converges faster in CAPE compared to the conv scheme.

Chapter 4

Improved Decentralized Differentially Private Principal Component Analysis

We propose an improved decentralized differentially private PCA algorithm that takes advantage of the CAPE protocol. We begin by introducing the problem of decentralized PCA. We then describe the proposed capePCA algorithm and present experimental results with real and synthetic data to compare against existing algorithms. We show that our new method outperforms previously proposed approaches, even under strong privacy constraints. For weaker privacy requirements, capePCA can achieve the same performance as a pooled-data scenario.

4.1 Decentralized Principal Component Analysis

We consider the same decentralized data model as in Section 3.1. We assume that the data is distributed in S sites, where each site $s \in [S]$ has a data matrix $\mathbf{X}_s \in \mathbb{R}^{D \times N_s}$. The data samples in the local sites are assumed to be disjoint. There is a central node that acts as an aggregator (see Figure 3.1). We denote $N = \sum_{s=1}^{S} N_s$ as the total number of samples over all sites. The data matrix

$$\mathbf{X}_s = [\mathbf{x}_{s,1} \ \ldots \ \mathbf{x}_{s,N_s}]$$

at site s is considered to contain the D-dimensional features of N_s individuals. Without loss of generality, we assume that $\|\mathbf{x}_{s,n}\|_2 \leq 1 \ \forall s \in [S]$ and $\forall n \in [N_s]$. If we had all the data in the aggregator (pooled data scenario), then the data matrix would be

$$\mathbf{X} = [\mathbf{X}_1 \ \ldots \ \mathbf{X}_S] \in \mathbb{R}^{D \times N}.$$

Our goal is to approximate the performance of the pooled data scenario using a decentralized differentially private algorithm.

We now formulate the problem of decentralized PCA. For simplicity, we assume that the observed samples are mean-centered. The $D \times D$ sample second-moment matrix at site s is

$$\mathbf{A}_s = \frac{1}{N_s}\mathbf{X}_s\mathbf{X}_s^\top.$$

In the pooled data scenario, the $D \times D$ positive semi-definite second-moment matrix is

$$\mathbf{A} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top.$$

According to the Schmidt approximation theorem [130], the rank-K matrix $\mathbf{A}_K$ that minimizes the difference $\|\mathbf{A} - \mathbf{A}_K\|_F$ can be found by taking the SVD of $\mathbf{A}$ as $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, where without loss of generality we assume $\mathbf{\Lambda}$ is a diagonal matrix with entries $\{\lambda_d(\mathbf{A})\}$ and $\lambda_1(\mathbf{A}) \geq \ldots \geq \lambda_D(\mathbf{A}) \geq 0$. Additionally, $\mathbf{V}$ is a matrix of eigenvectors corresponding to the eigenvalues. The top-K PCA subspace of $\mathbf{A}$ is the matrix

$$\mathbf{V}_K(\mathbf{A}) = [\mathbf{v}_1 \ldots \mathbf{v}_K].$$

Given $\mathbf{V}_K(\mathbf{A})$ and the eigenvalue matrix $\mathbf{\Lambda}$, we can form an approximation

$$\mathbf{A}_K = \mathbf{V}_K(\mathbf{A})\mathbf{\Lambda}_K\mathbf{V}_K(\mathbf{A})^\top$$

to $\mathbf{A}$, where $\mathbf{\Lambda}_K$ contains the K largest eigenvalues in $\mathbf{\Lambda}$. For a $D \times K$ matrix $\hat{\mathbf{V}}$ with orthonormal columns, the quality of $\hat{\mathbf{V}}$ in approximating $\mathbf{V}_K(\mathbf{A})$ can be measured by the *captured energy* of $\mathbf{A}$ as

$$q(\hat{\mathbf{V}}) = \mathrm{tr}(\hat{\mathbf{V}}^\top\mathbf{A}\hat{\mathbf{V}}). \tag{4.1}$$

The $\hat{\mathbf{V}}$, which maximizes $q(\hat{\mathbf{V}})$ is the subspace $\mathbf{V}_K(\mathbf{A})$. We are interested in approximating $\mathbf{V}_K(\mathbf{A})$ in a decentralized setting while guaranteeing differential privacy. One

Algorithm 4.1 Improved Decentralized Differentially Private PCA (capePCA)

Require: Data matrix $\mathbf{X}_s \in \mathbb{R}^{D \times N_s}$ with $\|\mathbf{x}_{s,n}\|_2 \leq 1$, local noise variances $\{\tau_s^2\}$ for $s \in [S]$; reduced dimension K

for $s = 1, 2, \ldots, S$ **do** ▷ at the local sites
  Generate $\mathbf{E}_s \in \mathbb{R}^{D \times D}$ using Algorithm 3.1 (element-wise) (Chapter 3)
  Compute $\mathbf{A}_s \leftarrow \frac{1}{N_s}\mathbf{X}_s\mathbf{X}_s^\top$
  Generate $D \times D$ symmetric $\mathbf{G}_s$, as described in the text
  Compute and send: $\hat{\mathbf{A}}_s \leftarrow \mathbf{A}_s + \mathbf{E}_s + \mathbf{G}_s$
end for
Compute $\hat{\mathbf{A}} \leftarrow \frac{1}{S}\sum_{s=1}^{S} \hat{\mathbf{A}}_s$ ▷ at the aggregator
Perform SVD: $\hat{\mathbf{A}} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$
Release / send to sites: $\mathbf{V}_K$
return $\mathbf{V}_K$

naïve approach (non-private) would be to send the data matrices from the sites to the aggregator. When D and/or N_s are large, this entails a huge communication overhead. In many scenarios the local data are also private or sensitive. As the aggregator is not trusted, sending the data to the aggregator can result in a significant privacy violation. Our goals are therefore to reduce the communication cost, ensure differential privacy, and provide a close approximation to the true PCA subspace $\mathbf{V}_K(\mathbf{A})$. We previously proposed a differentially private decentralized PCA scheme [78], but the performance of the scheme is limited by the larger variance of the additive noise at the local sites due to the smaller sample sizes. We intend to alleviate this problem using the correlated noise scheme shown in Chapter 3. The improved decentralized differentially private PCA algorithm (capePCA) we propose here achieves the same utility as the pooled data scenario in the symmetric setting.

4.2 Proposed capePCA Algorithm

We consider the same network structure as in Section 3.1. Recall that in the pooled data scenario, we have the data matrix $\mathbf{X}$ and the sample second-moment matrix $\mathbf{A} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$. We refer to the top-$K$ PCA subspace of this sample second-moment matrix as the true (or optimal) subspace $\mathbf{V}_K(\mathbf{A})$. At each site, we compute the sample second-moment matrix as $\mathbf{A}_s = \frac{1}{N_s}\mathbf{X}_s\mathbf{X}_s^\top$. The $\mathcal{L}_2$ sensitivity [50] of the function $f(\mathbf{X}_s) = \mathbf{A}_s$ is $\Delta_2^s = \frac{1}{N_s}$ [51]. In order to approximate $\mathbf{A}_s$ satisfying (ϵ, δ) differential privacy, we

can employ the AG algorithm [51] to compute $\hat{\mathbf{A}}_s = \mathbf{A}_s + \mathbf{G}_s$, where the symmetric matrix $\mathbf{G}_s$ is generated with entries i.i.d. $\sim \mathcal{N}(0, \tau_s^2)$ and $\tau_s = \frac{\Delta_2^s}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$. Note that, in the pooled data scenario, the $\mathcal{L}_2$ sensitivity of the function $f(\mathbf{X}) = \mathbf{A}$ is $\Delta_2^{\text{pool}} = \frac{1}{N}$. Therefore, the required additive noise standard deviation should satisfy $\tau_{\text{pool}} = \frac{\Delta_2^{\text{pool}}}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}} = \frac{\tau_s}{S}$, assuming equal number of samples in the sites. As we want the same utility as the pooled data scenario, we compute the following at each site s:

$$\hat{\mathbf{A}}_s = \mathbf{A}_s + \mathbf{E}_s + \mathbf{G}_s.$$

Here, the sites generate the $D \times D$ matrix $\mathbf{E}_s$ according to the scheme shown in Section 3.3. Additionally, the sites generate their own symmetric $D \times D$ matrix $\mathbf{G}_s$, where $[\mathbf{G}_s]_{ij}$ are drawn i.i.d. $\sim \mathcal{N}(0, \tau_g^2)$ and $\tau_g^2 = \frac{1}{S}\tau_s^2$ according to (3.2). Note that, these variance assignments can be readily modified to fit the unequal privacy/sample size scenario (see Section 3.3.7). However, for simplicity, we are considering the equal sample size scenario. Now, the sites send their $\hat{\mathbf{A}}_s$ to the aggregator and the aggregator computes

$$\begin{aligned}
\hat{\mathbf{A}} &= \frac{1}{S}\sum_{s=1}^{S}\hat{\mathbf{A}}_s \\
&= \frac{1}{S}\sum_{s=1}^{S}(\mathbf{A}_s + \mathbf{G}_s) + \frac{1}{S}\sum_{s=1}^{S}\mathbf{E}_s \\
&= \frac{1}{S}\sum_{s=1}^{S}(\mathbf{A}_s + \mathbf{G}_s),
\end{aligned}$$

where we used the relation $\sum_{s=1}^{S}\mathbf{E}_s = 0$. Note that at the aggregator, we achieve an estimator with noise variance exactly the same as that of the pooled data scenario (by Lemma 3.1). Next, we perform SVD on $\hat{\mathbf{A}}$ and release the top-K eigenvector matrix $\mathbf{V}_K$, which is the (ϵ, δ) differentially private approximate to the true subspace $\mathbf{V}_K(\mathbf{A})$. To achieve the same utility level as the pooled data case, we send the full matrix $\hat{\mathbf{A}}_s$ from the sites to the aggregator instead of the partial square root of it [78]. This increases the communication cost by $SD(D-R)$, where R is the intermediate dimension of the partial square root. We consider this as the cost of performance gain. Note that

capePCA can be readily extended to incorporate unequal privacy requirements/samples sizes at each site (shown in Section 3.3.7).

Theorem 4.1 (Privacy of capePCA Algorithm)**.** *Consider Algorithm 4.1 in the decentralized data setting of Section 3.1 with* $N_s = \frac{N}{S}$, $\tau_s = \tau$ *for all sites* $s \in [S]$. *Suppose that at most* $S_C = \left\lceil \frac{S}{3} \right\rceil - 1$ *sites can collude after execution. Then Algorithm 4.1 computes an* (ϵ, δ)*-differentially private approximation to the optimal subspace* $\mathbf{V}_K(\mathbf{A})$, *where* (ϵ, δ) *satisfy the relation* $\delta = 2\frac{\sigma_z}{\epsilon - \mu_z}\phi\left(\frac{\epsilon - \mu_z}{\sigma_z}\right)$, $\phi(\cdot)$ *is the density for standard Normal random variable and* (μ_z, σ_z) *are given by* (3.3) *and* (3.4)*, respectively.*

Proof. The proof of Theorem 4.1 follows from using the Gaussian mechanism [50], the AG algorithm [51], the bound on $\|\mathbf{A}_s - \mathbf{A'}_s\|_2$ and the privacy of CAPE as in Theorem 3.1. The computation of $\hat{\mathbf{A}}_s$ at each site is (ϵ, δ) differentially private. As differential privacy is post-processing invariant, we can combine the noisy second-moment matrices $\hat{\mathbf{A}}_s$ at the aggregator. By the correlated noise generation at the random noise generator, the noise $\mathbf{E}_s$ cancels out. We perform the SVD on $\hat{\mathbf{A}}$ and release $\mathbf{V}_K$. The released subspace $\mathbf{V}_K$ is thus the (ϵ, δ) differentially private approximate to the true subspace $\mathbf{V}_K(\mathbf{A})$. □

4.2.1 Performance Gain with Correlated Noise

The decentralized differentially private PCA algorithm of [78] essentially employs the conventional averaging (when each site sends the full $\hat{\mathbf{A}}_s$ to the aggregator). Therefore, the gain in performance of the proposed capePCA algorithm over the one in [78] is the same as shown in Proposition 3.1.

4.2.2 Theoretical Performance Guarantee

Due to the application of the correlated noise protocol, we achieve the same level of noise at the aggregator in the decentralized setting as we would have in the pooled data scenario. In essence, the proposed capePCA algorithm can achieve the same performance as the AG algorithm [51] modified to account for all the samples across all the sites. Here, we present three guarantees for the captured energy, closeness to the

true subspace and low-rank approximation. The guarantees are adopted from Dwork et al. [51] and modified to fit our symmetric setup and notation. Let us assume that the (ϵ, δ) differentially private subspace output from Algorithm 4.1 and the true subspace are denoted by $\hat{\mathbf{V}}_K$ and $\mathbf{V}_K$, respectively. We denote the singular values of $\mathbf{X}$ with $\sigma_1 \geq \ldots \geq \sigma_D$ and the un-normalized second-moment matrix with $\mathbf{A} = \mathbf{X}\mathbf{X}^\top$. Let $\mathbf{A}_K$ and $\hat{\mathbf{A}}_K$ be the true and the (ϵ, δ) differentially private rank-K approximates to $\mathbf{A}$, respectively. If we assume that the gap $\sigma_K^2 - \sigma_{K+1}^2 = \omega(\tau_{\text{pool}}\sqrt{D})$, then the following holds

- $\text{tr}\left(\hat{\mathbf{V}}_K^\top \mathbf{A} \hat{\mathbf{V}}_K\right) \geq \text{tr}\left(\mathbf{V}_K^\top \mathbf{A} \mathbf{V}_K\right) - O(\tau_{\text{pool}} K \sqrt{D})$
- $\left\| \mathbf{V}_K \mathbf{V}_K^\top - \hat{\mathbf{V}}_K \hat{\mathbf{V}}_K^\top \right\|_2 = O\left(\frac{\tau_{\text{pool}}\sqrt{D}}{\sigma_K^2 - \sigma_{K+1}^2}\right)$
- $\|\mathbf{A} - \hat{\mathbf{A}}_K\|_2 \leq \|\mathbf{A} - \mathbf{A}_K\|_2 + O(\tau_{\text{pool}}\sqrt{D})$.

The detailed proofs can be found in Dwork et al. [51].

4.2.3 Communication Cost

We quantify the total communication cost associated with the proposed capePCA algorithm. Recall that capePCA is an one-shot algorithm. For generating the zero-sum noise terms, we encounter $O(S+D^2)$ communication complexity of the sites and $O(S^2+SD^2)$ communication complexity of the aggregator [22]. Each site uses these to compute the noisy estimate of the local second-moment matrix $(D \times D)$ and sends that back to the aggregator. Therefore, the total communication cost is $O(S + D^2)$ for the sites and $O(S^2+SD^2)$ for the aggregator. This is expected as we are computing the global $D \times D$ second-moment matrix in a decentralized setting before computing the PCA subspace.

4.3 Experimental Results

We empirically compared the proposed capePCA, the existing DPdisPCA [78] and non-private PCA on pooled data ($\text{non} - \text{dp pool}$). We also included the performance of differentially private PCA [51] on local data (local) (i.e. data of a single site) and the conventional approach (conv) (i.e. without correlated noise). We designed the

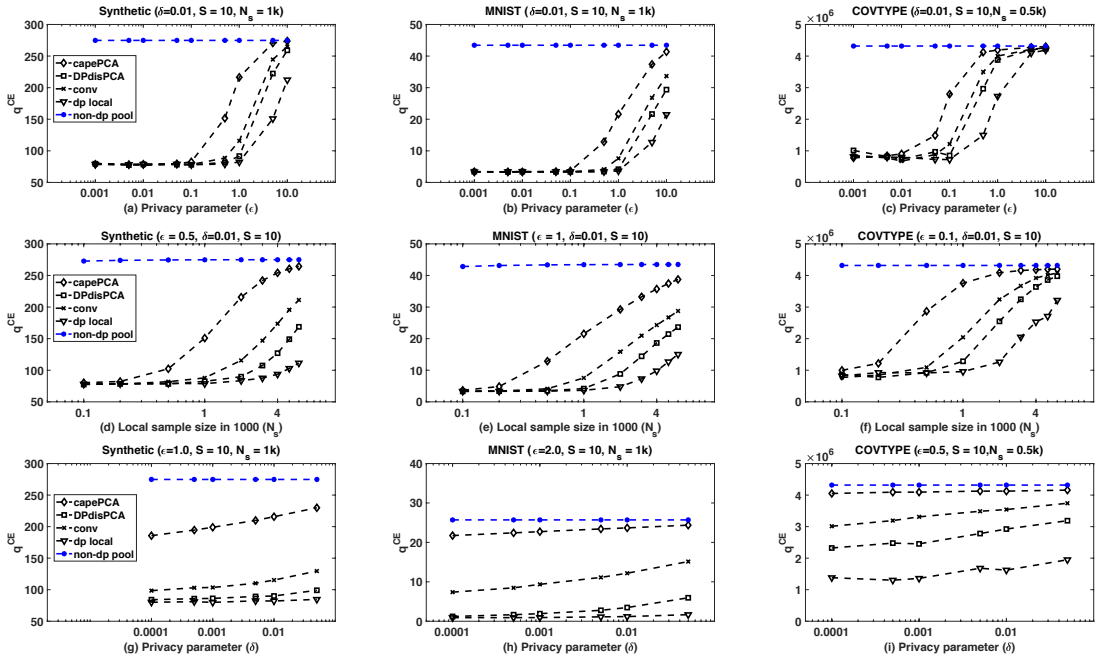

Figure 4.1: Variation of performance in distributed PCA for synthetic and real data: (a) - (c) with privacy parameter ϵ; (d) - (f) with sample size N_s and (g) - (i) with privacy parameter δ

experiments according to Imtiaz and Sarwate [78] using three datasets: a *synthetic* dataset ($D = 200$, $K = 50$) generated with zero mean and a pre-determined covariance matrix, the *MNIST* dataset ($D = 784$, $K = 50$) [91] (MNIST) and the *Covertype* dataset ($D = 54$, $K = 10$) [96] (COVTYPE). The MNIST consists of handwritten digits and has a training set of 60000 samples, each of size 28×28 pixels. The COVTYPE contains the forest cover types for 30×30 m^2 cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. We collected the dataset from the UC Irvine KDD archive [96]. For our experiments, we randomly selected 60000 samples from the COVTYPE. We preprocessed the data by subtracting the mean (centering) and normalizing the samples with the maximum $\mathcal{L}_2$ norm in each dataset to enforce the condition $\|\mathbf{x}_n\|_2 \leq 1 \quad \forall n$. We note that this preprocessing step is not differentially private, although it can be modified to satisfy differential-privacy at the cost of some utility. In all cases we show the average performance over 10 runs of the algorithms. As a performance measure of the produced subspace from the algorithm, we choose the captured energy: $q^{\text{CE}} = \text{tr}(\hat{\mathbf{V}}^\top \mathbf{A} \hat{\mathbf{V}})$, where $\hat{\mathbf{V}}$ is the subspace estimated

by an algorithm and $\mathbf{A}$ is the true second-moment matrix of the data. Note that, we can approximate the the captured energy in the true subspace as $\text{tr}(\mathbf{V}_K(\mathbf{A})^\top \mathbf{A}\mathbf{V}_K(\mathbf{A}))$, where $\mathbf{A}$ is achieved from the pooled-data sample second-moment matrix and $\mathbf{V}_K(\mathbf{A})$ is achieved from the non-private PCA.

4.3.1 Dependence on privacy parameter ϵ

First, we explore the trade-off between privacy and utility; i.e., between ϵ and q^{CE}. We note that the standard deviation of the added noise is inversely proportional to ϵ – bigger ϵ means higher privacy risk but less noise and thus, better utility. In Figure 4.1(a)-(c), we show the variation of q^{CE} of different algorithms for different values of ϵ. For this experiment, we kept the parameters δ, N_s and S fixed. For all the datasets, we observe that as ϵ increases (higher privacy risk), the captured energy increases. The proposed capePCA reaches the optimal utility ($\mathsf{non-dp\ pool}$) for some parameter choices and clearly outperforms the existing DPdisPCA, the conv, and the local algorithms. One of the reasons that capePCA outperforms conv is the smaller noise variance at the aggregator, as described before. Moreover, capePCA outperforms DPdisPCA because DPdisPCA suffers from a larger variance at the aggregator due to computation of the partial square root of $\hat{\mathbf{A}}_s$ [78]. However, DPdisPCA offers a much smaller communication overhead than capePCA. Achieving better performance than local is intuitive because including the information from multiple sites should always result in better estimates of population parameters than using the data from a single site only. An interesting observation is that for datasets with lower dimensional samples, we can use smaller ϵ (i.e., lower privacy risk) for the same utility.

4.3.2 Dependence on number of samples N_s

Next, we investigate the variation in performance with sample size N_s. Intuitively, it should be easier to guarantee smaller privacy risk ϵ and higher utility q^{CE}, when the number of samples is large. Figures 4.1(d)-(f) show how q^{CE} increases as a function of N_s. The variation with N_s reinforces the results seen earlier with variation of ϵ. For a fixed ϵ and δ, the utility increases as we increase N_s. For sufficiently large N_s, the

captured energy will reach that of $\mathsf{non-dp\ pool}$. Again, we observe a sharper increase in utility for lower-dimensional dataset.

4.3.3 Dependence on privacy parameter δ

Finally, we explore the variation of performance with the other privacy parameter δ. Recall that δ can be considered as the probability that the algorithm releases the private information without guaranteeing privacy. We, therefore, want this to be as small as possible. However, lower δ results in larger noise variance. In Figure 4.1(g)-(i), we show how q^{CE} vary with varying δ. We observe that if δ is not too small, the proposed algorithm can achieve very good utility, easily outperforming the other algorithms.

Chapter 5

Decentralized Differentially Private Orthogonal Tensor Decomposition

We propose an algorithm (capeAGN) for decentralized differentially private orthogonal tensor decomposition. The proposed algorithm takes advantage of the correlated noise scheme (Algorithm 3.2) [81]. To our knowledge, this is the first work on decentralized differentially private OTD. We begin by reviewing two applications of OTD [8] on centralized data, namely the single topic model (STM) and the mixture of Gaussian (MOG). We then describe the problem of differentially private OTD. We refer the reader to our previous work [79] for two differentially private OTD algorithms: AGN and AVN, for centralized settings. We employ the CAPE scheme for proposing the decentralized differentially private algorithm capeAGN. Finally, we present experimental results with synthetic data and compare the performance of our capeAGN with that of other algorithms. We show that capeAGN outperforms previously proposed approaches and can achieve the same performance as a pooled-data scenario under certain conditions.

5.1 Applications of Orthogonal Tensor Decomposition

We review two examples from Anandkumar et al. [8], which involve estimation of latent variables from observed samples. The lower-order moments obtained from the samples can be written as low-rank symmetric tensors.

5.1.1 Single Topic Model (STM)

Let us consider an exchangeable bag-of-words model [8] for documents. Such exchangeable models can be viewed as mixture models in which there is a latent variable h such

that the L words in the document $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_L$ are conditionally i.i.d. given h. Additionally, the conditional distributions are identical at all the nodes [8]. Let us assume that h is the only topic of a given document, and it can take only K distinct values. Let D be the number of distinct words in the vocabulary, and $L \geq 3$ be the number of words in each document. The generative process for a document is as follows: the document's topic is drawn according to the discrete distribution specified by the probability vector $\mathbf{w} = [w_1, w_2, \ldots, w_K]^\top$. This is modeled as a discrete random variable h such that $\Pr[h = k] = w_k$, for $k \in [K]$. Given the topic h, the document's L words are drawn independently according to the discrete distribution specified by the probability vector $\mathbf{a}_h \in \mathbb{R}^D$. We represent the L words in the document by D-dimensional random vectors $\mathbf{t}_l \in \mathbb{R}^D$. Specifically, if the l-th word is d, we set $\mathbf{t}_l = \mathbf{e}_d$ for $l \in [L]$, where $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_D$ are the standard coordinate basis vectors for $\mathbb{R}^D$. We observe that for any topic k

$$\begin{aligned}
\mathbb{E}[\mathbf{t}_1 \otimes \mathbf{t}_2 | h = k] &= \sum_{i,j} \Pr[\mathbf{t}_1 = i, \mathbf{t}_2 = j | h = k] \, \mathbf{e}_i \otimes \mathbf{e}_j \\
&= \mathbb{E}[\mathbf{t}_1 | h = k] \otimes \mathbb{E}[\mathbf{t}_2 | h = k] \\
&= \mathbf{a}_k \otimes \mathbf{a}_k.
\end{aligned}$$

Now, we can define two moments in terms of the outer products of the probability vectors $\mathbf{a}_k$ and the distribution of the topics h as

$$\mathbf{M}_2 = \sum_{k=1}^{K} w_k \mathbf{a}_k \otimes \mathbf{a}_k, \;\; \mathcal{M}_3 = \sum_{k=1}^{K} w_k \mathbf{a}_k \otimes \mathbf{a}_k \otimes \mathbf{a}_k. \tag{5.1}$$

The method proposed in [8] to recover $\mathbf{w}$ and $\{\mathbf{a}_k\}$ proceeds as follows: we observe N documents. Each of the documents has number of words $L \geq 3$. The way we record what we observe is: we form an $D \times D \times D$ tensor whose (d_1, d_2, d_3)-th entry is the proportion of times we see a document with first word d_1, second word d_2 and third word d_3. In this setting, we can estimate the moments $\mathbf{M}_2$ and $\mathcal{M}_3$, defined in (5.1), from the observed data as: $\mathbf{M}_2 = \mathbb{E}[\mathbf{t}_1 \otimes \mathbf{t}_2]$ and $\mathcal{M}_3 = \mathbb{E}[\mathbf{t}_1 \otimes \mathbf{t}_2 \otimes \mathbf{t}_3]$. We then need to perform orthogonal tensor decomposition on $\mathcal{M}_3$ to recover $\mathbf{w}$ and $\{\mathbf{a}_k\}$.

5.1.2 Mixture of Gaussians (MOG)

A similar example as the single topic model is the spherical mixture of Gaussians [8]. Let us assume that there are K components and the component mean vectors are given by the set $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_K\} \subset \mathbb{R}^D$. The probability of choosing component k is w_k. We assume that the common covariance matrix is $\sigma^2\mathbf{I}_D$. However, the model can be extended to incorporate different covariance matrices for different component as well [8, 73]. The n-th observation of the model can be written as $\mathbf{t}_n = \mathbf{a}_h + \mathbf{z}$, where h is a discrete random variable with $\Pr[h = k] = w_k$ and $\mathbf{z}$ is an D-dimensional random vector, independent from h, drawn according to $\mathcal{N}(0, \sigma^2\mathbf{I}_D)$. Let us denote the total number of observations by N. Without loss of generality, we assume that $\|\mathbf{a}_k\|_2 \leq 1$. Now, for $D \geq K$, it has been shown [73] that if we have estimates of the second and third order moments from the observations $\mathbf{t}_n$ as

$$\mathbf{M}_2 = \mathbb{E}[\mathbf{t} \otimes \mathbf{t}] - \sigma^2\mathbf{I}_D, \text{ and}$$
$$\mathcal{M}_3 = \mathbb{E}[\mathbf{t} \otimes \mathbf{t} \otimes \mathbf{t}] - \sigma^2 \sum_{d=1}^{D} \left(\mathbb{E}[\mathbf{t}] \otimes \mathbf{e}_d \otimes \mathbf{e}_d + \mathbf{e}_d \otimes \mathbb{E}[\mathbf{t}] \otimes \mathbf{e}_d + \mathbf{e}_d \otimes \mathbf{e}_d \otimes \mathbb{E}[\mathbf{t}]\right),$$

then these moments are decomposable as:

$$\mathbf{M}_2 = \sum_{k=1}^{K} w_k \mathbf{a}_k \otimes \mathbf{a}_k \text{ and}$$
$$\mathcal{M}_3 = \sum_{k=1}^{K} w_k \mathbf{a}_k \otimes \mathbf{a}_k \otimes \mathbf{a}_k.$$

5.1.3 Orthogonal Decomposition of $\mathcal{M}_3$

For both the STM and the MOG models, in order to decompose $\mathcal{M}_3$ using the tensor power method (2.2), we need the $\mathbf{a}_k$'s to be orthogonal to each other. But, in general, they are not. To employ the orthogonal tensor decomposition, we can project the tensor onto some subspace $\mathbf{W} \in \mathbb{R}^{D \times K}$ to ensure $\mathbf{W}^\top \mathbf{a}_k$'s are orthogonal to each other. We

note that, according to the multi-linear notation (see Section 2.2.2), we have

$$\mathcal{M}_3(\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3) = \sum_{k=1}^{K} w_k \left(\mathbf{V}_1^\top \mathbf{a}_k\right) \otimes \left(\mathbf{V}_2^\top \mathbf{a}_k\right) \otimes \left(\mathbf{V}_3^\top \mathbf{a}_k\right).$$

In order to find $\mathbf{W}$, we can compute the SVD(K) on the second-order moment $\mathbf{M}_2 \in \mathbb{R}^{D\times D}$ as $\mathbf{M}_2 = \mathbf{U}\mathbf{D}\mathbf{U}^\top$, where $\mathbf{U} \in \mathbb{R}^{D\times K}$ and $\mathbf{D} \in \mathbb{R}^{K\times K}$. We define $\mathbf{W} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}} \in \mathbb{R}^{D\times K}$ and then compute the projection $\tilde{\mathcal{M}}_3 = \mathcal{M}_3(\mathbf{W}, \mathbf{W}, \mathbf{W})$. We note that $\tilde{\mathcal{M}}_3 \in \mathbb{R}^{K\times K\times K}$ is now orthogonally decomposable. We use the tensor power iteration (2.2) on $\tilde{\mathcal{M}}_3$ to recover the weights $\{w_k\}$ and the component vectors $\{\mathbf{a}_k\}$. The detail of the tensor power method can be found in Anandkumar et al. [8].

