BAYESIAN APPROACHES FOR PRIVACY PRESERVING DATA SHARING

by

Beyza Ermiş

B.S., Computer Engineering, Bilkent University, 2010

M.S., Computer Engineering, Boğaziçi University, 2012

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2020

ACKNOWLEDGEMENTS

First, I would like to thank to my supervisor Prof. Ali Taylan Cemgil for all his support during my PhD. I would also like to thank to my PhD committee and jury, Prof. İlker Birbil, Assoc. Prof. Aybek Korugan, Prof. Cem Say and Assist. Prof. Sinan Yıldırım for their valuable feedback. Finally, I would like to thank the many people that helped through comments and discussions during the writing of this thesis. I would like to thank Ethem Alpaydın, Cédric Archambeau, Evrim Acar, Umut Şimsekli, Fatma Güney and Barıs Kurt.

ABSTRACT

BAYESIAN APPROACHES FOR PRIVACY PRESERVING DATA SHARING

In this thesis, we focus on the *data fusion* problem where we have heterogeneous data which is collected from different sources and stored in the form of matrices and higher-order tensors and propose coupled matrix and tensor factorization models to be able to jointly analyze these relational datasets. This method performs simultaneous factorization of matrices and tensors by extracting the common latent factors from the shared modes. We develop coupled models using various tensor models and cost functions for the missing link prediction problem and report the successful empirical results. Most of the time, the data matrices and tensors are distributed between several parties. Sharing information across those parties brings the *privacy protection* requirement, therefore the second problem we handle is protecting the privacy of distributed and heterogeneous datasets. We develop and evaluate a practical mechanism that ensures the privacy of individuals in a distributed setting, in which N data sites jointly estimate the parameters of a statistical model conditioned on all the data without sharing their input datasets. We exploit the connection between differential privacy and sampling from a Bayesian posterior to derive an efficient coupled tensor factorization algorithm. We empirically show that our methods are able to provide good prediction accuracy on synthetic and real datasets while providing provable privacy guarantee. Finally, we propose an approach to preserve the privacy of the neural network's training data due to the connection between tensor factorization and neural networks. We introduce a dropout technique that provides an elegant Bayesian interpretation to dropout, and show that the intrinsic noise added can be exploited to obtain a degree of differential privacy.

ÖZET

MAHREMİYETİ KORUYAN VERİ PAYLAŞIMINDA BAYESÇİ YÖNTEMLER

Bu tezde, farklı kaynaklardan toplanan matrisler ve yüksek mertebeli tensörler şeklinde depolanan heterojen verilerin birlikte analiz edilmesi ve veri füzyon problemine yoğunlaşıyoruz. Problemin çözümünde ise bağlaşımlı matris ve tensör ayışımı modelleri kullanmaktayız. Bu yöntem, paylaşılan modlardan ortak gizli faktörleri çıkararak matrislerin ve tensörlerin aynı anda bileşenlerine ayrılmasını sağlar. Biz de burada eksik bağlantı tahmini problemi için bağlaşımlı tensör modelleri geliştirerek, çeşitli model topolojileri ve çeşitli ıraksaylar kullanarak başarılı deneysel sonuçları rapor etmekteyiz. Çoğu zaman, veri matrisleri ve tensörler değişik taraflar arasında dağıtılır. Bu taraflar arasında bilgi paylaşımı gizlilik ve mahremiyeti koruma gereksinimini getirir, bu nedenle ele aldığımız ikinci sorun dağıtılmış ve heterojen veri kümelerinin mahremiyetini korumaktır. Dağıtık bir ortamda bireylerin gizliliğini sağlayan pratik bir mekanizma geliştirecek ve bu mekanizmayi çesitli gerçek veriler kullanarak değerlendirecegiz. Bu mekanizma için Bayesçi cıkarım ve diferansiyel mahremiyet arasındaki bağlantıdan faydalanarak etkili bir bağlaşımlı tensör ayrışım yöntemi geliştireceğiz. Yöntemlerimizin mahremiyet garantisi sağlarken sentetik ve gerçek veri kümelerinde iyi tahmin doğruluğu sağlayabildiğini deneysel olarak göstereceğiz. Son olarak, tensör ayrışımı ve yapay sinir ağları arasındaki bağlantıyı göstererek, yapay sinir ağlarının kullandığı verilerinin gizliliğini korumak için bir yaklaşım önereceğiz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathcal{A}$	Randomized algorithm
B	Minibatch size
$\mathcal{B}_t$	Minibatch at iteration t
C_v	Data size of $X^{(v)}$
$D_p(.)$	β divergence
D_α	Rényi divergence
$\mathcal{D}$	The complete dataset
e_d	Entity type
G	Gradient clipping norm
$\mathcal{G}$	Gamma distribution
$\mathbb{I}$	Identity matrix
i_d	Set of indices in Θ_d
i_v	Set of indices in relation v
K	Latent dimension size
$M^{(v)}$	Binary mask for $X^{(v)}$
$\mathcal{M}(.)$	Exponential mechanism
$\mathcal{N}$	Normal distribution
Ω_v	Full observation space of relation v
$\mathcal{PO}$	Poisson distribution
Q	Privatization mechanism
p_v	Power parameter for $X^{(v)}$
R	Response set of algorithm $\mathcal{A}$
$R_{v,d}$	Coupling matrix
S_v	Ordered set of entity types in relation v
x_{i_v}	Tensor element
$X^{(v)}$	Observed tensor
$\hat{X}^{(v)}$	Mean parameter for $X^{(v)}$
v	Data tensor index

$Z^{(v)}$	Perturbed observation of $X^{(v)}$
δ	probability of privacy guarantee
ϵ	privacy budget
$\theta_{i_d k}$	Rows of latent factor Θ_d
Θ_d	Latent factor
$\delta(.)$	Kronecker delta function
η_t	Step size at iteration t for DP case
κ_t	Step size at iteration t
ϕ_v	Dispersion parameter for $X^{(v)}$
$\Delta(.)$	Tensor valued collapse operation
∇	Gradient
Ω_v	Full observation space of relation v
$\xi^{(v)}_{(t)}$	Gaussian noise matrix at iteration t
∂	Partial derivative

LIST OF ACRONYMS/ABBREVIATIONS

AC	Advanced Composition
AUC	Area Under the Curve
BNN	Bayesian Neural Network
CDP	Centralized Differential Privacy
CMF	Coupled Matrix Factorization
CP	CANDECOMP/PARAFAC
CTF	Coupled Tensor Factorization
DNN	Deep Neural Network
DP	Differential Privacy
DP-CTF	Differentially Private Coupled Tensor Factorization
DPD	Differentially Private Dropout
EUC	Euclidean distance
GM	Gaussian Mechanism
IS	Itakura-Saito divergence
KL	Kullback-Leibler divergence
LDP	Local Differential Privacy
LL	Log-likelihood
mCDP	mean Concentrated Differential Privacy
MA	Moments Accountant
MAP	Maximum a Posteriori
MCMC	Markov Chain Monte Carlo
MH	Metropolis Hastings
ML	Maximum Likelihood
NP	Non-private
OPS	One Posterior Sample
RMSE	Root Mean Square Error
SGD	Stochastic Gradient Descent
SGLD	Stochastic Gradient Langevin Dynamics

VB	Variational Bayes
zCDP	zero Concentrated Differential Privacy

1. INTRODUCTION

In numerous applications, like healthcare, natural language processing, gene expression analysis and collaborative filtering, data can be arranged conveniently in terms of matrices. For instance, the users' ratings can be organized as a matrix with rows corresponding to the users and columns corresponding to the items in recommender systems. Similarly, in healthcare data, medical records can be formed as a matrix with rows corresponding to the patients and columns corresponding to the medical conditions (features). Note that, matrices are natural for describing binary relations (such as user-item) but many data analysis tasks call for more complex setups where the data is collected from heterogeneous data sources and consists of ternary or higher order relations (such as user-item-time). For instance, in retail recommender systems, in addition to retail data showing who has bought which items, we may also have access to customers' social networks, i.e., who is friends with whom. In such scenarios, datasets are represented by a collection of matrices and/or higher order tensors and jointly analyzing data from multiple sources has great potential to increase our ability for capturing the underlying structure in data.

Coupled analysis of the multi-relational heterogeneous data brings several challenges along with the benefits. In this thesis we address two main challenges. The *first* one is the *data fusion* problem. The data is generally collected from various sources by different parties at different times. Handling these heterogeneous data jointly may remarkably increase the performance if the relation between the data is properly modelled. In the literature, a number of algorithms have been proposed for data fusion and handling data-heterogeneity. Three of the main research topics that focus on this problem are transfer/multi-task learning that generalizes better on the original task by sharing information between related tasks [3–7], multiple-kernel learning that combines multiple kernels instead of using one kernel in kernel machine algorithms such as Support Vector Machines (SVM) [8–11] and coupled matrix and tensor factorization [12–18]. Each of these topics has different aims, so they approach the data fusion from a different perspective. Here, our main focus will be on data fusion with

coupled matrix and tensor factorization models.

1.1. Coupled Tensor Factorization

Applications that have successfully used coupled tensor factorization models include link prediction [18,19], computational biology [20,21], audio processing [17,22], chemometrics and neuroscience [23,24] where various information source is available. One example of such situations can be observed for movie recommender system where a user rating matrix can be augmented with movie information with specific features and/or social media connectivity information of the user.

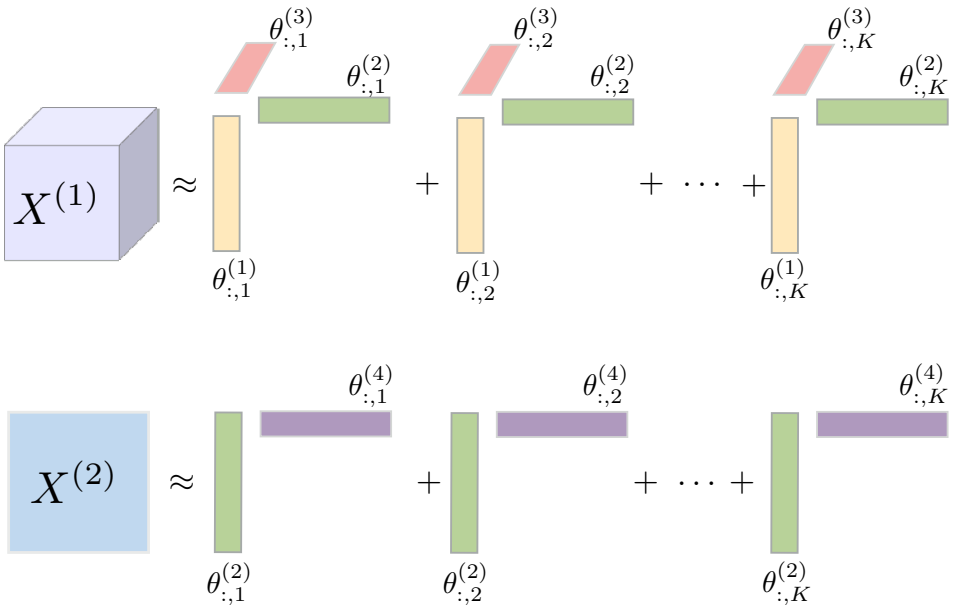


Figure 1.1. An illustration of coupled matrix and tensor factorization model. The observed tensor $X^{(1)}$ is approximated as the sum of inner products of the columns of $\theta^{(1)}$, $\theta^{(2)}$ and $\theta^{(3)}$. Similarly, the observed matrix $X^{(2)}$ is approximated as the sum of inner products of the columns of $\theta^{(2)}$ and $\theta^{(4)}$. Here, $\boldsymbol{\theta}^{(2)}_{i_2} = \theta^{(2)}_{i_2,1:k}$ exists in both $X^{(1)}$ and $X^{(2)}$, which makes this model coupled. In this notation, K denotes the size of the index $k \in \{1, \ldots, K\}$.

The key idea in coupled factorization models is extracting the latent structures that are shared across matrices and/or tensors, which enables transferring information between them. In order to better explain the method, we first illustrate a simple coupled matrix and tensor

factorization model in Figure 1.1 that is defined as follows:

$$X^{(1)} \approx \hat{X}^{(1)} = \sum_{k=1}^{K} \theta_{i_1 k}^{(1)} \, \theta_{i_2 k}^{(2)} \, \theta_{i_3 k}^{(3)} = \left\langle \boldsymbol{\theta}_{i_1}^{(1)}, \boldsymbol{\theta}_{i_2}^{(2)}, \boldsymbol{\theta}_{i_3}^{(3)} \right\rangle$$
$$X^{(2)} \approx \hat{X}^{(2)} = \sum_{k=1}^{K} \theta_{i_2 k}^{(2)} \, \theta_{i_4 k}^{(4)} = \left\langle \boldsymbol{\theta}_{i_2}^{(2)}, \boldsymbol{\theta}_{i_4}^{(4)} \right\rangle$$

where $X^{(1)}$ is the observed tensor and $X^{(2)}$ is the observed matrix decomposed by a coupled tensor factorization model. Since the factor $\boldsymbol{\theta}_{i_2}^{(2)}$ is shared in both decomposition, the overall model is coupled. Given the observed matrices, the coupled factorization problem estimates the factors $\boldsymbol{\theta}^{1:4} \equiv \{\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}, \boldsymbol{\theta}^{(4)}\}$ by solving the optimization problem given as:

$$(\boldsymbol{\theta}^{1:4})^* = \arg\min_{\boldsymbol{\theta}^{1:4}} D_1(X^{(1)} \parallel \hat{X}^{(1)}) + D_2(X^{(2)} \parallel \hat{X}^{(2)})$$

where $D_1(.)$ and $D_2(.)$ are (possibly different) divergences. We will define the coupled tensor factorization notation in Chapter 2 and describe the estimation methods of factor matrices in Chapter 3.

1.2. Privacy Protection

The *second* challenge is the *privacy protection* of the collected data that appears in sharing information across matrices and/or tensors. In modern data processing applications, data is often owned by multiple data sites (as illustrated in Figure 1.2); e.g. patients' records are distributed across insurance agencies and hospitals; financial data is distributed across various institutions, or retail data is distributed across online review sites. Statistical inference on such distributed data has many applications, such as in health care, personalized services, and social networks. However, several barriers, such as business policies, privacy requirements or legislative restrictions prohibit the direct unification of data in a single database for collective use. While there is still a mutual benefit for data owners to share data in some form, privacy concerns and perceived risk about implications of data sharing often outweigh the direct benefits.