5.2 Differentially Private OTD

We note that the key step in the orthogonal tensor decomposition algorithm is the mapping given by (2.2). In order to ensure differential privacy for the orthogonal decomposition, we may either add noise at each iteration step scaled to the $\mathcal{L}_2$ sensitivity [51] of the operation given by (2.2) or we can add noise to the tensor $\mathcal{X}$ itself just once. Adding noise in each iteration step might result in a poor utility/accuracy of the recovered eigenvectors and eigenvalues. We intend to add noise to the tensor itself prior to employing the tensor power method. In the following, we are showing the sensitivity calculations for the pooled data scenario. Extension to the decentralized case is straightforward (replacing N with N_s).

First, we focus on the exchangeable single topic model setup that we described in Section 5.1.1. We observe and record N documents. Let us consider two sets of documents, which differ in only one sample (e.g., the last one). Let the empirical second-order moment matrices be $\mathbf{M}_2$ and $\mathbf{M}_2'$ and the third-order moment tensors be $\mathcal{M}_3$ and $\mathcal{M}_3'$, respectively, for these two sets. We consider the two tensors, $\mathcal{M}_3$ and

$\mathcal{M}'_3$, as *neighboring*. We observe that

$$\begin{aligned}
\mathbf{M}_2 &= \frac{1}{N}\sum_{n=1}^{N} \mathbf{t}_{1,n}\mathbf{t}_{2,n}^\top \\
&= \frac{1}{N}\sum_{n=1}^{N-1} \mathbf{t}_{1,n}\mathbf{t}_{2,n}^\top + \frac{1}{N}\mathbf{t}_{1,N}\mathbf{t}_{2,N}^\top, \\
\mathbf{M}'_2 &= \frac{1}{N}\sum_{n=1}^{N-1} \mathbf{t}_{1,n}\mathbf{t}_{2,n}^\top + \frac{1}{N}\mathbf{t}'_{1,N}\mathbf{t}'^\top_{2,N},
\end{aligned}$$

where $\mathbf{t}_{l,n}$ denotes the l-th word of the n-th document. Similarly, we observe

$$\begin{aligned}
\mathcal{M}_3 &= \frac{1}{N}\sum_{d=1}^{D} \mathbf{t}_{1,n} \otimes \mathbf{t}_{2,n} \otimes \mathbf{t}_{3,n} \\
&= \frac{1}{N}\sum_{n=1}^{N-1} \mathbf{t}_{1,n} \otimes \mathbf{t}_{2,n} \otimes \mathbf{t}_{3,n} + \frac{1}{N}\mathbf{t}_{1,N} \otimes \mathbf{t}_{2,N} \otimes \mathbf{t}_{3,N}, \\
\mathcal{M}'_3 &= \frac{1}{N}\sum_{n=1}^{N-1} \mathbf{t}_{1,n} \otimes \mathbf{t}_{2,n} \otimes \mathbf{t}_{3,n} + \frac{1}{N}\mathbf{t}'_{1,N} \otimes \mathbf{t}'_{2,N} \otimes \mathbf{t}'_{3,N}.
\end{aligned}$$

As mentioned before, we perform the SVD on $\mathbf{M}_2$ first to compute $\mathbf{W}$. We intend to use the AG algorithm [51] to make this operation differentially private. We look at the quantity:

$$\begin{aligned}
\|\mathbf{M}_2 - \mathbf{M}'_2\|_2 &= \frac{1}{N}\|\mathbf{t}_{1,N}\mathbf{t}_{2,N}^\top - \mathbf{t}'_{1,N}\mathbf{t}'^\top_{2,N}\|_2 \\
&= \frac{1}{N}\sup_{\|\mathbf{u}\|_2,\|\mathbf{v}\|_2=1} \left\{\mathbf{u}^\top\left(\mathbf{t}_{1,N}\mathbf{t}_{2,N}^\top - \mathbf{t}'_{1,N}\mathbf{t}'^\top_{2,N}\right)\mathbf{v}\right\} \\
&\leq \frac{\sqrt{2}}{N} = \Delta_{2,S},
\end{aligned}$$

because of the encoding $\mathbf{t}_{l,n} = \mathbf{e}_d$. For the mixture of Gaussians model, we note that we assumed $\|\mathbf{a}_k\|_2 \leq 1$ for all $k \in \{1, 2, \ldots, K\}$. To find a bound on $\|\mathbf{M}_2 - \mathbf{M}'_2\|_2$, we consider the following: for identifiability of the $\{\mathbf{a}_k\}$, we have to assume that the $\mathbf{a}_k$'s are linearly independent. In other words, we are interested in finding the directions of the components specified by $\{\mathbf{a}_k\}$. In that sense, while obtaining the samples, we can divide the samples by a constant ζ such that $\|\mathbf{t}_n\|_2 \leq 1$ is satisfied. From the resulting second- and third-order moments, we will be able to recover $\{\mathbf{a}_k\}$ up to a scale factor.

It is easy to show using the definition of largest eigenvalue of a symmetric matrix [26] that

$$\begin{aligned}\|\mathbf{M}_2 - \mathbf{M'}_2\|_2 &= \frac{1}{N} \sup_{\|\mathbf{u}\|_2=1} \left\{ \mathbf{u}^\top \left(\mathbf{t}_N \mathbf{t}_N^\top - \mathbf{t'}_N \mathbf{t'}_N^\top \right) \mathbf{u} \right\} \\ &= \frac{1}{N} \sup_{\|\mathbf{u}\|_2=1} \left\{ \left| \mathbf{u}^\top \mathbf{t}_N \right|^2 - \left| \mathbf{u}^\top \mathbf{t'}_N \right|^2 \right\} \\ &\leq \frac{1}{N} = \Delta_{2,M},\end{aligned}$$

where the inequality follows from the relation $\|\mathbf{t}_n\|_2 \leq 1$. We note that the largest singular value of a matrix is the square root of the largest eigenvalue of that matrix. For the decentralized case, as mentioned before, the sensitivity of $\mathbf{M}_2^s$ depends only on the local sample size. We can therefore use the AG algorithm [51] (i.e., adding Gaussian noise with variance scaled to $\Delta_{2,S}$ or $\Delta_{2,S}$ to $\mathbf{M}_2$) to make the computation of $\mathbf{W}$ (ϵ_1, δ_1)-differentially private. Next, we focus on the tensor $\mathcal{M}_3$. We need to project $\mathcal{M}_3$ on $\mathbf{W}$ before using the tensor power method. We can choose between making the projection operation differentially private, or we can make the $\mathcal{M}_3$ itself differentially private before projection. We found that making the projection operation differentially private involves addition of a large amount of noise and more importantly, the variance of the noise to be added depends on the alphabet size (or feature dimension) D and the singular values of $\mathbf{M}_2$. Therefore, we choose to make the tensor itself differentially private. We are interested to find the sensitivity of the tensor valued function $f(\mathcal{M}_3) = \mathcal{M}_3$, which is simply the identity map. That is, we need to find the maximum quantity that this function can change if we replace the argument $\mathcal{M}_3$ with a neighboring $\mathcal{M'}_3$. For our exchangeable single topic model setup, we have

$$\begin{aligned}\|\mathcal{M}_3 - \mathcal{M'}_3\| &= \left\| \frac{1}{N} \mathbf{t}_{1,N} \otimes \mathbf{t}_{2,N} \otimes \mathbf{t}_{3,N} - \frac{1}{N} \mathbf{t'}_{1,N} \otimes \mathbf{t'}_{2,N} \otimes \mathbf{t'}_{3,N} \right\| \\ &\leq \frac{\sqrt{2}}{N} = \Delta_{3,S},\end{aligned}$$

because only one entry in the $D \times D \times D$ tensor $\mathbf{t}_{1,N} \otimes \mathbf{t}_{2,N} \otimes \mathbf{t}_{3,N}$ is non-zero (in fact,

the only non-zero entry is 1). Now, for the mixture of Gaussians model, we define

$$\mathcal{T} = \sigma^2 \sum_{d=1}^{D} (\mathbb{E}[\mathbf{t}] \otimes \mathbf{e}_d \otimes \mathbf{e}_d + \mathbf{e}_d \otimes \mathbb{E}[\mathbf{t}] \otimes \mathbf{e}_d + \mathbf{e}_d \otimes \mathbf{e}_d \otimes \mathbb{E}[\mathbf{t}])$$

Therefore, we have

$$\begin{aligned}
\mathcal{T} - \mathcal{T}' = \\
&\frac{\sigma^2}{N} \sum_{d=1}^{D} \Big(\big(\mathbf{t}_N - \mathbf{t}'_N\big) \otimes \mathbf{e}_d \otimes \mathbf{e}_d + \mathbf{e}_d \otimes \big(\mathbf{t}_N - \mathbf{t}'_N\big) \otimes \mathbf{e}_d + \mathbf{e}_d \otimes \mathbf{e}_d \otimes \big(\mathbf{t}_N - \mathbf{t}'_N\big) \Big) \\
&\implies \|\mathcal{T} - \mathcal{T}'\| \leq \frac{3D\sigma^2}{N} \|\mathbf{t}_N - \mathbf{t}'_N\|_2 \leq \frac{6D\sigma^2}{N},
\end{aligned}$$

where the last inequality follows from $\|\mathbf{t}_n\|_2 \leq 1$. We have

$$\begin{aligned}
\|\mathcal{M}_3 - \mathcal{M}'_3\| &= \Big\| \frac{1}{N} \mathbf{t}_N \otimes \mathbf{t}_N \otimes \mathbf{t}_N - \frac{1}{N} \mathbf{t'}_N \otimes \mathbf{t'}_N \otimes \mathbf{t'}_N + \mathcal{T} - \mathcal{T}' \Big\| \\
&\leq \frac{2}{N} + \frac{6D\sigma^2}{N} = \Delta_{3,M},
\end{aligned}$$

because $\|\mathbf{t}_N \otimes \mathbf{t}_N \otimes \mathbf{t}_N\| = 1$ in our setup. Again, we note that in the decentralized setting, the sensitivity of the local $\mathcal{M}_3^s$ depends only on the local sample size. We refer the reader to our previous work [79] where we proposed two algorithms for centralized differentially private OTD. The first one uses a symmetric tensor made with i.i.d. entries from a Gaussian distribution, while the second proposed method uses a symmetric tensor made with entries taken from a sample vector drawn from an appropriate distribution. Both of the algorithms guarantee (ϵ, δ)-differential privacy.

5.3 Proposed capeAGN Algorithm

We assume the same decentralized data setting as in Section 3.1: each site $s \in [S]$ has a data matrix $\mathbf{X}_s \in \mathbb{R}^{D \times N_s}$. The data samples in the local sites are assumed to be disjoint. Each site s computes the sample second-order moment matrix $\mathbf{M}_2^s \in \mathbb{R}^{D \times D}$ and the third-order moment tensor $\mathcal{M}_3^s \in \mathbb{R}^{D \times D \times D}$ using the local data samples. There is a central node that acts as an aggregator (see Figure 3.1). We denote $N = \sum_{s=1}^{S} N_s$

Algorithm 5.1 Decentralized Differentially Private OTD (capeAGN)

Require: Sample second-order moment matrices $\mathbf{M}_2^s \in \mathbb{R}^{D\times D}$ and third-order moment tensors $\mathcal{M}_3^s \in \mathbb{R}^{D\times D\times D}$ $\forall s \in [S]$, local noise variances $\{\tau_2^s, \tau_3^s\}$, reduced dimension K

for $s = 1, \ldots, S$ **do** ▷ at the local sites
  Generate $\mathbf{E}_2^s \in \mathbb{R}^{D\times D}$, as described in the text
  Generate $\mathbf{G}_2^s \in \mathbb{R}^{D\times D}$, as described in the text
  Compute and send $\hat{\mathbf{M}}_2^s \leftarrow \mathbf{M}_2^s + \mathbf{E}_2^s + \mathbf{G}_2^s$
end for
Compute $\hat{\mathbf{M}}_2 \leftarrow \frac{1}{S}\sum_{s=1}^{S} \hat{\mathbf{M}}_2^s$ and then SVD(K) of $\hat{\mathbf{M}}_2$ as $\hat{\mathbf{M}}_2 = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ ▷ at the aggregator
Compute and send to sites: $\mathbf{W} \leftarrow \mathbf{U}\mathbf{D}^{-\frac{1}{2}}$
for $s = 1, \ldots, S$ **do** ▷ at the local sites
  Generate $\mathcal{E}_3^s \in \mathbb{R}^{D\times D\times D}$, as described in the text
  Generate symmetric $\mathcal{G}_3^s \in \mathbb{R}^{D\times D\times D}$ from the entries of $\mathbf{b} \in \mathbb{R}^{D_{\text{sym}}}$, where $[\mathbf{b}]_d \sim \mathcal{N}(0, \tau_{3g}^2)$ and $\tau_{3g}^2 = \frac{1}{S}\tau_3^{s2}$
  Compute $\hat{\mathcal{M}}_3^s \leftarrow \mathcal{M}_3^s + \mathcal{E}_3^s + \mathcal{G}_3^s$ and $\tilde{\mathcal{M}}_3^s \leftarrow \hat{\mathcal{M}}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W})$
  Send $\tilde{\mathcal{M}}_3^s$ to aggregator
end for
Compute $\tilde{\mathcal{M}}_3 \leftarrow \frac{1}{S}\sum_{s=1}^{S} \tilde{\mathcal{M}}_3^s$ ▷ at the aggregator
return The differentially private orthogonally decomposable tensor $\tilde{\mathcal{M}}_3$, projection subspace $\mathbf{W}$

as the total number of samples over all sites. We refer the reader to Section 2.2.2 and the survey by Kolda and Bader [87] for related basic definitions.

As mentioned before, we consider the decomposition of symmetric tensors that appear in several latent variable models. Such tensors can be orthogonally decomposed efficiently. We start with recalling that the orthogonal decomposition of a 3-rd order symmetric tensor $\mathcal{X} \in \mathbb{R}^{D\times D\times D}$ is a collection of orthonormal vectors $\{\mathbf{v}_k\}$ together with corresponding positive scalars $\{\lambda_k\}$ such that $\mathcal{X} = \sum_{k=1}^{K} \lambda_k \mathbf{v}_k \otimes \mathbf{v}_k \otimes \mathbf{v}_k$. A unit vector $\mathbf{u} \in \mathbb{R}^D$ is an **eigenvector** of $\mathcal{X}$ with corresponding **eigenvalue** λ if $\mathcal{X}(\mathbf{I}, \mathbf{u}, \mathbf{u}) = \lambda\mathbf{u}$, where $\mathbf{I}$ is the $D \times D$ identity matrix [8]. To see this, one can observe

$$\begin{aligned}\mathcal{X}(\mathbf{I}, \mathbf{u}, \mathbf{u}) &= \sum_{k=1}^{K} \lambda_k \left(\mathbf{I}^\top \mathbf{v}_k\right) \otimes \left(\mathbf{u}^\top \mathbf{v}_k\right) \otimes \left(\mathbf{u}^\top \mathbf{v}_k\right) \\ &= \sum_{k=1}^{K} \lambda_k \left(\mathbf{u}^\top \mathbf{v}_k\right)^2 \mathbf{v}_k.\end{aligned}$$

By the orthogonality of the $\mathbf{v}_k$, it is clear that $\mathcal{X}(\mathbf{I}, \mathbf{v}_k, \mathbf{v}_k) = \lambda_k \mathbf{v}_k$ $\forall k$. Now, the

orthogonal tensor decomposition proposed in [8] is based on the mapping

$$\mathbf{u} \mapsto \frac{\mathcal{X}(\mathbf{I}, \mathbf{u}, \mathbf{u})}{\|\mathcal{X}(\mathbf{I}, \mathbf{u}, \mathbf{u})\|_2}, \tag{5.2}$$

which can be considered as the tensor equivalent of the well-known matrix power method. Obviously, all tensors are not orthogonally decomposable. As the tensor power method requires the eigenvectors $\{\mathbf{v}_k\}$ to be orthonormal, we need to perform *whitening* - that is, projecting the tensor on a subspace such that the eigenvectors become orthogonal to each other.

We note that the proposed algorithm applies to both of the STM and MOG problems. As the correlated noise scheme only works with Gaussian noise, the proposed capeAGN employs the AGN algorithm [79] at its core. In-line with our setup in Section 5.2, the sample second-order moment matrix and the third-order moment tensor at site s are denoted as $\mathbf{M}_2^s \in \mathbb{R}^{D\times D}$ and $\mathcal{M}_3^s \in \mathbb{R}^{D\times D\times D}$, respectively. The noise standard deviation required for computing the (ϵ_1, δ_1) differentially private approximate to $\mathbf{M}_2^s$ is given by

$$\tau_2^s = \frac{\Delta_2^s}{\epsilon_1}\sqrt{2\log\left(\frac{1.25}{\delta_1}\right)}, \tag{5.3}$$

where the sensitivity Δ_2^s is inversely proportional to the sample size N_s and (ϵ_1, δ_1) pair corresponds to the noise variance τ_2^s. To be more specific, we can write $\Delta_{2,S}^s = \frac{\sqrt{2}}{N_s}$ for STM and $\Delta_{2,M}^s = \frac{1}{N_s}$ for MOG. The detailed derivation of the sensitivity of $\mathbf{M}_2^s$ for both STM and MOG are shown in Section 5.2. Additionally, at site s, the noise standard deviation required for computing the (ϵ_2, δ_2) differentially private approximate to $\mathcal{M}_3^s$ is given by

$$\tau_3^s = \frac{\Delta_3^s}{\epsilon_2}\sqrt{2\log\left(\frac{1.25}{\delta_2}\right)}. \tag{5.4}$$

Again, we can write $\Delta_{3,S}^s = \frac{\sqrt{2}}{N_s}$ for STM and $\Delta_{3,M}^s = \frac{2}{N_s} + \frac{6D\sigma^2}{N_s}$ for MOG. Section 5.2 contains the detailed algebra for calculating the sensitivity of $\mathcal{M}_3^s$ for both STM and MOG. We note that, as in the case of $\mathbf{M}_2^s$, the sensitivity depends only on the

sample size N_s. Now, in the pooled-data scenario, the noise standard deviations would be given by:

$$\tau_2^{\text{pool}} = \frac{\Delta_2^{\text{pool}}}{\epsilon_1}\sqrt{2\log\left(\frac{1.25}{\delta_1}\right)}$$
$$\tau_3^{\text{pool}} = \frac{\Delta_3^{\text{pool}}}{\epsilon_2}\sqrt{2\log\left(\frac{1.25}{\delta_2}\right)},$$

where $\Delta_2^{\text{pool}} = \frac{\Delta_2^s}{S}$ and $\Delta_3^{\text{pool}} = \frac{\Delta_3^s}{S}$, assuming equal number of samples in the sites. We need to compute the $D \times K$ whitening matrix $\mathbf{W}$ and the $D \times D \times D$ tensor $\hat{\mathcal{M}}_3$ in a decentralized way while satisfying differential privacy. Although we could employ our previous differentially private decentralized PCA algorithm [78] to compute $\mathbf{W}$, to achieve the same level of accuracy as the pooled data scenario we compute the following matrix at site s:

$$\hat{\mathbf{M}}_2^s = \mathbf{M}_2^s + \mathbf{E}_2^s + \mathbf{G}_2^s,$$

where $\mathbf{E}_2^s \in \mathbb{R}^{D\times D}$ is generated following to Algorithm 3.1 element-wise, according to Section 3.3. Additionally, $\mathbf{G}_2^s \in \mathbb{R}^{D\times D}$ is a symmetric matrix generated at site s where $\{[\mathbf{G}_2^s]_{ij} : i \in [D], j \leq i\}$ are drawn i.i.d. $\sim \mathcal{N}(0, \tau_{2g}^2)$, $[\mathbf{G}_2^s]_{ij} = [\mathbf{G}_2^s]_{ji}$ and $\tau_{2g}^2 = \frac{1}{S}\tau_2^{s2}$. The aggregator computes

$$\hat{\mathbf{M}}_2 = \frac{1}{S}\sum_{s=1}^{S}\hat{\mathbf{M}}_2^s = \frac{1}{S}\sum_{s=1}^{S}\left(\mathbf{M}_2^s + \mathbf{G}_2^s\right),$$

where we used the relation $\sum_{s=1}^{S}\mathbf{E}_2^s = 0$. Note that the variance of the additive noise in $\hat{\mathbf{M}}_2$ is exactly the same as the pooled data scenario (please see Lemma 3.1). At the aggregator, we can then compute the SVD(K) of $\hat{\mathbf{M}}_2$ as $\hat{\mathbf{M}}_2 = \mathbf{U}\mathbf{D}\mathbf{U}^\top$. We compute the matrix $\mathbf{W} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}}$ and send it to the sites.

Next, we focus on computing $\hat{\mathcal{M}}_3$ in the decentralized setting. For this purpose, we can follow the same steps as computing $\hat{\mathbf{M}}_2$. Each site generates $\mathcal{E}_3^s \in \mathbb{R}^{D\times D\times D}$ following to Algorithm 3.1 element-wise, according to Section 3.3. Then, each site generates their own $\mathcal{G}_3^s \in \mathbb{R}^{D\times D\times D}$ in the following way: site s draws a vector $\mathbf{b} \in \mathbb{R}^{D_{\text{sym}}}$

with $D_{\text{sym}} = \binom{D+2}{3}$ and entries i.i.d. $\sim \mathcal{N}(0, \tau_{3g}^2)$, where $\tau_{3g}^2 = \frac{1}{S}\tau_3^{s\,2}$. The tensor $\mathcal{G}_3^s$ is generated with the entries from $\mathbf{b}$ such that $\mathcal{G}_3^s$ is symmetric. Again, for both $\hat{\mathbf{M}}_2^s$ and $\hat{\mathcal{M}}_3^s$, we are considering the equal sample size scenario for simplicity. Our framework requires only a small modification to incorporate the unequal privacy/sample size (see Section 3.3.7). Now, site s computes

$$\hat{\mathcal{M}}_3^s = \mathcal{M}_3^s + \mathcal{E}_3^s + \mathcal{G}_3^s \text{ and } \tilde{\mathcal{M}}_3^s = \hat{\mathcal{M}}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}).$$

We note that $\tilde{\mathcal{M}}_3^s$ is a $K \times K \times K$ dimensional tensor. Each site sends this to the aggregator. This saves a lot of communication overhead as typically $K \ll D$. To see how this results in the same estimate of $\tilde{\mathcal{M}}_3$ as the pooled data scenario, we observe

$$\begin{aligned}\tilde{\mathcal{M}}_3^s &= \hat{\mathcal{M}}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}) \\ &= \mathcal{M}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}) + \mathcal{E}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}) + \mathcal{G}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}).\end{aligned}$$

Additionally, at the aggregator, we compute

$$\begin{aligned}\tilde{\mathcal{M}}_3 &= \frac{1}{S}\sum_{s=1}^{S}\Big(\mathcal{M}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}) + \mathcal{E}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}) + \mathcal{G}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W})\Big) \\ &= \frac{1}{S}\sum_{s=1}^{S}\Big(\mathcal{M}_3^s\Big(\mathbf{W}, \mathbf{W}, \mathbf{W}\Big) + \mathcal{G}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W})\Big) + \left(\frac{1}{S}\sum_{s=1}^{S}\mathcal{E}_3^s\right)(\mathbf{W}, \mathbf{W}, \mathbf{W}) \\ &= \frac{1}{S}\sum_{s=1}^{S}\left(\mathcal{M}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}) + \mathcal{G}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W})\right) \\ &= \left(\frac{1}{S}\sum_{s=1}^{S}\mathcal{M}_3^s + \mathcal{G}_3^s\right)(\mathbf{W}, \mathbf{W}, \mathbf{W}).\end{aligned}$$

We used the associativity of the multi-linear operation [8] and the relation $\sum_{s=1}^{S}\mathcal{E}_3^s = 0$. Note that the $\tilde{\mathcal{M}}_3$ we achieve in this scheme is exactly the same $\tilde{\mathcal{M}}_3$ we would have achieved if all the data samples were present in the aggregator. Moreover, this is also the quantity that the aggregator would get if the sites send the full $\hat{\mathcal{M}}_3^s$ to the aggregator instead of $\tilde{\mathcal{M}}_3^s$. The complete capeAGN algorithm is shown in Algorithm 5.1. Note that, capeAGN can be readily extended to incorporate unequal privacy requirements/samples

sizes at each site (shown in Section 3.3.7).

Theorem 5.1 (Privacy of capeAGN Algorithm)**.** *Consider Algorithm 5.1 in the decentralized data setting of Section 3.1 with* $N_s = \frac{N}{S}$, $\tau_2^s = \tau_2$ *and* $\tau_3^s = \tau_3$ *for all sites* $s \in [S]$. *Suppose that at most* $S_C = \lceil \frac{S}{3} \rceil - 1$ *sites can collude after execution. Then Algorithm 5.1 computes an* $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$*-DP orthogonally decomposable tensor* $\tilde{\mathcal{M}}_3$, *where* $\delta_1 + \delta_2 = 2\frac{\sigma_z}{\epsilon_1 + \epsilon_2 - \mu_z}\phi\left(\frac{\epsilon_1 + \epsilon_2 - \mu_z}{\sigma_z}\right)$, $\phi(\cdot)$ *is the density for standard Normal random variable and* (μ_z, σ_z) *are given by* (3.3) *and* (3.4)*, respectively. Additionally, the computation of the projection subspace* $\mathbf{W}$ *is* (ϵ_1, δ_1) *differentially private.*

Proof. The proof of Theorem 5.1 follows from using the Gaussian mechanism [50], the sensitivities of $\mathbf{M}_2^s$ and $\mathcal{M}_3^s$ and the privacy of CAPE algorithm as in Theorem 3.1. The release of $\hat{\mathbf{M}}_2^s$ from each site s is (ϵ_1, δ_1) differentially private, where ϵ_1 and δ_1 satisfy the relation $\delta_1 = 2\frac{\sigma_z}{\epsilon_1 - \mu_z}\phi\left(\frac{\epsilon_1 - \mu_z}{\sigma_z}\right)$. As differential privacy is closed under post-processing and the samples in each site are disjoint, the computation of $\mathbf{W}$ at the aggregator also satisfies (ϵ_1, δ_1) differential privacy. We can similarly show by the composition theorem [50], the computation $\tilde{\mathcal{M}}_3^s = \hat{\mathcal{M}}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W})$ at each site is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ differentially private. By the post-processing invariability, the computation of $\tilde{\mathcal{M}}_3$ at the aggregator is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ differentially private. □

5.3.1 Performance Gain with Correlated Noise

As we mentioned before, this is the first work that proposes an algorithm for decentralized differentially private OTD. As we employ the CAPE scheme for our computations, the gain in the performance over a conventional decentralized differentially private OTD is therefore the same as in the case of decentralized differentially private averaging, as described in Proposition 3.1.

5.3.2 Theoretical Performance Guarantee

Although our proposed capeAGN algorithm can reach the performance of the pooled data scenario (that is, same as that of the AGN algorithm with all data samples from all sites stored in the aggregator), it is hard to characterize how the estimated $\{\hat{\mathbf{a}}_k\}$ and

$\{\hat{w}_k\}$ would deviate from the true $\{\mathbf{a}_k\}$ and $\{w_k\}$, respectively. We note that although we are adding symmetric noise to the third-order moment tensor, an orthogonal decomposition need not exist for the perturbed tensor, even though the perturbed tensor is symmetric [8, 86]. Anandkumar et al. [8] provided a bound on the error of the recovered decomposition in terms of the operator norm of the tensor perturbation. For our proposed algorithm, the perturbation includes the effect of estimating the third-order moment tensor from the samples as well as the noise added for differential-privacy. Even without accounting for the error in estimating the moments from observable samples, the operator norm of the effective noise at the aggregator: $\|\mathcal{G}\|_{\text{op}} = \frac{1}{S}\left\|\sum_{s=1}^{S}\mathcal{G}_3^s\right\|_{\text{op}}$, is a random quantity, and requires new measure concentration results to analyze. Relating these bounds to the error in estimating recovering the $\{\mathbf{a}_k\}$ and $\{w_k\}$ is nontrivial. However, very recently Esmaeili and Huang [56] proposed differentially private OTD-based Latent Dirichlet Allocation (LDA) for topic modeling. The authors consider the sensitivities of different functions at different points in the flow of the LDA algorithm and propose to employ Gaussian mechanism [50] to the point with the smallest sensitivity, conditioned on some constraints. This enables the DP-LDA algorithm to achieve better utility bounds. The extension of the techniques introduced in [56] to STM and MOG is nontrivial and we defer that for future work.

5.3.3 Communication Cost

We note that capeAGN is a two-step algorithm: it computes the projection subspace $\mathbf{W} \in \mathbb{R}^{D\times K}$ and then the orthogonally decomposable tensor $\tilde{\mathcal{M}}_3$. We need to two zero-sum noise terms: $\mathbf{E}_2^s \in \mathbb{R}^{D\times D}$ and $\mathcal{E}_3^s \in \mathbb{R}^{D\times D\times D}$. For this, we encounter $O(S+D^2)$ and $O(S+D^3)$ communication complexity of the sites, respectively and $O(S^2+SD^2)$ and $O(S^2+SD^3)$ communication complexity of the aggregator [22], respectively. Therefore, the total communication cost is $O(S+D^3)$ for the sites and $O(S^2+SD^3)$ for the aggregator. This is expected as we are computing the global $D\times D\times D$ third-order moment tensor in a decentralized setting.