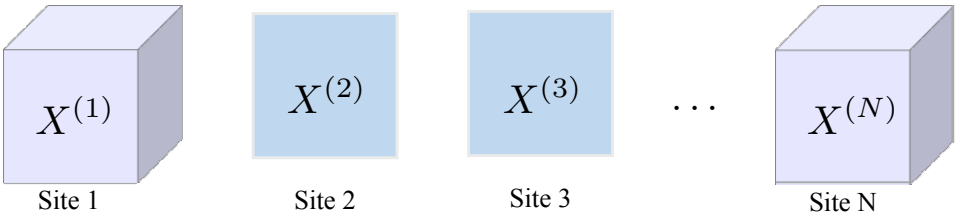


Figure 1.2. An illustration of multiple data sites that each possess data $X^{(1)}, X^{(2)}, \ldots, X^{(N)}$.

Privacy concerns are justified, as apart from the exceptional cases of completely hiding or completely revealing the data, the mechanisms for reliable data sharing are far from trivial. It is noted that apparently viable intermediate solutions such as "anonymization" or just releasing summary statistics may allow an adversary to breach the data privacy, possibly in the presence of side information. That's why anonymization techniques are evolving to always bring more privacy guarantees. The privacy guarantee is required to be mathematically proven and does not depend on any other information that an adversary could have. Into the bargain, the level of the privacy guarantee should be manageable.

Differential privacy (DP) is currently a widely accepted privacy definition [25,26] to formalize the privacy protection of algorithms. The key principle of DP is to ensure that an adversary should be unable to reliably infer whether or not a particular individual is participating in a database, even with unlimited computational power and access to every entry except for that particular individual's data. This can be accomplished through adding noise into an algorithm at different stages such as adding noise to the data itself or changing the objective function to be optimized. In order to design efficient differentially private algorithms, where there is a better tradeoff between privacy and utility, one needs to design a noise addition system. However, iterative algorithms such as stochastic gradient descent (SGD) accumulate the privacy loss at each iteration and large number of iterations cause high cumulative privacy loss because of the potential of each access to leak more information. In this thesis, we employ recent privacy notions to develop efficient algorithms to overcome the high cumulative privacy loss.

We first explain why privacy protection is required in most of the practical real-world

applications and we will point out how this problem is highly related with the first challenge, which is data fusion. First, we investigate a scenario where each data site concurrently acts both as a data provider and a data consumer to learn a model from the overall data while preserving the privacy of individual sites and/or each individual's data points. In both cases, *N* distinct parties (say, companies) hold sensitive data from several individuals, and they wish to jointly estimate parameters of a statistical model conditioned on all data available.

We shall assume that data instances that are partitioned between several parties and they are arranged as matrices and/or higher-order tensors. A natural statistical model for such relational data is coupled tensor factorization model (CTF) [15,16]. We can give the recommender systems as an example to relational data. In recommender systems, each party holds a subset of the total number of users, containing the data from the same set of items. For sharing the sensitive data between parties, we introduce a privacy preserving CTF approach to predict missing information. To provide and evaluate the privacy guarantees, we use the connection [27,28] between differential privacy and Stochastic Gradient Langevin Dynamics (SGLD) [29]. The CTF models can be easily designed in compliance with the privacy requirements of the use case, so we don't need a custom-design for each new CTF model to provide differential privacy.

Furthermore, we address the *privacy protection in Bayesian deep neural networks* (BNN)s. Large data collections required for the training of neural networks often contain sensitive information such as the clinical histories of patients, biomedical images or user profiles; and the privacy of the training data must be preserved in that case. We consider the privacy issues in neural networks due to the connection between tensor factorization and estimating the parameters of a deep neural network [30–33]. We summarize this connection in the following. CP decomposition [34] is the most basic and commonly used tensor decomposition method that decomposes a tensor into a sum of outer products of vectors. The CP decomposition of a 3-way tensor X of size $I^{(1)} \times I^{(2)} \times I^{(3)}$ is written as:

$$X \approx \sum_{k=1}^{K} \lambda_k u_k^{(1)} \circ u_k^{(2)} \circ u_k^{(3)} = \hat{X} \tag{1.1}$$

where K is the rank and $\{U^{(i)}\}_{i=1}^{3}$ are the factor matrices of sizes $I^{(1)} \times K$, $I^{(2)} \times K$, $I^{(3)} \times K$ respectively. Then, the CP decomposition solves the optimization problem given below to estimate the factor matrices $\{U^{(i)}\}_{i=1}^{3}$.

$$\min_{\hat{X}} \parallel X - \hat{X} \parallel \quad \text{with} \quad \hat{X} = \sum_{k=1}^{K} \lambda_k u_k^{(1)} \circ u_k^{(2)} \circ u_k^{(3)} \tag{1.2}$$

where $\parallel \cdot \parallel$ is the Frobenius norm. We note that the CP decomposition can be extended to N-way tensors for any $N \geq 3$. Here, we show that when we include several nonlinearities we can generalize tensor factorizations to corresponding neural networks. For this, we consider the following score function:

$$\ell(Z) = \langle X, \Phi(Z) \rangle \tag{1.3}$$

where $Z = \{z_i\}_{i=1}^{N}$ is a dataset with sequential instances z_i's and:

$$\Phi(Z) = f(z_1) \circ f(z_2) \ldots \circ f(z_N) \tag{1.4}$$

where f is a nonlinear activation function such as $f(z) = \sigma(Az + b)$. Now, we substitute the N-way generalized CP decomposition into Equation(1.3) and see the corresponding neural network formalization:

$$\begin{aligned} \ell(Z) &= \sum_{k=1}^{K} \lambda_k \left[\langle f(z_1), u_k^{(1)} \rangle \circ \langle f(z_2), u_k^{(2)} \rangle \circ \ldots \circ \langle f(z_N), u_k^{(N)} \rangle \right] \\ &= \sum_{k=1}^{K} \lambda_k \prod_{i=1}^{N} \langle f(z_i), u_k^{(i)} \rangle \end{aligned} \tag{1.5}$$

where the parameters of network are $\Theta = \left(\{\lambda_k\}_{k=1}^{K} \in \mathbb{R}, \{u_k^{(i)}\}_{k=1,i=1}^{K,N} \right)$. Equation(1.5) represents only the multiplicative nonlinearities. It is also possible to include other types of nonlinearities by replacing the outer product $\circ$, which is a scalar product, with a kernel function $\circ_\xi$ [33]. In this thesis, we use the Stochastic Gradient Langevin Dynamics method to estimate the parameters of a network differentially privately like in CTF method. We also show that our method preserves data privacy, represents uncertainty and performs regularization in

neural networks.

1.3. Related Work

In this section, we briefly survey the related studies on coupled tensor factorization and differentially private inference methods.

1.3.1. Coupled Tensor Factorization

For analysis of multi-relational data, Singh and Gordon [12] as well as Long *et al.* [35] have introduced collective matrix factorizations (CMFs). Afterwards, collective matrix factorization has been extended to coupled analysis of multi-relational data in the form of matrices and higher-order tensors [36,37] since in many disciplines, relations can be defined among more than two entities, e.g., when a user engages in an activity at a certain location, a relation can be defined over user, activity and location entities. For instance, Zheng *et al.* [38] model the user-location-activity relations with a tensor representation, and propose a matrix and tensor decomposition solution for collaborative location and activity filtering. Banerjee *et al.* [36] introduce a multi-way clustering approach for relational and multi-relational data where coupled analysis of heterogeneous data is studied using minimum Bregman information. Lin *et al.* [39] also discuss coupled matrix and tensor factorizations using KL-divergence modeling higher-order tensors by fitting a CP model [34]. While these studies use alternating algorithms, Acar *et al.* [16] propose an all-at-once optimization approach for coupled analysis. Recently Choi *et al* [40] proposed a fast and scalable algorithm for coupled matrix and tensor factorization that uses parallel SGD and Tucker decomposition method [41].

A number of studies examined coupled matrix and tensor factorization in Bayesian formulation. First time, Yilmaz *et al* [15] introduced a probabilistic framework - Generalised Coupled Tensor Factorization (GCTF) - for coupled matrix and tensor factorization models that jointly factorizes data from multiple heterogeneous information sources by extracting common factors. GCTF is used by latter studies for audio source separation [17] and link prediction [18]. Then, similar Bayesian multiple tensor factorization methods are proposed to decompose multiple cooccurring matrices and tensors into a set of underlying factors that can

be shared between any subset of them [42–45]. Recently, Kaya *et al* [46] extended distributed incremental gradient descent method and proposed an incremental, distributed and parallel algorithm (HAMSI) that incorporates second order information into the to optimization steps for a faster convergence where the second order information comes from an approximation to the Hessian of the objective function. Tang *et al* [47] proposed a variational Bayesian tensor factorization method for multiway analysis of brain states by processing EEG tensors. In [48], the authors used coupled tensor factorization to extract and analyze the spatiotemporal activities of people and use this information to control traffic lights or organize public transport better. Wu *et al* [49] used collective tensor factorization for health data analysis.

1.3.2. Bayesian Differential Privacy

In the literature, there are a number of works that address Bayesian approaches to machine learning under differential privacy. Williams and McSherry [50] use probabilistic inference to produce a posterior distribution that integrates all available information about the private and observed datasets. Dimitrikakis *et al* [51] and Zhang *et al* [52] introduce a posterior sampling mechanism and showed that under moderate sensitivity condition on the log-likelihood, Bayesian posterior sampling provides differential privacy automatically. Foulds *et al* [53] proposed a simple Laplace mechanism method for exponential family distributions that samples from posterior distribution differentially privately. Park *et al* [54] provided a general framework for privacy-preserving variational Bayes for a large class of probabilistic models by perturbation of the expected sufficient statistics. Later, they proposed a practical algorithm that outputs expectation-maximization (EM) parameter estimates privately by using moment perturbation [55]. Most recently, Heikkela *et al* [56] applied the gradient perturbation mechanism to devise a generic differentially private variational inference method, which is also applicable to non-conjugate models. Finally, Schein *et al* [57] proposed a method to privatize the Poisson factorization by establishing the objective of Bayesian inference under local-DP.

1.3.3. Local Differential Privacy

The connection between differential privacy and information theory has been studied to measure the privacy leakage [58]. In particular, Kairouz *et al* [59] study the tradeoff between information theoretic utility functions and local differential privacy. Duchi *et al* propose information theoretic bounds for privacy preserving learning [60] and provide general techniques to derive bounds under local privacy constraints [61]. In [62], Barber *et al* establish a connection between privacy constraints and statistical estimation. Most recently, Schein *et al* [57] proposed a locally differentially private approach where the latent variables are inferred from the privatized data (which is privatized by a randomized response mechanism) and the posterior distribution is built over the latent variables. In addition, Joseph *et al* [63] propose a method to track the timely statistics over time and use this method for frequency estimation.

1.3.4. Multiparty Differential Privacy

Several differentially-private learning algorithms from multiparty data were proposed, including secure multiparty computation [64], empirical risk minimization with private gradient interchange [65], matrix factorization [66] and private Bayesian inference in a distributed setting which is based on a secure multiparty communication technique [56]. In [67] and in [68], they propose a global classifier that protects differential privacy by combining separate classifiers that are trained locally by performing privacy constraints. Most recently, multiparty settings have been started to be used in deep learning. Shokri *et al* [69] introduced a method that enables multiple parties to train a neural network without sharing the datasets that they own and Papernot *et al* [70] proposed a method that combines private models which are learned locally from independent datasets differentially privately. Besides, Gupta *et al* studied a differentially private multitask learning in [71] by utilizing from a noisy task relation matrix. Lastly, Xie *et al* [72] propose a differentially private multitask learning framework to protect the privacy of the distributed data. In this study, they introduced a privacy-preserving proximal gradient algorithm that solves a general class of multitask learning formulations with asynchronous updates.

1.3.5. Differentially Private Matrix and Tensor Factorization

Matrix and tensor factorization models are the key methods for recommender systems and privacy of the users inherently brings privacy concerns to this area. There is an increasing number of studies on differentially private matrix factorization approaches in the literature. Nikolaenko *et al* [73] used a a cryptographic method which is named as garbled circuits to provide differential privacy in matrix factorization. In [74], the authors used sketching techniques and utilizes from the inherent randomness of the data structure to guarantee (ϵ, δ) differential privacy. Friedman *et al* [75] proposed a private matrix factorization approach that provides privacy different perturbation methods: the input, gradients of SGD and the output. Recently, two studies considered differential privacy for tensor decomposition methods. In [76], they proposed an online differentially private tensor factorization algorithm that based on tensor power method framework and Gaussian mechanism to perturb the objective to provide privacy. Imtiaz and Sarwate [66] presented a privacy preserving distributed tensor factorization methods for PCA and OTD that uses an efficient correlated noise to manage the same noise level as the centralized tensor factorization scenario.

The proposed differentially private coupled tensor factorization algorithm is most closely related to the works by Liu *et al* [2] and Hua *et al* [1]. Hua *et al* propose a differentially private SGD algorithm for matrix factorization (MF) where each user is accepted as a party; so the rows of the user factor are kept as private. Then, the data aggregator (recommender) uses the item factor to share preferences between users using a Laplace mechanism. The recommender has access to the intermediate results from all users after each iteration and it can easily eliminate the noise vectors' effect. So, the noise vector is needed to be decomposed and determined after each iteration using a fairly complicated procedure. Liu *et al* [2] exploit the fact which is proved in [27] and propose a DP-collaborative filtering approach. They focus on the *single-party* setting where *no information sharing between data sites* is considered. In this thesis, we propose a simple MCMC algorithm for multiparty prediction systems where each party can hold data of a different modality. Our method allows information sharing while preserving both the *user-level* DP and the *site-level* DP that will be explained in Chapter 4.

1.3.6. Differentially Private Deep Learning

There are a number of works that address deep learning under differential privacy. Recently, Shokri and Shmatikov [69] designed a method that permits multiple parties to train a neural network model without sharing their datasets. Their key contribution is based on perturbing the gradients of the SGD by fussy sharing of model parameters during training. Phan *et al.* [77] introduced a different approach towards differentially private deep learning that focuses on learning autoencoders by perturbing the objective functions of them. Papernot *et al.* [70] proposed a method where privacy-preserving models are learned locally from disjoint datasets and then combined in a privacy-preserving fashion. Most recently, Phan *et al.* [78] developed a method that uses Laplace mechanism to preserve differential privacy in deep neural networks. Their method doesn't depend on the number of epochs in the training phase where the privacy budget is consumed; but it adds more (Laplace) noise into the less relevant input features with the model output. In [79], Acs *et al.* proposed a mixture of multiple generative DNNs model where the data is divided into N clusters. In this work, the DNNs are trained collectively to learn the distribution that generates the given data using a differentially private gradient descent. Most recently, Phan *et al.* [80] developed a differentially private convolutional deep belief network that preserves privacy by perturbing the energy-based objective functions. They basically focus on the healthcare field that have significant privacy issues.