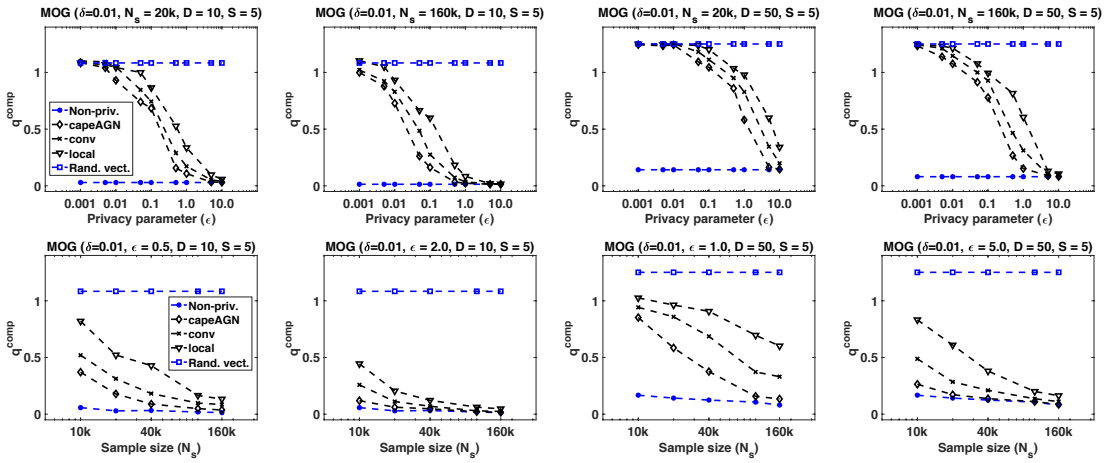

Figure 5.1: Variation of performance in the MOG setup: top-row – with privacy parameter ϵ; bottom-row – with sample size N_s

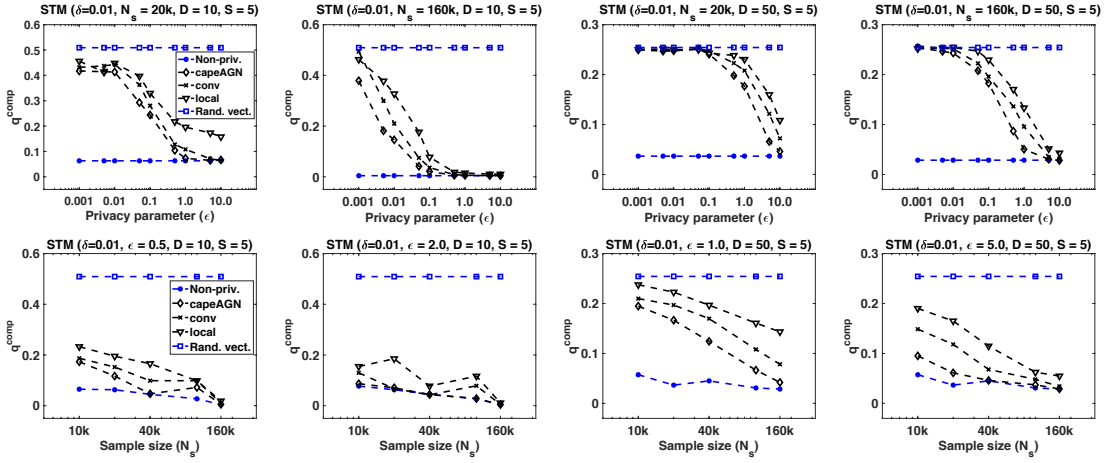

Figure 5.2: Variation of performance in the STM setup: top-row – with privacy parameter ϵ; bottom-row – with sample size N_s

5.4 Experimental Results

For the capeAGN algorithm, we focus on measuring how well the output of the proposed algorithm approximate the true components $\{\mathbf{a}_k\}$ and $\{w_k\}$. Let the recovered component vectors be $\{\hat{\mathbf{a}}_k\}$. We use the same error metric as our previous work [79], q^{comp}, to capture the disparity between $\{\mathbf{a}_k\}$ and $\{\hat{\mathbf{a}}_k\}$:

$$q^{\text{comp}} = \frac{1}{K}\sum_{k=1}^{K} ED^k_{\min}, \text{ where } ED^k_{\min} = \min_{k' \in [K]} \|\hat{\mathbf{a}}_k - \mathbf{a}_{k'}\|_2.$$

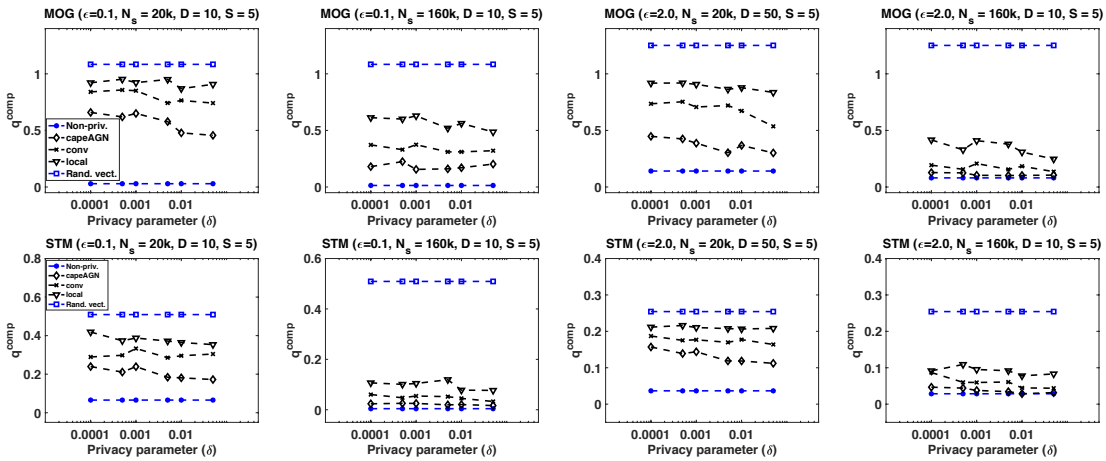

Figure 5.3: Variation of performance with privacy parameter δ: top-row – in MOG setup; bottom-row – in STM setup

For comparison, we show the error resulting from the $\hat{\mathbf{a}}_k$'s achieved from the proposed capeAGN algorithm, a conventional (but never proposed anywhere to the best of our knowledge) distributed differentially private OTD algorithm that does not employ correlated noise (conv), a differentially private OTD [79] on local data (local) and the non-private tensor power method [8] on pooled data ($\mathsf{Non-priv.}$). We also show the error considering random vectors as $\hat{\mathbf{a}}_k$'s (Rand. vect.). The reason [79] to show (Rand. vect.) is the following: this error corresponds to the worst possible results, as we are not taking any information from data into account to estimate $\hat{\mathbf{a}}_k$'s. As recovering the component vectors is closely related with recovering the selection probabilities $\{w_k\}$, we only show the error of recovering the component vectors. We studied the dependence of q^{comp} on the privacy parameters ϵ, δ and the sample size N_s. In all cases we show the average performance over 10 runs of each algorithm. We note that the capeAGN algorithm adds noise in two stages for ensuring differential-privacy: one for estimating $\mathbf{W}$ and another for estimating $\mathcal{M}_3$. We equally divided ϵ and δ to set ϵ_1, ϵ_2 and δ_1, δ_2 for the two stages. Optimal allocation of ϵ and δ in multi-stage differentially private algorithms is still an open question.

5.4.1 Performance variation in the MOG setup

First, we present the performance of the aforementioned algorithms in the setting of the mixture of Gaussians. We use two *synthetic data* sets of different feature dimensions ($D = 10$, $K = 5$ and $D = 50$, $K = 10$), where the common covariance is $\sigma^2 = 0.05$ and the components $\{\mathbf{a}_k\}$ satisfy $\|\mathbf{a}_k\|_2 \leq 1$.

Performance Variation with ϵ. We first explore the *privacy-utility tradeoff* between ϵ and q^{comp}. For the capeAGN algorithm, the variance of the noise is inversely proportional to ϵ^2 – smaller ϵ means more noise (lower utility) and lower privacy risk. In the top-row of Figure 5.1, we show the variation of q^{comp} with ϵ for a fixed $\delta = 0.01$ and $S = 5$ for two different feature dimensions, each with two different samples sizes. For both of the feature dimensions, we observe that as ϵ increases (higher privacy risk), the errors decrease and the proposed capeAGN algorithm outperforms the conv and local methods. capeAGN matches the performance of Non − priv. method for larger ϵ values. For a particular feature dimension, we notice that if we increase N_s, the performance of capeAGN gets even better. This is expected as the variance of the noise for capeAGN is inversely proportional to square of the sample size.

Performance Variation with N_s. Next, we consider the performance variation with N_s. Intuitively, it should be easier to guarantee a smaller privacy risk for the same ϵ and a higher utility (lower error) when the number of samples is large. In the bottom row of Figure 5.1, we show how the errors vary as a function of N_s for the MOG model for two different feature dimensions, while keeping $\delta = 0.01$ and $S = 5$ fixed. The variation with the sample size reinforces the results seen earlier with variation in ϵ: the proposed capeAGN outperforms the other algorithms under investigation for both $D = 10$ and $D = 50$. In general, capeAGN approaches the performance of Non − priv. as the sample size increases. When ϵ is large enough, the capeAGN algorithm achieves as much utility as Non − priv. method.

Performance Variation with δ. Finally, we show the variation of performance with the other privacy parameter δ. Recall that δ can be interpreted as the probability that the privacy-preserving algorithm releases the private information "out in the wild"

without any additive noise. Therefore, we want to ensure that δ is small. However, the smaller the δ is the larger the noise variance becomes. Thus smaller δ dictates loss in utility. We observe this in our experiments as well. In the top-row of Figure 5.3, we show the variation of q^{comp} with δ for two different feature dimensions and two different sample sizes. We kept $S = 5$ fixed. We observe that when ϵ is small, we need larger δ to achieve meaningful performance. This can be explained in the following way: for Gaussian mechanism, we need to ensure that the standard deviation of the noise σ satisfies $\sigma \geq \frac{\Delta}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$, where Δ is the $\mathcal{L}_2$ sensitivity of the function under consideration. This inequality can be satisfied with infinitely many (ϵ, δ) pairs and one can keep the noise level the same for a smaller ϵ with a larger δ. We observe from the figure that when both ϵ and N_s are larger, the proposed capeAGN can achieve very close performance to the non-private one even for very small δ values.

5.4.2 Performance variation in the STM setup

We performed experiments on two *synthetic* datasets of different feature dimensions ($D = 10$, $K = 5$ and $D = 50$, $K = 10$) generated with pre-determined $\mathbf{w}$ and $\{\mathbf{a}_k\}$. It should be noted here that the recovery of $\{\mathbf{a}_k\}$ is difficult [79], because the recovered word probabilities from the tensor decomposition, whether private or non-private, may not always be valid probability vectors (i.e., no negative entries and sum to 1). Therefore, prior to computing the q^{comp}, we ran a post-processing step (0-out negative entries and then normalize by summation) to ensure that the recovered vectors are valid probability vectors. This process is non-linear and potentially makes the recovery error worse. However, for practical STM, D is not likely to be 10 or 50, rather it may be of the order of thousands, simulating which is a huge computational burden. In general, if we want the same privacy level for higher dimensional data, we need to increase the sample size. We refer the reader to some efficient (but non-privacy-preserving) implementations [74] for tensor factorization.

Performance Variation with ϵ. As in the case of the MOG model, we first explore the *privacy-utility tradeoff* between ϵ and q^{comp}. In the top-row of Figure 5.2, we show the variation of q^{comp} with ϵ for a fixed $\delta = 0.01$ and $S = 5$ for two different feature

dimensions. For both of the feature dimensions, we observe that as ϵ increases (higher privacy risk), the errors decrease. The proposed capeAGN outperforms conv and local methods in all settings; and match the performance of $\mathsf{Non-priv.}$ for large enough ϵ. Increasing N_s makes the proposed algorithm perform even better.

Performance Variation with N_s. Next, in the bottom-row of Figure 5.2, we show how the errors vary as a function of N_s for two different feature dimensions, while keeping $\delta = 0.01$ and $S = 5$ fixed. The variation with N_s reiterates the results seen earlier. The proposed capeAGN outperforms all other algorithms (except the $\mathsf{Non-priv.}$) for both $D = 10$ and $D = 50$. For larger N_s, it achieves almost the same utility as the $\mathsf{Non-priv.}$ algorithm. Even for smaller ϵ with a proper sample size, the error is very low. For the $D = 10$ case, the capeAGN always performs very closely with the $\mathsf{Non-priv.}$ algorithm.

Performance Variation with δ. Lastly, we show the variation of q^{comp} with δ in the bottom-row of Figure 5.3. We observe similar performance trend as in the MOG setting. For smaller ϵ and sample size, we need to compensate with larger δ to achieve a performance closer to $\mathsf{Non-priv.}$ one. However, when sample size is larger, we can get away with a smaller ϵ and δ. This is intuitive as hiding one individual among a large group is easier – the additive noise variance need not be very large and hence the performance does not take a large hit.

Chapter 6

Decentralized Differentially Private Joint Independent Component Analysis

In this chapter, we first formulate the problem of ICA in the decentralized setting. We then propose a new algorithm, capeDJICA, for (ϵ, δ)-differentially private decentralized joint ICA. The algorithm significantly improves upon our earlier work [80] by taking advantage of a recently proposed correlation assisted private estimation (CAPE) protocol. We also analyze the privacy guarantees of the proposed algorithm using conventional approach [48] and the moments accountant [2]. Finally, we present experimental results to analyze the variation of utility with different privacy levels, number of samples and some other key parameters. The results show that our algorithm achieves very good utility on both synthetic and real datasets, while providing strong privacy guarantees.

6.1 The ICA Model

In this work we consider the generative ICA model as in [11]. In the centralized scenario, the independent sources $\mathbf{S} \in \mathbb{R}^{R \times N}$ are composed of N observations from R statistically independent components. We have a linear mixing process defined by a mixing matrix $\mathbf{A} \in \mathbb{R}^{D \times R}$, which forms the observed data $\mathbf{X} \in \mathbb{R}^{D \times N}$ as a product $\mathbf{X} = \mathbf{AS}$. Many ICA algorithms propose to recover the unmixing matrix $\mathbf{W} = \mathbf{A}^{-1}$, assuming $\mathbf{A}$ is invertible [11], by trying to maximize independence between rows of the product $\mathbf{WX}$. The maximal information transfer (infomax) [17] is a popular heuristic for estimating $\mathbf{W}$ that maximizes an entropy functional related to $\mathbf{WX}$. More specifically, the objective of Infomax ICA can be expressed as

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmax}}\, h(g(\mathbf{WX})). \tag{6.1}$$

Here, $g(\cdot)$ is the sigmoid function and is given by:

$$g(z) = \frac{1}{1 + \exp(-z)}.$$

Additionally, $h(\mathbf{z})$ is the differential entropy of a random vector $\mathbf{z}$ with joint density q:

$$h(\mathbf{z}) = -\int q(\mathbf{z}) \log q(\mathbf{z}) d\mathbf{z}.$$

Note that, we apply $g(\cdot)$ to a matrix $\mathbf{Z}$ element-wise, i.e., $g(\mathbf{Z})$ is a matrix with the same size as $\mathbf{Z}$ and $[g(\mathbf{Z})]_{ij} = g([\mathbf{Z}]_{ij})$.

The Decentralized Data Problem. We consider a similar decentralized-data model with S sites as Section 3.1 (see Figure 3.1). There is a central node that acts as an aggregator. We assume that all parties are "honest but curious". That is, all parties follow the protocol honestly but a subset can collude (maybe with an external adversary) to learn other site's data/function output. Each site s has a collection of data matrices $\{\mathbf{X}_{s,m} \in \mathbb{R}^{D \times N_t} : m = 1, \ldots, M_s\}$ consisting of a total time course of length N_t time points over D voxels for M_s individuals. The data samples in the local sites are assumed to be disjoint. Sites concatenate their local data matrices temporally to form a $D \times N_t M_s$ data matrix $\mathbf{X}_s \in \mathbb{R}^{D \times N_s}$, where $N_s = N_t M_s$. Let $N = \sum_{s=1}^{S} N_s$ be the total number of samples and $M = \sum_{s=1}^{S} M_s$ be the total number of individuals (across all sites). We assume a global mixing matrix $\mathbf{A} \in \mathbb{R}^{D \times R}$ generates the time courses in $\mathbf{X}_s$ from underlying sources $\mathbf{S}_s \in \mathbb{R}^{R \times N_s}$ at each site. This yields the following model:

$$\mathbf{X} = [\mathbf{A}\mathbf{S}_1 \ldots \mathbf{A}\mathbf{S}_S] = [\mathbf{X}_1 \ldots \mathbf{X}_S] \in \mathbb{R}^{D \times N}. \tag{6.2}$$

We are interested to compute the global unmixing matrix $\mathbf{W} \in \mathbb{R}^{R \times D}$ corresponding to the Moore-Penrose pseudo-inverse of $\mathbf{A}$, denoted $\mathbf{A}^+$, in the decentralized setting. As sharing raw data across sites is prohibited in most cases due to privacy concerns and communication cost, we employ differential privacy [50]. More specifically, our goal is to use differentially private estimates of the local gradients to compute the global unmixing matrix $\mathbf{W}$ such that it closely approximates the true global unmixing matrix.

Algorithm 6.1 Improved Differentially Private Decentralized Joint ICA (capeDJICA)

Require: data $\{\mathbf{X}_s^r \in \mathbb{R}^{R\times N_s} : s \in [S]\}$, tolerance level $t = 10^{-6}$, maximum iterations J, $\|\Delta_{\mathbf{W}}(0)\|_2^2 = t$, initial learning rate $\rho = 0.015/\log(R)$, local noise standard deviations $\{\tau_G^s, \tau_h^s\}$, gradient bounds $\{B_G, B_h\}$

Initialize $j = 0$, $\mathbf{W} \in \mathbb{R}^{R\times R}$ ▷ for example, $\mathbf{W} = \mathbf{I}$
while $j < J$, $\|\Delta_{\mathbf{W}}(j)\|_2^2 \geq t$ **do**
  for all sites $s = 1, 2, \ldots, S$ **do**
    Generate $\mathbf{E}_s^G \in \mathbb{R}^{R\times R}$ and $\mathbf{e}_s^h \in \mathbb{R}^R$ using Algorithm 3.1 (entrywise)
    Generate $\mathbf{K}_s^G \in \mathbb{R}^{R\times R}$ and $\mathbf{k}_s^h \in \mathbb{R}^R$, as described in the text
    Compute $\mathbf{Z}_s(j) = \mathbf{W}(j-1)\mathbf{X}_s + \mathbf{b}(j-1)\mathbf{1}^\top$
    Compute $\hat{\mathbf{Y}}_s(j) = \mathbf{1} - 2g(\mathbf{Z}_s(j))$
    Compute $\mathbf{G}_s(j) = \frac{1}{N_s}\sum_{n=1}^{N_s} \frac{\left(\mathbf{I}+\hat{\mathbf{y}}_{s,n}\mathbf{z}_{s,n}^\top\right)\mathbf{W}(j-1)}{\max\left(1, \frac{1}{B_G}\left\|\left(\mathbf{I}+\hat{\mathbf{y}}_{s,n}\mathbf{z}_{s,n}^\top\right)\mathbf{W}(j-1)\right\|_F\right)}$
    Compute $\mathbf{h}_s(j) = \frac{1}{N_s}\sum_{n=1}^{N_s} \frac{\hat{\mathbf{y}}_{s,n}}{\max\left(1, \frac{1}{B_h}\left\|\hat{\mathbf{y}}_{s,n}\right\|_2\right)}$
    Compute $\hat{\mathbf{h}}_s(j) = \mathbf{h}_s(j) + \mathbf{e}_s^h + \mathbf{k}_s^h$
    Send $\hat{\mathbf{G}}_s(j)$ and $\hat{\mathbf{h}}_s(j)$ to the aggregator
  end for
  Compute $\Delta_{\mathbf{W}}(j) = \rho\frac{1}{S}\sum_{s=1}^{S}\hat{\mathbf{G}}_s(j)$ ▷ at the aggregator, update global variables
  Compute $\Delta_{\mathbf{b}}(j) = \rho\frac{1}{S}\sum_{s=1}^{S}\hat{\mathbf{h}}_s(j)$
  Compute $\mathbf{W}(j) = \mathbf{W}(j-1) + \Delta_{\mathbf{W}}(j)$
  Compute $\mathbf{b}(j) = \mathbf{b}(j-1) + \Delta_{\mathbf{b}}(j)$
  Check upper bound and perform learning rate adjustment (if needed)
  Send global $\mathbf{W}(j)$ and $\mathbf{b}(j)$ back to each site
  $j \leftarrow j + 1$
end while
return The current $\mathbf{W}(j)$

6.2 Improved Differentially Private djICA

In this work, we focus on the djICA algorithm [11] for performing temporal ICA of fMRI data. Our goal is to compute the privacy-preserving global spatial maps utilizing all samples across all the sites. To that end, we need to perform a multi-round decentralized gradient descent for solving (6.1). One option is to employ the conventional DP gradient descent [16, 127]: computing the sensitivity [50] of the gradient and then adding noise scaled to the sensitivity. However, this would lead to a significantly noisier estimate of the spatial maps, as outlined in Section 3.2. For an efficient spatial map estimation that can achieve the same level of noise as the pooled-data scenario, we can employ the recently proposed CAPE scheme. CAPE can benefit a broad class of functions whose sensitivities satisfy some conditions [81]. Examples include the empirical average loss

functions used in machine learning and deep neural networks. In the following, we demonstrate that we can adapt the CAPE scheme to develope a DP djICA algorithm that offers significant improvements over our previous method [80].

Recall our decentralized data setup: there are S sites and a central aggregator (see Figure 3.1(b)). Each site s has data from M_s individuals, which are concatenated to form the data matrix $\mathbf{X}_s \in \mathbb{R}^{D\times N_s}$. The global mixing matrix $\mathbf{A} \in \mathbb{R}^{D\times R}$ is assumed to generate the time courses in $\mathbf{X}_s$ from underlying sources $\mathbf{S}_s \in \mathbb{R}^{R\times N_s}$ at each site. That is: $\mathbf{X} = [\mathbf{A}\mathbf{S}_1 \dots \mathbf{A}\mathbf{S}_S] \in \mathbb{R}^{D\times N}$. We estimate the DP global unmixing matrix $\mathbf{W} \in \mathbb{R}^{R\times D} \approx \mathbf{A}^+$ in the decentralized setting with a multi-round gradient descent that employs the CAPE protocol.

As mentioned before, we employ the capePCA algorithm [77] (Algorithm 4.1) to estimate an efficient and privacy-preserving row-rank subspace and thereby reduce the dimension of the samples before solving the optimization problem of (6.1). Let the output of capePCA to be $\mathbf{V}_R \in \mathbb{R}^{D\times R}$, which is sent to the sites from the aggregator. Then the reduced dimensional $(R \times N_s)$ data matrix at site s is denoted by: $\mathbf{X}_s^r = \mathbf{V}_R^\top \mathbf{X}_s$. These projected samples are the inputs to the proposed capeDJICA algorithm that estimates the unmixing matrix $\mathbf{W}$. Note that, even though the preprocessing is performed satisfying differential privacy, the maximization to solve (6.1) itself may leak information about the local data since it relies on iterative message-passing with the central aggregator. More specifically, the maximization is performed through a gradient descent [6], where the gradient is to be computed in a decentralized fashion. That is, the algorithm depends on the data samples through the gradients. Our proposed capeDJICA algorithm employs the CAPE protocol to perform the privacy-preserving decentralized gradient descent to solve for $\mathbf{W}$.

The gradient of the *empirical average* loss function with respect to $\mathbf{W}$ at site s is given [80, 11] by

$$\mathbf{G}_s = \frac{1}{N_s}\left(N_s\mathbf{I} + (\mathbf{1} - 2\mathbf{Y}_s)\,\mathbf{Z}_s^\top\right)\mathbf{W}, \tag{6.3}$$

where $\mathbf{Z}_s = \mathbf{W}\mathbf{X}_s^r + \mathbf{b}\mathbf{1}^\top$, $\mathbf{Y}_s = g(\mathbf{Z}_s)$; $\mathbf{b} \in \mathbb{R}^R$ is the bias and $\mathbf{1}$ is a vector of ones. If

we denote $\mathbf{1} - 2\mathbf{Y}_s$ with $\hat{\mathbf{Y}}_s$ then we have

$$\mathbf{G}_s = \frac{1}{N_s}\left(N_s\mathbf{I} + \hat{\mathbf{Y}}_s\mathbf{Z}_s^\top\right)\mathbf{W} = \frac{1}{N_s}\sum_{n=1}^{N_s}\left(\mathbf{I} + \hat{\mathbf{y}}_{s,n}\mathbf{z}_{s,n}^\top\right)\mathbf{W}.$$

We can consider $\left(\mathbf{I} + \hat{\mathbf{y}}_{s,n}\mathbf{z}_{s,n}^\top\right)\mathbf{W}$ to be the gradient of the loss function corresponding to a single data sample. Therefore, the gradient of the average loss function at site s is essentially the average of the gradient of the loss function corresponding to each sample. Note that this gradient estimate is needed to be sent to the aggregator from the site. Therefore, we need to approximate this gradient in a differentially private way. To that end, let us consider that the gradient due to each sample satisfies

$$\left\|\left(\mathbf{I} + \hat{\mathbf{y}}_{s,n}\mathbf{z}_{s,n}^\top\right)\mathbf{W}\right\|_F \leq B_G, \tag{6.4}$$

where B_G is some constant. It is easy to see that by changing one data sample (i.e., for a neighboring dataset), the gradient at site s can change by at most $\frac{2B_G}{N_s}$. Therefore, the $\mathcal{L}_2$ sensitivity of the function $f(\mathbf{X}_s) = \mathbf{G}_s$ is

$$\Delta_G^s = \frac{2B_G}{N_s}. \tag{6.5}$$

In addition to the unmixing matrix $\mathbf{W}$, we update a bias term $\mathbf{b}$ using a gradient descent [11]. The gradient of the empirical average loss function with respect to the bias at site s is given [11] by

$$\mathbf{h}_s = \frac{1}{N_s}\sum_{n=1}^{N_s}\hat{\mathbf{y}}_{s,n}. \tag{6.6}$$

Similar to the case of $\mathbf{G}_s$, we can find the $\mathcal{L}_2$ sensitivity of the function $f(\mathbf{X}_s) = \mathbf{h}_s$ as

$$\Delta_h^s = \frac{2B_h}{N_s}, \tag{6.7}$$

where

$$\|\hat{\mathbf{y}}_{s,n}\|_2 \leq B_h. \tag{6.8}$$

In order to approximate $\mathbf{G}_s$ and $\mathbf{h}_s$ satisfying (ϵ, δ) differential-privacy, we can employ Gaussian mechanism [50]: the standard deviations τ_G^s and τ_h^s of the additive noise for $\mathbf{G}_s$ and $\mathbf{h}_s$ should satisfy:

$$\tau_G^s = \frac{\Delta_G^s}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}},\ \tau_h^s = \frac{\Delta_h^s}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}}. \tag{6.9}$$

We employ the CAPE protocol to combine the gradients from the sites at the aggregator to achieve the same utility level as that of the pooled data scenario. More specifically, each site generates two noise terms: $\mathbf{E}_s^G \in \mathbb{R}^{R\times R}$ and $\mathbf{e}_s^h \in \mathbb{R}^R$, collectively among all sites (entrywise, according to Algorithm 3.1) at each iteration round. Additionally, each site s generates the following two noise terms locally at each iteration:

- $\mathbf{K}_s^G \in \mathbb{R}^{R\times R}$ with $[\mathbf{K}_s^G]_{ij}$ drawn i.i.d. $\sim \mathcal{N}(0, \tau_{Gk}^2)$ and $\tau_{Gk}^2 = \frac{1}{S}{\tau_G^s}^2$
- $\mathbf{k}_s^h \in \mathbb{R}^R$ with $[\mathbf{k}_s^h]_i$ drawn i.i.d. $\sim \mathcal{N}(0, \tau_{hk}^2)$ and $\tau_{hk}^2 = \frac{1}{S}\tau_h^{s2}$.

At each iteration round, the sites compute the noisy estimates of the gradients of $\mathbf{W}$ and $\mathbf{b}$:

$$\hat{\mathbf{G}}_s = \mathbf{G}_s + \mathbf{E}_s^G + \mathbf{K}_s^G,\ \hat{\mathbf{h}}_s = \mathbf{h}_s + \mathbf{e}_s^h + \mathbf{k}_s^h.$$

These two terms are then sent to the aggregator and the aggregator computes

$$\Delta_{\mathbf{W}} = \rho\frac{1}{S}\sum_{s=1}^{S}\hat{\mathbf{G}}_s,\ \Delta_{\mathbf{b}} = \rho\frac{1}{S}\sum_{s=1}^{S}\hat{\mathbf{h}}_s,$$

where ρ is the learning rate. These gradient estimates are then used to update the variables $\mathbf{W}$ and $\mathbf{b}$. The complete algorithm is shown in Algorithm 6.1. Note that, one does not need to explicitly find the bounds in (6.4) and (6.8). Instead, the gradients due to each sample can be clipped to some pre-determined B_G or B_h in $\mathcal{L}_2$ norm sense.

That is, we can replace $\mathbf{G}_{s,n} = \left(\mathbf{I} + \hat{\mathbf{y}}_{s,n}\mathbf{z}_{s,n}^{\top}\right)\mathbf{W}$ with

$$\mathbf{G}_{s,n} = \frac{\mathbf{G}_{s,n}}{\max\left(1, \frac{\|\mathbf{G}_{s,n}\|_F}{B_G}\right)}.$$

Similarly, we can replace $\mathbf{h}_{s,n} = \hat{\mathbf{y}}_{s,n}$ with

$$\mathbf{h}_{s,n} = \frac{\mathbf{h}_{s,n}}{\max\left(1, \frac{\|\mathbf{h}_{s,n}\|_2}{B_h}\right)}.$$

We note that this norm clipping has a few consequences [2]. It destroys the unbiasedness of the gradient estimate. If we choose B_G and B_h to be too small, the average clipped gradient may be a poor estimate of the true gradient. Moreover, B_G and B_h dictate the additive noise level. In general, clipping prescribes taking a smaller step downhill towards the optimal point [16] and may slow down the convergence.

6.3 Privacy Analysis of capeDJICA

In this section, we first present a theorem that provides the privacy guarantee of the capeDJICA algorithm using the conventional composition theorem [48]. However, we note that the capeDJICA algorithm involves a multi-round gradient descent and conventional privacy analysis may exaggerate the privacy loss. For a better characterization of the privacy guarantee, we analyze the capeDJICA algorithm with Rényi Differential Privacy (RDP) [107]. Finally, we compute the overall privacy loss of the capeDJICA algorithm using the *moments accountant* [2] to keep a better track of the privacy loss at each iteration.

Theorem 6.1 (Privacy of capeDJICA Algorithm)**.** *Consider Algorithm 6.1 in the decentralized data setting of Chapter 3 Section 3.1 with* $N_s = \frac{N}{S}$, $\tau_G^s = \tau_G$ *and* $\tau_h^s = \tau_h$ *for all sites* $s \in [S]$. *Suppose that at most* $S_C = \left\lceil \frac{S}{3} \right\rceil - 1$ *sites can collude after execution and the required number of iterations is* J^*. *Then Algorithm 6.1 computes an* $(2J^*\epsilon, 2J^*\delta)$*-DP approximation to the optimal unmixing matrix* $\mathbf{W}^*$, *where* (ϵ, δ) *satisfy the relation* $\delta = 2\frac{\sigma_z}{\epsilon - \mu_z}\phi\left(\frac{\epsilon - \mu_z}{\sigma_z}\right)$, $\phi(\cdot)$ *is the density for standard Normal random variable and* (μ_z, σ_z) *are given by* (3.3) *and* (3.4)*, respectively.*

Proof. The proof of Theorem 6.1 follows from using the Gaussian mechanism [50], the decentralized Stochastic Gradient Descent algorithm [2, 16, 127], the $\mathcal{L}_2$ sensitivities of the functions $f(\mathbf{X}_s) = \mathbf{G}_s$ and $f(\mathbf{X}_s) = \mathbf{h}_s$ and the privacy of the CAPE scheme [81]. We recall that the data samples in each site are disjoint. By the CAPE scheme (see Theorem 3.1), each iteration round is $(2\epsilon, 2\delta)$-DP. If the required total number of iterations is J^* then by composition theorem of differential privacy [48], the capeDJICA algorithm satisfies $(2J^*\epsilon, 2J^*\delta)$-DP, where (ϵ, δ) satisfy the relation $\delta = 2\frac{\sigma_z}{\epsilon - \mu_z}\phi\left(\frac{\epsilon-\mu_z}{\sigma_z}\right)$. □

6.3.1 Privacy Analysis using Rényi Differential Privacy

In this section, we analyze the capeDJICA algorithm with Rényi Differential Privacy (RDP) [107]. Analyzing the total privacy loss of a multi-shot algorithm, each stage of which is differentially private, is a challenging task. It has been shown [2, 107] that the advanced composition theorem [48] for (ϵ, δ)-differential privacy can be loose in the sense that it may exaggerate the privacy loss. The main reason is that one can formulate infinitely many (ϵ, δ) differentially private algorithms for a given noise variance. RDP offers a much simpler composition rule that is shown to be tight [107]. We start our analysis of the capeDJICA algorithm by reviewing some properties of RDP [107].

Proposition 6.3 (From RDP to DP [107])**.** *If $\mathcal{A}$ is an (α, ϵ_r)-RDP mechanism, then it also satisfies $\left(\epsilon_r + \frac{\log\frac{1}{\delta_r}}{\alpha-1}, \delta_r\right)$-differential privacy for any $0 < \delta_r < 1$.*

Proposition 6.4 (Composition of RDP [107])**.** *Let $\mathcal{A} : \mathbb{D} \mapsto \mathbb{T}_1$ be (α, ϵ_{r1})-RDP and $\mathcal{B} : \mathbb{T}_1 \times \mathbb{D} \mapsto \mathbb{T}_2$ be (α, ϵ_{r2})-RDP, then the mechanism defined as (X, Y), where $X \sim \mathcal{A}(D)$ and $Y \sim \mathcal{B}(X, D)$, satisfies $(\alpha, \epsilon_{r1} + \epsilon_{r2})$-RDP.*

Proposition 6.5 (RDP and Gaussian Mechanism [107])**.** *If $\mathcal{A}$ has $\mathcal{L}_2$ sensitivity 1, then the Gaussian mechanism $\mathbf{G}_\sigma\mathcal{A}(D) = \mathcal{A}(D) + E$, where $E \sim \mathcal{N}(0, \sigma^2)$ satisfies $(\alpha, \frac{\alpha}{2\sigma^2})$-RDP. Additionally, a composition of J Gaussian mechanisms, each with parameter σ will have the RDP curve of a Gaussian mechanism with parameter $\frac{\sigma}{\sqrt{J}}$.*

The proofs of the Propositions 6.3, 6.4 and 6.5 are provided in [107]. Now, we are in a position to analyze the proposed capeDJICA algorithm with the composition theorem for RDP. Recall that, at each iteration j of capeDJICA, we compute the noisy estimates

of the gradients: $\Delta_{\mathbf{W}}(j)$ and $\Delta_{\mathbf{b}}(j)$. Because of our correlated noise scheme in the symmetric setting, the variance of noise at the aggregator for $\Delta_{\mathbf{W}}(j)$ is:

$$\sigma^2_{\mathbf{W}} = \rho^2 {\tau_G^{\text{pool}}}^2.$$

Similarly, the variance of noise at the aggregator for $\Delta_{\mathbf{b}}(j)$ is:

$$\sigma^2_{\mathbf{b}} = \rho^2 {\tau_h^{\text{pool}}}^2.$$

From Proposition 6.5, we have that the computation of $\Delta_{\mathbf{W}}(j)$ is $(\alpha, \frac{\alpha}{2\sigma^2_{\mathbf{W}}})$-RDP. Similarly, the computation of $\Delta_{\mathbf{b}}(j)$ is $(\alpha, \frac{\alpha}{2\sigma^2_{\mathbf{b}}})$-RDP. By Proposition 6.4, we have that each iteration step of capeDJICA is $\left(\alpha, \frac{\alpha}{2}\left(\frac{1}{\sigma^2_{\mathbf{W}}} + \frac{1}{\sigma^2_{\mathbf{b}}}\right)\right)$-RDP. If we denote the number of required iterations for reaching convergence in capeDJICA by J^* then, under J^*-fold composition of RDP, the overall capeDJICA algorithm is $(\alpha, \frac{\alpha J^*}{2\sigma^2_{\text{RDP}}})$-RDP, where

$$\frac{1}{\sigma^2_{\text{RDP}}} = \left(\frac{1}{\sigma^2_{\mathbf{W}}} + \frac{1}{\sigma^2_{\mathbf{b}}}\right).$$

We conclude that the capeDJICA algorithm satisfies $\left(\frac{\alpha J^*}{2\sigma^2_{\text{RDP}}} + \frac{\log \frac{1}{\delta_r}}{\alpha - 1}, \delta_r\right)$ differential-privacy for any $0 < \delta_r < 1$ (from Proposition 6.3). For a given δ_r, we find the optimal α_{opt} as:

$$\alpha_{\text{opt}} = 1 + \sqrt{\frac{2}{J^*}\sigma^2_{\text{RDP}} \log \frac{1}{\delta_r}}. \tag{6.10}$$

Therefore, the capeDJICA algorithm is $\left(\frac{\alpha_{\text{opt}} J^*}{2\sigma^2_{\text{RDP}}} + \frac{\log \frac{1}{\delta_r}}{\alpha_{\text{opt}} - 1}, \delta_r\right)$-differentially private for any $0 < \delta_r < 1$.

6.3.2 Privacy Accounting using Moments Accountant

In this section, we compute the overall privacy loss of the capeDJICA algorithm following the *moments accountant* [2]. The moments accountant method keeps a better track of the privacy loss at each iteration and can be used to achieve a much smaller total ϵ

than the strong composition theorem [48]. As mentioned before, naïvely employing the additive nature of the privacy loss results in the worst case analysis, i.e., assumes that each iteration step exposes the worst privacy risk and this exaggerates the total privacy loss. However, in practice, the privacy loss is a random variable that depends on the dataset and is typically well-behaved (concentrated much closer to its expected value). Let us consider the randomized mechanism $\mathcal{A} : \mathbb{D} \mapsto \mathbb{T}$. We denote the neighboring datasets with $D, D' \in \mathbb{D}$. For a particular outcome $o \in \mathbb{T}$ of the mechanism, the privacy loss random variable is defined [2] as

$$Z = \log \frac{\Pr[\mathcal{A}(D) = o]}{\Pr[\mathcal{A}(D') = o]} \text{ w.p. } \Pr[\mathcal{A}(D) = o]. \tag{6.11}$$

Note that the basic idea of [2] for accounting for the total privacy loss is to compute the moment generating function (MGF) of Z for each iteration, use composition to get the MGF of the complete algorithm and then use that to compute final privacy parameters (see Theorem 2 of [2]). The stepwise moment for any t at iteration j is defined [2] as

$$\alpha_j(t) = \sup_{D,D'} \log \mathbb{E}\left[\exp(tZ)\right]. \tag{6.12}$$

If total number of iterations is J^* then the overall moment is upper bounded as

$$\alpha(t) \leq \sum_{j=1}^{J^*} \alpha_j(t).$$

Finally, for any given $\epsilon > 0$, the overall mechanism is (ϵ, δ) differentially private for

$$\delta = \min_t \exp\left(\alpha(t) - t\epsilon\right).$$

We now employ the framework to our capeDJICA algorithm and find the best ϵ for a given δ. For a Gaussian mechanism $\mathbf{G}_\sigma\mathcal{A}(D) = \mathcal{A}(D) + E$, where $E \sim \mathcal{N}(0, \sigma^2)$, the

privacy loss random variable defined in (6.11) can be written as

$$\begin{aligned} Z &= \log \frac{\exp\left(-\frac{1}{2\sigma^2}\left(o - f_D\right)^2\right)}{\exp\left(-\frac{1}{2\sigma^2}\left(o - f'_D\right)^2\right)} \\ &= \frac{1}{2\sigma^2}\left(2o(f_D - f'_D) - (f_D^2 - {f'_D}^2)\right). \end{aligned}$$

Now,

$$\begin{aligned} \mathbb{E}[\exp(tZ)] &= \int_o \exp\left(\frac{t}{2\sigma^2}\left(2o(f_D - f'_D) - (f_D^2 - {f'_D}^2)\right)\right) \\ &\quad \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(o - f_D\right)^2\right) do \\ &= \frac{\sigma}{\sqrt{f_D - f'_D}} \exp\left((f_D - f'_D)\frac{t^2\sigma^2 + t(f_D - f'_D)}{2\sigma^2}\right), \end{aligned}$$

where the last equality follows from the moment generating function of generalized Gaussian and some simple algebra. If the $\mathcal{L}_2$ sensitivity of the function $\mathcal{A}(D)$ is Δ then

$$\begin{aligned} \alpha_j(t) &= \sup_{D,D'} \log \frac{\sigma}{\sqrt{f_D - f'_D}} + \frac{f_D - f'_D}{2\sigma^2}\left(t^2\sigma^2 + t(f_D - f'_D)\right) \\ &= \log \frac{\sigma}{\sqrt{\Delta}} + \frac{\Delta}{2\sigma^2}\left(t^2\sigma^2 + t\Delta\right). \end{aligned}$$

We can compute the upper bound of the overall moment

$$\begin{aligned} \alpha(t) &\leq \sum_{j=1}^{J^*} \alpha_j(t) \\ &= J^*\left(\log \frac{\sigma}{\sqrt{\Delta}} + \frac{\Delta}{2\sigma^2}\left(t^2\sigma^2 + t\Delta\right)\right). \end{aligned}$$

Now, for any given $\epsilon > 0$, we have

$$\begin{aligned} \delta &= \min_t \exp\left(\alpha(t) - t\epsilon\right) \\ &= \min_t \exp\left(J^*\left(\log \frac{\sigma}{\sqrt{\Delta}} + \frac{\Delta}{2\sigma^2}\left(t^2\sigma^2 + t\Delta\right)\right) - t\epsilon\right). \end{aligned}$$

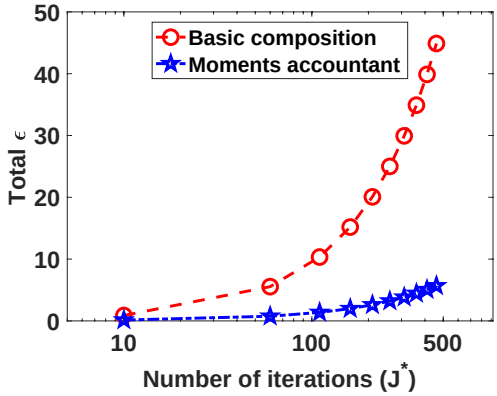

Figure 6.1: Variation of total ϵ with number of iterations J^*: $\sigma^2_{\mathbf{W}} = \sigma^2_{\mathbf{b}} = 0.001$ and $\delta = 1/N$. Moments accountant method provides a much smaller total ϵ than the basic composition.

We compute the minimizing t as

$$t_{\text{opt}} = \frac{\epsilon}{J^*\Delta} - \frac{\Delta}{2\sigma^2}.$$

Using this, we find δ_{opt}

$$\delta_{\text{opt}} = \exp\left(J^* \log\frac{\sigma}{\sqrt{\Delta}} + \frac{J^*\Delta}{2}\left(\frac{\epsilon}{J^*\Delta} - \frac{\Delta}{2\sigma^2}\right)^2 + \left(\frac{J^*\Delta^2}{2\sigma^2} - \epsilon\right)\left(\frac{\epsilon}{J^*\Delta} - \frac{\Delta}{2\sigma^2}\right)\right).$$

As we are interested in finding the best ϵ for a given δ, we rearrange the above equation to solve for ϵ as

$$\epsilon = \frac{1}{2a}\left(-b \pm \sqrt{b^2 - 4ac}\right),$$

where $a = \frac{1}{2J^*\Delta}$, $b = -\frac{\Delta}{2\sigma^2}$ and $c = \log\delta - J^*\log\frac{\sigma}{\sqrt{\Delta}} + \frac{J^*\Delta^3}{8\sigma^4}$. For our proposed capeDJICA algorithm, we release two noisy gradients: $\Delta_{\mathbf{W}}(j)$ and $\Delta_{\mathbf{b}}(j)$, at iteration j with noise variances $\sigma^2_{\mathbf{W}}$ and $\sigma^2_{\mathbf{b}}$, respectively. Adjusting for this, we plot the total ϵ against the total iterations J^* for the basic composition and the moments accountant in Figure 6.1. We observe that the moments accountant method provides a much smaller total ϵ than the basic composition (which grows linearly with J^*).

6.4 Performance Analysis of capeDJICA

6.4.1 Performance Gain with Correlated Noise

The existing differentially private djICA algorithm [80] achieved $J^*\epsilon$-differential privacy (where J^* is the total number of iterations required for convergence) by adding a noise term to the local estimate of the source (i.e., $\mathbf{Z}_s(j)$). Although the algorithm offered a "pure" differentially private djICA procedure, there are a few shortcomings. The cost of achieving pure differential-privacy (i.e., employing the Laplace mechanism [50]) was that the noise variance was dependent on R and the norm of the most-recent $\mathbf{W}$. Moreover, the neighboring dataset condition was met by restricting the $\mathcal{L}_2$-norm of the samples to satisfy $\|\mathbf{x}_n\|_2 \leq \frac{1}{2\sqrt{D}}$, which can be too limiting for datasets with large ambient dimensions. Last but not the least, the differentially private PCA preprocessing step was less fault tolerant than the one employed in this work. By employing the CAPE protocol in the preprocessing stage and also in the optimization process, we expect to gain a significant performance boost. We validate the performance gain in the Experimental Results (Section 6.5).

6.4.2 Convergence of capeDJICA Algorithm

We note that the gradient estimate at the aggregator (Step 13 in Algorithm 6.1) essentially contains the noise $\frac{\rho}{S}\sum_{s=1}^{S}\mathbf{K}_s^G$, which is zero mean. Therefore, the convergence of Algorithm 6.1 is guaranteed [25]. Since the total additive noise variance is smaller than the conventional case by a factor of S, the convergence rate is faster than the conventional case.

6.4.3 Communication Cost

We analyze the total communication cost associated with the proposed capeDJICA algorithm. At each iteration round, we need to generate two zero-sum noise terms, which entails $O(S + R^2)$ communication complexity of the sites and $O(S^2 + SR^2)$ communication complexity of the aggregator [22]. Each site computes the noisy gradient and sends one $R\times R$ matrix and one R dimensional vector to the aggregator. And finally,

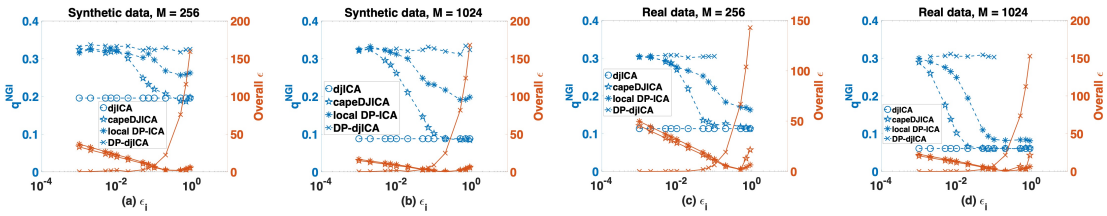

Figure 6.2: Variation of q^{NGI} and overall ϵ with privacy parameter $\epsilon_i$for: (a)-(b) synthetic fMRI data, (c)-(d) real fMRI data. Fixed parameters: $S = 4$, $\delta = 10^{-5}$. For a given privacy budget (performance requirement), the user can use the overall ϵ plot on the right y-axis, shown with solid lines, (q^{NGI} plot on the left y-axis, shown with dashed lines) to find the required ϵ_i on the x-axis and thereby, find the corresponding performance (overall ϵ). For capeDJICA, higher ϵ_i results a smaller q^{NGI}, but not necessarily a small overall ϵ, i.e., an optimal ϵ_i can be chosen based on q^{NGI} or overall ϵ requirement.

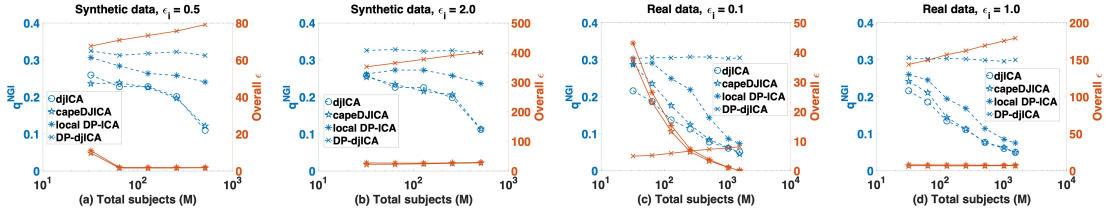

Figure 6.3: Variation of q^{NGI} and overall ϵ with total number of subjects M for: (a)-(b) synthetic fMRI data, (c)-(d) real fMRI data. Fixed parameters: $S = 4$, $\delta = 10^{-5}$. For a given privacy budget (performance requirement), the user can use the overall ϵ plot on the right y-axis, shown with solid lines, (q^{NGI} plot on the left y-axis, shown with dashed lines) to find the required M on the x-axis and thereby, find the corresponding performance (overall ϵ). For capeDJICA, higher M results a smaller q^{NGI} and a smaller overall ϵ.

the aggregator sends the $R \times R$ updated weight matrix and R dimensional bias estimate to the sites. If the number of iterations required to achieve convergence is denoted by J^* then the total communication cost is proportional to $4J^*(SR^2 + SR)$ or $O(S + R^2)$ for the sites and $O(S^2 + SR^2)$ for the central node. This is expected as we are estimating an $R \times R$ matrix in a decentralized setting.

6.5 Experimental Results

In this section, we empirically show the effectiveness of the proposed capeDJICA algorithm. We note the intricate relationship between ϵ and δ (see Theorem 6.1) due to the correlated noise scheme and the challenge of characterizing the overall privacy loss in our multi-round capeDJICA algorithm. We designed the experiments to better

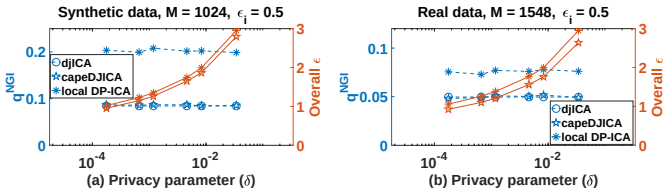

Figure 6.4: Variation of q^{NGI} and overall ϵ with privacy parameter δ: (a) synthetic and (b) real fMRI data. Fixed parameters: $S = 4$, $\epsilon_i = 0.5$. capeDJICA achieves very close utility to the non-private djICA with small overall ϵ.

demonstrate the trade-off between performance and several parameters: ϵ, δ and M. We show the simulation results to compare the performance of our capeDJICA algorithm with the existing DP djICA algorithm [80] (DP – djICA), the non-private djICA algorithm [11] and a DP ICA algorithm operating on only local data (local). We modified the base non-private djICA algorithm to incorporate the gradient bounds B_G and B_h. Although we are proposing an algorithm for decentralized setting, we included the performance indices for the local setting to show the effect of smaller sample sizes on the performance. We note that the DP – djICA algorithm [80] offers ϵ-differential privacy as opposed to (ϵ, δ)-differential privacy offered by capeDJICA. For both synthetic and real datasets, we consider the symmetric setting (i.e., $N_s = \frac{N}{S}$, $\tau_G^s = \tau_G$ and $\tau_h^s = \tau_h$) and show the average performance over 10 runs of the algorithms.

Synthetic Data. We generated the synthetic data from the same model as [80, 11]. The source signals $\mathbf{S}$ were simulated using the generalized autoregressive (AR) conditional heteroscedastic (GARCH) model [53, 20]. We have utilized $M = 1024$ simulated subjects in our experiments. For each subject, we generated $R = 20$ time courses with 250 time points. The data samples are equally divided into $S = 4$ sites. For each subject, the fMRI images are 30×30 dimensional. We employ the capePCA algorithm [77] as a preprocessing stage to reduce the sample dimension from $D = 900$ to $R = 20$. The capeDJICA is carried out upon the R-dimensional samples.

Real Data. The data were collected using a 3-T Siemens Trio scanner with a 12-channel radio frequency coil, according to the protocol in [5]. In the dataset, the resting-state scan durations range from 2 min 8 sec to 10 min 2 sec, with an average of 5 min 16 sec [11]. We used a total of $M = 1548$ subjects from the dataset and

estimated $R = 50$ independent components using the algorithms under consideration. Pre-processing of the data was performed according to [11] – the data underwent rigid body alignment for head motion, slice-timing correction, spatial normalization to MNI space, regression of 6 motion parameters and their derivatives in addition to any trends (up to cubic or quintic), and spatial smoothing using a $10mm^3$ full-width at half-maximum Gaussian kernel. We also projected the data onto a 50-dimensional PCA subspace estimated using pooled non-private PCA. As we do not have the ground truth for the real data, we computed a *pseudo* ground truth [11] by performing a pooled non-private analysis on the data and estimating the unmixing matrix. The performance of capeDJICA, djICA, $\mathsf{DP}-\mathsf{djICA}$ and local algorithms are evaluated against this pseudo ground truth.

Performance Indices. We set $\tau_G^s = \frac{\Delta_G^s}{\epsilon_i}\sqrt{2\log\frac{1.25}{10^{-2}}}$ and $\tau_h^s = \frac{\Delta_h^s}{\epsilon_i}\sqrt{2\log\frac{1.25}{10^{-2}}}$ for our experiments, where ϵ_i is the privacy parameter per iteration, Δ_G^s and Δ_h^s are the $\mathcal{L}_2$ sensitivities of $\mathbf{G}_s$ and $\mathbf{h}_s$, respectively. To evaluate the performance of the algorithms, we consider the quality of the estimated unmixing matrix $\mathbf{W}$. More specifically, we utilize the normalized gain index q^{NGI} [11, 114] that quantizes the quality of $\mathbf{W}$. The normalized gain index q^{NGI} varies from 0 to 1, with 0 indicating that the unmixing matrix is an identity matrix [114].

Note that, in addition to a small q^{NGI}, we want to attain a strict privacy guarantee, i.e. small overall (ϵ, δ). Recall from Section 6.3.2 that the overall ϵ is a function of the number of iterations, the overall δ and $\{\tau_G^s, \tau_h^s\}$. For all of our experimental analyses, we plotted the overall ϵ (with solid lines on the right y-axis) along with q^{NGI} (with dashed lines on the left y-axis) as a means for visualizing how the privacy-utility trade-off varies with different parameters.

6.5.1 Performance Variation with privacy parameter ϵ

First, we explore how the privacy-utility tradeoff between q^{NGI} and the overall "privacy risk" ϵ varies with ϵ_i. In Figs. 6.2(a) - (b), we show the variation of q^{NGI} and overall ϵ for different algorithms with ϵ_i on synthetic data. We kept the number of sites $S = 4$ and the target $\delta = 10^{-5}$ fixed. As mentioned before, we compare the performance of

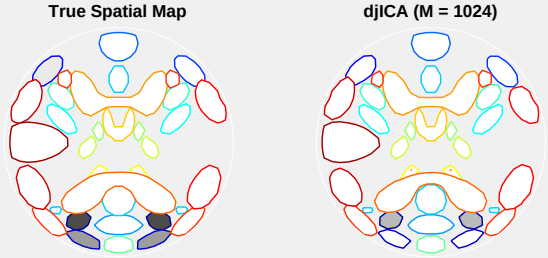

Figure 6.5: Spatial maps (synthetic data): true and resulting from djICA.

capeDJICA with those of the djICA, the DP − djICA and local. We show the performance indices for two different M values, namely $M = 256$ and $M = 1024$. We observe from the figures that the proposed capeDJICA outperforms the existing DP − djICA by a large margin. This is expected as DP − djICA suffers from too much noise (see Section 6.4.1 for the explanation). capeDJICA also guarantees the smallest overall ϵ among the privacy-preserving methods. capeDJICA can reach the utility level of the non-private djICA for some parameter choices and naturally outperforms local as estimation of the sources is much accurate when more samples are available. For the same privacy loss (i.e., for a fixed ϵ), one can achieve better performance by increasing the number of subjects. In Figs. 6.2(c) - (d), we show the variation of q^{NGI} and overall ϵ for different algorithms with ϵ_i on real data. We show the performance indices for $M = 256$ and $M = 1024$. We observe that, similar to the synthetic data, the proposed capeDJICA outperforms the existing DP − djICA by a large margin. The proposed capeDJICA can reach the utility level of the non-private djICA even for small overall ϵ values and outperforms local. Again we observe that, for a fixed ϵ, we can achieve better performance by increasing the number of subjects. For both synthetic and real data, we note that assigning a higher ϵ_i may provide a good q^{NGI} but does not guarantee a small overall ϵ. The user needs to choose the ϵ_i based on the "privacy budget" and the required performance.