Our differentially private BNN method is most closely related to the study in [81] where they developed the moments accountant method, which is closely related to the notion of concentrated-DP, to sum up the privacy loss that provides a cramped bound for privacy loss compared to the standard composition methods such as sequential and advanced composition. By using the moments accountant (MA), Abadi *et al* propose a differentially private SGD algorithm to train a neural network by perturbing the gradients of parameters in SGD. The gradients are perturbed by adding Gaussian distribution noise *separately*. To protect privacy in deep learning methods, noise addition is generally used [81, 82]. In this thesis, we focus on the BNNs and introduce a dropout technique that provides an elegant Bayesian interpretation to dropout. We show that the intrinsic noise added, with the primary goal of regularization, can be exploited to achieve a certain amount of differential privacy. In addition, we prove

that there is a one to one relation between dropout rate and the Gaussian noise, so we say that dropout preserves (ϵ, δ)-DP in each BNN model for free when we calibrate the dropout rate carefully. Conversely, the amount of noise computed to provide (ϵ, δ)-DP also finds a proper dropout rate. We exploit the intrinsic randomized noise of the dropout by using two connections: i) Dropout and SGLD and ii) SGLD and DP. Furthermore, we apply the zero concentrated DP [83, 84] (zCDP) to gradient perturbation mechanism. We note that the zCDP achieves similar tight bound on privacy budget with the moments accountant but easier to convert into DP. We also note that the moments accountant requires that the intermediate computations after each iteration are not revealed. Eventually, we derive similar bounds to the moments accountant by using zCDP to strengthen the privacy guarantee in Bayesian DNNs. We will describe the method in detail and make the privacy analysis of the method in Chapter 5.

1.4. Contributions

In this thesis, we address the problems defined as below:

- *Completing missing entries in a relational dataset represented by several matrices and tensors:* We address the completion of missing matrix and tensor entries/parts by data fusion formulated as simultaneous factorization of several observation tensors where latent factors are shared among each observation. We study various tensor models and loss functions for this problem.
- *Estimation of the latent factors:* In matrix and tensor factorization models, the latent factors are the crucial variables to reconstruct the matrices and tensors. They are used for the analysis of observed data and prediction of the missing entries/parts of the dataset. For this reason, accurate estimation of the latent factors is crucial to the success of the prediction/completion of the missing dataset.
- *Privacy protection of the data that appears in sharing information across matrices and/or tensors:* In modern data processing applications, data is often owned by multiple data sites. We shall assume that the data instances, which are partitioned between those sites, are arranged as matrices and/or higher-order tensors. In this setting, the information is shared between sites via the *shared latent factors* and privacy protection

is required in the inference phase.

- *Privacy protection of neural network training:* The datasets that are used to train neural networks are often collected from individuals and contain sensitive information. Applying deep learning methods to these records are restricted by privacy requirements or legislative regulations. Thus, sharing and usage of the data about individuals require methods that provide precise privacy guarantees while meeting the demands of the applications.

Our main contributions in this thesis can be summarized as follows:

- We address the missing link prediction problem as filling the missing entries in a relational dataset using joint analysis of heterogeneous data based on different tensor models, i.e., CP, Tucker and some arbitrary tensor factorization models, as well as different loss functions, i.e., KL-divergence, IS (Itakura–Saito)- divergence, EUC distance and various other cost functions based on β-divergences.
- We first use the Maximum Likelihood (ML) and Maximum-a-Posteriori (MAP) estimation of the latent variables. We then develop a novel variational Bayesian algorithm for making inference on coupled matrix and tensor factorization models. In this method, the exact characterization of the approximating distribution and full conditionals are observed as a product of multinomial distributions, leading to a richer approximation distribution than a naive mean field. Finally, we present how the SGLD is used as an inference method for coupled matrix and tensor factorization models.
- In the first part of the thesis, our main application focus will be on link prediction. We develop several factorization models to demonstrate that coupled tensor factorizations outperform low-rank approximations of a single tensor and the selection of the tensor model as well as the loss function is significant. We also handle the *cold start problem* that arises when a new entity enters the dataset.
- In the second part, we focus on the private estimation of the latent factors in coupled matrix and tensor factorization. We develop and evaluate a practical mechanism that ensures the privacy of individuals in a distributed setting, in which N data sites jointly estimate parameters of a statistical model conditioned on all the data without sharing their input datasets.

- We describe a provably differentially private treatment of the CMF models for distributed multi-party setting by using the two notions of privacy: i) centralized differential privacy and ii) local differential privacy.
- We propose the CTF as a method to share information between distrustful parties with strong theoretical guarantees on the privacy of the released data. We use the connection between DP and SGLD [27], and propose differentially private coupled tensor factorization (DP-CTF) model by using two privacy definitions *site-level* and *user-level*.
- We claim that sharing the latent structures across the tensors enables transferring information between them and increases the prediction accuracy. However, in differentially private settings, sharing multiple factors may cause a considerable amount of noise added. We study under which conditions sharing latent factors helps to increase prediction accuracy for DP-CTF.
- In the final part, we propose an approach that preserves data privacy, represents uncertainty and performs regularization in deep neural networks. We use the recently proposed connection between Gaussian dropout and Stochastic Gradient Langevin Dynamics (SGLD) [85], and then analyze that under which conditions dropout provides provable privacy guarantee for the training data of Bayesian DNNs.
- For all proposed methods, we analyze the privacy of the algorithm with Advanced Composition (AC) [26] as a baseline approach. We then use the zCDP composition which remarkably reduces the amount of noise added and ensures to use the privacy budget more effective over multiple iterations.
- With experiments on coupled tensor factorization problems, we empirically show that as the number of sites that participate in the framework increases, we can support differentially private estimates while still preserving good utility and achieving better prediction performance over learning at a single site.
- With experiments on Bayesian neural networks, we empirically show that dropout helps to regularize network and improves prediction accuracy while providing (ϵ, δ)-DP and zCDP. As our experiments illustrate, dropout with zCDP outperforms both the standard DP and the state-of- the-art algorithms, especially when the privacy budget is low.

1.5. Thesis Outline

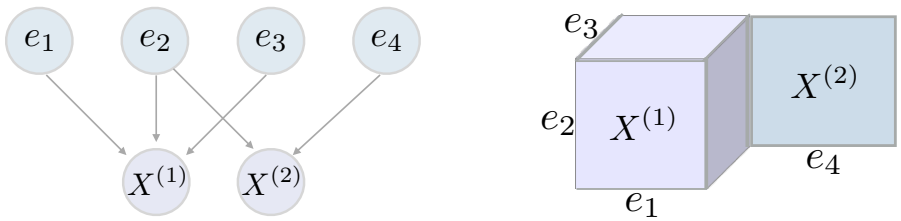
The content of the thesis is organized as follows. In Chapter 2, we provide the backgroud information and an overview of the relevant ingredients of the thesis. We formalize the coupled tensor factorization, provide the notation and describe the inference methods that is used for the estimation of the latent factors in Chapter 3. In Chapter 4, we present our differentially private coupled matrix (CMF) and tensor (CTF) models. In the first part of this chapter, we define the *user-level* and *site-level* DP and present the CMF models under *centralized* and *local* DP constraints. In the second part, we analyze the privacy of several CTF methods by using Advanced Composition (AC) and zCDP. Chapter 6 shows the results of the inference algorithms defined in Chapter 3 on synthetic and real-world datasets. In Chapter 7, we demonstrate our differentially private methods presented in Chapter 4 on different models and datasets. First, we show the results of differentially private CMF models with different privacy settings. Then, we show the differentially private CTF results on three different models with different notions of privacy on both synthetic and real world datasets. In Chapter 8, we evaluate the performance of our differentially private BNNs on several real world datasets. Finally, we conclude with some open directions in Chapter 9.

2. BACKGROUND

In this chapter, we provide an overview of the background material on which the ideas in subsequent chapters are based; namely coupled tensor factorization, differential privacy and its variants, stochastic gradient Langevin dynamics (SGLD) and Bayesian neural networks.

2.1. Coupled Tensor Factorization

In many applications, data from different sources are available. In these cases, single factorization models are not sufficient and the data from different sources should be combined to exploit all the information. The comprehensive modeling of multiple sources is called *coupled tensor factorization* and the goal is to jointly decompose numerous observed tensors that share a set of latent factors.



(a) (b)

Figure 2.1. Example of two relations among four types of entities on Flickr dataset. The dataset is represented with bipartite graph in (a) and represented in the form of a matrix $X^{(1)}$ and a third-order tensor $X^{(2)}$ in (b). $X^{(1)}$ is the user-item-tag relations and $X^{(2)}$ is the item-feature relations. The indices $e_1, \ldots, e_4$ indicates user, item, tag and feature respectively; e_2 is the common mode between two relations.

Given a set of V tensors $\{X^{(v)}\}_{v=1}^{V}$ that describe the relations between D sets of entities; the goal of the CTF is to jointly approximate the tensors with low-rank factorization. Each relation $v \in \{1, \ldots, V\}$ is associated with the ordered set S_v of the entity types involved in relation v, that is $S_v = (S_{v1}, \ldots, S_{v|S_v|})$ with $S_{vj} \in \{1, \ldots, D\}$. Let $\Omega_v := \{(e_1, \ldots, e_{|S_v|})\ ;\ e_d \in \{1, \ldots, n_d\}\ ; \forall d \in \{1, \ldots, |S_v|\}$ denote the full observation space of relation v (the set of $|S_v|$-uplets). For each of these $|S_v|$-uplets, we observe an entry $x_{i_v}^{(v)}$

indexed by $i_v \in \Omega_v$. Typically, $|S_v|= 1$ will refer to vectors, $|S_v|= 2$ will correspond to matrices and $|S_v|= 3$ will correspond to third-order tensors. For a simple matrix factorization model for matrix X, we have V = 1, D = 2 and a relation $S = (e_1, e_2)$ with $|S|= 2$.

A more complicated model is illustrated in Figure 2.1 with our notation. We have two relations (V=2) and four entity types (D=4) where the indices $e_1, \ldots, e_4$ indicates these entities. The first relation forms a 3-dimensional tensor $X^{(1)}$ and the second relation forms a matrix $X^{(2)}$. The set of relations $\mathcal{S}$ is defined as $\{S_1, S_2\}$ where $S_1 = (e_1, e_2, e_3)$, $|S_1|= 3$ and $S_2 = (e_2, e_4)$, $|S_2|= 2$.

Our objective is to predict the value of some specific entry $x^{(v)}_{i_v}$ of the tensor $X^{(v)}$. To do this, we approximate each tensor by the multilinear product of rank-K factors represented in the rows of some latent matrices $\Theta_d = [\theta^{(d)}_{i_d k}] \in \Re^{n_d \times K}$, $d \in \{1, \ldots, D\}$:

$$X^{(v)}_{i_v} \approx \hat{X}^{(v)}_{i_v} \quad := \quad \sum_{k=1}^{K} \prod_{d \in S_v} \theta^{(d)}_{i_d k} := \left\langle \boldsymbol{\theta}^{(1)}_{i_1}, \ldots, \boldsymbol{\theta}^{(d)}_{i_d} \right\rangle$$

where Θ_d is the factor associated to entity type d, and d is an element of set S_v. This data representation is very flexible, and we can handle relations of arbitrary number and/or arbitrary order.

Coupled factorization models simultaneously factorize data from multiple information sources by extracting common factors. We first consider the following model which is illustrated in Figure 2.1. This model is formalized in our notation as follows:

$$X^{(1)} \approx \hat{X}^{(1)} = \sum_{k=1}^{K} \theta^{(1)}_{i_1 k}\, \theta^{(2)}_{i_2 k}\, \theta^{(3)}_{i_3 k} = \left\langle \boldsymbol{\theta}^{(1)}_{i_1}, \boldsymbol{\theta}^{(2)}_{i_2}, \boldsymbol{\theta}^{(3)}_{i_3} \right\rangle \tag{2.1}$$

$$X^{(2)} \approx \hat{X}^{(2)} = \sum_{k=1}^{K} \theta^{(2)}_{i_2 k}\, \theta^{(4)}_{i_4 k} = \left\langle \boldsymbol{\theta}^{(2)}_{i_2}, \boldsymbol{\theta}^{(4)}_{i_4} \right\rangle \tag{2.2}$$

Given the observed tensor $X^{(1)}$ and matrix $X^{(2)}$, the coupled factorization problem estimates the factors, i.e., Θ_d where $d = \{1, 2, 3, 4\}$ by minimizing the total discrepancy

between the observations and the model output. This problem results in the following optimization problem:

$$(\Theta_{1:4})^* = \underset{(\Theta_{1:4})}{\arg\min} \frac{1}{\phi_1} d_1(X^{(1)} \parallel \hat{X}^{(1)}) + \frac{1}{\phi_2} d_2(X^{(2)} \parallel \hat{X}^{(2)}) \tag{2.3}$$

where $d_1(.)$ and $d_2(.)$ are (possibly different) divergences, usually chosen as the Euclidean, Kullback-Leibler (KL) or the Itakura-Saito (IS) divergences. On the other hand, our framework is defined for a large family of loss functions called the β-divergences, which generalizes these commonly-used divergences. β-divergences are defined as [86]:

$$d_p(X; \hat{X}) = \frac{X^{2-p}}{(1-p)(2-p)} - \frac{X\hat{X}^{1-p}}{1-p} + \frac{\hat{X}^{2-p}}{2-p} \tag{2.4}$$

where p determines the cost function. Note that $p = \{0, 1, 2\}$ corresponds to EUC, KL, and IS cost functions, respectively. Besides, the (inverse) weights ϕ_1 and ϕ_1 will be called dispersion parameters in our context and they are used to balance the information obtained from different modalities.

2.2. Differential Privacy

Differential privacy (DP) is currently a widely accepted privacy definition [25,26,87–89] to formalize the privacy protection of algorithms. The key principle of DP is to ensure that an adversary should be unable to reliably infer whether or not a particular individual is participating in a database, even with unlimited computational power and access to every entry except for that particular individual's data.