6.5.2 Performance Variation with number of subjects M

Next, in Figure 6.3(a) - (b), we show the variation of q^{NGI} and the overall ϵ with the total number of subjects M for two different ϵ_i values on synthetic data. We kept the number of sites $S = 4$ and target $\delta = 10^{-5}$ fixed. We observe similar trends in

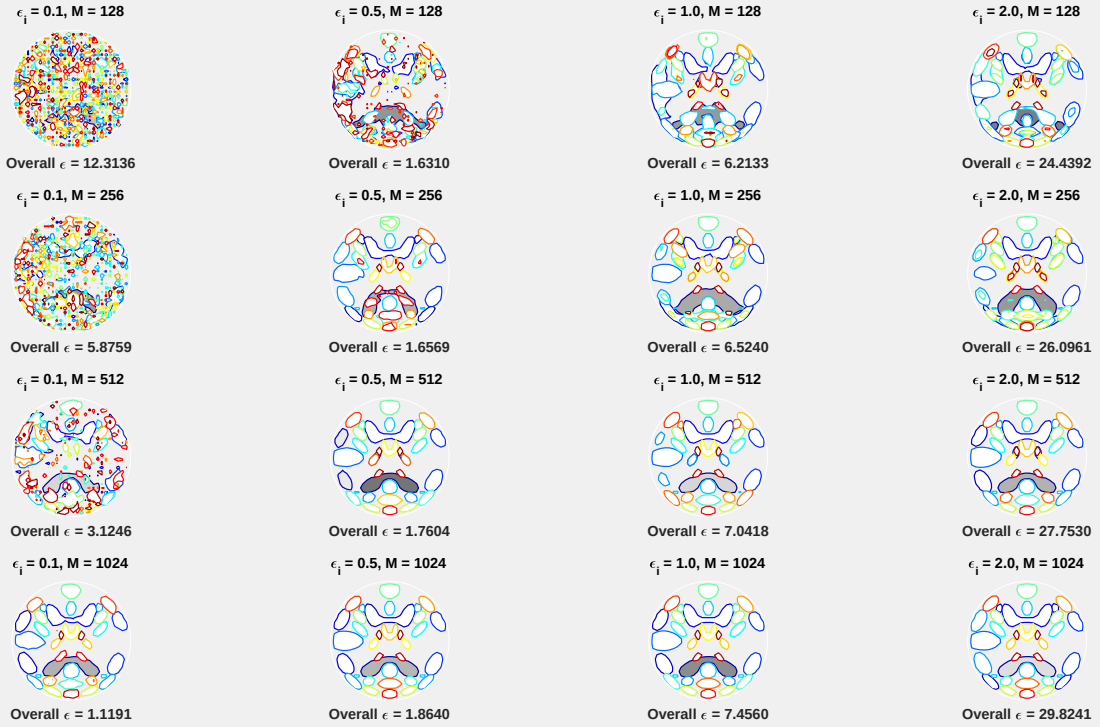

Figure 6.6: Spatial maps (synthetic data) resulting from capeDJICA for different parameters. capeDJICA estimates spatial maps that closely resemble the true ones, even for strict privacy guarantee (small overall ϵ).

performance as in the case of varying ϵ_i. The capeDJICA algorithm outperforms the DP – djICA and the local: with respect to both q^{NGI} and the overall ϵ. For the q^{NGI}, the capeDJICA performs very closely to the non-private djICA. The performance gain over DP – djICA is particularly noteworthy. For a fixed number of subjects, increasing ϵ results in a slightly better utility, albeit at the cost of greater privacy loss. In Figure 6.3(c) - (d), we show the variation of q^{NGI} and overall ϵ with the total number of subjects M for $\epsilon_i = 0.1$ and $\epsilon_i = 1.0$ on real data. We observe a very similar trend as the synthetic data simulations: the capeDJICA algorithm comfortably outperforms the DP – djICA and the local and achieves utility close to the non-private djICA even for moderate M values, while guaranteeing the smallest overall ϵ.

6.5.3 Performance Variation with privacy parameter δ

Recall that, the proposed capeDJICA algorithm guarantees (ϵ, δ) differential privacy, where (ϵ, δ) satisfy the relation $\delta = 2\frac{\sigma_z}{\epsilon - \mu_z}\phi\left(\frac{\epsilon - \mu_z}{\sigma_z}\right)$. In Figure 6.4, we show the variation

of q^{NGI} with overall δ on synthetic and real data. Recall that δ is essentially the probability of failure of a DP algorithm. Therefore, we want δ to be small. However, a smaller δ also results in a larger noise variance, which affects the utility. From the figure, we can observe how the performance of the proposed capeDJICA algorithm varies with δ. We demonstrate the performance indices keeping $\epsilon_i = 0.5$ and $S = 4$ fixed. We set the number of colluding sites to be $S_C = \lceil \frac{S}{3} \rceil - 1$. The proposed algorithm achieves very close utility to the non-private djICA for both synthetic and real data. For both cases, the overall ϵ is also very small. However, we can opt for even smaller δ values at the cost of performance.

6.5.4 Reconstructed Spatial Maps

Finally, we intend to demonstrate how the estimated spatial maps actually look like, as interpretability is one of the most important concerns for fMRI applications. In Figure 6.5, we show the true spatial map and the one estimated from the non-private djICA [11] algorithm. In Figure 6.6, we show the estimated spatial maps resulting from the proposed capeDJICA algorithm along with the overall ϵ for a variety of combinations of ϵ_i and M. We observe that when sufficiently large number of subjects are available, the estimated spatial maps closely resemble the true one, even for strict privacy guarantee (small overall ϵ). For smaller number of samples, we may need to compensate by allowing larger ϵ values to achieve good utility. In general, we observe that capeDJICA can achieve very good approximate to the true spatial map, almost indistinguishable from the non-private spatial map. This emphasizes the effectiveness of the proposed capeDJICA in the sense that very meaningful utility can be achieved even with strict privacy guarantee.

Chapter 7

Decentralized Differentially Private Canonical Correlation Analysis

In this chapter, we first formulate the problem of decentralized CCA. We then employ the CAPE protocol to propose a decentralized version of our centralized differentially private CCA [76]. To our knowledge, this work proposes the first differentially private CCA algorithm for decentralized settings. Using synthetic and real datasets, we demonstrate how the utility/performance is affected by the privacy risk, number of samples and some other key parameters. Simulation results show that our capeCCA algorithm can achieve the same utility as the pooled data scenario satisfying (ϵ, δ)-differential privacy. For some parameter choices, our algorithm can achieve almost as much utility as the non-private algorithm, showing that meaningful privacy can (almost) come for free.

7.1 Decentralized Canonical Correlation Analysis

Consider a system similar to the one shown in Section 3.1: S different sites, each holding disjoint data sets, and an untrusted central node or aggregator (see Fig. 3.1(a)). In site $s \in [S]$, the data is a pair of sample matrices $\mathbf{X}_s \in \mathbb{R}^{D_x \times N_s}$ and $\mathbf{Y}_s \in \mathbb{R}^{D_y \times N_s}$ corresponding to the two "views" of the same physical phenomena. The n-th column of $\mathbf{X}_s$ and $\mathbf{Y}_s$, denoted $\mathbf{x}_{s,n}$ and $\mathbf{y}_{s,n}$, respectively, are the n-th samples from view 1 and view 2. For simplicity, we assume that the observed samples are mean-centered. The sample size in site s is N_s and we denote $N = \sum_{s=1}^{S} N_s$ as the total number of samples over all sites. If we had all the samples at the central aggregator (pooled data scenario), then the data matrices would be $\mathbf{X} = [\mathbf{X}_1 \dots \mathbf{X}_S] \in \mathbb{R}^{D_x \times N}$ and $\mathbf{Y} = [\mathbf{Y}_1 \dots \mathbf{Y}_S] \in \mathbb{R}^{D_y \times N}$. The CCA projection vectors are defined to be the columns of the matrices

$\mathbf{U} \in \mathbb{R}^{D_x \times K}$ and $\mathbf{V} \in \mathbb{R}^{D_y \times K}$ that solve the following problem [71, 63, 9]:

$$\begin{aligned} \underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad & \|\mathbf{U}^\top \mathbf{X} - \mathbf{V}^\top \mathbf{Y}\|_F^2 \\ \text{subject to} \quad & \frac{1}{N}\mathbf{U}^\top \mathbf{X}\mathbf{X}^\top \mathbf{U} = \mathbf{I}, \frac{1}{N}\mathbf{V}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{V} = \mathbf{I}, \\ & \frac{1}{N}\mathbf{U}^\top \mathbf{X}\mathbf{Y}^\top \mathbf{V} = \mathbf{I}, \end{aligned}$$

where $\mathbf{I}$ is the $K \times K$ identity matrix with $K \leq \min\{D_x, D_y\}$. The solution to the optimization problem [67] is given as follows: $\mathbf{U}^*$ and $\mathbf{V}^*$ contain the top-K eigenvectors of the matrices $\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}$ and $\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}$, respectively. Here, the sample covariance and cross-covariance matrices are given by $\mathbf{C}_{xx} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$, $\mathbf{C}_{yy} = \frac{1}{N}\mathbf{Y}\mathbf{Y}^\top$ and $\mathbf{C}_{xy} = \frac{1}{N}\mathbf{X}\mathbf{Y}^\top = \mathbf{C}_{yx}^\top$. We assume that we obtain samples as $\mathbf{z}_{s,n} = \begin{bmatrix}\mathbf{x}_{s,n}^\top & \mathbf{y}_{s,n}^\top\end{bmatrix}^\top \in \mathbb{R}^D$, where $D = D_x + D_y$. We compute the $D \times D$ positive semi-definite sample covariance matrix of $\mathbf{Z} = [\mathbf{Z}_1 \dots \mathbf{Z}_S] \in \mathbb{R}^{D \times N}$ as

$$\mathbf{C} = \frac{1}{N}\mathbf{Z}\mathbf{Z}^\top = \frac{1}{N}\sum_{s=1}^{S}\sum_{n=1}^{N_s}\mathbf{z}_{s,n}\mathbf{z}_{s,n}^\top \text{ and } \mathbf{C} = \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{C}_{yy} \end{bmatrix}.$$

Without loss of generality, we can ensure that $\|\mathbf{z}_{s,n}\|_2 \leq 1$, because canonical correlations are invariant with respect to affine transformations of the variables [23]. We are interested in approximating $\mathbf{U}^*$ and $\mathbf{V}^*$ in a decentralized setting while guaranteeing differential privacy. A naïve approach (non-privacy-preserving) would be to send the data matrices $\mathbf{X}_s$ and $\mathbf{Y}_s$ from the sites to the aggregator. The aggregator can then compute $\mathbf{C}$ and subsequently $\mathbf{U}^*$ and $\mathbf{V}^*$. However, when D_x, D_y and/or N_s are large, this results in a huge communication overhead. Additionally, in many scenarios, the local data are private or sensitive. As the aggregator is not trusted, sending the data to the aggregator can result in a significant privacy violation. Our goals are therefore to (i) ensure differential privacy, (ii) achieve the same utility as the pooled data scenario in a decentralized setting and (iii) provide close approximations to the true CCA subspaces $\mathbf{U}^*$, $\mathbf{V}^*$.

Algorithm 7.1 Decentralized Differentially Private CCA (capeCCA)

Require: **0**-centered samples $\mathbf{X}_s \in \mathbb{R}^{D_x \times N_s}$ and $\mathbf{Y}_s \in \mathbb{R}^{D_y \times N_s}$ as $\mathbf{Z}_s = \left[\mathbf{X}_s^\top \ \mathbf{Y}_s^\top\right]^\top$ with $\|\mathbf{z}_{s,n}\|_2 \leq 1$ for $s \in [S]$; local noise variances $\{\tau_s^2\}$; reduced dimension K

for $s = 1, 2, \ldots, S$ **do** ▷ at the local sites
  Generate $\mathbf{E}_s$ element-wise according to Algorithm 3.1
  Generate $D \times D$ symmetric $\mathbf{G}_s$, as described in text
  Compute and send: $\hat{\mathbf{C}}_s \leftarrow \frac{1}{N_s}\mathbf{Z}_s\mathbf{Z}_s^\top + \mathbf{E}_s + \mathbf{G}_s$
end for
Compute $\hat{\mathbf{C}} \leftarrow \frac{1}{S}\sum_{s=1}^{S} \hat{\mathbf{C}}_s$ ▷ at the aggregator
Extract sub-matrices from $\hat{\mathbf{C}}$ to compute $\hat{\mathbf{U}}^*$ and $\hat{\mathbf{V}}^*$
return $\hat{\mathbf{U}}^*$ and $\hat{\mathbf{V}}^*$

7.2 Proposed Decentralized Differentially Private CCA

Our proposed method, capeCCA, is described in Algorithm 7.1 and exploits the CAPE scheme (see Fig.3.1(b)). More specifically, our approach employs a correlated noise design to achieve the same utility of the pooled data case (i.e., $\tau_{\text{ag}} = \tau_c$) in the decentralized setting. We assume the same honest-but-curious setting as Section 3.1. Recall that in the pooled data scenario with no privacy requirements, we have the data matrices $\mathbf{X}$ and $\mathbf{Y}$. The samples are assumed to be the columns of the matrix $\mathbf{Z} = \left[\mathbf{X}^\top \ \mathbf{Y}^\top\right]^\top$. We can compute $\mathbf{C} = \frac{1}{N}\mathbf{Z}\mathbf{Z}^\top$, extract the sub-matrices $\mathbf{C}_{xx}$, $\mathbf{C}_{xy}$, $\mathbf{C}_{yx}$ and $\mathbf{C}_{yy}$ and compute the optimal CCA subspaces $\mathbf{U}^*$ and $\mathbf{V}^*$. In our decentralized setting, we need to add noise to preserve privacy. Recall that we designed the noise addition procedure in such a way that we can ensure differential privacy for the output from each site and achieve the noise level of the pooled data scenario in the final output from the aggregator. Each site generates the $D \times D$ matrix $\mathbf{E}_s$ following Algorithm 3.1 according to Section 3.3. The sites also generate the symmetric $D \times D$ matrix $\mathbf{G}_s$, where $[\mathbf{G}_s]_{ij}$ are drawn i.i.d. $\sim \mathcal{N}(0, \tau_g^2)$. At each site s, we compute the sample second-moment matrix $\mathbf{C}_s = \frac{1}{N_s}\mathbf{Z}_s\mathbf{Z}_s^\top$ and release (or send to the central aggregator): $\hat{\mathbf{C}}_s = \mathbf{C}_s + \mathbf{E}_s + \mathbf{G}_s$. The noise variances should ensure that the variance of $\mathbf{E}_s + \mathbf{G}_s$ is sufficient to guarantee (ϵ, δ)-differential privacy to $\mathbf{C}_s$. For our approach of computing the noise $\mathbf{E}_s$, we have $\tau_e^2 = \left(1 - \frac{1}{S}\right)\tau_s^2$. We also set $\tau_g^2 = \frac{1}{S}\tau_s^2$. Recall that, for a given pair of (ϵ, δ), we can calculate a noise variance τ_s^2 such that adding Gaussian noise of variance τ_s^2 will guarantee (ϵ, δ)-differential privacy. Since there are

many (ϵ, δ) pairs that yield the same τ_s^2, we parameterized our method using τ_s^2 [77]. Now, the aggregator computes

$$\hat{\mathbf{C}} = \frac{1}{S}\sum_{s=1}^{S}\hat{\mathbf{C}}_s = \frac{1}{S}\sum_{s=1}^{S}(\mathbf{C}_s + \mathbf{G}_s)\,, \text{ as } \sum_{s=1}^{S}\mathbf{E}_s = 0.$$

At the aggregator, the variance of the estimator is exactly the same as if all the data were present at the aggregator (see Lemma 3.1). Next, we extract the sub-matrices $\hat{\mathbf{C}}_{xx}$, $\hat{\mathbf{C}}_{xy}$, $\hat{\mathbf{C}}_{yx}$ and $\hat{\mathbf{C}}_{yy}$ and compute the (ϵ, δ)-differentially private CCA subspaces $\hat{\mathbf{U}}^*$ and $\hat{\mathbf{V}}^*$. The privacy guarantee of capeCCA is given in Theorem 7.1. Note that capeCCA can be readily extended to incorporate unequal privacy requirements/samples sizes at each site (shown in Section 3.3.7).

Theorem 7.1 (Privacy of capeCCA)**.** *Consider Algorithm 7.1 in the decentralized data setting of Section 3.1 with $N_s = \frac{N}{S}$, $\tau_s = \tau$ for all sites $s \in [S]$. Suppose that at most $S_C = \lceil \frac{S}{3} \rceil - 1$ sites can collude after execution. Then Algorithm 7.1 computes an (ϵ, δ)-differentially private approximation to the optimal subspaces $\mathbf{U}^*$ and $\mathbf{V}^*$, where (ϵ, δ) satisfy the relation $\delta = 2\frac{\sigma_z}{\epsilon - \mu_z}\phi\left(\frac{\epsilon - \mu_z}{\sigma_z}\right)$, $\phi(\cdot)$ is the density for standard Normal random variable and (μ_z, σ_z) are given by* (3.3) *and* (3.4)*, respectively.*

Proof sketch. The proof of Theorem 7.1 follows from using the AG algorithm [51], the bound on $\|\mathbf{C}_s - \mathbf{C'}_s\|_2$ and the privacy of CAPE as in Theorem 3.1. The computation of $\hat{\mathbf{C}}_s$ at each site is (ϵ, δ)-differentially private. As differential privacy is invariant under post-processing, we can combine the matrices $\{\hat{\mathbf{C}}_s\}$ at the aggregator while subtracting $\mathbf{F}_s$ for each site. We extract the sub-matrices $\hat{\mathbf{C}}_{xx}$, $\hat{\mathbf{C}}_{xy}$, $\hat{\mathbf{C}}_{yx}$ and $\hat{\mathbf{C}}_{yy}$ and compute the subspaces $\hat{\mathbf{U}}^*$ and $\hat{\mathbf{V}}^*$, which are (ϵ, δ)-differentially private approximates to the true CCA subspaces $\mathbf{U}^*$ and $\mathbf{V}^*$. □

7.2.1 Performance gain with correlated noise

This is the first work that proposes an algorithm for decentralized differentially private CCA. It can be shown that as we employ the correlated noise scheme, the gain in the performance over a conventional decentralized differentially private CCA is atleast S,

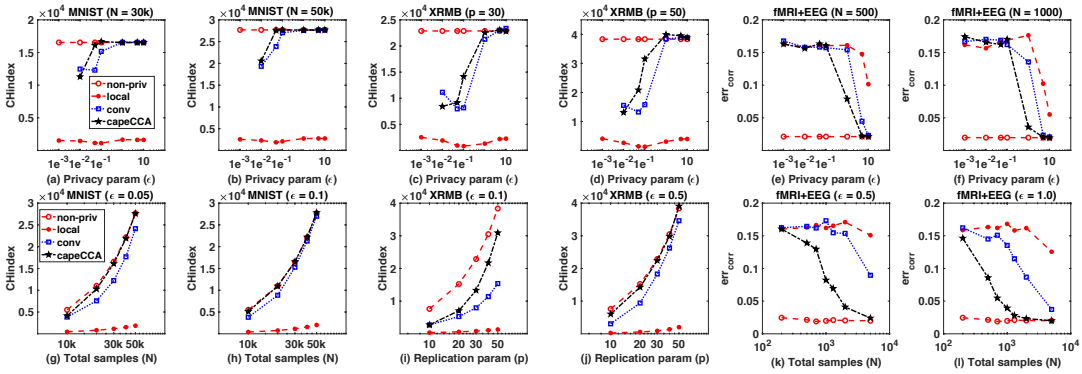

Figure 7.1: Variation of performance with privacy parameter ϵ and total samples N. Fixed parameters: $\delta = 0.01$, $S = 10$.

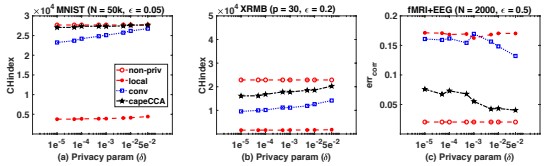

Figure 7.2: Variation of performance with δ. Fixed parameter: $S = 10$.

even when we do not know N_s for $s \in [S]$ (see Proposition 3.1). Moreover, in case of site drop-out, the performance of capeCCA would fall back to that of the conventional scheme [77]: the output from each site remains (ϵ, δ)-differentially private, irrespective of the number of dropped-out sites.

7.2.2 Communication cost

We note that capeCCA is an one-shot algorithm. For generating the zero-sum noise terms, we encounter $O(S+D^2)$ communication complexity of the sites and $O(S^2+SD^2)$ communication complexity of the aggregator [22]. Each site uses these to compute the noisy estimate of the local second-moment matrix $(D \times D)$ and sends that back to the aggregator. Therefore, the total communication cost is $O(S + D^2)$ for the sites and $O(S^2+SD^2)$ for the aggregator. This is expected as we are computing the global $D \times D$ second-moment matrix in a decentralized setting for computing the CCA subspaces.

7.3 Experimental Results

We consider measuring how well the output subspaces of capeCCA algorithm, $\hat{\mathbf{U}}^*$ and $\hat{\mathbf{V}}^*$, approximate the true subspaces $\mathbf{U}^*$ and $\mathbf{V}^*$ achieved from pooled non-private CCA ($\mathsf{non-dp\ pool}$). We have also compared the performance of capeCCA against a conventional (but never proposed before) decentralized differentially private CCA algorithm with no correlated noise (conv) and a centralized differentially private CCA [76] on local data (local). We used three data sets for our experiments: the MNIST data set [91], the University of Wisconsin X-ray Microbeam data set (XRMB) [142] and a simulated fMRI and EEG data set (fMRI+EEG) [43]. For MNIST, we chose the two views to be the top- and bottom-halves of the images, preprocessed by projecting each view onto 50-dimensional subspaces using PCA. For XRMB, we chose two speakers, JW16 and JW18, with the first view being the pellet coordinates and the second view containing acoustic features including the normalized 13-dimensional mel-frequency cepstral coefficients (MFCCs) and their first and second derivatives [9]. Each view is projected onto a 25-dimensional subspace using PCA [76]. We replicate the sample matrices p times in our experiments. For generating the fMRI+EEG data set we follow [43]. A simulated fMRI-like set of components was generated following [80] using the `simTB` toolbox [113]. For EEG features, an event-related potential (ERP)-like set of components was generated using the `EEGIFT` toolbox [112]. We mixed each set with different sets of modulation profiles to achieve the simulated fMRI and EEG signals. The modulation profiles are orthogonal within each modality and correlated across different modalities. The relation between the fMRI and EEG signals are due to these correlations. We used $K = 5$ simulated components for both fMRI and EEG signals. Each fMRI component is a 50×50 pixel image, whereas each ERP component is a 6360 time-point segment.

For the real data sets, we evaluate the quality of the subspaces produced from the algorithms by using one of the most common applications of CCA: clustering. We employ the popular K-means clustering algorithm on the reduced-dimension samples (achieved by projecting onto the CCA subspaces). We measure the performance of clustering using [76] the Caliński-Harabasz (CH) index [33, 104] for N data points and

K clusters:

$$\mathsf{CH} = \frac{\frac{1}{K-1}\sum_{k=1}^{K} N_k \|\mathbf{z}_k - \mathbf{z}\|_2^2}{\frac{1}{N-K}\sum_{k=1}^{K}\sum_{n\in\mathbb{S}_k} \|\mathbf{z}_{nk} - \mathbf{z}_k\|_2^2},$$

where $\mathbf{z}_k$ is the centroid of the k-th cluster, $\mathbf{z}$ is the centroid for all of the samples, N_k denotes the size of cluster k, $\mathbb{S}_k$ is the set of indices of the members of cluster k and $\mathbf{z}_{nk}$ is the n-th point of the k-th cluster. For the fMRI+EEG data set, we are interested to see how our algorithm can estimate the correlation between the two modalities. Therefore, we use the following performance index:

$$\text{err}_{\text{corr}} = \frac{1}{K}\|\mathbf{r}^* - \hat{\mathbf{r}}^*\|_2,$$

where $\mathbf{r}^* \in \mathbb{R}^K$ and $\hat{\mathbf{r}}^* \in \mathbb{R}^K$ contain the true correlation scores and the estimated correlation scores between the corresponding modulation profiles of the two modalities. We refer the reader to Correa et al. [43] for more details. In all cases we show the average performance over 10 independent runs of the algorithms.

7.3.1 Privacy-utility trade-offs

First, we explore the *privacy-utility trade-off* between the privacy risk ϵ and the aforementioned performance indices. Recall that the standard deviation of the noise in capeCCA is inversely proportional to both ϵ and N_s. Larger ϵ values indicate higher privacy risk but smaller noise and therefore better utility. We observe this in our experiments as well. In Figure 7.1(a)-(f), we show how the CH index and err_{corr} vary with ϵ for MNIST, XRMB and fMRI+EEG data sets, while keeping δ and S fixed. For each data set, we show the performance variation with ϵ for two different sample sizes. In all cases, the proposed capeCCA approaches the performance of non − dp pool as we increase ϵ, outperforming the conv and local. One of the reasons that capeCCA outperforms conv is the smaller noise variance at the aggregator that we can achieve due to the correlated noise scheme. Achieving better performance than local is intuitive because including the information from multiple sites to estimate a population parameter always results in a better performance than using the data from a single site only. For a particular

data set, we notice that if we increase the total sample size N (and hence the sample size per site N_s), the performance of capeCCA gets even better. This is expected as the variance of the noise for capeCCA is inversely proportional to square of N_s.

7.3.2 Learning rates and impact of δ

To better understand the impact of sample size, we tested the algorithms on data sets of increasing size. Intuitively, it should be easier to guarantee a smaller privacy risk for the same ϵ and a higher utility (lower error) when the number of samples is large. In Figure 7.1(g)-(l), we show the performance of capeCCA as a function of total sample size N for synthetic and real data sets with different values of ϵ. Increasing sample size improves the performance of all algorithms. Again, we observe that even for small ϵ values, capeCCA performs nearly as well as $\mathsf{non-dp\ pool}$, comfortably outperforming conv and local. For a particular data set, increasing ϵ dictates even better performance. Note that for the XRMB data set, we plotted the CH index vs. sample size plot with the replication parameter p.

Finally, we investigate the variation of performance with δ. Recall that δ can be interpreted as the probability that the privacy-preserving algorithm releases the private information "out in the wild". Therefore, we want δ to be small. However, the smaller the δ is the larger the noise variance becomes, resulting in loss of utility. In Figure 7.2, we show the performance indices as a function of δ for the synthetic and real data sets. As expected, we observe that increasing δ results in improved performance. However, choosing a $\delta \leq \frac{1}{N}$ may not provide meaningful utility without a sufficiently large ϵ. The proposed capeCCA algorithm achieves similar performance as $\mathsf{non-dp\ pool}$ for moderate δ values (~ 0.01).

Chapter 8

Decentralized Differentially Private Computation of Functions

As mentioned in Chapter 3, the CAPE framework can benefit any decentralized differentially private function computation, as long as the sensitivity of the function satisfies the conditions outlined in Proposition 3.2. In this chapter, we propose an algorithm that is specifically suited for privacy-preserving computation of cost functions in decentralized settings. We begin by reviewing the functional mechanism. We propose an improved functional mechanism by invoking a tighter characterization of the sensitivities and demonstrate the advantages with two examples. We then describe our capeFM algorithm in detail. Finally, we present experimental results to validate the effectiveness of our proposed algorithm over existing methods on real and synthetic data.

8.1 Functional Mechanism

We first review the functional mechanism [143] in the pooled data case. Let us consider a cost function $f_D(\mathbf{w}) : \mathbb{R}^D \mapsto \mathbb{R}$. Our goal is to find the minimizer $\mathbf{w}^*$ of $f_D(\mathbf{w})$. The cost incurred by a $\mathbf{w} \in \mathbb{R}^D$ due to one data sample $\mathbf{x}_n$ is $f(\mathbf{x}_n; \mathbf{w}) : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$. We need to minimize the average cost to find the optimal $\mathbf{w}^*$. The empirical average cost for a particular $\mathbf{w}$ over all the samples is expressed as

$$f_D(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x}_n; \mathbf{w}). \tag{8.1}$$

Therefore, we have

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} f_D(\mathbf{w}) = \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x}_n; \mathbf{w}).$$

For centralized optimization, we can find a differentially private approximate to $\mathbf{w}^*$ using output perturbation [38] or objective perturbation [38, 115]. We propose using the *functional mechanism* [143], which is a form of objective perturbation and is more amenable to decentralized implementation. It uses a polynomial representation of the cost function and can be used for any differentiable and continuous cost function. We perturb each term in the polynomial representation of $f_D(\mathbf{w})$ to get modified cost function $\hat{f}_D(\mathbf{w})$. The minimizer $\hat{\mathbf{w}}^* = \arg\min_{\mathbf{w}} \hat{f}_D(\mathbf{w})$ guarantees differential privacy.

Suppose $\phi(\mathbf{w})$ is a monomial function of the entries of $\mathbf{w}$: $\phi(\mathbf{w}) = w_1^{c_1} w_2^{c_2} \dots w_D^{c_D}$, for some set of exponents $\{c_d : d \in [D]\} \subseteq \mathbb{N}$. Let us define the set Φ_j of all $\phi(\mathbf{w})$ with degree j as

$$\Phi_j = \left\{ w_1^{c_1} w_2^{c_2} \dots w_D^{c_D} : \sum_{d=1}^{D} c_d = j \right\}.$$

For example, $\Phi_0 = \{1\}$, $\Phi_1 = \{w_1,\ w_2,\ \dots,\ w_D\}$, $\Phi_2 = \{w_{d_1} w_{d_2} : d_1, d_2 \in [D]\}$, etc. Now, from the Stone-Weierstrass Theorem [121], any differentiable and continuous cost function $f(\mathbf{x}_n; \mathbf{w})$ can be written as a (potentially infinite) sum of monomials of $\{w_d\}$:

$$f(\mathbf{x}_n; \mathbf{w}) = \sum_{j=0}^{J} \sum_{\phi \in \Phi_j} \lambda_{n\phi} \phi(\mathbf{w}),$$

for some $J \in [0, \infty]$. Here, $\lambda_{n\phi}$ denotes the coefficient of $\phi(\mathbf{w})$ in the polynomial and is a function of the n-th data sample. Plugging the expression of $f(\mathbf{x}_n; \mathbf{w})$ in (8.1), we can express the empirical average cost over all N samples as

$$f_D(\mathbf{w}) = \sum_{j=0}^{J} \sum_{\phi \in \Phi_j} \left(\frac{1}{N} \sum_{n=1}^{N} \lambda_{n\phi} \right) \phi(\mathbf{w}). \tag{8.2}$$

The function $f_D(\mathbf{w})$ depends on the data samples only through $\{\lambda_{n\phi}\}$. As the goal is to approximate $f_D(\mathbf{w})$ in a differentially private way, one can compute these $\lambda_{n\phi}$ to satisfy differential privacy [143]. Let us consider two "neighboring" datasets: $\mathbb{D} = \{\mathbf{x}_1,\ \dots,\ \mathbf{x}_{N-1},\ \mathbf{x}_N\}$ and $\mathbb{D}' = \{\mathbf{x}_1,\ \dots,\ \mathbf{x}_{N-1},\ \mathbf{x}'_N\}$, differing in a single data point

Algorithm 8.1 Improved Functional Mechanism

Require: Data samples $\{\mathbf{x}_n\}$; cost function $f_D(\mathbf{w})$ as in (8.3); privacy parameters ϵ, δ

1: **for** $j = 0,\ 1,\ \ldots,\ J$ **do**
2: Compute Λ_j as shown in Section 8.2
3: Compute $\Delta_j \leftarrow \max_{\mathbb{D},\mathbb{D}'} \|\Lambda_j^{\mathbb{D}} - \Lambda_j^{\mathbb{D}'}\|_F$
4: Compute $\tau_j = \frac{\Delta_j}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$
5: Compute $\hat{\Lambda}_j \leftarrow \Lambda_j + e_j$, where e_j have same dimensions as Λ_j and entries of e_j are i.i.d. $\sim \mathcal{N}(0, \tau_j^2)$
6: **end for**
7: Compute $\hat{f}_D(\mathbf{w}) = \sum_{j=0}^{J} \langle \hat{\Lambda}_j, \bar{\phi}_j \rangle$
8: **return** $\hat{f}_D(\mathbf{w})$

(i.e., the last one). Zhang et al. [143] computed the $\mathcal{L}_1$ sensitivity $\Delta^{\mathrm{dp-fm}}$ as

$$\left\| \sum_{j=0}^{J} \sum_{\phi\in\Phi_j} \frac{1}{N} \left(\sum_{\mathbb{D}} \lambda_{n\phi} - \sum_{\mathbb{D}'} \lambda_{n\phi} \right) \right\|_1 \leq \frac{2}{N} \max_n \sum_{j=0}^{J} \sum_{\phi\in\Phi_j} \|\lambda_{n\phi}\|_1 \triangleq \Delta^{\mathrm{dp-fm}},$$

and proposed an ϵ-differentially private method that adds Laplace noise with variance $2\left(\frac{\Delta^{\mathrm{dp-fm}}}{\epsilon}\right)^2$ to each $\sum_{\mathbb{D}} \lambda_{n\phi}$ for all $\phi \in \Phi_j$ and for all $j \in \{0, \ldots, J\}$. In the following, we propose an improved functional mechanism that employs a tighter characterization of the sensitivities and can be extended to incorporate the CAPE protocol for the decentralized settings.

8.2 Improved Functional Mechanism

Our method is an improved version of the functional mechanism [143]. We use the Gaussian mechanism [48] for computing the (ϵ, δ)-differentially private approximate of $f_D(\mathbf{w})$. This gives a weaker privacy guarantee than the original functional mechanism [143], which used Laplace noise for ϵ-differential privacy. Our decentralized function computation method (described in Section 8.3.2) benefits from the fact that linear combinations of Gaussians are Gaussian. In other words, the proposed CAPE and the decentralized functional mechanism rely on the Gaussianity of the noise. Now, instead of computing the $\mathcal{L}_2$-sensitivity of $\lambda_{n\phi}$, we define an *array* Λ_j that contains $\frac{1}{N}\sum_{n=1}^{N} \lambda_{n\phi}$

as its entries for all $\phi(\mathbf{w}) \in \Phi_j$. We used the term "array" instead of vector or matrix or tensor because the dimension of Λ_j depends on the cardinality $|\Phi_j|$ of the set Φ_j. We can represent Λ_j as a scalar for $j = 0$ (because $\Phi_0 = \{1\}$), as a D-dimensional vector for $j = 1$ (because $\Phi_1 = \{w_d : d \in [D]\}$), and as a $D \times D$ matrix for $j = 2$ (because $\Phi_2 = \{w_{d_1} w_{d_2} : d_1, d_2 \in [D]\}$). Additionally, we use $\bar{\phi}_j$ to denote the array containing all $\phi(\mathbf{w}) \in \Phi_j$ as its entries. The arrays $\bar{\phi}_j$ and Λ_j have the same dimensions and number of elements. Rewriting the objective, we observe

$$f_D(\mathbf{w}) = \sum_{j=0}^{J} \sum_{\phi \in \Phi_j} \left(\frac{1}{N} \sum_{n=1}^{N} \lambda_{n\phi} \right) \phi(\mathbf{w}) = \sum_{j=0}^{J} \langle \Lambda_j, \bar{\phi}_j \rangle. \tag{8.3}$$

We define the $\mathcal{L}_2$-sensitivity of Λ_j as

$$\Delta_j = \max_{\mathbb{D}, \mathbb{D}'} \| \Lambda_j^{\mathbb{D}} - \Lambda_j^{\mathbb{D}'} \|_F, \tag{8.4}$$

where $\Lambda_j^{\mathbb{D}}$ and $\Lambda_j^{\mathbb{D}'}$ are computed on neighboring datasets $\mathbb{D}$ and $\mathbb{D}'$, respectively. Now, we use the Gaussian mechanism to get an (ϵ, δ)-differentially private approximation to Λ_j as: $\hat{\Lambda}_j = \Lambda_j + e_j$, where e_j is a noise array with the same dimensions as Λ_j and its entries are drawn i.i.d. $\sim \mathcal{N}(0, \tau_j^2)$. Here $\tau_j = \frac{\Delta_j}{\epsilon} \sqrt{2 \log \frac{1.25}{\delta}}$. As the function $f_D(\mathbf{w})$ depends on the data only through $\{\Lambda_j\}$, the computation

$$\hat{f}_D(\mathbf{w}) = \sum_{j=0}^{J} \langle \hat{\Lambda}_j, \bar{\phi}_j \rangle \tag{8.5}$$

satisfies (ϵ, δ) differential privacy. We show the proposed improved functional mechanism in Algorithm 8.1.

Theorem 8.1. *Consider Algorithm 8.1 with input parameters (ϵ, δ) and the function $f_D(\mathbf{w})$ represented as* (8.3)*. Then Algorithm 8.1 computes an (ϵ, δ) differentially private approximation $\hat{f}_D(\mathbf{w})$ to $f_D(\mathbf{w})$. Consequently, the minimizer $\hat{\mathbf{w}}^* = \arg\min_{\mathbf{w}} \hat{f}_D(\mathbf{w})$ satisfies (ϵ, δ)-differential privacy.*

Proof. The proof of Theorem 8.1 follows from the fact that the function $\hat{f}_D(\mathbf{w})$ depends on the data samples only through $\{\hat{\Lambda}_j\}$. The computation of $\{\hat{\Lambda}_j\}$ is (ϵ, δ)-differentially

private by the Gaussian mechanism [50, 48], so the release of $\hat{f}_D(\mathbf{w})$ satisfies (ϵ, δ)-differential privacy. One way to rationalize this is to consider that the probability of the event of selecting a particular set of $\{\hat{\Lambda}_j\}$ is the same as the event of formulating a function $\hat{f}_D(\mathbf{w})$ with that set of $\{\hat{\Lambda}_j\}$. Therefore, it suffices to consider the joint density of the $\{\hat{\Lambda}_j\}$ and find an upper bound on the ratio of the joint densities of the $\{\hat{\Lambda}_j\}$ under $\mathbb{D}$ and $\mathbb{D}'$. As we employ the Gaussian mechanism to compute $\{\hat{\Lambda}_j\}$, the ratio is upper bounded by $\exp(\epsilon)$ with probability at least $1 - \delta$. Therefore, the release of $\hat{f}_D(\mathbf{w})$ satisfies (ϵ, δ)-differential privacy. Furthermore, differential privacy is closed under post processing. Therefore, the computation of the minimizer $\hat{\mathbf{w}}^* = \arg\min_{\mathbf{w}} \hat{f}_D(\mathbf{w})$ also satisfies (ϵ, δ)-differential privacy. □

8.2.1 Example – Linear Regression

In this section, we demonstrate the proposed improved functional mechanism in the centralized setting using a linear regression problem. We show that Algorithm 8.1 achieves much better utility at the price of a weaker privacy guarantee ((ϵ, δ)-differential privacy vs ϵ-differential privacy). The main reason for this performance improvement is due to defining the sensitivities of Λ_j separately for each j, instead of using an uniform conservative upper-bound $\Delta^{\text{dp}-\text{fm}}$ (as in [143]).

We have a dataset $\mathbb{D}$ with N samples of the form $(\mathbf{x}_n, y_n)$, where $\mathbf{x}_n \in \mathbb{R}^D$ is the feature vector and $y_n \in \mathbb{R}$ is the response. Without loss of generality, we assume that $\|\mathbf{x}_n\|_2 \leq 1$ and $y_n \in [-1, 1]$. We want to find a vector $\mathbf{w} \in \mathbb{R}^D$ such that $\mathbf{x}_n^\top \mathbf{w} \approx y_n$ for all $n \in [N]$. The cost function $f : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}^+$ due to each sample and a particular $\mathbf{w}$ is the squared loss: $f(\mathbf{x}_n, \mathbf{w}) = \left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2$. The empirical average cost function is defined as

$$f_D(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x}_n, \mathbf{w}) = \frac{1}{N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_F^2 , \tag{8.6}$$

where $\mathbf{y} \in \mathbb{R}^N$ contains all y_n and $\mathbf{X} \in \mathbb{R}^{N \times D}$ contains $\mathbf{x}_n^\top$ as its rows. By simple

algebra, it can be shown that

$$f_D(\mathbf{w}) = \left(\frac{1}{N}\sum_{n=1}^{N} y_n^2\right) + \sum_{d=1}^{D}\left(-\frac{2}{N}\sum_{n=1}^{N} y_n x_{nd}\right) w_d + \sum_{d_1=1}^{D}\sum_{d_2=1}^{D}\left(\frac{1}{N}\sum_{n=1}^{N} x_{nd_1}x_{nd_2}\right) w_{d_1}w_{d_2},$$

where x_{pq} is the q-th element of the p-th sample vector or $x_{pq} = [\mathbf{X}]_{pq}$. Now, we have

$$\Lambda_0 = \frac{1}{N}\sum_{n=1}^{N} y_n^2 = \frac{1}{N}\|\mathbf{y}\|_2^2$$

$$\Lambda_1 = -\frac{2}{N}\begin{bmatrix} \sum_{n=1}^{N} y_n x_{n1} \\ \vdots \\ \sum_{n=1}^{N} y_n x_{nD} \end{bmatrix} = -\frac{2}{N}\left(\mathbf{1}^\top \mathrm{diag}(\mathbf{y})\mathbf{X}\right)^\top$$

$$\Lambda_2 = \frac{1}{N}\begin{bmatrix} \sum_{n=1}^{N} x_{n1}^2 & \cdots & \sum_{n=1}^{N} x_{n1}x_{nD} \\ \vdots & \ddots & \vdots \\ \sum_{n=1}^{N} x_{nD}x_{n1} & \cdots & \sum_{n=1}^{N} x_{nD}^2 \end{bmatrix} = \frac{\mathbf{X}^\top\mathbf{X}}{N},$$

where $\mathbf{1}$ is an N-dimensional vector of all 1's and $\mathrm{diag}(\mathbf{y})$ is an $N \times N$ diagonal matrix with the elements of $\mathbf{y}$ as diagonal entries. Additionally, we write out the $\{\bar{\phi}_j\}$ for

completeness

$$\bar{\phi}_0 = 1,$$

$$\bar{\phi}_1 = \begin{bmatrix} w_1 \\ \vdots \\ w_D \end{bmatrix}, \text{ and}$$

$$\bar{\phi}_2 = \begin{bmatrix} w_1^2 & w_1 w_2 & \cdots & w_1 w_D \\ \vdots & \vdots & \ddots & \vdots \\ w_D w_1 & w_D w_2 & \cdots & w_D^2 \end{bmatrix}.$$

Therefore, we can express $f_D(\mathbf{w})$ as $f_D(\mathbf{w}) = \sum_{j=0}^{2} \langle \Lambda_j, \bar{\phi}_j \rangle$. Now, we focus on finding the sensitivities of $\{\Lambda_j\}$. Let us consider a neighboring dataset $\mathbb{D}'$ which contains the same tuples as $\mathbb{D}$, except for the last one, i.e. $(\mathbf{x}'_N, y'_N)$. We have

$$\left| \Lambda_0^{\mathbb{D}} - \Lambda_0^{\mathbb{D}'} \right| = \frac{1}{N} \left(y_N^2 - {y'_N}^2 \right) \leq \frac{1}{N} \triangleq \Delta_0,$$

where the inequality follows from the fact that $y_n \in [-1, 1]$. Next, we observe

$$\left\| \Lambda_1^{\mathbb{D}} - \Lambda_1^{\mathbb{D}'} \right\|_2 = \frac{2}{N} \|\mathbf{a} - \mathbf{a}'\|_2 \leq \frac{4}{N} \triangleq \Delta_1,$$

where $\mathbf{a} = y_N \mathbf{x}_N$ and $\mathbf{a}' = y'_N \mathbf{x}'_N$. Here, we used the inequality $\|\mathbf{a}\|_2 \leq 1$ as $-1 \leq y_n \leq 1$, $\|\mathbf{x}_n\|_2 \leq 1$. Similarly, $\|\mathbf{a}'\|_2 \leq 1$. Finally, we observe

$$\begin{aligned} \left\| \Lambda_2^{\mathbb{D}} - \Lambda_2^{\mathbb{D}'} \right\|_F &= \frac{1}{N} \left\| \mathbf{X}^\top \mathbf{X} - \mathbf{X}'^\top \mathbf{X}' \right\|_F \\ &= \frac{1}{N} \left\| \mathbf{x}_N \mathbf{x}_N^\top - \mathbf{x}'_N {\mathbf{x}'_N}^\top \right\|_F \\ &\leq \frac{\sqrt{2}}{N} \triangleq \Delta_2, \end{aligned}$$

where the last inequality follows from realizing that the $D \times D$ symmetric matrix

$\mathbf{x}_N\mathbf{x}_N^\top - \mathbf{x}'_N{\mathbf{x}'_N}^\top$ is at most rank-2. Therefore, we can write its eigen-decomposition as:

$$\begin{aligned}\left\|\mathbf{x}_N\mathbf{x}_N^\top - \mathbf{x}'_N{\mathbf{x}'_N}^\top\right\|_F &= \left\|\mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top\right\|_F \\ &\leq \|\mathbf{U}\|_F^2\|\mathbf{\Sigma}\|_F \\ &= \|\mathbf{\Sigma}\|_F \\ &\leq \sqrt{2}.\end{aligned}$$

Now that we have computed the $\mathcal{L}_2$-sensitivities of $\{\Lambda_j\}$, we can compute $\{\hat{\Lambda}_j\}$ and thus, the (ϵ, δ)-differentially private approximation $\hat{f}_D(\mathbf{w})$ following Algorithm 8.1. Note that if we employed the sensitivity computation technique as in [143], the sensitivity for each entry of Λ_j would be $\Delta^{\text{dp-fm}} = \frac{2(D+1)^2}{N}$, which is orders of magnitude larger than Δ_j for any meaningful $D > 1$ and for all $j \in \{0, \ldots, J\}$. Therefore, with the sensitivity computation proposed in this work, we can achieve $\hat{f}_D(\mathbf{w})$ with much less noise. This results in a more accurate privacy-preserving estimate of the optimal $\hat{\mathbf{w}}^* = \arg\min_{\mathbf{w}} \hat{f}_D(\mathbf{w})$, which we demonstrate empirically in Section 8.4.

8.2.2 Example – Logistic Regression

In this section, we demonstrate the proposed improved functional mechanism for a logistic regression problem in the centralized setting. As in Section 8.2.1, we show that Algorithm 8.1 achieves much better utility at the price of a weaker privacy guarantee ((ϵ, δ)-differential privacy vs ϵ-differential privacy). The main reason for this performance improvement is due to defining the sensitivities of Λ_j separately for each j, instead of using an uniform conservative upper-bound $\Delta^{\text{dp-fm}}$ (as in [143]).

We have a dataset $\mathbb{D}$ with N samples. Each sample is a tuple $(\mathbf{x}_n, y_n)$, where $\mathbf{x}_n \in \mathbb{R}^D$ is the feature vector and $y_n \in \{0, 1\}$ is the label. Without loss of generality, we assume that $\|\mathbf{x}_n\|_2 \leq 1$. We want to find a vector $\mathbf{w} \in \mathbb{R}^D$ such that $\mathcal{I}(\mathbf{x}_n^\top \mathbf{w} \geq 0)$ gives the label y_n for all $n \in [N]$, where $\mathcal{I}(\cdot)$ denotes the indicator function. The cost function due to each sample and a particular $\mathbf{w}$ is $f : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$ and is defined as

the logistic loss:

$$f(\mathbf{x}_n, \mathbf{w}) = \log\left(1 + \exp\left(\mathbf{x}_n^\top \mathbf{w}\right)\right) - y_n \mathbf{x}_n^\top \mathbf{w}. \tag{8.7}$$

The empirical average cost function is defined as

$$f_D(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \log\left(1 + \exp\left(\mathbf{x}_n^\top \mathbf{w}\right)\right) - y_n \mathbf{x}_n^\top \mathbf{w}. \tag{8.8}$$

It is not readily apparent how to express this logistic loss function in the form of (8.2). Zhang et al. [143] derived an approximate polynomial form of $f_D(\mathbf{w})$, denoted by $\tilde{f}_D(\mathbf{w})$, using a Taylor series expansion. Each function $f(\mathbf{x}_n, \mathbf{w})$ can be represented as

$$f(\mathbf{x}_n, \mathbf{w}) = \sum_{m=1}^{M} f_m\left(g_m\left(\mathbf{x}_n, \mathbf{w}\right)\right)$$

for some functions $f_m(\cdot)$ and $g_m(\cdot)$, where each $g_m(\cdot)$ is a monomial of $\{w_d\}$. Using the Taylor expansion of $f_m(\cdot)$ around any real z_m, we have

$$f(\mathbf{x}_n, \mathbf{w}) = \sum_{m=1}^{M}\sum_{j=0}^{\infty} \frac{f_m^{(j)}(z_m)}{j!}\left(g_m\left(\mathbf{x}_n, \mathbf{w}\right) - z_m\right)^j,$$

where $f_m^{(j)}(\cdot)$ is the j-th derivative of $f_m(\cdot)$. Now, for the logistic loss given in (8.7), we have [143] that

$$\begin{aligned} g_1(\mathbf{x}_n, \mathbf{w}) &= \mathbf{x}_n^\top \mathbf{w} \\ f_1(z) &= \log(1 + \exp(z)) \\ g_2(\mathbf{x}_n, \mathbf{w}) &= y_n \mathbf{x}_n^\top \mathbf{w} \\ f_2(z) &= -z. \end{aligned}$$

Therefore, the empirical average cost can be written as

$$f_D(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{2}\sum_{j=0}^{\infty} \frac{f_m^{(j)}(z_m)}{j!}\left(g_m\left(\mathbf{x}_n, \mathbf{w}\right) - z_m\right)^j.$$

Zhang et al. [143] approximated this infinite sum for $j = 0, 1, 2$ and showed analysis for the excess error of the approximation. For $J = 2$, the approximation error is a small constant. Now, for $m = 1, 2$ and $j = 0, 1, 2$, the approximate empirical average cost function can be written as

$$\tilde{f}_D(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{2} \sum_{j=0}^{2} \frac{f_m^{(j)}(z_m)}{j!} \left(g_m \left(\mathbf{x}_n, \mathbf{w} \right) - z_m \right)^j .$$

Using the expressions of $f_m^{(j)}$ and some simple algebra, it can be shown that

$$\begin{aligned} \tilde{f}_D(\mathbf{w}) = \log 2 &+ \sum_{d=1}^{D} \left(\frac{1}{N} \sum_{n=1}^{N} \left(\frac{1}{2} - y_n \right) x_{nd} \right) w_d \\ &+ \sum_{d_1=1}^{D} \sum_{d_2=1}^{D} \left(\frac{1}{8N} \sum_{n=1}^{N} x_{nd_1} x_{nd_2} \right) w_{d_1} w_{d_2}, \end{aligned}$$

where x_{pq} is the q-th element of the p-th sample vector. Using our definitions of Λ_j as before, we have

$$\Lambda_0 = \log 2$$

$$\Lambda_1 = \frac{1}{N} \begin{bmatrix} \sum_{n=1}^{N} \left(\frac{1}{2} - y_n \right) x_{n1} \\ \vdots \\ \sum_{n=1}^{N} \left(\frac{1}{2} - y_n \right) x_{nD} \end{bmatrix} = \frac{\left(\mathbf{1}^\top \mathrm{diag} \left(\frac{1}{2} - \mathbf{y} \right) \mathbf{X} \right)^\top}{N}$$

$$\Lambda_2 = \frac{1}{8N} \begin{bmatrix} \sum_{n=1}^{N} x_{n1}^2 & \cdots & \sum_{n=1}^{N} x_{n1} x_{nD} \\ \vdots & \ddots & \vdots \\ \sum_{n=1}^{N} x_{nD} x_{n1} & \cdots & \sum_{n=1}^{N} x_{nD}^2 \end{bmatrix} = \frac{\mathbf{X}^\top \mathbf{X}}{8N},$$

where $\mathbf{1}$ is an N-dimensional vector of all 1's and $\mathrm{diag}\left(\frac{1}{2} - \mathbf{y} \right)$ is an $N \times N$ diagonal matrix with the elements of $\frac{1}{2} - \mathbf{y}$ as the diagonal entries. If we express the $\{\bar{\phi}_j\}$ as in

the linear regression example (Section 8.2.1), we can write $\tilde{f}_D(\mathbf{w})$ as

$$\tilde{f}_D(\mathbf{w}) = \sum_{j=0}^{2} \langle \Lambda_j, \bar{\phi}_j \rangle. \tag{8.9}$$

Now, we focus on finding the sensitivities of $\{\Lambda_j\}$. Let us consider a neighboring dataset $\mathbb{D}'$ which contains the same tuples as $\mathbb{D}$, except for the last one, i.e. $(\mathbf{x}'_N, y'_N)$. We have

$$\left| \Lambda_0^{\mathbb{D}} - \Lambda_0^{\mathbb{D}'} \right| = 0 \triangleq \Delta_0.$$

Next, we observe

$$\begin{aligned}
\left\| \Lambda_1^{\mathbb{D}} - \Lambda_1^{\mathbb{D}'} \right\|_2 &= \frac{1}{N} \left\| \left(\frac{1}{2} - y_N \right) \mathbf{x}_N - \left(\frac{1}{2} - y'_N \right) \mathbf{x}'_N \right\|_2 \\
&\leq \frac{2}{N} \left\| \left(\frac{1}{2} - y_N \right) \mathbf{x}_N \right\|_2 \\
&\leq \frac{2}{N} \frac{1}{2} \|\mathbf{x}_N\|_2 \\
&\leq \frac{1}{N} \triangleq \Delta_1,
\end{aligned}$$

where we used the inequality $\left\| \left(\frac{1}{2} - y_N \right) \mathbf{x}_N \right\|_2 \leq \frac{1}{2} \|\mathbf{x}_N\|_2 \leq \frac{1}{2}$ because, $y_n \in \{0, 1\}$ and $\|\mathbf{x}_n\|_2 \leq 1$. Finally, we observe

$$\begin{aligned}
\left\| \Lambda_2^{\mathbb{D}} - \Lambda_2^{\mathbb{D}'} \right\|_F &= \frac{1}{8N} \left\| \mathbf{X}^\top \mathbf{X} - \mathbf{X}'^\top \mathbf{X}' \right\|_F \\
&= \frac{1}{8N} \left\| \mathbf{x}_N \mathbf{x}_N^\top - \mathbf{x}'_N \mathbf{x}'_N{}^\top \right\|_F \\
&\leq \frac{\sqrt{2}}{8N} \triangleq \Delta_2,
\end{aligned}$$

where the last inequality follows from realizing that the $D \times D$ symmetric matrix $\mathbf{x}_N \mathbf{x}_N^\top - \mathbf{x}'_N \mathbf{x}'_N{}^\top$ is at-most rank-2 and $\left\| \mathbf{x}_N \mathbf{x}_N^\top - \mathbf{x}'_N \mathbf{x}'_N{}^\top \right\|_F \leq \sqrt{2}$, as shown before (Section 8.2.1). Now that we have computed the $\mathcal{L}_2$-sensitivities of $\{\Lambda_j\}$, we can compute $\{\hat{\Lambda}_j\}$ and thus, the (ϵ, δ)-differentially private approximation of $\tilde{f}_D(\mathbf{w})$ following Algorithm 8.1 ($\tilde{f}_D(\mathbf{w})$ is the input to Algorithm 8.1 for logistic regression). Note that as with linear regression, the sensitivity computation technique for ϵ-differential privacy [143] for each entry of Λ_j would be $\Delta^{\text{dp-fm}} = \frac{1}{N} \left(\frac{D^2}{4} + 3D \right)$, which is orders of

magnitude larger than Δ_j for any $D > 1$ and for all $j \in \{0, \ldots, J\}$. Therefore, with the sensitivity computation proposed in this work, we can achieve $\hat{f}_D(\mathbf{w})$ from $\tilde{f}_D(\mathbf{w})$ with much less noise. This would certainly result in a more accurate privacy-preserving estimate of the optimal $\hat{\mathbf{w}}^* = \arg\min_{\mathbf{w}} \hat{f}_D(\mathbf{w})$. However, one cost of the performance improvement, for both linear and logistic regression applications, is the weakening of the privacy guarantee from ϵ-differential privacy to (ϵ, δ)-differential privacy.

8.3 Decentralized Functional Mechanism

In this section, we first we describe a conventional approach. We show that the conventional approach would always result in sub-optimal performance due to larger noise added to the cost function. Then we present the proposed algorithm (capeFM) in detail (see Algorithm 8.2).

Let us consider a cost function $f_D(\mathbf{w}) : \mathbb{R}^D \mapsto \mathbb{R}$ that depends on private data distributed across S sites. A central aggregator (see Figure 3.1) wishes to find the minimizer $\mathbf{w}^*$ of $f_D(\mathbf{w})$. This is a common scenario in decentralized machine learning. Now, the aggregator is not trusted and the sites may collude with an adversary to learn information about the other sites. Since computing the $\mathbf{w}^*$ by minimizing the expected cost/loss involves the sensitive information of the local datasets, we need to ensure that $\mathbf{w}^*$ is computed in a privacy-preserving way. In particular, we want to develop an algorithm to compute the (ϵ, δ) differentially private approximate to $\mathbf{w}^*$, denoted $\hat{\mathbf{w}}^*$, in a decentralized setting that produces a result as close as possible to the non-private pooled $\mathbf{w}^*$.

Similar to the setting in Section 3.1, let us assume that each site $s \in [S]$ holds a dataset $\mathbb{D}_s$ of N_s samples $\mathbf{x}_{s,n} \in \mathbb{R}^D$. The total sample size across all sites is $N = \sum_{s=1}^{S} N_s$. The cost incurred by a $\mathbf{w} \in \mathbb{R}^D$ due to one data sample $\mathbf{x}_{s,n}$ is $f(\mathbf{x}_{s,n}; \mathbf{w}) : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$. We need to minimize the average cost to find the optimal $\mathbf{w}^*$. The empirical average cost for a particular $\mathbf{w}$ over all the samples is expressed as (8.1).

Therefore, we have

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} f_D(\mathbf{w}) = \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{s=1}^{S} \sum_{n=1}^{N_s} f(\mathbf{x}_{s,n}; \mathbf{w}).$$

8.3.1 Conventional Approach

Recall the decentralized setting, where we have S different sites and a central aggregator, as depicted in Figure 3.1. The aggregator is not trusted. The goal is to compute the differentially private minimizer $\hat{\mathbf{w}}^*$ in this decentralized setting. We assume that the datasets in the sites ($\mathbb{D}_s$ for $s \in [S]$) are disjoint and $N_s = \frac{N}{S}$. Recall that the cost function $f_D(\mathbf{w})$ depends on the data only through the $\{\Lambda_j\}$ for $j \in \{0, \ldots, J\}$. One naïve way to compute $\hat{\mathbf{w}}^*$ at the aggregator is the following: the sites use local data to compute and send the (ϵ, δ) differentially private approximates of $\{\Lambda_j^s\}$, denoted $\{\hat{\Lambda}_j^s\}$, to the aggregator. The aggregator combines these quantities to compute $\{\hat{\Lambda}_j\}$ and hence $\hat{f}_D(\mathbf{w})$. The aggregator can then compute and release $\hat{\mathbf{w}}^* = \arg\min_{\mathbf{w}} \hat{f}_D(\mathbf{w})$. More specifically, each site s computes $\hat{\Lambda}_j^s = \Lambda_j^s + e_j^s$, where e_j^s is an array with the same dimensions as Λ_j^s and the entries are drawn i.i.d. $\sim \mathcal{N}(0, {\tau_j^s}^2)$. Here,

$$\tau_j^s = \frac{\Delta_j^s}{\epsilon} \sqrt{2 \log \frac{1.25}{\delta}},$$

and $\Delta_j^s = \max_{\mathbb{D}_s, \mathbb{D}'_s} \|\Lambda_j^{s,\mathbb{D}_s} - \Lambda_j^{s,\mathbb{D}'_s}\|_F$, where $\Lambda_j^{s,\mathbb{D}_s}$ and $\Lambda_j^{s,\mathbb{D}'_s}$ are computed on neighboring datasets $\mathbb{D}_s$ and $\mathbb{D}'_s$, respectively. Recall that $\Lambda_j^{s,\mathbb{D}_s}$ (and similarly $\Lambda_j^{s,\mathbb{D}'_s}$) is an array of $|\Phi_j|$ elements containing $\frac{1}{N_s} \sum_{n=1}^{N_s} \lambda_{n\phi}$ as its entries for all $\phi(\mathbf{w}) \in \Phi_j$. The sites send these $\{\hat{\Lambda}_j^s\}$ to the aggregator. The aggregator then computes

$$\hat{\Lambda}_j^{\text{conv}} = \frac{1}{S} \sum_{s=1}^{S} \hat{\Lambda}_j^s = \frac{1}{S} \sum_{s=1}^{S} \Lambda_j^s + \frac{1}{S} \sum_{s=1}^{S} e_j^s.$$

The variance of the estimator $\hat{\Lambda}_j^{\text{conv}}$ is $S \cdot \frac{{\tau_j^s}^2}{S^2} = \frac{{\tau_j^s}^2}{S} \triangleq {\tau_j^{\text{conv}}}^2$. However, if we had all the data samples at the aggregator (centralized/pooled data scenario), we could compute the (ϵ, δ)-differentially private approximates of $\{\Lambda_j\}$ as $\hat{\Lambda}_j = \Lambda_j + e_j$, where e_j is an array with the same dimensions as Λ_j and the entries are drawn i.i.d. $\sim \mathcal{N}(0, {\tau_j^{\text{pool}}}^2)$.