Formally, DP prevents inference about specific records by requiring a randomized query response mechanism that yields similar distributions on responses of similar datasets. For any two possible input datasets X and Y with the edit distance or Hamming distance $d(X, Y)$, and any subset of possible responses R, a randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ) differential

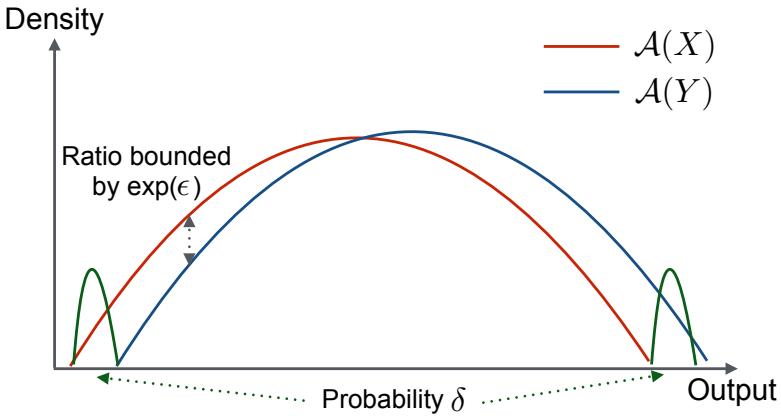


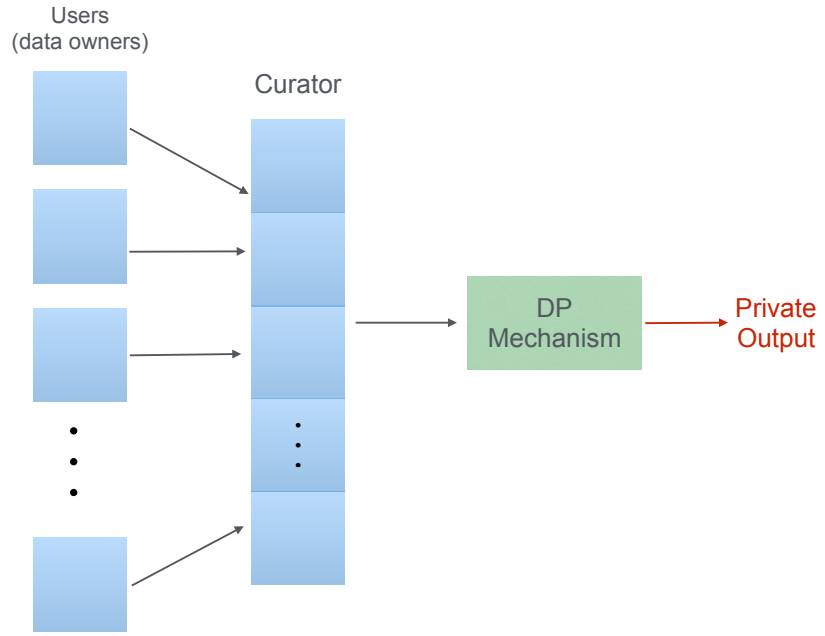
Figure 2.2. An illustration of (ϵ,δ)-DP of an algorithm $\mathcal{A}$ on all pairs of neighboring datasets X and Y.

privacy if:

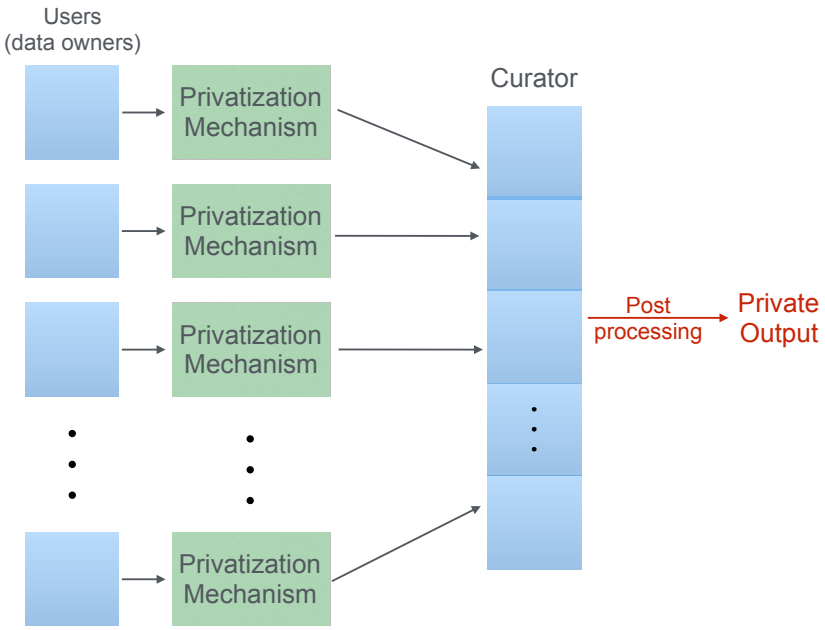
$$P(\mathcal{A}(X) \in R) \leq e^{\epsilon} P(\mathcal{A}(Y) \in R) + \delta \tag{2.5}$$

If X and Y are different only for one single data point, then $d(X, Y) = 1$. For all adjacent X, Y; (ϵ, δ)-differential privacy ensures that with probability $1 - \delta$ (at least), the value of the privacy loss is bounded by ϵ. Here, δ accounts for exceptional cases that might result in high privacy losses. When $\delta = 0$, $\mathcal{A}$ is called ϵ-differential private. In this formula, ϵ manages the amount of information gain for an individual, given the output of the algorithm. When the positive parameter ϵ is smaller, the mechanism provides stronger privacy guarantee [87].

In the literature, differential privacy is examined in two different contexts as illustrated in Figure 2.3. First is in *centralized* (global) privacy context, where the data analyst/curator needs to collect raw data on the server which is required for centralized-DP (CDP). Second is in *local* privacy context, where the data is privatized by users before sending it to the server. In most cases, DP focuses on the case where the users provide data records to a data curator, and assumes that the data aggregation is reliable. The classical definition of DP is based on this scenario and formulated in Equation (2.5). In the literature, this is the most widely used privacy notion and it is also referred as *centralized*-DP [59, 89, 90]. In this thesis, we generally use centralized-DP and call it DP except for Section 4.1. However, sometimes a



(a) Centralized (Global) Differential Privacy



(b) Local Differential Privacy

Figure 2.3. Differential privacy (DP) is examined in two different contexts: a) first is in *centralized* (global) privacy context and b) second is in *local* privacy context.

stronger variant of differential privacy is required when the users do not trust the data curator. In this case, the users need to apply some privatization mechanism to their records in case the users' private information cannot be inferred even by the data curator. Local differential privacy [57, 58, 60, 62, 91] (*LDP*) is inspired by *randomized response* method [92] and classic

DP definition, but eliminates the trusted curator assumption. In *LDP*, an untrusted algorithm is allowed to access a perturbed (or noised) version of a sensitive dataset through a privatization mechanism and must use this perturbed data to perform some estimation [91]. This definition naturally adapted to the multiple party setting [67, 68, 70] where numerous distinct parties hold sensitive data from several individuals, and they wish to jointly estimate parameters of a statistical model conditioned on complete data. The amount of perturbation is controlled by an ϵ parameter. The explicit definition of local differential privacy is given as:

Definition 2.1. *For a given privacy parameter $\epsilon \geq 0$ and a privatization (randomized response) mechanism $\mathcal{R}$ is ϵ-locally differentially private if for all pairs of observations $x, x' \in X$ and for all subsets $\mathcal{S}$ in the range of $\mathcal{R}(\cdot)$:*

$$\frac{P(\mathcal{R}(x) \in \mathcal{S})}{P(\mathcal{R}(x') \in \mathcal{S})} \leq e^{\epsilon} \tag{2.6}$$

where $x \in X$ denote a user's record where X is the data universe. A privatization mechanism $\mathcal{R}$ is a stochastic mapping from X to some output domain Z. If an algorithm perceives only the ϵ-private responses of the observations, then this algorithm satisfies ϵ-LDP.

In Section 4.1, we will explain our LDP setting in detail.

2.2.1. Gaussian Mechanism

There are several possibilities how to make an algorithm differentially private. Input perturbation, objective perturbation and output perturbation are some of those possibilities. For perturbation, one widely used method is using *Gaussian mechanism* [87] that adds Gaussian noise to the revealed information. The amount of the Gaussian noise is determined by the privacy parameters (ϵ, δ) and the sensitivity to changes in the dataset. In this thesis, we use objective perturbation and Gaussian mechanism as a perturbation method.

Theorem 2.2. ***(Gaussian Mechanism (GM), Theorem 3.22 in [93]):*** *Let $\epsilon \in (0, 1)$ be arbitrary.* Gaussian Mechanism *states that given function f with L_2 sensitivity of $\triangle_2 f$, releasing $f(X) + Z$ where $Z \sim \mathcal{N}(0, \sigma^2)$ is (ϵ, δ)-DP when $\sigma \geq \triangle_2 f \sqrt{2 \log(1.25/\delta)}/\epsilon$.*

Definition 2.3. *(L_2-sensitivity): Given two adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$, the important L_2-sensitivity of a function f is defined as:*

$$\triangle_2 f = \sup_{\mathcal{D},\mathcal{D}',\|\mathcal{D}-\mathcal{D}'\|=1} \| f(\mathcal{D}) - f(\mathcal{D}') \|_2$$

In this thesis, this theorem will be our main workhorse. We will recall this theorem in Chapter 4 and Chapter 5.

2.2.2. Composition Theorems

There are several useful properties of differential privacy that make it practical to design sophisticated privacy-preserving machine learning algorithms. One of thesel features is *composability* that enables to combine multiple differentially private algorithms or repeatedly use the same data set. Modular design of algorithms and using an algorithm multiple times on a dataset lower the privacy guarantee because of the information leakage at each time. In the following, we present the composition theorems that show how the privacy parameters compose when differentially private modules are combined. Firstly, we give the definition of the composition rule.

Theorem 2.4. ***(Composition rule, Theorem 3.14 in [89])*** *If algorithm $\mathcal{A}_1$ is (ϵ_1,δ_1)-DP, and $\mathcal{A}_1$ is (ϵ_1,δ_1)-DP, then ($\mathcal{A}_1 \otimes \mathcal{A}_2$) is ($\epsilon_1 + \epsilon_2$,$\delta_1 + \delta_2$)-DP.*

The composition rule can be applied repeatedly to obtain the basic composition theorem: *sequential composition*.

Theorem 2.5. ***(Sequential Composition, Theorem 3.16 in [89])*** *Suppose a set of privacy algorithms $\boldsymbol{\mathcal{A}} = \{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ are sequentially performed on a dataset, and each $\mathcal{A}_i$ provides (ϵ_i,δ_i) privacy guarantee. Then, $\boldsymbol{\mathcal{A}}$ will provide ($\sum_{i=1}^{k} \epsilon_i$,$\sum_{i=1}^{k} \delta_i$)-DP.*

Currently, the simplest way to analyze the privacy budget consumption of a differentially private algorithm is the sequential composition theorem. However, some sophisticated

techniques have been proposed to handle more complicated forms of composition. *Advanced Composition* is the most popular composition theorem in the literature that gives slower growth of ϵ, but has a larger total δ on the composition.

Theorem 2.6. ***(Advanced composition, Theorem 3.20 in [89])*** *For all ϵ_{iter}, δ_{iter}, $\delta' \geq 0$, the class of $(\epsilon_{iter}, \delta_{iter})$-DP mechanisms satisfy $(\epsilon_{tot}, \delta_{tot})$-DP under k-fold adaptive composition for*

$$\epsilon_{tot} = \sqrt{2k\log(1/\delta')}\epsilon_{iter} + k\epsilon_{iter}(e^{\epsilon_{iter}} - 1) \qquad \delta_{tot} = k\delta_{iter} + \delta' \tag{2.7}$$

When the data is iteratively used over many iterations, the advanced composition provides more strict ϵ_{tot}. In this thesis, we will use the data iteratively, so using the advanced composition is important. We will recall this theorem in Chapter 4 and Chapter 5. Now, we briefly introduce the most recently proposed privacy notion: *Concentrated Differential Privacy*.

2.2.3. Concentrated Differential Privacy

Concentrated differential privacy [83] is a recent variation of DP that provides high probability bounds for cumulative privacy loss. It is proposed to make privacy-preserving iterative algorithms viable to the real world privacy protection scenarios while still providing strong privacy guarantees. For a certain amount of privacy guarantee, concentrated DP requires much less additional noise compared to the DP. The concentrated DP framework treats the privacy loss of an outcome,

$$L^{(o)}_{(\mathcal{A}(\mathcal{D})\|\mathcal{A}(\mathcal{D}'))} = \log \frac{P(\mathcal{A}(\mathcal{D}) = o)}{P(\mathcal{A}(\mathcal{D}') = o)} \tag{2.8}$$

as a random variable. Two concentrated DP methods are proposed in the literature. The first one is (μ, τ)-mCDP [83] where μ is the mean of this privacy loss and the second one is τ-zCDP which is proposed by Bun and Steinke in [84]. The DP mechanisms and applications can be converted to zCDP mechanisms, since they are conjugates. We will define the prepositions

and lemmas in the following sections that are used to compute DP in terms of zCDP. However, DP and mCDP are not conjugates and they cannot be converted to each other. For a fair comparison with (ϵ, δ)-DP, we will use zCDP as our second privacy definition.

Zero Concentrated Differential Privacy . The relation between the moment generating function of $L(o)$ and Rényi divergence between $\mathcal{A}_{(\mathcal{D})}$ and $\mathcal{A}_{(\mathcal{D}')}$ reveals the τ-zCDP [84]. The Rényi Divergence between the probability distributions P_1 and P_2 is defined of order $\alpha \in (1, \infty)$ as:

$$D_\alpha(P_1 \parallel P_2) = \frac{1}{\alpha - 1} \log \left(\mathop{\mathbb{E}}_{x \sim P_1} \left[\left(\frac{P_1(x)}{P_2(x)} \right)^{(\alpha-1)} \right] \right)$$

The definition of zCDP where D_α is the Rényi Divergence is given as: A randomized mechanism $\mathcal{A}$ is (ξ,ρ)-zCDP, if for all $x, x' \in \mathcal{X}^n$ such that $\parallel x - x' \parallel \leq 1$ and all $\alpha \in (1, \infty)$,

$$D_\alpha(\mathcal{A}(x) \parallel \mathcal{A}(x')) \leq \xi + \rho\alpha$$

There is a correspondence between the classical definition of DP and zCDP; where zCDP satisfies the key basic properties of DP such as composition and postprocessing. Most importantly, zCDP satisfies the Gaussian mechanism, and we will use this property in the following Chapters for our privacy analyses.

Moments Accountant . In the same way as zCDP, the moments accountant (MA) method is used to effectively keep track of the cumulative distribution of the privacy loss random variables by bounding the moments of the privacy loss $L^{(o)}$ where the λ^{th} moment is

defined as the log of the moment generating function evaluated at λ [81]:

$$\alpha_{\mathcal{A}}(\lambda; X, X') = \log \mathbb{E}_{o\sim\mathcal{A}(X)} \left[e^{\lambda L^{(o)}} \right] \tag{2.9}$$

Then the moments accountant is formalized as below:

Definition 2.7. *(Moments accountant) The moments accountant with an integer parameter λ on neighboring datasets X and X' is defined as:*

$$\alpha_{\mathcal{A}} = \max_{X,X'} \alpha_{\mathcal{A}}(\lambda; X, X') \tag{2.10}$$

The λ^{th} moment in Equation (2.9) composes linearly, which yields the composability theorem which is defined below.