Algorithm 8.2 Proposed Decentralized Functional Mechanism (capeFM)

Require: Data samples $\{\mathbf{x}_{s,n}\}$; cost function $f_D(\mathbf{w})$ as in (8.3); local noise variances $\{\tau_j^s\}$ for all $j \in \{0, \ldots, J\}$

for $s = 1, \ldots, S$ **do**
  for $j = 0, 1, \ldots, J$ **do**
    Compute Λ_j as described in Section 8.2
    Generate e_j^s according to Algorithm 3.1 (entrywise)
    Compute ${\tau_{jg}^s}^2 \leftarrow \frac{{\tau_j^s}^2}{S}$
    Generate g_j^s with entries i.i.d. $\sim \mathcal{N}(0, {\tau_{jg}^s}^2)$
    Compute $\hat{\Lambda}_j^s \leftarrow \Lambda_j^s + e_j^s + g_j^s$
  end for
end for
At the central aggregator, compute for all $j \in \{0, \ldots, J\}$: $\hat{\Lambda}_j \leftarrow \frac{1}{S}\sum_{s=1}^{S} \hat{\Lambda}_j^s$
Compute $\hat{f}_D(\mathbf{w}) \leftarrow \sum_{j=0}^{J} \langle \hat{\Lambda}_j, \bar{\phi}_j \rangle$
return $\hat{f}_D(\mathbf{w})$

The noise standard deviation τ_j^{pool} is given by:

$$\tau_j^{\text{pool}} = \frac{\Delta_j^{\text{pool}}}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}},$$

where $\Delta_j^{\text{pool}} = \max_{\mathbb{D},\mathbb{D}'} \|\Lambda_j^{\mathbb{D}} - \Lambda_j^{\mathbb{D}'}\|_F$. Now, $\Lambda_j^{\mathbb{D}}$ is an $|\Phi_j|$-dimensional array with entries $\frac{1}{N}\sum_{n=1}^{N} \lambda_{n\phi}$ for all $\phi(\mathbf{w}) \in \Phi_j$. Clearly $\frac{\Delta_j^{\text{pool}}}{\Delta_j^s} = \frac{1/N}{1/N_s} = \frac{1}{S}$, which implies $\tau_j^{\text{pool}} = \frac{\tau_j^s}{S}$. For this case, we observe the ratio $\frac{{\tau_j^{\text{pool}}}^2}{{\tau_j^{\text{conv}}}^2} = \frac{{\tau_j^s}^2/S^2}{{\tau_j^s}^2/S} = \frac{1}{S}$, which is exactly the same as that of the decentralized averaging problem of Section 3.2.

8.3.2 Proposed capeFM Algorithm

Our proposed method, capeFM, is described in Algorithm 8.2 and is based on the CAPE scheme described in Section 3.3. We exploit the correlated noise to achieve the same performance of the pooled data case (Lemma 3.1) in the symmetric decentralized setting under the honest-but-curious model for the sites. Recall our assumption that all parties follow the protocol and the number of colluding sites is not more than $\lceil S/3 \rceil - 1$. The sites collectively generate the noise e_j^s with entries i.i.d. $\sim \mathcal{N}(0, {\tau_{je}^s}^2)$, according to Algorithm 3.1, such that $\sum_{s=1}^{S} e_j^s = 0$ holds for all $j \in \{0, \ldots, J\}$. The local sites also generate noise g_j^s with entries i.i.d. $\sim \mathcal{N}(0, {\tau_{jg}^s}^2)$. From each site s, we release (or send

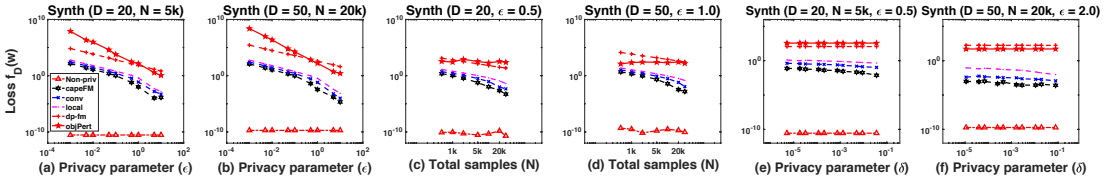

Figure 8.1: Variation of loss $f_D(\mathbf{w})$ at $\mathbf{w}^*$ for synthetic datasets. (a)–(b): with ϵ. (c)–(d): with total samples N. (e)–(f): with δ. Fixed param.: $S = 5$.

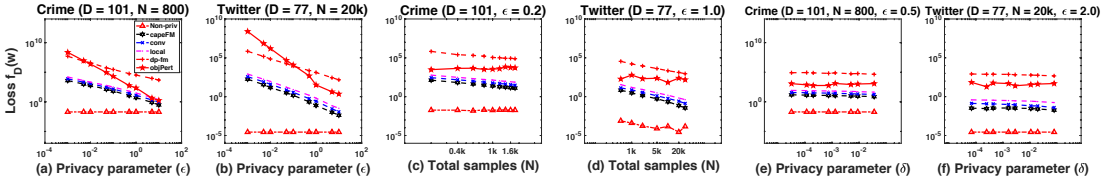

Figure 8.2: Variation of loss $f_D(\mathbf{w})$ at $\mathbf{w}^*$ for two real datasets. (a)–(b): with ϵ. (c)–(d): with total samples N. (e)–(f): with δ. Fixed param.: $S = 5$.

to the central aggregator): $\hat{\Lambda}_j^s = \Lambda_j^s + e_j^s + g_j^s$ for all $j \in \{0, \ldots, J\}$. Note that e_j^s and g_j^s are arrays of the same dimension as Λ_j^s. As described in Section 3.3, we have

$$\tau_{je}^{s\,2} = \left(1 - \frac{1}{S}\right) \tau_j^{s2}, \text{ and } \tau_{jg}^{s\,2} = \frac{\tau_j^{s2}}{S}. \tag{8.10}$$

Now, at the aggregator we compute the following quantity

$$\hat{\Lambda}_j = \frac{1}{S} \sum_{s=1}^{S} \hat{\Lambda}_j^s = \frac{1}{S} \sum_{s=1}^{S} \Lambda_j^s + \frac{1}{S} \sum_{s=1}^{S} g_j^s,$$

because $\sum_s e_j^s = 0$. The aggregator then uses these $\{\hat{\Lambda}_j\}$ to compute $\hat{f}_D(\mathbf{w})$ and release $\hat{\mathbf{w}}^* = \arg\min_{\mathbf{w}} \hat{f}_D(\mathbf{w})$. Privacy of capeFM follows directly from Theorem 3.1. In the symmetric setting (i.e., $N_s = \frac{N}{S}$ and $\tau_j^s = \tau_j$ for all sites $s \in [S]$ and all $j \in \{0, 1, \ldots, J\}$), the noise variance at the aggregator is exactly the same as that of the pooled data scenario (see Lemma 3.1 and Proposition 3.2). Additionally, the performance gain of capeFM over any conventional decentralized functional mechanism is given by Proposition 3.1.

8.4 Experimental Results

In this section, we empirically show the effectiveness of the proposed capeFM algorithm with applications in a linear regression problem in decentralized settings. Our capeFM algorithm is well-suited for decentralized optimization, as we can compute the differentially private approximate $\hat{f}_D(\mathbf{w})$ of the loss function and then use any off-the-shelf optimizer. If the function can be represented as (8.3), we can employ the capeFM algorithm for an even better utility. We present experimental results to show empirical comparison of performance of the proposed capeFM with the existing $\mathsf{dp-fm}$ [143], objective perturbation (objPert) [38] and non-private linear regression ($\mathsf{non-dp\ pool}$) on pooled-data. Note that both $\mathsf{dp-fm}$ and objPert offer the stronger ϵ-differential privacy and are applied to the pooled-data scenario. We also included the performance variation of a conventional differentially private distributed scheme with no correlated noise (conv) and a differentially private linear regression on local (single site) data (local). For both the neural network based classifier and the linear regression problem, we consider the symmetric setting (i.e., $N_s = \frac{N}{S}$ and $\tau_s = \tau$) and show the average performance over 10 independent runs.

We performed experiments on 3 datasets: a *synthetic* dataset ($D = 20$ or $D = 50$) (Synth) generated with a random $\mathbf{w}^*$ and random samples $\mathbf{X}$, the *Communities and Crime* dataset ($D = 101$) [96] (Crime) and the *Buzz in social media* dataset ($D = 77$) [96] (Twitter). We refer the reader to [96] for a detailed description of these real datasets. Now, for each of the datasets, we normalized each feature across the samples to ensure that each feature lies in the range $[-1, 1]$. We also normalized the samples with the maximum $\mathcal{L}_2$ norm in each dataset to ensure $\|\mathbf{x}_n\|_2 \leq 1 \ \forall n$. This preprocessing step is not differentially private but can be modified to satisfy privacy at the cost of some utility. For capeFM, we set $\tau_j^s = \frac{\Delta_j^s}{\epsilon}\sqrt{2\log\frac{1.25}{10^{-5}}}$ for experiments on the synthetic datasets and $\tau_j^s = \frac{\Delta_j^s}{\epsilon}\sqrt{2\log\frac{1.25}{10^{-3}}}$ for experiments on the real datasets, where Δ_j^s is the $\mathcal{L}_2$ sensitivity of the corresponding Λ_j. We use two performance indices for the

synthetic dataset. The first one is the empirical loss

$$f_D(\mathbf{w}) = \frac{1}{N}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2, \tag{8.11}$$

where $\mathbf{X}$ and $\mathbf{y}$ contains all data tuples $(\mathbf{X}_s, \mathbf{y}_s)$ from $s \in [S]$. We compute $f_D(\mathbf{w})$ at the final $\mathbf{w}$, which is found from minimizing the output function of the algorithm. The second one is closeness to the true $\mathbf{w}^*$. If the final $\mathbf{w}$ achieved by minimizing the output function of the algorithm is denoted by $\hat{\mathbf{w}}^*$ then we define: $\text{err}_w = \frac{1}{D}\|\mathbf{w}^* - \hat{\mathbf{w}}^*\|_2$. As the true $\mathbf{w}^*$ is unknown for the real datasets, we only use $f_D(\mathbf{w})$ as the performance index for Crime and Twitter datasets.

8.4.1 Dependence on Privacy Parameter ϵ

First, we explore the trade-off between privacy and utility. We note that, as we employ the Gaussian mechanism, the standard deviation of the added noise is inversely proportional to ϵ – bigger ϵ means higher privacy risk but less noise and thus, better utility. We observe this in our experiments as well. In Figure 8.1(a)–(b), we show the variation of $f_D(\mathbf{w})$ of different algorithms for different values of ϵ on synthetic data. For this experiment, we kept the number of total samples N and the number of sites S fixed. We show the plots for two different feature dimensions: $D = 20$ and $D = 50$. For both of the synthetic datasets, we observe that as ϵ increases (higher privacy risk), the loss $f_D(\mathbf{w})$ decreases. The proposed capeFM reaches very small $f_D(\mathbf{w})$ for some parameter choices and clearly outperforms the dp − fm, objPert, conv and local. One of the reasons that capeFM outperforms conv is the smaller noise variance at the aggregator that we can achieve due to the correlated noise scheme. Moreover, capeFM outperforms dp − fm because dp − fm suffers from a much larger variance at the aggregator (due to the conservative sensitivity computation of $\lambda_{n\phi}$). On the other hand, objPert also entails addition of noise with large variance as the sensitivity of the optimal $\mathbf{w}^*$ is large (to be exact, the sensitivity is 2). Achieving better performance than local is intuitive because including the information from multiple sites to estimate a population parameter always results in better performance than using the data from a single site

only. Additionally, we observe that for datasets with lower dimensional samples, we can use smaller ϵ (i.e., to guarantee lower privacy risk) for the same utility. In Figure 8.2(a)–(b), we show the variation of $f_D(\mathbf{w})$ with ϵ for the Crime and the Twitter datasets. We observe similar variation characteristic of $f_D(\mathbf{w})$ for the real datasets as we observed for the synthetic datasets. Note that, for real datasets, we chose larger τ_s values than synthetic datasets to achieve the similar utility.

8.4.2 Dependence on Total Sample Size N

Next, we investigate the variation in performance with the total sample size N. Intuitively, it should be easier to guarantee smaller privacy risk ϵ and higher utility, when N is large. Figure 8.1(c)–(d) show how $f_D(\mathbf{w})$ decreases as a function of N on synthetic data. The variation with N reinforces the results seen earlier with variation of ϵ. For a fixed ϵ and S, the $f_D(\mathbf{w})$ decreases as we increase N. For sufficiently large N and ϵ, $f_D(\mathbf{w})$ will reach that of the non-private pooled case ($\mathsf{non-dp\ pool}$). We observe a sharper decrease in $f_D(\mathbf{w})$ for lower-dimensional datasets. In Figure 8.2(c)–(d), we show the variation of $f_D(\mathbf{w})$ with N for the Crime and the Twitter datasets. Again, we observe similar variation of $f_D(\mathbf{w})$ for the real datasets as we observed for the synthetic datasets.

8.4.3 Dependence on Privacy Parameter δ

Note that, the proposed capeFM algorithm guarantees (ϵ, δ) differential privacy where (ϵ, δ) satisfy the relation $\delta = 2\frac{\sigma_z}{\epsilon - \mu_z}\phi\left(\frac{\epsilon-\mu_z}{\sigma_z}\right)$. Recall that δ can be considered as the probability that the algorithm releases the private information without guaranteeing privacy. Therefore, we want this to be as small as possible. However, smaller δ also dictates larger noise variance. We explore the variation of performance with different δ with a fixed number of colluding sites $S_C = \lceil \frac{S}{3} \rceil - 1$. In Figure 8.1(e)–(f), we show how $f_D(\mathbf{w})$ varies with varying δ on synthetic data. We observe that if N and δ are too small, the proposed capeFM, conv and local algorithms perform poorly. However, our capeFM algorithm can achieve very good utility for moderate N and δ values, easily

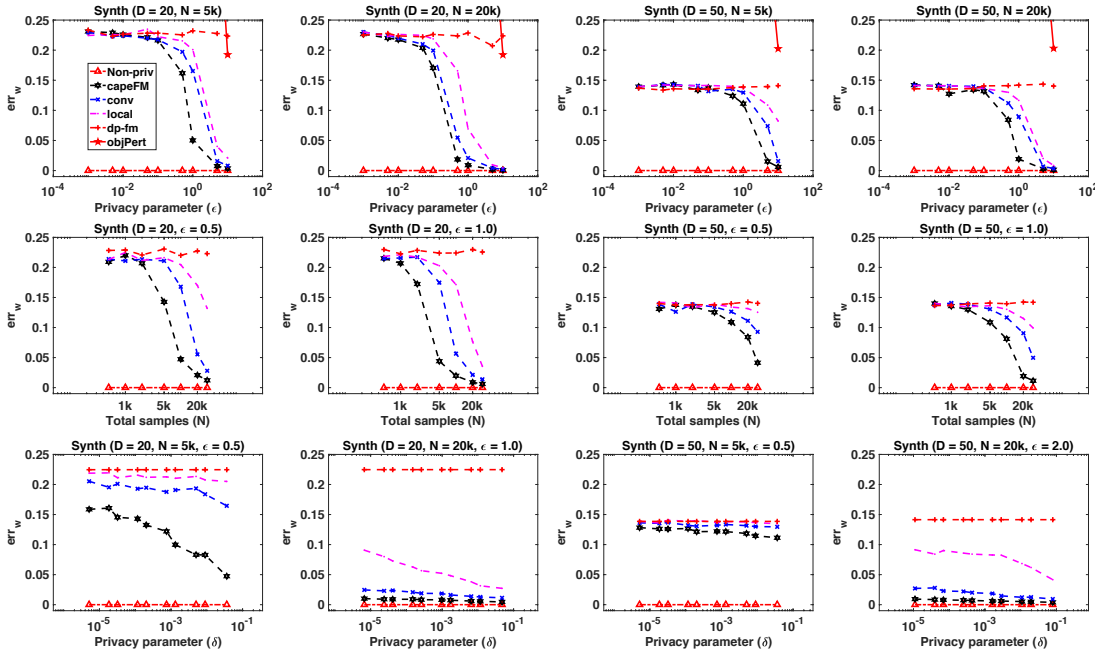

Figure 8.3: Variation of err_w for synthetic datasets. Top-row: with ϵ. Mid-row: with total samples N. Bottom-row: with δ. Fixed parameter: $S = 5$.

outperforming the other differentially private algorithms. We observe in Figure 8.2(e)–(f) similar characteristic for the two real datasets, although the variation of $f_D(\mathbf{w})$ is not as pronounced as for the synthetic datasets. Recall that the $\mathsf{dp-fm}$ and the $\mathsf{objPert}$ algorithms offer pure ϵ-differential privacy guarantee and, therefore, do not vary with δ.

8.4.4 Variation of err_w

Dependence on Privacy Parameter ϵ. In the top-row of Figure 8.3, we show the variation of err_w of different algorithms for different values of ϵ on synthetic data. For this experiment, we kept the number of total samples N and the number of sites S fixed. We show the plots for two different feature dimensions: $D = 20$ and $D = 50$, each with two different sample sizes. For both of the synthetic datasets, we observe that as ϵ increases (higher privacy risk), the err_w decreases. The proposed capeFM reaches very small err_w for some parameter choices and clearly outperforms the $\mathsf{dp-fm}$, $\mathsf{objPert}$, conv and local. One of the reasons that capeFM outperforms conv is the smaller noise variance at the aggregator that we can achieve due to the correlated noise scheme. Moreover,

capeFM outperforms dp − fm because dp − fm suffers from a much larger variance at the aggregator (due to the conservative sensitivity computation of $\lambda_{n\phi}$). On the other hand, objPert also entails addition of noise with large variance as the sensitivity of the optimal $\mathbf{w}^*$ is large (to be exact, the sensitivity is 2). Achieving better performance than local is intuitive because including the information from multiple sites to estimate a population parameter always results in better performance than using the data from a single site only. Additionally, we observe that for datasets with lower dimensional samples, we can use smaller ϵ (i.e., to guarantee lower privacy risk) for the same utility.

Dependence on Total Sample Size N. Next, we investigate the variation in performance with the total sample size N. The middle-row of Figure 8.3 shows how err_w decreases as a function of total sample size N on synthetic data. The variation with N reinforces the results seen earlier with variation of ϵ. For a fixed ϵ and S, err_w decreases as we increase N. For sufficiently large N and ϵ, err_w will reach that of the non-private pooled case (non − dp pool). Again, we observe a sharper decrease in err_w for lower-dimensional datasets. Note that, for the synthetic datasets, the error err_w for objPert is too large to show on the same scale as other algorithms. That is why the err_w curves for objPert do not appear in Figure 8.3 (middle-row).

Dependence on Privacy Parameter δ. In the bottom-row of Figure 8.3, we show how err_w varies with varying δ on synthetic data. We observe that if N and δ are too small, the proposed capeFM, conv and local algorithms perform poorly. However, the proposed capeFM algorithm can achieve very good utility for moderate N and δ values, easily outperforming the other differentially private algorithms. Recall that the dp − fm and the objPert algorithms offer pure ϵ-differential privacy and, therefore, do not vary with δ. Again we observe that, for the synthetic datasets, the error err_w for objPert is too large and do not appear in Figure 8.3 (bottom row).

Chapter 9

Conclusion and Future Directions

This work proposes a novel protocol, CAPE, for decentralized differentially private computations. CAPE is best suited for applications in which private data must be held locally, e.g. in health care research with legal and ethical limitations on the degree of sharing the "raw" data. CAPE can greatly improve the privacy-utility tradeoff when (a) all parties follow the protocol and (b) the number of colluding sites is not more than $\lceil S/3 \rceil - 1$. Our proposed CAPE protocol is based on an estimation-theoretic analysis of the noise addition process for differential privacy and therefore, provides different guarantees than cryptographic approaches such as SMC. We analyzed the privacy guarantees of CAPE in details and provided a scheme for extending CAPE to asymmetric network/privacy-requirements scenarios. In addition to CAPE, we proposed several new algorithms for differentially private matrix/tensor factorization and computation of functions in decentralized settings.

Our capePCA, capeAGN and capeCCA algorithms are well suited for decentralized subspace learning and achieving the same utility as the pooled-data scenario in certain regimes. We empirically compared the performance of the proposed algorithms with those of existing (if any) and conventional decentralized algorithms on synthetic and real data sets. We varied privacy parameters and relevant dataset parameters. The proposed algorithms outperformed the existing and conventional algorithms comfortably and matched the performance of corresponding non-private algorithms for proper parameter choices.

Our proposed capeDJICA algorithm can be applied to jointly learn spatial maps from decentralized neuroimaging data. capeDJICA can offer significant improvement upon our earlier work and achieves the same level of additive noise variance as the pooled

data scenario in the symmetric network setting. We analyzed our capeDJICA algorithm using Rényi differential privacy and provided a better account of the privacy loss per iteration using the moments accountant method. We presented empirical comparison of the performance of capeDJICA with those of existing, pooled and local algorithms on synthetic and real datasets. We varied privacy parameters and relevant dataset parameters to show that the proposed algorithm outperformed the existing and local algorithms comfortably and matched the performance of the non-private algorithm for some parameter choices. By analyzing our proposed algorithm using the moments accountant, we show that we can achieve performance very close to that of non-private algorithm, even for strict privacy requirements.

Finally, our capeFM algorithm can be employed to compute any continuous and differentiable function in the decentralized data scenario. As mentioned before, approximation of the empirical average cost is required in many decentralized optimization problems. We proposed an improved functional mechanism with a new way to compute the associated sensitivities. We analytically showed that the proposed approach for computing the sensitivities offers much less additive noise for two common regression problems – linear regression and logistic regression. Our proposed capeFM can approximate the privacy-preserving empirical average cost such that we can achieve the same utility level as the pooled data scenario in certain regimes. We empirically compared the performance of the proposed algorithms with those of existing and conventional algorithms for a neural-network based classification problem and a decentralized linear regression problem. We varied privacy parameters and relevant dataset (synthetic and real) parameters and showed that the proposed algorithms outperformed the existing and conventional algorithms comfortably – matching the performance of the non-private algorithm for some parameter choices. In general, the proposed algorithms offered very good utility indicating that meaningful privacy can be attained without losing much performance by the virtue of algorithm design.

As for future directions, deployment of the capeDJICA and capeCCA algorithms on existing neuroimaging analysis toolboxes (e.g. COINSTAC [119]) would be of great

practical value. The primary challenge in this task would be to implement the communication heavy zero-sum noise generation. We believe another very interesting future work could be to extend the CAPE framework to fit the optimal Staircase Mechanism [60] or the Podium mechanism [118] for differential privacy. Another possible direction is to extend CAPE to be employable in arbitrary tree-structured networks. As mentioned in Remark 2 (in Section 3.3), we generated the zero-sum noise terms by mapping floating point numbers to a finite field. Another direction of future work would be to address the floating point implementation issues to fit our protocol. Additionally, analyzing the sample complexity bounds for the capeAGN and capeFM algorithms would also be interesting.

Bibliography

[1] *How One of Apple's Key Privacy Safeguards Falls Short*, 2017. [Online]. Available: https://www.wired.com/story/apple-differential-privacy-shortcomings/

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep Learning with Differential Privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 308–318. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978318

[3] J. M. Abowd, "The U.S. Census Bureau Adopts Differential Privacy," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, London, UK, August 19–23 2018. [Online]. Available: https://doi.org/10.1145/3219819.3226070

[4] D. Achlioptas and F. McSherry, "On Spectral Learning of Mixtures of Distributions," in *Int. Conf. on Computational Learning Theory*. Springer, 2005, pp. 458–469. [Online]. Available: http://dx.doi.org/10.1007/11503415_31

[5] E. A. Allen, E. B. Erhardt, E. Damaraju, W. Gruner, J. M. Segall, R. F. Silva, M. Havlicek, S. Rachakonda, J. Fries, R. Kalyanam, A. M. Michael, A. Caprihan, J. A. Turner, R. Eichele, S. Adelsheim, A. D. Bryan, J. Bustillo, V. P. Clark, S. W. Feldstein Ewing, F. Filbey, C. C. Ford, K. Hutchison, R. E. Jung, K. A. Kiehl, P. Kodituwakku, Y. M. Komesu, A. R. Mayer, G. D. Pearlson, J. P. Phillips, J. R. Sadek, M. Stevens, U. Teuscher, R. J. Thoma, and V. D. Calhoun, "A Baseline for the Multivariate Comparison of Resting State Networks," *Frontiers in Systems Neuroscience*, vol. 5, no. 2, 2011. [Online]. Available: http://dx.doi.org/10.3389/fnsys.2011.00002

[6] S. Amari, A. Cichocki, and H. H. Yang, "A New Learning Algorithm for Blind Signal Separation," in *Advances in Neural Information Processing Systems*, 1996, pp. 757–763. [Online]. Available: http://dl.acm.org/citation.cfm?id=2998828.2998935

[7] B. Anandan and C. Clifton, "Laplace Noise Generation for Two-party Computational Differential Privacy," in *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, July 2015, pp. 54–61. [Online]. Available: https://doi.org/10.1109/PST.2015.7232954

[8] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor Decompositions for Learning Latent Variable Models," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2773–2832, Jan. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2697055

[9] R. Arora and K. Livescu, "Multi-view Learning with Supervision for Transformed Bottleneck Features," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 2499–2503. [Online]. Available: https://doi.org/10.1109/ICASSP.2014.6854050

[10] Z.-J. Bai, R. H. Chan, and F. T. Luk, *Principal Component Analysis for Distributed Data Sets with Updating*, Berlin, Heidelberg, 2005, pp. 471–483. [Online]. Available: https://doi.org/10.1007/11573937_51

[11] B. T. Baker, A. Abrol, R. F. Silva, E. Damaraju, A. D. Sarwate, V. D. Calhoun, and S. M. Plis, "Decentralized Temporal Independent Component Analysis: Leveraging fMRI Data in Collaborative Settings," *NeuroImage*, vol. 186, pp. 557 – 569, 2019. [Online]. Available: https://doi.org/10.1016/j.neuroimage.2018.10.072

[12] M.-F. Balcan, V. Kanchanapally, Y. Liang, and D. Woodruff, "Improved Distributed Principal Component Analysis," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, ser. NIPS'14, 2014, pp. 3113–3121. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969033.2969174

[13] V. Balcer and S. Vadhan, "Differential Privacy on Finite Computers," in *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 94. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 43:1–43:21. [Online]. Available: https://doi.org/10.4230/LIPIcs.ITCS.2018.43

[14] B. Balle, J. Bell, A. Gascón, and K. Nissim, "The Privacy Blanket of the Shuffle Model," in *Advances in Cryptology – CRYPTO 2019*, A. Boldyreva and D. Micciancio, Eds. Cham: Springer International Publishing, 2019, pp. 638–667. [Online]. Available: https://doi.org/10.1007/978-3-030-26951-7_22

[15] R. Bassily, K. Nissim, U. Stemmer, and A. Guha Thakurta, "Practical Locally Private Heavy Hitters," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 2288–2296. [Online]. Available: http://papers.nips.cc/paper/6823-practical-locally-private-heavy-hitters.pdf

[16] R. Bassily, A. Smith, and A. Thakurta, "Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds," in *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, ser. FOCS '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 464–473. [Online]. Available: http://dx.doi.org/10.1109/FOCS.2014.56

[17] A. J. Bell and T. J. Sejnowski, "An Information-maximization Approach to Blind Separation and Blind Deconvolution," *Neural Computation*, vol. 7, no. 6, pp. 1129–1159, 1995. [Online]. Available: http://dx.doi.org/10.1162/neco.1995.7.6.1129

[18] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, , and R. Rogers., "Protection Against Reconstruction and its Applications in Private Federated Learning,"

ArXiV, Tech. Rep. arXiv:1812.00984 [stat.ML], June 2018. [Online]. Available: https://arxiv.org/abs/1812.00984

[19] A. Bittau, U. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld, "Prochlo: Strong Privacy for Analytics in the Crowd," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: ACM, 2017, pp. 441–459. [Online]. Available: http://doi.acm.org/10.1145/3132747.3132769

[20] T. Bollerslev, "Generalized Autoregressive Conditional Heteroskedasticity," *J Econometrics*, vol. 31, pp. 307–327, 1986. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.161.7380

[21] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical Secure Aggregation for Federated Learning on User-Held Data," *CoRR*, vol. abs/1611.04482, 2016. [Online]. Available: http://arxiv.org/abs/1611.04482

[22] ——, "Practical Secure Aggregation for Privacy-Preserving Machine Learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 1175–1191. [Online]. Available: http://doi.acm.org/10.1145/3133956.3133982

[23] M. Borga, "Canonical Correlation A Tutorial," 1999. [Online]. Available: https://www.cs.cmu.edu/~tom/10701_sp11/slides/CCA_tutorial.pdf

[24] Y. L. Borgne, S. Raybaud, and G. Bontempi, "Distributed Principal Component Analysis for Wireless Sensor Networks," *CoRR*, vol. abs/1003.1967, 2010. [Online]. Available: http://arxiv.org/abs/1003.1967

[25] L. Bottou, "On-line Learning in Neural Networks," D. Saad, Ed. New York, NY, USA: Cambridge University Press, 1998, ch. On-line Learning and Stochastic Approximations, pp. 9–42. [Online]. Available: http://dl.acm.org/citation.cfm?id=304710.304720

[26] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[27] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1561/2200000016

[28] R. C. Buck, "Nomographic Functions are Nowhere Dense," *Proceedings of the American Mathematical Society*, vol. 85, no. 2, pp. 195–199, June 1982. [Online]. Available: https://dx.doi.org/10.2307/2044280

[29] V. D. Calhoun, V. K. Potluru, R. Phlypo, R. F. Silva, B. A. Pearlmutter, A. Caprihan, S. M. Plis, and T. Adalı, "Independent Component Analysis for Brain fMRI Does Indeed Select for Maximal Independence," *PLoS ONE*, vol. 8, p. e73309, 2013. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0073309

[30] V. Calhoun, T. Adali, N. Giuliani, J. Pekar, K. Kiehl, and G. Pearlson, "Method for Multimodal Analysis of Independent Source Differences in Schizophrenia: Combining Gray Matter Structural and Auditory Oddball Functional Data," *Human Brain Mapping*, vol. 27, no. 1, pp. 47 – 62, 2006. [Online]. Available: http://dx.doi.org/10.1002/hbm.20166

[31] V. D. Calhoun and T. Adali, "Multisubject Independent Component Analysis of fMRI: a Decade of Intrinsic Networks, Default Mode, and Neurodiagnostic Discovery," *IEEE Reviews in Biomedical Engineering*, vol. 5, pp. 60–73, 2012. [Online]. Available: https://doi.org/10.1109/RBME.2012.2211076

[32] V. D. Calhoun, T. Adali, G. D. Pearlson, and J. Pekar, "A Method for Making Group Inferences from Functional MRI Data Using Independent Component Analysis," *Human Brain Mapping*, vol. 14, no. 3, pp. 140–151, 2001. [Online]. Available: https://doi.org/10.1002/hbm.10044

[33] T. Caliński and J. Harabasz, "A Dendrite Method for Cluster Analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974. [Online]. Available: https://doi.org/10.1080/03610927408827101

[34] J. D. Carroll and J.-J. Chang, "Analysis of Individual Differences in Multidimensional Scaling via an n-way Generalization of "Eckart-Young" Decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970. [Online]. Available: http://dx.doi.org/10.1007/BF02310791

[35] K. W. Carter, R. W. Francis, K. Carter, R. Francis, M. Bresnahan, M. Gissler, T. Grønborg, R. Gross, N. Gunnes, G. Hammond *et al.*, "ViPAR: a Software Platform for the Virtual Pooling and Analysis of Research Data," *International journal of epidemiology*, vol. 45, no. 2, pp. 408–416, 2015. [Online]. Available: https://doi.org/10.1093/ije/dyv193

[36] K. Chaudhuri and C. Monteleoni, "Privacy-Preserving Logistic Regression," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 289–296.