Theorem 2.8. *(Composability) Suppose that $\mathcal{A}$ consist of a sequence of mechanisms $\mathcal{A}_1, \cdots \mathcal{A}_k$. Then for any λ, the sum of each upper bound on $\alpha_{\mathcal{A}_i}$ is an upper bound on the total λ^{th} moment after k compositions:*

$$\alpha_{\mathcal{A}}(\lambda) = \sum_{i=1}^{k} \alpha_{\mathcal{A}_i}$$

Besides, it is possible to convert the λ^{th} moment to the (ϵ, δ)-DP using Markov's inequality. For any $\epsilon > 0$, the algorithm $\mathcal{A}$ is (ϵ, δ)-DP for:

$$\delta = \min_{\lambda} \exp\left(\alpha_{\mathcal{A}}(\lambda) - \lambda\epsilon\right) \tag{2.11}$$

2.2.4. Private Bayesian Inference

Bayesian inference provides a methodology for parameter estimation where inference refers to the calculation of the posterior distribution of the desired parameters. Unfortunately, the exact calculation becomes intractable for most useful models. Monte Carlo methods, which constitute a broad group of Bayesian inference methods, are typically used to solve intractable integration problems by generating random samples from the posterior distribution, either exactly or approximately. One of the most popular Monte Carlo methods for Bayesian inference is Markov chain Monte Carlo (MCMC) [94], where one generates a sequence of samples from a Markov chain that has the target posterior as its stationary distribution. The MCMC methods produce random outputs; therefore they are the natural candidates for privacy preserving methods by their inherent randomness. The connection between MCMC methods and differential privacy has recently been studied by Wang *et al* [27].

Let us assume that we want to draw model parameter θ differentially privately from a posterior distribution $p(\theta|X) \propto p(\theta)\prod_{i=1}^{N}(x_i|\theta)$ where the data items are $X = \{x_i\}_{i=1}^{N}$, $p(x|\theta)$ is the probability of data item x and $p(\theta)$ is the prior distribution of the given model. The claim in [27] is that sampling from the posterior distribution is intrinsically differentially private.

Theorem 2.9. ***(One posterior sample (OPS), Theorem 1 in [27])*** *If* $\max_{x\in\mathcal{X},\theta\in\Theta}|\log p(x|\theta)|\leq B$*, releasing one sample from the posterior distribution* $p(\theta|X)$ *with any prior is* $4B$*-DP.*

The proof is straightforward when we apply the exponential mechanism [95]. First, we briefly introduce the exponential mechanism that maps input-output pairs to a real valued scores when the utility function u is given. Formally, the exponential mechanism $\mathcal{M}(X, u, \epsilon)$ produces outputs as:

$$p(\mathcal{M}(X, u, \epsilon) = R) \propto \exp\left(\frac{\epsilon u(X, R)}{2\Delta u}\right)$$

where $\Delta u = \max_{R,(X,X')} \parallel u(X, R) - u(X', R) \parallel_1$ is the sensitivity of u and $d(X, X')$ = 1.

Now, we can explain the posterior sampling as an instance of the exponential mechanism with utility function $u(X, \theta) = \log p(\theta, X)$. The sensitivity of the OPS method is $\Delta \log p(\theta, X)$ that equals to $\max_{\theta,(X,X')} \parallel \log p(\theta, X) - \log p(\theta, X') \parallel_1$. The prior is independent of the data, so we have $\Delta \log p(\theta, X) = \max_{\theta,x,x'} \parallel \log p(x'|\theta) - \log p(x|\theta) \parallel$. This result gives the privacy cost of posterior sampling. Under exponential mechanism, sampling θ from the posterior $P(\theta|X)$ when $\epsilon = 2\Delta \log p(\theta, X)$ is exactly the same with standard sampling from a posterior distribution. For the detailed proof, see [27].

In this thesis, we use the connection between privacy protection and SGLD and the locally private Bayesian inference that is proposed to sample from a posterior distribution under local privacy constraints. So, we introduces the privacy of sampling from a posterior distribution here and we explain the locally private Bayesian Inference and SGLD in the following.

Locally Private Bayesian Inference . In this section, we briefly show the objective of locally private Bayesian inference. In this case the Bayesian posterior is formed with a generative model, observed data X, latent variables Θ and a privatization mechanism $\mathcal{R}$ that generates the privatized datasets $Z \sim P_{\mathcal{R}}(Z \mid X, \epsilon)$. Eventually we write the posterior as in the following:

$$P_{\mathcal{R}}(\Theta \mid Z, \epsilon) = \mathbb{E}_{P_{\mathcal{R}}(Z|X,\epsilon)} \left[P(\Theta \mid X) \right] = \int P(\Theta \mid X) P_{\mathcal{R}}(X \mid Z, \epsilon) dX \tag{2.12}$$

The observed data X is treated as a latent variable and it is marginalized over with the privatization distribution $P_{\mathcal{R}}(X \mid Z, \epsilon)$. The privatization distribution is also a posterior that characterizes the uncertainty about X when Z, $\mathcal{R}$ and ϵ are given. Here, we first generate samples from $P_{\mathcal{R}}(X \mid Z, \epsilon)$, then we use them to approximate the expectation in Equation (2.12) by using an approximation method that is used to sample from the original posterior $P(\Theta \mid X)$. In Section 4.1.4, we proposed an approach (Gibbs sampling) that is used to sample from $P_{\mathcal{R}}(X \mid \Theta, Z, \epsilon)$ in order to obtain ϵ-LDP.

2.3. Stochastic Gradient Langevin Dynamics

In this section, we begin to explain SGLD with the *maximum a posteriori* (MAP) estimation of model parameter vector θ from the posterior $p(\theta|X)$. In general, stochastic optimization method is used to find the MAP estimation of θ_t as follows:

$$\theta_t = \theta_{t-1} + \frac{\kappa_t}{2}\left(\frac{K}{B}\sum_{k\in\mathcal{B}_t}\nabla \log p(x_k \mid \theta_{t-1}) + \nabla \log p(\theta_{t-1})\right)$$

where $\mathcal{B}_t \subset \{1,\ldots,N\}$ is a random data subsample that is drawn with or without replacement, $B = \mid \mathcal{B}_t \mid$ is the number of data points in $\mathcal{B}_t$ and κ_t is the step size. For convergence to a local maximum, the step size should satisfy the following requirements: i) $\sum_{t=1}^{\infty}\kappa_t = \infty$ and ii) $\sum_{t=1}^{\infty}{\kappa_t}^2 < \infty$.

As we mention in Section 2.2.4, MCMC techniques are used to capture parameter uncertainty. Langevin Dynamics [96] is a gradient-based MCMC method that produces samples from the posterior by taking gradient steps and injecting a Gaussian noise into the parameter updates:

$$\theta_t = \theta_{t-1} + \kappa\left(\sum_{i=1}^{N}\nabla \log p(x_i \mid \theta_{t-1}) + \nabla \log p(\theta_{t-1})\right) + \psi_t \qquad \psi_t \sim \mathcal{N}(0,\kappa) \tag{2.13}$$

Here, the mean of the newly proposed θ is in the direction of increasing log posterior by cause of the gradient. To prevent the samples from collapsing to a single local maximum, we need some amount of added noise. The amount of the noise can be explained by the Metropolis-Hastings (MH) algorithm [97, 98] that is one of the most generally applicable MCMC algorithms. In this algorithm, the proposed samples θ' are drawn from a proposal distribution $q(\theta'|\theta^t)$ at iteration t with acceptance probability $\min\{1, \frac{p(\theta'|X)q(\theta^t|\theta')}{p(\theta^t|X)q(\theta'|\theta^t)}\}$ where θ^t is the current parameter from the posterior $p(\theta^t|X)$. When κ in Equation 2.13 goes to 0, the acceptance probability of MH goes to 1 where the Markov chain has $p(\theta|X)$ as its stationary distribution [99].

An important attempt for scaling up MCMC techniques was made by [29], where the

authors combined the ideas from statistical Langevin Dynamics and stochastic optimization, and developed a scalable MCMC framework called the SGLD. In algorithmic sense, SGLD is very similar to Stochastic Gradient Descent (SGD); the only difference is that SGLD injects a Gaussian noise to the gradient at each iteration. SGLD exploits the assumption that the data samples $X = \{x_n\}_{n=1}^{N}$ are i.i.d. and it asymptotically generates a sample $\theta^{(t)}$ from the posterior distribution $p(\theta \mid X) \propto p(\theta)\, p(X \mid \theta)$ by iteratively applying the following update equation at iteration t:

$$\theta_t = \theta_{t-1} + \frac{\kappa_t}{2}\left(\frac{K}{B}\sum_{k\in\mathcal{B}_t} \nabla \log p(x_k \mid \theta_{t-1}) + \nabla \log p(\theta_{t-1})\right) + \psi_t$$

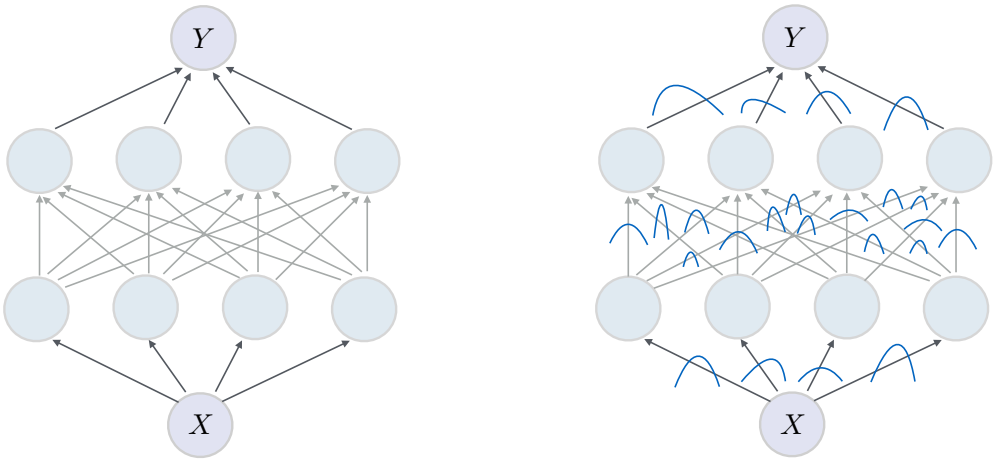
where ψ_t is Gaussian noise: $\psi_t \sim \mathcal{N}(\kappa_t \mathbb{I})$ where $\mathbb{I}$ stands for the identity matrix.

We began with an overview of the SGLD algorithm that is relevant to the understanding of our approach. In the following, we will state the specialized version of the SGLD algorithm that guarantees differential privacy and present our methods that guarantees differential privacy for CMF, CTF models in Chapter 4 and BNN in Chapter 5.

2.4. Bayesian Neural Networks

In this thesis, we assume that the input data provided to the neural networks is $\mathcal{D} = \{d_i\}_{i=1}^{N}$ where $d_i = (x_i, y_i)$, with input object/feature $x_i \in \mathcal{R}^D$ and output label $y_i \in \mathcal{Y}$, with $\mathcal{Y}$ being the output discrete label space. A model characterizes the relationship from x to y with parameters (or weights) θ. Our goal is to tune the parameters θ of a model $p(y|x, \theta)$ that predicts y given x and θ. As we show in the previous sections, Bayesian inference in such a model consists of updating some initial belief over parameters θ in the form of a prior distribution $p(\theta)$, after observing data $\mathcal{D}$, into an updated belief over these parameters in the form of the posterior distribution $p(\theta|\mathcal{D})$. The posterior distribution of a set of N items is $p(\theta|\mathcal{D}) \propto p(\theta)\, p(\mathcal{D}|\theta)$ where the corresponding data likelihood is $p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(d_i|\theta)$.

Computing the posterior is often difficult in practice as it requires the computation of analytically intractable integrals, so we need to use approximation techniques. For the



(a) Standard DNN (b) Bayesian DNN

Figure 2.4. Illustration of standard DNN and Bayesian DNN with two-hidden-layers model. All weights in DNNs are represented with fixed values in 2.4(a) and all weights are represented as distributions in 2.4(b).

inference, we leverage the advances in SGLD to train deep neural networks (DNNs). All weights in Bayesian DNNs are represented as distributions (as shown in Figure 2.4(a)); rather than having fixed values (as shown in Figure 2.4(b)). The SGLD algorithm learns the weight uncertainty jointly on all parameters. Our aim is to infer a distribution over parameters θ on a given dataset $\mathcal{D}$ by following the Bayesian approach. This distribution encapsulates the prior belief as to which parameters are likely to be generated before observing any data. Given weight matrices θ, standard Gaussian prior distributions are usually placed over the weight matrices, $p(\theta) = \mathcal{N}(\theta; 0, \mathbb{I})$. We also need to describe a distribution over the outputs $p(y|x, \theta)$ that captures how likely the function parameters are, given our observed data. The form of relationships from x to y are parameterized as a composition of a set of nonlinear functions in Bayesian DNNs. There are several of nonlinear functions such as the sigmoid function used in logistic regression; but a softmax function is generally used by an M-layer DNN for multiclass classification:

$$p(y|x, \theta) = \text{softmax}(g_{\theta_M} \circ \ldots \circ g_{\theta_0}(x)), \tag{2.14}$$

Here, $\circ$ denotes the function composition, $softmax(x) = e^x/(\sum_i e^{x_i})$ and $g_\theta(x)$ can be the

sigmoid function, hyperbolic tangent or the Rectified Linear Unit (ReLU) [100]. In this paper, we use feed-forward neural networks. In these networks, the feature representations [101,102] are learned by forwarding the data from the bottom layer to the top layer. There are nonlinear functions between the layers. In the m^{th} layer, we choose the ReLU function for g_{θ_m} in Equation 2.14 which is defined as:

$$g_{\theta_m} = \max(0, \theta_m^T x)$$

3. COUPLED TENSOR FACTORIZATION

As mentioned in Chapter 1, an effective way of including side information and faithful representation of multiple attributes without loss of structure is Coupled Tensor Factorization (CTF) models. The real world datasets that are represented by several tensors/matrices need custom models and CTF models bring the modeling flexibility which is required. In this chapter, we describe the inference methods in order to estimate the latent factors for all possible model topologies and coupled factorization models in compliance with the privacy requirements of the use case.

3.1. Inference

Most of the methods on matrix and tensor factorization in the literature focus on obtaining point estimates such as Maximum A-Posteriori Estimations [12, 16, 103–105]. However, it has been proven that the full-Bayesian approaches are more robust to overfitting, incorporates confidence intervals and side information easily and allows model selection inherently [106–108]. Here, starting with the Maximum A-Posteriori Estimation, we develop full-Bayesian approaches that allows more powerful modeling and inference, such as Variational Inference that approximates the intractable posterior distribution and Monte Carlo methods that sample from the intractable posterior.

3.1.1. Maximum Likelihood and A-Posteriori Estimation

The estimation of the shared latent factors Θ_d, can be achieved via iterative optimization methods (see [15]) applied to the posterior density given below:

$$P(\{\Theta_d\}_{d\in S_v}|X^{(v)}) \propto P(X^{(v)}|\{\Theta_d\}_{d\in S_v}) \prod_{d\in S_v} P(\Theta_d) \tag{3.1}$$

For non-negative data and factors, one can obtain the following compact fixed point equation where each Θ_d is updated in an alternating fashion fixing the other factors $\Theta_{d'}$ for $d' \neq d$

$$\Theta_d \leftarrow \Theta_d \circ \frac{\sum_v R^{v,d} \phi_v^{-1} \Delta_{d,v}(M^{(v)} \circ (\hat{X}^{(v)})^{-p_v} \circ X^{(v)})}{\sum_v R^{v,d} \phi_v^{-1} \Delta_{d,v}(M^{(v)} \circ (\hat{X}^{(v)})^{1-p_v})} \tag{3.2}$$

where $\circ$ is the Hadamard (element-wise) product, $M^{(v)}$ is a 0-1 mask array with $M^{(v)}_{i_v} = 1$ ($M^{(v)}_{i_v} = 0$) if $X^{(v)}_{i_v}$ is observed (missing) and $R^{v,d}$ is a *coupling matrix* that is defined as:

$$R^{v,d} = \begin{cases} 1 & \text{if } X^{(v)} \text{ and } \Theta_d \text{ connected} \\ 0 & \text{otherwise} \end{cases} . \tag{3.3}$$

In this iteration, the key quantity is the $\Delta_{d,v}$ function that is defined as follows:

$$\Delta_{d,v}(A) = \left[\sum_{i_v \cap \bar{i}_d} A^{(v)}_{i_v} \sum_k \prod_{d \notin S_v} \theta^{(d)}_{i_d k} \right] \tag{3.4}$$

where $\bar{i}_d = i_v - i_d$ and $A = M^{(v)} \circ (\hat{X}^{(v)})^{1-p_v}$.

Lastly, p_v determines the cost function while dispersion parameter ϕ_v is used for data driven regularization and weighting in coupled factorization of heterogeneous datasets. In [17], they tackle learning the dispersion parameters ϕ_v when $p \in \{0, 1, 2, 3\}$ by using a probabilistic approach, which makes use of the relation between the β-divergence and the family of Tweedie distributions and enables to find the dispersion parameters by maximizing the likelihood.

It is possible to choose different cost functions (different p_v) for each observed data in a coupled model if each $X^{(v)}$ is modeled by a different type of distribution. Here, we solved update equations under the assumption of each observation tensor is modeled by the same type of distribution having the same dispersion parameter. This results in the same cost function (p_v) for all the observed tensors $X^{(v)}$ and we can cancel out the dispersion parameters from the update equations. See Table 3.1 for update rules for different p_v values.

Table 3.1. Update rules for different p_v values.

p_ν	Cost Function	Multiplicative Update Rule
0	Euclidean	$\Theta_d \leftarrow \Theta_d \circ \frac{\sum_v R^{v,d}\phi_v^{-1}\Delta_{d,v}(M^{(v)} \circ X^{(v)})}{\sum_v R^{v,d}\phi_v^{-1}\Delta_{d,v}(M^{(v)} \circ \hat{X}^{(v)})}$
1	Kullback-Leibler	$\Theta_d \leftarrow \Theta_d \circ \frac{\sum_v R^{v,d}\phi_v^{-1}\Delta_{d,v}(M^{(v)} \circ (\hat{X}^{(v)})^{-1} \circ X^{(v)})}{\sum_v R^{v,d}\phi_v^{-1}\Delta_{d,v}(M^{(v)})}$
2	Itakura-Saito	$\Theta_d \leftarrow \Theta_d \circ \frac{\sum_v R^{v,d}\phi_v^{-1}\Delta_{d,v}(M^{(v)} \circ (\hat{X}^{(v)})^{-2} \circ X^{(v)})}{\sum_v R^{v,d}\phi_v^{-1}\Delta_{d,v}(M^{(v)} \circ (\hat{X}^{(v)})^{-1})}$

3.1.2. Variational Inference

In this section, we present a variational Bayesian method to make inference on the coupled tensor factorization models (CTF-VB) and to derive update equations for these models that handles the simultaneous tensor factorizations where multiple observations tensors are available. Here, we use the Kullback-Leibler (KL) divergence as the cost function which is equivalent to selecting the Poisson observation model [109,110], while our approach can be extended to other costs where a composite structure is present. The overall probabilistic model where the symbols refer to Poisson and Gamma distributions respectively is defined at below.

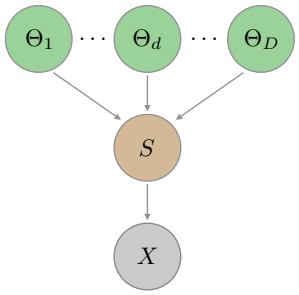


Figure 3.1. The generative model of the tensor factorization framework as a Bayesian network. The directed acyclic graph describes the dependency structure of the variables: the full joint distribution can be written as $p(X, S, \Theta_{1:D}) = p(X|S)p(S|\Theta_{1:D}) \prod_d p(\Theta_d)$.

$$\boldsymbol{\theta}_{i_d}^{(d)} \sim \mathcal{G}(\boldsymbol{\theta}_{i_d}^{(d)}; A_{i_d}^{(d)}, B_{i_d}^{(d)}) \quad \text{(factor priors)}$$

$$\Lambda = \prod_d \boldsymbol{\theta}_{i_d}^{(d)} \quad \text{(intensity)}$$

$$S \sim \mathcal{PO}(S; \Lambda) \quad \text{(components)}$$

$$X_{i_v} = \sum_k S \quad \text{(observation)}$$

$$\hat{X}_{i_v} = \sum_k \Lambda \quad \text{(parameter)}$$

$$\mathcal{PO}(s; \lambda) = e^{-\lambda} \frac{\lambda^s}{s!} \quad \text{(poisson)}$$

$$\mathcal{G}(\theta; a, b) = e^{-b\theta} \frac{\theta^{a-1} b^a}{\Gamma(a)}. \quad \text{(gamma)}$$

In this model, we choose the Gamma prior on the factors in order to preserve conjugacy. The graphical model that represents a single tensor factorization is depicted in Figure 3.1. Note that $p(X|S)$ is a degenerate distribution that is defined as: $p(X|S) = \prod_{i_v} \delta(X_{i_v} - \sum_k S)$. Here, $\delta(.)$ is the Kronecker delta function where $\delta(x) = 1$ when $x = 0$ and $\delta(x) = 0$ otherwise.

For a Bayesian point of view, a model is associated with a random variable Φ and interacts with the observed data X simply as $p(\Phi|X) \propto p(X|\Phi)p(\Phi)$. The quantity $p(X|\Phi)$ is called *marginal likelihood* [111] and it is average over the space of the parameters, in our case, S and Θ as [109].

$$p(X|\Phi) = \int_\Theta d\Theta \sum_S p(X|S, \Theta, \Phi)p(S, \Theta|\Phi) \tag{3.5}$$

On the other hand, computation of this integral is itself a difficult task that requires averaging on several models and parameters. There are several approximation methods such as sampling or deterministic approximations such as Gaussian approximation. One other approximation method is to bound the log marginal likelihood by using *variational inference* [109, 111, 112] where an approximating distribution q is introduced into the log

marginal likelihood equation:

$$\log p(X|\Phi) \geq \int_{\Theta} d\Theta \sum_{S} q(S|\Theta) \log \frac{p(X, S, \Theta|\Phi)}{q(S, \Theta)} \tag{3.6}$$

where the bound attains its maximum and becomes equal to the log marginal likelihood whenever $q(S, \Theta)$ is set as $p(S, \Theta|X, \Phi)$, that is the exact posterior distribution. However, the posterior is usually intractable, and rather, inducing the approximating distribution becomes easier. Here, the approximating distribution q is chosen such that it assumes no coupling between the hidden variables such that it factorizes into independent distributions as $q(S, \Theta) = q(S)q(\Theta)$. As exact computation is intractable, we will resort to standard variational Bayes approximations [111, 112]. The interesting result is that we get a belief propagation algorithm for marginal intensity fields rather than marginal probabilities.

We present variational Bayesian coupled tensor factorization as an approach to exploiting side information, i.e., each decomposition is coupled by sharing some factor matrices. As in Section 3.1.1, each observed tensor $X^{(v)}$ has a corresponding index set i_v and we define the $V \times D$ coupling matrix R. Additionally, we define another particular configuration for index set of $S^{(v)}$ will be denoted by $\bigcup_{R_{v,d}=1} i_d \equiv r_v$. For the coupled factorization (CTF-VB), we get the following expression of the cell probabilities $P_v(r_v)$ here as:

$$P_v(r_v) = \frac{\exp(\sum_d \langle \log \Theta_d \rangle)}{\sum_k \exp(\sum_d \langle \log \Theta_d \rangle)} = \frac{\prod_d \exp(\langle \log \Theta_d \rangle)}{\sum_k \prod_d \exp(\langle \log \Theta_d \rangle)} = \frac{\prod_d L_d}{(\hat{X}_L)^{(v)}_{i_v}} \tag{3.7}$$

Then the sufficient statistics $\langle S_v(r_v) \rangle$ turns to

$$\langle S_v(r_v) \rangle = X^{(v)}_{i_v} P_v(r_v) = \frac{X^{(v)}_{i_v}}{(\hat{X}_L)^{(v)}_{i_v}} \prod_d L_d \tag{3.8}$$

Now we turn to formulating $q(\Theta)$. The distribution q_{Θ_d} is obtained similarly as after we

expand the log and drop irrelevant terms it becomes proportional to

$$
\begin{aligned}
q_{\Theta_d} &\propto \exp\left(\langle \log p(S|\Theta) + \log p(\Theta|\Phi)\rangle_{q/q_{\Theta_d}}\right) \\
&\propto \log \Theta_d(A_d - 1 + \sum_v R^{v,d} \sum_k \langle S_v(r_v)\rangle^{R^{v,d}}) \\
&- \Theta_d(\frac{A_d}{B_d} + \sum_v R^{v,d} \sum_k \prod_{d\prime \neq d} \langle \Theta_{d\prime}\rangle^{R^{v,d}})
\end{aligned}
\tag{3.9}
$$

which is the distribution $q_{\Theta_d} \sim \mathcal{G}(C_d, D_d)$ where the shape and scale parameters for q_{Θ_d} are

$$
C_d = A_d + \sum_v R^{v,d} L_d \circ \Delta_d(M^{(v)} \circ X^{(v)} / (\hat{X}_L)^{(v)}) \tag{3.10}
$$

$$
D_d = (A_d/B_d + \sum_v R^{v,d} \Delta_d(M^{(v)}))^{-1} \tag{3.11}
$$

that, in turn, since $\langle \Theta_d \rangle$ is $C_d \circ D_d$, the sufficient statistics for $q(\Theta_d)$ become:

$$
\langle \Theta_d \rangle = E_d \leftarrow \frac{A_d + \sum_v R^{v,d} L_d \circ \Delta_d(M^{(v)} \circ X^{(v)} / (\hat{X}_L)^{(v)})}{\frac{A_d}{B_d} + \sum_v R^{v,d} \Delta_d(M^{(v)})} \tag{3.12}
$$

3.1.3. Markov Chain Monte Carlo

In this section we present how to use SGLD as an inference method to develop our CTF algorithm. Recently *Ahn et al* [113] proposed a distributed stochastic gradient MCMC algorithm (SG-MCMC) for the matrix factorization framework that uses the same model in Bayesian probabilistic matrix factorization (BPMF) [106]. In this section, we extend the existing work and propose a generalized approach that can be used for different Bayesian models and for the inference and estimation of various coupled tensor factorization models.

In the following, we consider the general structure of a probabilistic model to be able to cover several likelihood functions. In Chapter 4 we will provide two specific probabilistic models, namely Poisson-Gamma and Normal-Normal for arbitrary collections of tensors and

matrices. In MCMC, we are interested in approximating posterior distribution as follows:

$$P(X_{i_v}^{(v)^*}|X_{i_v}^{(v)}) \approx \frac{1}{T}\sum_{t=1}^{T} P(X_{i_v}^{(v)^*}|\{\Theta_d{}^{(t)}\}_{d\in S_v}) \tag{3.13}$$

where $\{\Theta_d^{(t)}\}$ is the t^{th} sample from the posterior distribution: $P(\{\Theta_d\}_{d\in S_v}|X^{(v)})$. We will now describe how SGLD is used to generate these samples in CTF models. We will consider the coupled model that is illustrated in Figure 2.1.

$$X^{(1)} \approx \hat{X}^{(1)} = \left\langle \boldsymbol{\theta}_{i_1}^{(1)}, \boldsymbol{\theta}_{i_2}^{(2)}, \boldsymbol{\theta}_{i_3}^{(3)} \right\rangle \tag{3.14}$$

$$X^{(2)} \approx \hat{X}^{(2)} = \left\langle \boldsymbol{\theta}_{i_2}^{(2)}, \boldsymbol{\theta}_{i_4}^{(4)} \right\rangle \tag{3.15}$$

As we mentioned in Section 2.3, SGLD is very similar to the SGD algorithm. There is only one difference that SGLD adds a Gaussian noise to SGD at each iteration. For CTF models, the update rules used for estimation of the independent and the shared factors are different. In each iteration, SGLD applies the update rule given below to get the samples of the private factors:

$$\Theta_d^{(t)} = \Theta_d{}^{(t-1)} + \nabla\Theta_d{}^{(t)} \tag{3.16}$$

for $d = \{1, 3, 4\}$ and the shared factor:

$$\Theta_d^{(t)} = \Theta_d{}^{(t-1)} + \sum_{v\in\mathcal{V}} \nabla\Theta_d{}^{(t)} \tag{3.17}$$

for $d = 2$ and $V = \{1, 2\}$ (Because, the common latent structure Θ_2 participates in both $X^{(1)}$ and $X^{(2)}$ in our specific model). In these equations $\Delta\Theta_d{}^{(t)}$ s are computed as:

$$\Delta\Theta_d{}^{(t)} = \kappa_t\Big(\frac{C_v}{B}\sum_{i_v\in\mathcal{B}^{(t)}} \nabla_{\theta_{i_dk}} \log P(X_{i_v}^{(v)} \mid \{\theta_{i_dk}{}^{(t-1)}\}_{d\in S_v}) + \nabla_{\theta_{i_dk}} \log P(\theta_{i_dk}{}^{(t-1)})\Big) + \psi^{(t)} \tag{3.18}$$

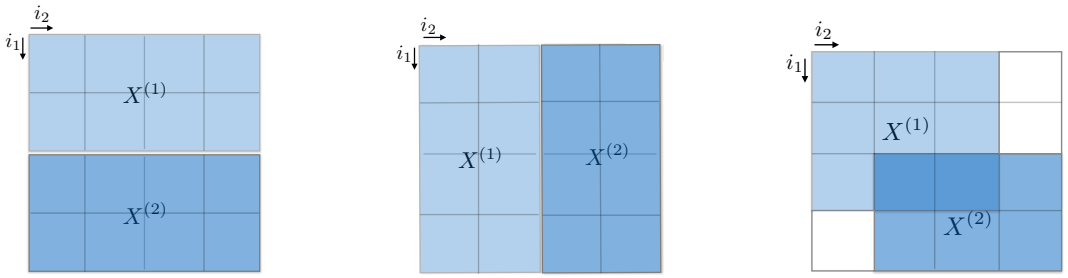
where the conditional distribution of the observed dataset $X^{(v)}$ (the likelihood term) and the

prior distributions over Θ_d s are given as: $P(X^{(v)}|\{\Theta_d\}_{d\in S_v})$ and $P(\Theta_d|\tau_{\Theta_d})$ respectively (Here, τ_{Θ_d} is the set of hyperparameters of the prior distribution). Besides, C_v is the number of elements in $X^{(v)}$, i_v is the row indices of $X^{(v)}$, B is the minibatch size, ∇ denotes the gradients and $\mathcal{B}_{(t)}^{(v)} \subset n_1 \times \ldots \times n_{|S_v|}$ is the set of entries that are subsampled from $X^{(v)}$ at iteration t. The entries of the noise matrix $\psi^{(t)}$ are independently sampled from a Normal distribution: $\psi^{(t)} \sim \mathcal{N}(0, 2\kappa_t \mathbb{I})$. For convergence, the step size κ_t must satisfy:

$$\sum_{t=0}^{\infty} \kappa_t = \infty \qquad \text{and} \qquad \sum_{t=0}^{\infty} \kappa_t^2 < \infty$$

4. DIFFERENTIALLY PRIVATE COUPLED TENSOR FACTORIZATION

Data matrices are the fundamental structure in several applications, like recommender systems, text mining and gene expression. Note that, matrices are natural for describing binary relations (such as user-item) but many data analysis tasks call for more complex setups where the data is collected from heterogeneous data sources and consists of ternary or higher order relations (such as user-item-time) and they are represented by a collection of matrices and/or tensors. In this chapter, we present an approach that allows us making useful inferences from a distributed dataset organized as a collection of matrices and tensors without explicitly revealing individual data items. Each matrix/tensor in our dataset represents a relation between multiple entities. Specifically, we allow parts of each data to be owned by a different party (for example ratings for the same movie by different users on different sites); hence data is naturally distributed. In such a distributed scenario, the dataset can be jointly analyzed by CTF models.