[37] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan, "Multi-view Clustering via Canonical Correlation Analysis," in *Proc. of the 26th Annual Int. Conf. on Machine Learning*, ser. ICML '09. ACM, 2009, pp. 129–136. [Online]. Available: https://doi.org/10.1145/1553374.1553391

[38] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially Private Empirical Risk Minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1069–1109, Jul. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2021036

[39] K. L. Clarkson and D. P. Woodruff, "Low-Rank Approximation and Regression in Input Sparsity Time," *J. ACM*, vol. 63, no. 6, pp. 54:1–54:45, Jan. 2017. [Online]. Available: http://doi.acm.org/10.1145/3019134

[40] P. Comon, "Independent Component Analysis, A New Concept?" *Signal Processing*, vol. 36, no. 3, pp. 287 – 314, 1994. [Online]. Available: http://dx.doi.org/10.1016/0165-1684(94)90029-9

[41] P. Comon, G. Golub, L.-H. Lim, and B. Mourrain, "Symmetric Tensors and Symmetric Tensor Rank," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1254–1279, 2008. [Online]. Available: http://dx.doi.org/10.1137/060661569

[42] N. Correa, Y.-O. Li, T. Adali, and V. D. Calhoun, "Examining Associations Between fMRI and EEG Data Using Canonical Correlation Analysis," in *2008 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, May 2008, pp. 1251–1254. [Online]. Available: https://doi.org/10.1109/ISBI.2008.4541230

[43] N. M. Correa, Y. Li, T. Adali, and V. D. Calhoun, "Canonical Correlation Analysis for Feature-Based Fusion of Biomedical Imaging Modalities and Its Application to Detection of Associative Networks in Schizophrenia," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 6, pp. 998–1007, Dec 2008. [Online]. Available: https://doi.org/10.1109/JSTSP.2008.2008265

[44] S. Dasgupta, "Learning Mixtures of Gaussians," *Found. of Comp. Sci.*, pp. 634 – 644, 1999. [Online]. Available: http://dl.acm.org/citation.cfm?id=795665.796496

[45] F. Deleus and M. M. V. Hulle, "Functional Connectivity Analysis of fMRI Data Based on Regularized Multiset Canonical Correlation Analysis," *Journal of Neuroscience Methods*, vol. 197, no. 1, pp. 143 – 157, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jneumeth.2010.11.029

[46] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting Telemetry Data Privately," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3571–3580. [Online]. Available: https://arxiv.org/abs/1712.01524

[47] J. Duchi and R. Rogers, "Lower Bounds for Locally Private Estimation via Communication Complexity," in *Proceedings of the Thirty-Second Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, A. Beygelzimer and D. Hsu, Eds., vol. 99. PMLR, 25–28 Jun 2019, pp. 1161–1191. [Online]. Available: https://arxiv.org/abs/1902.00582

[48] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2013. [Online]. Available: http://dx.doi.org/10.1561/0400000042

[49] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our Data, Ourselves: Privacy Via Distributed Noise Generation," in *Advances in Cryptology (EUROCRYPT 2006)*, vol. 4004. Saint Petersburg, Russia: Springer Verlag, May 2006, pp. 486–503. [Online]. Available: http://dx.doi.org/10.1007/11761679_29

[50] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *Proc. of the Third Conf. on Theory of Cryptography*, 2006, pp. 265–284. [Online]. Available: http://dx.doi.org/10.1007/11681878_14

[51] C. Dwork, K. Talwar, A. Thakurta, and L. Zhang, "Analyze Gauss: Optimal Bounds for Privacy-preserving Principal Component Analysis," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, 2014, pp. 11–20. [Online]. Available: http://dx.doi.org/10.1145/2591796.2591883

[52] F. Eigner, A. Kate, M. Maffei, F. Pampaloni, and I. Pryvalov, "Differentially Private Data Aggregation with Optimal Utility," in *Proceedings of the 30th Annual Computer Security Applications Conference*, ser. ACSAC '14. New York, NY, USA: ACM, 2014, pp. 316–325. [Online]. Available: https://doi.org/10.1145/2664243.2664263

[53] R. Engle, "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation," *Econometrica*, vol. 50, no. 4, pp. 987–1007, 1982. [Online]. Available: https://www.jstor.org/stable/1912773

[54] U. 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. New York, NY, USA: ACM, 2014, pp. 1054–1067. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660348

[55] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2019, pp. 2468–2479. [Online]. Available: https://doi.org/10.1137/1.9781611975482.151

[56] S. A. Esmaeili and F. Huang, "An end-to-end Differentially Private Latent Dirichlet Allocation Using a Spectral Algorithm," *ArXiv e-prints*, May 2018. [Online]. Available: https://ui.adsabs.harvard.edu/link_gateway/2018arXiv180510341E/arxiv:1805.10341

[57] D. Feldman, M. Schmidt, and C. Sohler, "Turning Big Data into Tiny Data: Constant-size Coresets for K-means, PCA and Projective Clustering," in *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '13, 2013, pp. 1434–1453. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627817.2627920

[58] S. Gade and N. H. Vaidya, "Private Learning on Networks," *CoRR*, vol. abs/1612.05236, 2016. [Online]. Available: http://arxiv.org/abs/1612.05236

[59] A. Gaye, Y. Marcon, J. Isaeva, P. LaFlamme, A. Turner, E. M. Jones, J. Minion, A. W. Boyd, C. J. Newby, M.-L. Nuotio *et al.*, "DataSHIELD: Taking the Analysis to the Data, Not the Data to the Analysis," *International Journal of Epidemiology*, vol. 43, no. 6, pp. 1929–1944, 2014. [Online]. Available: https://doi.org/10.1093/ije/dyu188

[60] Q. Geng, P. Kairouz, S. Oh, and P. Viswanath, "The Staircase Mechanism in Differential Privacy," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 7, pp. 1176–1184, 2015. [Online]. Available: https://doi.org/10.1109/JSTSP.2015.2425831

[61] M. Goldenbaum and S. Stanczak, "Robust Analog Function Computation via Wireless Multiple-Access Channels," *IEEE Transactions on Communications*, vol. 61, no. 9, pp. 3863–3877, September 2013. [Online]. Available: https://dx.doi.org/10.1109/TCOMM.2013.072913.120815

[62] M. Goldenbaum, H. Boche, and S. Stańczak, "Harnessing Interference for Analog Function Computation in Wireless Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 61, no. 20, pp. 4893–4906, October 2013. [Online]. Available: https://dx.doi.org/10.1109/TSP.2013.2272921

[63] G. H. Golub and H. Zha, "The Canonical Correlations of Matrix Pairs and Their Numerical Computation," Stanford, CA, USA, Tech. Rep., 1992. [Online]. Available: https://dl.acm.org/citation.cfm?id=891630

[64] S. Goryczka, L. Xiong, and V. Sunderam, "Secure Multiparty Aggregation with Differential Privacy: A Comparative Study," in *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, ser. EDBT '13. New York, NY, USA: ACM, 2013, pp. 155–163. [Online]. Available: http://doi.acm.org/10.1145/2457317.2457343

[65] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, May 2011. [Online]. Available: http://dx.doi.org/10.1137/090771806

[66] S. Han, U. Topcu, and G. J. Pappas, "Differentially Private Distributed Constrained Optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 50–64, Jan 2017. [Online]. Available: https://doi.org/10.1109/TAC.2016.2541298

[67] D. R. Hardoon, S. R. Szedmak, and J. R. Shawe-taylor, "Canonical Correlation Analysis: An Overview with Application to Learning Methods," *Neural Comput.*, vol. 16, no. 12, pp. 2639–2664, Dec. 2004. [Online]. Available: https://doi.org/10.1162/0899766042321814

[68] R. A. Harshman, "Foundations of the PARAFAC Procedure: Models and Conditions for an 'Explanatory' Multi-modal Factor Analysis," *UCLA Working Papers in Phonetics*, vol. 16, no. 1, 1970. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.144.5652

[69] M. Heikkilä, E. Lagerspetz, S. Kaski, K. Shimizu, S. Tarkoma, and A. Honkela, "Differentially Private Bayesian Learning on Distributed Data," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3229–3238. [Online]. Available: https://ui.adsabs.harvard.edu/link_gateway/2017arXiv170301106H/arxiv:1703.01106

[70] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. New York, NY, USA: Cambridge University Press, 2012.

[71] H. Hotelling, "Relations Between Two Sets of Variates," *Biometrika*, vol. 28, no. 3/4, pp. 321–377, 1936. [Online]. Available: https://www.jstor.org/stable/2333955

[72] D. Hsu, S. M. Kakade, and T. Zhang, "A Spectral Algorithm for Learning Hidden Markov Models," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1460 – 1480, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.jcss.2011.12.025

[73] D. J. Hsu and S. M. Kakade, "Learning Mixtures of Spherical Gaussians: Moment Methods and Spectral Decompositions," *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, pp. 11–20, 2013. [Online]. Available: http://doi.acm.org/10.1145/2422436.2422439

[74] F. Huang, S. Matusevych, A. Anandkumar, N. Karampatziakis, and P. Mineiro, "Distributed Latent Dirichlet Allocation via Tensor Factorization," in *NIPS Optimization Workshop*, 2014. [Online]. Available: http://www.opt-ml.org/papers/opt2014_submission_15.pdf

[75] Z. Huang, S. Mitra, and N. Vaidya, "Differentially Private Distributed Optimization," in *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, ser. ICDCN '15. New York, NY, USA: ACM, 2015, pp. 4:1–4:10. [Online]. Available: http://doi.acm.org/10.1145/2684464.2684480

[76] H. Imtiaz and A. D. Sarwate, "Differentially-Private Canonical Correlation Analysis," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Nov 2017, pp. 283–287. [Online]. Available: https://doi.org/10.1109/GlobalSIP.2017.8308649

[77] ——, "Distributed Differentially-Private Algorithms for Matrix and Tensor Factorization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 6, pp. 1449–1464, December 2018. [Online]. Available: https://doi.org/10.1109/JSTSP.2018.2877842

[78] ——, "Differentially Private Distributed Principal Component Analysis," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 2206–2210. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8462519

[79] ——, "Improved Algorithms for Differentially Private Orthogonal Tensor Decomposition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 2201–2205. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8461303

[80] H. Imtiaz, R. Silva, B. Baker, S. M. Plis, A. D. Sarwate, and V. Calhoun, "Privacy-Preserving Source Separation for Distributed Data using Independent Component Analysis," in *2016 Annual Conference on Information Science and Systems (CISS)*, March 2016, pp. 123–127. [Online]. Available: https://dx.doi.org/10.1109/CISS.2016.7460488

[81] H. Imtiaz, J. Mohammadi, and A. D. Sarwate, "Distributed Differentially Private Computation of Functions with Correlated Noise," *CoRR*, 2019. [Online]. Available: http://arxiv.org/abs/1904.10059

[82] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential Privacy and Machine Learning: a Survey and Review," *CoRR*, vol. abs/1412.7584, 2014. [Online]. Available: http://arxiv.org/abs/1412.7584

[83] L. Jing, "Differentially Private M-estimators," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, ser. NIPS'11. USA: Curran Associates Inc., 2011, pp. 361–369. [Online]. Available: http://dl.acm.org/citation.cfm?id=2986459.2986500

[84] P. Kairouz, S. Oh, and P. Viswanath, "Secure Multi-party Differential Privacy," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2008–2016. [Online]. Available: https://papers.nips.cc/paper/6004-secure-multi-party-differential-privacy.pdf

[85] R. Kannan, H. Salmasian, and S. Vempala, "The Spectral Method for General Mixture Models," in *Int. Conf. on Computational Learning Theory*. Springer, 2005, pp. 444–457. [Online]. Available: https://doi.org/10.1007/11503415_3

[86] T. G. Kolda, "Symmetric Orthogonal Tensor Decomposition is Trivial," in *eprint arXiv:1503.01375*, 2015. [Online]. Available: https://arxiv.org/abs/1503.01375

[87] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," *SIAM REVIEW*, vol. 51, no. 3, pp. 455–500, 2009. [Online]. Available: http://dx.doi.org/10.1137/07070111X

[88] A. Kolmogorov, "On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of one Variable and Addition (in Russian)," *Doklady Akademii Nauk*, vol. 114, no. 5, pp. 953–956, 1957. [Online]. Available: http://mi.mathnet.ru/eng/dan22050

[89] L. Lathauwer, B. D. Moor, and J. Vandewalle, "On the Best Rank-1 and Rank-(R1 ,R2 ,. . .,RN) Approximation of Higher-Order Tensors," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1324–1342, Mar. 2000. [Online]. Available: http://dx.doi.org/10.1137/S0895479898346995

[90] J. Le Ny and M. Mohammady, "Differentially Private MIMO Filtering for Event Streams," *IEEE Transactions on Automatic Control*, vol. 63, no. 1, pp. 145–157, Jan 2018. [Online]. Available: https://doi.org/10.1109/TAC.2017.2713643

[91] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998. [Online]. Available: http://dx.doi.org/10.1109/5.726791

[92] C. Li, P. Zhou, L. Xiong, Q. Wang, and T. Wang, "Differentially Private Distributed Online Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, no. 99, pp. 1–1, 2018. [Online]. Available: https://doi.org/10.1109/TKDE.2018.2794384

[93] Y. O. Li, W. Wang, T. Adali, and V. D. Calhoun, "CCA for Joint Blind Source Separation of Multiple Datasets with Application to Group fMRI Analysis," in *2008 IEEE International Conference on Acoustics, Speech*

and Signal Processing, March 2008, pp. 1837–1840. [Online]. Available: https://doi.org/10.1109/ICASSP.2008.4517990

[94] Y. O. Li, T. Adali, W. Wang, and V. D. Calhoun, "Joint Blind Source Separation by Multiset Canonical Correlation Analysis," *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3918–3929, Oct 2009. [Online]. Available: https://doi.org/10.1109/TSP.2009.2021636

[95] Y. Liang, M.-F. Balcan, and V. Kanchanapally, "Distributed PCA and k-Means Clustering." [Online]. Available: http://www.cs.cmu.edu/~ninamf/papers/distributedPCAandCoresets.pdf

[96] M. Lichman, "UCI Machine Learning Repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[97] K. Ligett, S. Neel, A. Roth, B. Waggoner, and S. Z. Wu, "Accuracy First: Selecting a Differential Privacy Level for Accuracy Constrained ERM," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 2563–2573.

[98] S. Limmer, J. Mohammadi, and S. Stańczak, "A Simple Algorithm for Approximation by Nomographic Functions," in *Proceedings of the Fifty-third Annual Allerton Conference on Communication, Control, and Computation*, September 25 – October 2 2015, pp. 453–458. [Online]. Available: http://arxiv.org/abs/1504.05474

[99] J. Liu and V. Calhoun, "Parallel Independent Component Analysis for Multimodal Analysis: Application to fMRI and EEG Data," in *4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, April 2007, pp. 1028–1031. [Online]. Available: http://dx.doi.org/10.1109/ISBI.2007.357030

[100] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber, "Privacy: Theory Meets Practice on the Map," in *2008 IEEE 24th International Conference on Data Engineering*, April 2008, pp. 277–286. [Online]. Available: https://doi.org/10.1109/ICDE.2008.4497436

[101] S. V. Macua, P. Belanovic, and S. Zazo, "Consensus-based Distributed Principal Component Analysis in Wireless Sensor Networks," in *2010 IEEE 11th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, June 2010, pp. 1–5. [Online]. Available: http://dx.doi.org/10.1109/SPAWC.2010.5671089

[102] S. Malluri and V. K. Pamula, "Gaussian Q-function and its Approximations," in *Communication Systems and Network Technologies (CSNT), 2013 International Conference on*. IEEE, 2013, pp. 74–77.

[103] A. W. Marshall, l. Olkin, and B. C. Arnold, *Inequalities: Theory of Majorization and Its Applications*. Springer-Verlag New York, 1979.

[104] U. Maulik and S. Bandyopadhyay, "Performance Evaluation of Some Clustering Algorithms and Validity Indices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, pp. 1650–1654, Dec 2002. [Online]. Available: https://doi.org/10.1109/TPAMI.2002.1114856

[105] F. McSherry and K. Talwar, "Mechanism Design via Differential Privacy," in *Foundations of Computer Science, 2007. FOCS '07. 48th Annual IEEE Symposium on*, Oct 2007, pp. 94–103. [Online]. Available: https://doi.org/10.1109/FOCS.2007.66

[106] I. Mironov, "On Significance of the Least Significant Bits for Differential Privacy," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 650–661. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382264

[107] ——, "Rényi Differential Privacy," *CoRR*, vol. abs/1702.07476, 2017. [Online]. Available: http://arxiv.org/abs/1702.07476

[108] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A Survey of Distributed Optimization and Control Algorithms for Electric Power Systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2941–2962, Nov 2017. [Online]. Available: https://doi.org/10.1109/TSG.2017.2720471

[109] A. Narayanan and V. Shmatikov, "How To Break Anonymity of the Netflix Prize Dataset," *CoRR*, vol. abs/cs/0610105, 2006. [Online]. Available: http://arxiv.org/abs/cs/0610105

[110] B. Nazer and M. Gastpar, "Computation Over Multiple-Access Channels," *IEEE Transactions on Information Theory*, vol. 53, no. 10, pp. 3498–3516, Oct 2007. [Online]. Available: https://doi.org/10.1109/TIT.2007.904785

[111] A. Nedic and A. Ozdaglar, "Distributed Subgradient Methods for Multi-Agent Optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, Jan 2009. [Online]. Available: https://doi.org/10.1109/TAC.2008.2009515

[112] M. R. Network, "Group ICA of EEG Toolbox." [Online]. Available: http://mialab.mrn.org/software/eegift/index.html

[113] ——, "fMRI Simulation Toolbox." [Online]. Available: http://mialab.mrn.org/software/simtb/index.html

[114] K. Nordhausen, E. Ollila, and H. Oja, "On the Performance Indices of ICA and Blind Source Separation," in *Proceedings of the 12th IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, June 2011, pp. 486–490. [Online]. Available: http://dx.doi.org/10.1109/SPAWC.2011.5990458

[115] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially Private Distributed Convex Optimization via Objective Perturbation," in *2016 American Control Conference (ACC)*, July 2016, pp. 2061–2066. [Online]. Available: https://doi.org/10.1109/ACC.2016.7525222

[116] M. Ohlson, M. R. Ahmad, and D. von Rosen, "The Multilinear Normal Distribution: Introduction and Some Basic Properties," *Journal of Multivariate Analysis*, vol. 113, pp. 37 – 47, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.jmva.2011.05.015

[117] M. Pathak, S. Rane, and B. Raj, "Multiparty Differential Privacy via Aggregation of Locally Trained Classifiers," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 1876–1884.

[118] V. Pihur, "The Podium Mechanism: Improving on the Laplace and Staircase Mechanisms," *CoRR*, vol. abs/1905.00191, 2019. [Online]. Available: http://arxiv.org/abs/1905.00191

[119] S. M. Plis, A. D. Sarwate, D. Wood, C. Dieringer, D. Landis, C. Reed, S. R. Panta, J. A. Turner, J. M. Shoemaker, K. W. Carter, P. Thompson, K. Hutchison, and V. D. Calhoun, "COINSTAC: A Privacy Enabled Model and Prototype for Leveraging and Processing Decentralized Brain Imaging Data," *Frontiers in Neuroscience*, vol. 10, p. 365, 2016. [Online]. Available: https://doi.org/10.3389/fnins.2016.00365

[120] A. Rajkumar and S. Agarwal, "A Differentially Private Stochastic Gradient Descent Algorithm for Multiparty Classification," in *Artificial Intelligence and Statistics*, 2012, pp. 933–941.

[121] W. Rudin, *Principles of Mathematical Analysis*. McGraw-Hill Higher Education, 1976. [Online]. Available: https://www.mheducation.com/highered/product/principles-mathematical-analysis-rudin/M007054235X.html

[122] A. Shamir, "How to Share a Secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359176

[123] Y. Shoukry, K. Gatsis, A. Alanwar, G. J. Pappas, S. A. Seshia, M. Srivastava, and P. Tabuada, "Privacy-Aware Quadratic Optimization using Partially Homomorphic Encryption," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 5053–5058. [Online]. Available: https://doi.org/10.1109/CDC.2016.7799042

[124] M. Singulla, M. R. Ahmad, and D. von Rosen, "More on the Kronecker Structured Covariance Matrix," *Communications in Statistics - Theory and Methods*, vol. 41, no. 13-14, pp. 2512–2523, 2012. [Online]. Available: http://dx.doi.org/10.1080/03610926.2011.615971

[125] A. Smith, "Privacy-preserving Statistical Estimation with Optimal Convergence Rates," in *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, ser. STOC '11. New York, NY, USA: ACM, 2011, pp. 813–822. [Online]. Available: http://doi.acm.org/10.1145/1993636.1993743

[126] J. So, B. Guler, A. Salman Avestimehr, and P. Mohassel, "CodedPrivateML: A Fast and Privacy-Preserving Framework for Distributed Machine Learning," *arXiv e-prints*, Feb 2019. [Online]. Available: http://arxiv.org/abs/1902.00641

[127] 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*, Dec 2013, pp. 245–248. [Online]. Available: https://doi.org/10.1109/GlobalSIP.2013.6736861

[128] D. A. Sprecher, "On Computational Algorithms for Real-valued Continuous Functions of Several Variables," *Neural Networks*, vol. 59, pp. 16–22, 2014. [Online]. Available: https://doi.org/10.1016/j.neunet.2014.05.015

[129] ——, "A Representation Theorem for Continuous Functions of Several Variables," *Proceedings of the American Mathematical Society*, vol. 16, no. 2, pp. 200–203, April 1965. [Online]. Available: http://dx.doi.org/10.2307/2033845

[130] G. W. Stewart, "On the Early History of the Singular Value Decomposition," *SIAM Rev.*, vol. 35, no. 4, pp. 551–566, Dec. 1993. [Online]. Available: http://dx.doi.org/10.1137/1035134

[131] J. Sui, T. Adalı, G. D. Pearlson, and V. D. Calhoun, "An ICA-based Method for the Identification of Optimal fMRI Features and Components using Combined Group-discriminative Techniques," *NeuroImage*, vol. 46, no. 1, pp. 73 – 86, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.neuroimage.2009.01.026

[132] A. T. Suresh, F. X. Yu, H. B. McMahan, and S. Kumar, "Distributed Mean Estimation with Limited Communication," *CoRR*, vol. abs/1611.00429, 2016. [Online]. Available: http://arxiv.org/abs/1611.00429

[133] L. Sweeney, "Only You, Your Doctor, and Many Others May Know," *Technology Science*, vol. 2015092903, no. 9, p. 29, 2015. [Online]. Available: https://techscience.org/a/2015092903

[134] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang, "Privacy loss in Apple's implementation of differential privacy on MacOS 10.12," ArXiV, Tech. Rep. arXiv:1709.02753 [cs.CR], September 2017. [Online]. Available: https://arxiv.org/abs/1709.02753

[135] P. M. Thompson, O. A. Andreassen, A. Arias-Vasquez, C. E. Bearden, P. S. Boedhoe, R. M. Brouwer, R. L. Buckner, J. K. Buitelaar, K. B. Bulayeva, D. M. Cannon *et al.*, "ENIGMA and the Individual: Predicting Factors that Affect the Brain in 35 Countries Worldwide," *Neuroimage*, vol. 145, pp. 389–408, 2017. [Online]. Available: https://doi.org/10.1016/j.neuroimage.2015.11.057

[136] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, Sep 1986. [Online]. Available: https://doi.org/10.1109/TAC.1986.1104412

[137] L. R. Tucker, "Some Mathematical Notes on Three-mode Factor Analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966. [Online]. Available: http://dx.doi.org/10.1007/BF02289464

[138] C. A. Uribe, S. Lee, A. Gasnikov, and A. Nedić, "Optimal Algorithms for Distributed Optimization," *arXiv preprint arXiv:1712.00232*, 2017. [Online].

Available: https://ui.adsabs.harvard.edu/link_gateway/2017arXiv171200232U/arxiv:1712.00232

[139] S. Vempala and G. Wang, "Special Issue on FOCS 2002 A Spectral Algorithm for Learning Mixture Models," *Journal of Computer and System Sciences*, vol. 68, no. 4, pp. 841 – 860, 2004. [Online]. Available: http://dx.doi.org/10.1016/j.jcss.2003.11.008

[140] D. Wang, M. Ye, and J. Xu, "Differentially Private Empirical Risk Minimization Revisited: Faster and More General," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 2719–2728. [Online]. Available: http://arxiv.org/abs/1802.05251

[141] Y. Wang and A. Anandkumar, "Online and Differentially-Private Tensor Decomposition," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. USA: Curran Associates Inc., 2016, pp. 3539–3547. [Online]. Available: http://dl.acm.org/citation.cfm?id=3157382.3157493

[142] J. R. Westbury, "X-ray Microbeam Speech Production Database User's Handbook: Madison," *WI: Waisman Center, University of Wisconsin*, 1994. [Online]. Available: http://www.haskins.yale.edu/staff/gafos_downloads/ubdbman.pdf

[143] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional Mechanism: Regression Analysis Under Differential Privacy," *Proc. VLDB Endow.*, vol. 5, no. 11, pp. 1364–1375, Jul. 2012. [Online]. Available: http://dx.doi.org/10.14778/2350229.2350253

[144] J. Zhu, C. Xu, J. Guan, and D. O. Wu, "Differentially Private Distributed Online Algorithms Over Time-Varying Directed Networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 4–17, March 2018. [Online]. Available: https://doi.org/10.1109/TSIPN.2018.2797806