(a) (b) (c)

Figure 4.1. Matrix partitioning scenarios: a) row-distributed, b) column-distributed, c) arbitrarily distributed entries.

For the models presented in this chapter, we consider that the target dataset $\mathcal{X}$ is partitioned into N subsets as $\{X^{(v)}\}_{v=1}^{N}$ (or N subsets composes $\mathcal{X}$) where the subsets are located in N distinct data sites. Each data subset $X^{(v)} \in \mathbb{R}^{n_1 \times \ldots \times n_{|S_v|}}$ is approximated as the outer product of rank-K factors $\Theta_d^{(v)} \in \mathbb{R}^{n_d \times K}$, $\forall d \in \{1, \ldots, |S_v|\}$. Several scenarios can

appear subject to the distribution of matrix entries across distinct data sites. In the simplest scenario of a matrix, we can identify three cases that have practical privacy implications: *i)* row-distributed, *ii)* column-distributed and *iii)* arbitrarily-distributed (possibly overlapping) as shown in Figure 4.1. The row-distributed scenario is appropriate for recommender systems where disjoint sets of users rate a common set of items. The healthcare systems naturally fit to the column-distributed scenario where medical conditions (test results, symptoms, etc.) of a patient are distributed among several hospitals and insurance companies. In this case, the hospitals/companies have common patients' records on disjoint sets of medical conditions and they may require to protect the privacy of medical conditions on the records.

Besides, we may have more complex interactions in practice. Some of the items and users can be common to both sites, but each site may have specific data not shared by the other peer. In this mixed scenario, we have multiple data partitions from multiple companies, and can encounter common users and/or items as in the example given by Figure 4.1(c). In case of more complex relational data, we may have tensors with distributed entries. Such tensor data with a richer relational structure calls for more flexible arrangements in terms of entity type and particular privacy requirement.

In this chapter, we will propose a flexible method that covers all kinds of entry distributions. However, our *main* focus is to preserve the users' privacy; so we design a setting that the data owners do not inform the other sites about their users' identities. To clarify, each user will be assumed to be unique while items can be common at different sites. In the matrix case, this corresponds to a row-distributed scenario, yet for tensor data we can have other common entity types. For example, an entire slice of a tensor can have specific information about a user over several entities, that we assume are private only at a given site, however other entities are common and known by the other parties.

As we mentioned in Section 2.2, differential privacy is examined in two different context: *centralized-DP* (CDP) and *local-DP* (LDP). In Section 4.1, we propose a provably differentially private treatment of CMF models for these two notions of privacy. To provide the CDP, we use the connection between differential privacy and SGLD. Then, we propose two different CMF models that provide LDP by using the privatization method introduced

in [57] and Gibbs sampling [114,115] for inference in the locally differentially private models. In Section 4.2, we propose CTF models that provides CDP with advanced composition (AC) and composition under zCDP constraints.

In addition to the centralized-DP and the local-DP, the proposed methods allow us to define two levels of privacy protection: *i) user-level* and *ii) site-level*. The *user-level* DP provides privacy protection for the user-definite sensitive information for each user in the data site while using accumulated information for the profit of all users. This privacy notion protect each user's privacy against every single user even if they are in the same data site. It means the individuals don't feel free to allow their raw data to be used even by their own data sites, so the local data is not fully accessible to its owner. This is a stronger form of privacy protection than the *site-level* privacy.

Besides, the need for *site-level* privacy appears in distributed setting where each one of the N parties holds some private data and desires to jointly process the all data to obtain some valuable output. The important thing is that each data site respects the privacy of the others and the site-definite data privacy is provably protected for all data sites. In this setting, it can be acceptable that each site can use their own data without privacy protection (the local data is accessible to its owner), but it should release the data by providing ϵ-DP for its users independently. Thus, for a data site n, *user-level* DP is not protected in *site-*n but it is protected against the other $N-1$ data sites.

We will first develop our differentially private CMF models in Section 4.1 and then we describe more generalized differentially private CTF models in Section 4.2. We give specific names to the methods based on these privacy definitions in the following sections.

4.1. Coupled Matrix Factorization

In this section, we will present a coupled matrix factorization (CMF) model which is a subset of CTF model in a distributed scenario. We will start with the CMF model in order to make the model and notation simpler. In this way, the differentially private data sharing model via matrix and tensor factorization becomes easier to understand. Afterwards, we generalize

the differentially private algorithms for CMF models to CTF models to cover all possible model topologies and data sharing settings.

First, we will describe a provably differentially private treatment of several CMF models for the two notions of privacy: i) centralized differential privacy (CDP) and ii) local differential privacy (LDP). Centralized-DP is the classical definition of differential privacy and some studies refer it as global differential privacy in the literature [59, 89, 90]. Throughout this section, we call it centralized-DP (CDP). To provide the CDP, we use the novel connection between differential privacy and sampling from the posterior distribution of the parameters of a statistical model via SGLD [28]. We propose two different CMF models that provide LDP [57, 60, 61], which is a stronger form of privacy compared to the CDP. We use Gibbs sampling [114, 115] for inference in the locally differentially private models.

4.1.1. Problem Definition and CMF Notation

We first consider the following simple CMF model and, in the next section, we construct various CTF models based on a more complicated data sharing scenarios. For the models presented in this section, we consider that the target dataset is represented as an $I \times J$ matrix $\mathbf{X}$ where the rows of $\mathbf{X}$ represent one type of entity (generally user) and the columns of $\mathbf{X}$ represent the other type of entity (generally item). In this notation, the element $x_{i,j}$ expresses the value of the relation between entities i and j for $i = 1, \cdots, I$ and $j = 1, \ldots, J$.

We assume the data matrix $\mathbf{X}$ is partitioned into N submatrices, these submatrices are distributed among N different data sites and each site n for $n = 1, \ldots, N$ has access to the submatrix $X^{(n)}$. Each data subset $X^{(n)} \in \mathbb{R}^{I_n \times J_n}$ (with cardinalities I_n and J_n for $n = 1, \ldots, N$) is approximated as: $X^{(n)} \approx U^{(n)} V^{(n)}$. Here, $U^{(n)} \in \mathbb{R}^{I_n \times D}$ and $V^{(n)} \in \mathbb{R}^{D \times J_n}$ are the rank-D matrices with vectors $U^{(n)}_{i_n}$ (that represents user-specific latent feature vector) and $V^{(n)}_{j_n}$ (that represents the item-specific latent feature vector) where $i_n = 1, \ldots, I_n$ and $j_n = 1, \ldots, J_n$. Our objective is to predict the value of some specific entry $x^{(n)}_{i_n, j_n} = \sum_{d=1}^{D} U^{(n)}_{i_n, d} V^{(n)}_{d, j_n}$ of the data site n.

In this section, our main goal is to preserve the users' privacy; so we design a setting

that the data owners do not inform the other sites about their users' identities. Eventually, each user is accepted unique even if it appears in different data sites while the items can be common for different sites. Therefore, we use the *row-distributed* scenario shown in Figure 4.1(a) where the dataset is partitioned based on the users' data and the item set is common for all the sites.

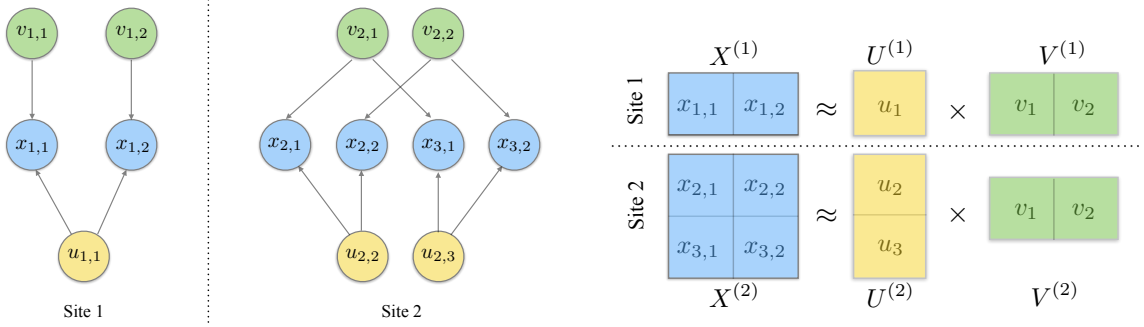


Figure 4.2. Graphical model and matrix factorization representation of individual learning without data sharing.

The default situation is no data sharing (that is shown in Figure 4.2). Here, suppose the users' ratings on an item set are placed in two distinct data sites where the first data site (site 1) has access to $X^{(1)}$ and the second data site (site 2) has access to $X^{(2)}$. In this case, each site keeps its own user factor and item factor in its local; e.g. data site 1 keeps $U^{(1)}$, $V^{(1)}$ and data site 2 keeps $U^{(2)}$, $V^{(2)}$. Then, the sites decompose their data as follows:

$$X^{(1)} \approx \hat{X}^{(1)} = U^{(1)}\, V^{(1)} \qquad\qquad X^{(2)} \approx \hat{X}^{(2)} = U^{(2)}\, V^{(2)}$$

On the other hand, protecting privacy permits a pooled analysis. We present a coupled matrix factorization model that transfers information across the data sites over a shared latent factor. In our design shown in Figure 4.3, $X^{(1)}$ and $X^{(2)}$ are collectively decomposed as $X^{(1)} \approx U^{(1)}V$ and $X^{(2)} \approx U^{(2)}V$. Here, $U^{(1)}$ and $U^{(2)}$ are the user profile factors that are *private* to the data sites $X^{(1)}$ and $X^{(2)}$ respectively. V is a *shared* item profile factor that is common between the two data sites, so we do not put a superscript to describe the data site number. Our main objective is to design a privacy preserving scheme that only privately

releases the shared factor(s) to public and allows the individual data sites to infer the private factor(s) locally. Accordingly, once the final item profile matrix V is learned and published differentially privately, a data site n infers the user profile matrix $U^{(n)}$. Afterwards, the data site n can further predict its ratings on all the items privately by using $U^{(n)}$ and V.

In the privacy-preserving model, the data sites are allowed to inform each other about the items they have; thus V contains all the items in the model. As a result of this, each data site owns a subset of the whole item set and only updates the corresponding subset of V. (In our experiments, we assume that all the data sites have the complete item set.) On the other hand, the sites are uninformed about the users' identities that each other have, and some of the users may possibly be common for both data sites. (But, the data sites do not know that they have the same users' data.) These users are represented by the corresponding rows of $U^{(1)}$ in the local of site 1 and with the corresponding rows of $U^{(2)}$ in the local of site 2. ($U^{(1)}$ and $U^{(2)}$ are updated privately.)

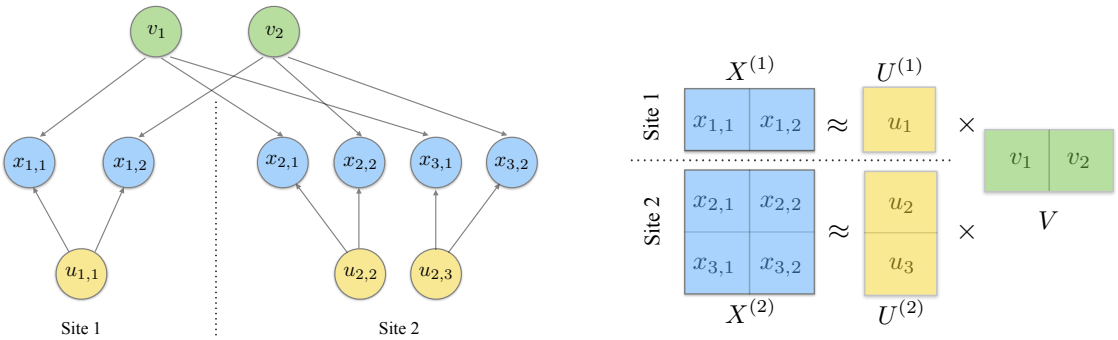


Figure 4.3. *CDP model*. Graphical model and matrix factorization representation of private learning by sharing a latent common factor. The common factor V is learned collectively by $X^{(1)}$ and $X^{(2)}$.

4.1.2. Probabilistic Model

We instantiate the general model by specifying the conditional distribution of the observed data $X^{(1)} \in \mathbb{R}^{I_1 \times J}$ (the likelihood term) and the prior distribution over $U^{(1)} \in \mathbb{R}^{I_1 \times D}$

and $V \in \mathbb{R}^{D\times J}$ as:

$$p(X^{(1)} \mid U^{(1)}, V) = \prod_{i_1=1}^{I_1} \prod_{j=1}^{J} \mathcal{PO}(x_{i_1,j}^{(1)} \mid U_{i_1}^{(1)} V_j) \tag{4.1}$$

$$p(U^{(1)} \mid \theta_{U^{(1)}}) = \prod_{i_1=1}^{I_1} \mathcal{G}(U_{i_1}^{(1)} \mid \alpha_{U^{(1)}}, \beta_{U^{(1)}}) \tag{4.2}$$

$$p(V \mid \theta_V) = \prod_{j=1}^{J} \mathcal{G}(V_j \mid \alpha_V, \beta_V) \tag{4.3}$$

where $\mathcal{PO}$ denotes the Poisson distribution and $\mathcal{G}(\cdot \mid \alpha, \beta)$ denotes the Gamma distribution with hyperparameters α and β which reflect our prior knowledge about the specific problem. We assume that $\theta_{U^{(1)}} = (\alpha_{U^{(1)}}, \beta_{U^{(1)}})$ and $\theta_V = (\alpha_V, \beta_V)$ can be treated as constant during training. In this model, we chose a Gamma distribution as prior for computational convenience where Gamma is the conjugate prior of the Poisson distribution.

The likelihood term for $X^{(2)} \in \mathbb{R}^{I_2\times J}$, the priors over $U^{(2)} \in \mathbb{R}^{I_2\times D}$ are defined similarly. Throughout this paper, we will use the same likelihood and the prior distributions. Finally, we define the posterior distributions $P^{(1)}$ and $P^{(2)}$ for data site 1 and data site 2 as:

$$P^{(1)} \propto p(U^{(1)}, V \mid X^{(1)}, \Theta^{(1)}) = p(X^{(1)} \mid U^{(1)}, V)\, p(U^{(1)} \mid \theta_{U^{(1)}})\, p(V \mid \theta_V) \tag{4.4}$$

$$P^{(2)} \propto p(U^{(2)}, V \mid X^{(2)}, \Theta^{(2)}) = p(X^{(2)} \mid U^{(2)}, V)\, p(U^{(2)} \mid \theta_{U^{(2)}})\, p(V \mid \theta_V) \tag{4.5}$$

where $\Theta^{(1)} = (\theta_{U^{(1)}}, \theta_V)$ and $\Theta^{(2)} = (\theta_{U^{(2)}}, \theta_V)$.

In the following, we propose several approaches to protect privacy in different scenarios involving information sharing. First, we develop an MCMC method for matrix factorization based on SGLD that protects CDP [28] for the model shown in Figure 4.3. Then, we propose an alternative approach for the models presented in Figure 4.4 and Figure 4.5 to guarantee differential privacy under the local privacy constraints.

4.1.3. Centralized Differential Privacy

In this section, we use the connection between SGLD and differential privacy and propose the specialized version of the SGLD algorithm of Li *et al* [28] in order to guarantee

(ϵ, δ)-DP for CMF models. Here, we recall the DP definition for the ease of convenience:

$$P(\mathcal{A}(X) \in R) \leq e^{\epsilon} P(\mathcal{A}(Y) \in R) + \delta \tag{4.6}$$

In our approach to CDP, the abstract randomized algorithm $\mathcal{A}$ that is defined in Equation (4.6) is chosen to be an SGLD sampler. When the SGLD sampler's parameters (that are used to determine the *stepsize* and the noise) are chosen appropriately and the gradient norm is clipped to satisfy Lipschitz condition, the SGLD sampler can produce multiple samples from a posterior distribution differentially privately with a fixed privacy budget. In [28], Li *et al* propose two theorems, Theorem 4.1 and Theorem 4.2 that restrict the selection of the stepsize parameter. These theorems imply that one can approximate the posterior mean using SGLD privately. We first recap these Theorems, then we give Theorem 4.3 that claims the latent factors U and V are updated in each iteration differentially privately.

Theorem 4.1. ***(Theorem 3 in [28])****: When the stepsize decreases at the rate of* $O(t^{-1/3})$*, there exist positive constants* c_1 *and* c_2 *such that given the sampling probability* $q = B/S$*, number of iterations* T*, gradient clipping norm* G *and* $\epsilon < c_1 q^2 T^{2/3}$*, then the SGLD algorithm updates the model parameter* θ *to enable* (ϵ, δ)*-differential privacy as long as stepsize* η_t *satisfies:*

(1) $\eta_t \leq \frac{S}{G^2}$,
(2) $\eta_t > \frac{q^2 S}{256 L^2}$,
(3) $\eta_t < \frac{\epsilon^2 S t^{-1/3}}{c_2^2 G^2 T^{2/3} \log(1/\delta)}$.

Theorem 4.2. ***(Theorem 4 in [28])****: When the stepsize is fixed to* $\eta_t = \eta$ *under the same setting as Theorem 4.1, the SGLD algorithm updates the model parameter* θ *to enable* (ϵ, δ)*-differential privacy as long as* $\eta < \frac{\epsilon^2 S}{c^2 G^2 T \log(1/\delta)}$ *for a constant* c*.*

Li *et al* theoretically showed that they improved the bound of η_t given by Wang *et al* [27] by a factor of $T^{(1/3)} \log(T/\delta)$ at the first iteration. They also empirically compared the stepsizes when S=50000, T=10000, G=1, ϵ =0.1 and δ=10^{-5}. For the same privacy loss, the stepsize is computed as $\eta_t < 0.103$ with Theorem 4.1 while Wang *et al* [27] gives

$\eta_t < 1.54 \times 10^{-6}$. It is obvious that the stepsize obtained by Wang *et al* is quite small to be used in practice, so the new method [28] is able to achieve higher accuracy for a certain amount of privacy.

To provide CDP, we adopt the theorems given above and make a minor modification to derive a novel differentially private CMF model. In our setting, there are two model parameters U and V. The loss function $\ell(X|U,V)$ corresponds to $-\sum_{i=1}^{I}\sum_{j=1}^{J}\log p(x_{i,j}|U_i,V_j)$; the regularizers $r(U)$ and $r(V)$ correspond to $-\sum_{i=1}^{I}\log p(U_i)$ and $-\sum_{j=1}^{J}\log p(V_j)$ respectively under the ERM. To ensure DP, the loss function is required to be Lipschitz smooth with constant L [27,91,116]. If a function f is Lipschitz smooth with constant L, then its derivatives (in our case the computed gradients of the loss function) are Lipschitz continuous with constant L. Formally, L-Lipschitz condition is satisfied [117,118] for a function $f:\mathbb{R}\rightarrow\mathbb{R}$ for any x and y if:

$$|f'(x)| \leq L \qquad \text{and}$$
$$\parallel \nabla f(x) - \nabla f(y) \parallel \leq L \parallel x - y \parallel$$

In Theorem 4.3, we claim that U and V are updated in each iteration differentially privately, and we prove this theorem in Appendix B.

Theorem 4.3. *Suppose the initial components $U^{(0)}$, $V^{(0)}$ are chosen independently from the data and the loss function $\ell(X|U,V)$ is L-smooth with respect to $\|\cdot\|_2$ for any $X\in\mathbb{R}^{I\times J}$, $U\in\mathbb{R}^{I\times D}$ and $V\in\mathbb{R}^{D\times J}$ where the data size $S=|I|\times|J|$. Let q, T, G, c, c_1, c_2 and ϵ be chosen such that $\epsilon < c_1 q^2 T^{2/3}$. Then, Algorithm 2 updates the model parameters U and V to enable (ϵ,δ)-differential privacy as long as stepsize η_t satisfies the conditions in Theorem 4.1 when the stepsize decreases at the rate of $O(t^{-1/3})$ or in Theorem 4.2 when the stepsize is fixed.*

To ensure the L-Lipschitz condition, we apply gradient norm clipping that is commonly used to normalize the gradients of a parameter vector when its L_2 norm exceeds a certain threshold. When we use a constant value G to clip the gradients, the sensitivity of $\sum_{(i,j)\in\mathcal{B}^{(t)}}\nabla\log p(x_{i,j}|U_i^{(t-1)},V_j^{(t-1)})$ becomes at most $2G$.

For the proposed matrix factorization models, we will apply the SGLD [29] to derive the update rules iteratively for the two kinds of factors. We obtain the samples of the *private* factors $U^{(n,t)}$s for $n = \{1, \cdots, N\}$ and the *common* factor $V^{(t)}$ as follows:

$$U^{(n,t)} = U^{(n,t-1)} + \Delta U^{(n,t)} \qquad V^{(t)} = V^{(t-1)} + \sum_{n=1}^{N} \Delta V^{(n,t)}$$

where n and t denote the site number and the iteration number respectively. The site-specific *private* factors $U^{(n,t)}$s are computed by using each data site's own data $X^{(n)}$, and the *shared* factor $V^{(t)}$ is computed by all sites that share this common factor. In the following, we show the detailed computations of the factors.

First, we apply the following update rule in order to obtain the samples of the *shared* factor. It must be noted that only the data sites that share a common factor V need to compute the gradients; then all these local gradients are summed to update V. In our specific model, the common latent structure V participates in all the data partitions for $n = \{1, \cdots, N\}$. We first compute the gradients $\Delta V^{(n,t)}$ for each n, then clip the gradient norm to ensure that the computed gradients satisfy the Lipschitz condition in Theorem 4.3 and finally update V as follows:

$$\Delta V^{(n,t)} = \frac{S^{(n)}}{B} \sum_{(i_n,j)\in\mathcal{B}^{(n,t)}} \nabla_{V_j} \log p(x^{(n)}_{i_n,j} | U^{(n,t-1)}_{i_n}, V^{(t-1)}_j) + \nabla_{V_j} \log p(V^{(t-1)}) \Big) \tag{4.7}$$

$$\Delta \tilde{V}^{(n,t)} = \Delta V^{(n,t)} / \max\left(1, \frac{\| \Delta V^{(n,t)} \|_2}{G}\right) \tag{4.8}$$

$$V^{(t)}_j = V^{(t-1)}_j + \sum_{n=1}^{N} \left(\eta_t \Delta \tilde{V}^{(n,t)} + \xi^{(n,t)}\right) \tag{4.9}$$

where ∇ denotes the gradients, $S^{(n)}$ is the number of elements in $X^{(n)}$, i_n and j are the row and column indices of $X^{(n)}$ respectively. When using a minibatch of size B, $\mathcal{B}^{(n,t)} \subset I_n \times J$ is the set of entries that are subsampled from $X^{(n)}$ at iteration t. A privacy-preserving CMF scheme should guarantee that the execution of the algorithm exposes only the shared factor by providing (ϵ, δ)-DP. The privacy is protected by adjusting the stepsize η_t and the noise matrix $\xi^{(n,t)}$ that is of the same size as V and its entries are independently Gaussian distributed. The Gaussian noise is added into gradients of parameter V at every update step. We give the

details how we adjust these parameters in Appendix B.

Note that the index j that corresponds to the common mode should be the same for all relations during the computation of the *shared* factor. In this method, each data site stores its private factors in its local memory. For the *shared* factors, there are two options that are equivalent from a privacy perspective: *i)* each data site keeps copies of the *shared* factors and updates them after collecting the gradients from the *other* data sites or *ii)* an independent data site keeps the *shared* factors and updates them after collecting the gradients from *all* the data sites. We use the second option in our experiments.

Next, we derive the update rules in order to obtain the samples of the *private* factor $U^{(n,t)}$ that is specific for the data site n. The update rule differs for *site-level* and *user-level* DP. To protect *site-level* DP, the algorithm publish the shared factor V to satisfy (ϵ, δ)-DP while each private factor $U^{(n,t)}$ is kept in the local of a data site and is updated without privacy constraints. The update rule that is used to protect site-level DP is:

$$\Delta U^{(n,t)} = \kappa_t \Big(\frac{S^{(n)}}{B} \sum_{(i_n,j)\in\mathcal{B}^{(n,t)}} \nabla_{U^{(n)}_{i_n}} \log p(x^{(n)}_{i_n,j} | U^{(n,t-1)}_{i_n}, V^{(t-1)}_{j}) + \nabla_{U^{(n)}_{i_n}} \log p(U^{(n,t-1)}_{i_n}) \Big) + \psi^{(t)} \tag{4.10}$$

$$U^{(n,t)}_{i_n} = U^{(n,t-1)}_{i_n} + \Delta U^{(n,t)} \tag{4.11}$$

Here, the noise matrix $\psi^{(t)}$ is of the same size as $U^{(n,t)}$ and its entries are independently Gaussian distributed: $\psi^{(t)}_{i_n,d} \sim \mathcal{N}(0, 2\kappa_t)$. For convergence, the step size κ_t must satisfy: $\sum_{t=0}^{\infty} \kappa_t = \infty$ and $\sum_{t=0}^{\infty} (\kappa_t)^2 < \infty$.

To protect *user-level* privacy, in addition to the shared factors, all the private factors $U^{(n,t)}$ should satisfy (ϵ, δ)-DP. This means no user can learn private ratings of others even in

Algorithm 1 Differentially Private SGLD for Coupled Matrix Factorization

Inputs: Number of parties N, number of iterations T, minibatch size B, privacy parameters ϵ and δ, gradient clipping norm G, data matrices: $\{X^{(n)}\}_{n=1}^{N}$ and initial factors: $\{U^{(n,0)}\}_{n=1}^{N}, V^{(0)}$

Outputs: $\{U^{(1,t)}, \cdots, U^{(N,t)}\}_{t=1}^{T}, \{V^{(t)}\}_{t=1}^{T}$

for $t \leftarrow 1$ to T **do**

  for Each data site $n = 1 \dots N$ **do**

    Set the step size κ_t and η_t

    Sample each coord. of $\psi^{(t)}$ i.i.d from $\mathcal{N}(0, 2\kappa_t)$ and $\xi^{(n,t)}$ i.i.d from $\mathcal{N}(0, \sigma_t^2)$ where $\sigma_t^2 = \frac{c_2^2 G^2 T^{2/3} t^{1/3} \log(1/\delta)}{\epsilon^2 S^2} \eta_t^2 \mathbb{I} \vee \sigma_t^2 = \frac{\eta_0^2 G^2 \eta_t^2}{q^2} \mathbb{I}$

    Sample a minibatch $\mathcal{B}^{(n,t)}$ from $X^{(n)}$ of size B

    for each i_n and j in $\mathcal{B}^{(n,t)}$ **do**

      if site-level privacy **then**

        Update *private* factors $U^{(n,t)}$ using Eqn. (4.11)

      else if user-level privacy **then**

        Compute (4.12) and clip gradients (4.13) for *private* factors $U^{(n,t)}$

        Update *private* factors $U^{(n,t)}$ using Eqn. (4.14)

      end if

      Compute gradients for *shared* factors using Eqn. (4.7)

      Clip gradients for *shared* factors using Eqn. (4.8)

    end for

  end for

  Update *shared* factor $V^{(t)}$ using Eqn. (4.9)

  Return $U^{(n,t)}$ and $V^{(t)}$ as a posterior sample

end for

the same data site. The following update rule is applied for this purpose.

$$\Delta U^{(n,t)} = \frac{S^{(n)}}{B} \sum_{(i_n,j)\in\mathcal{B}^{(n,t)}} \nabla_{U_{i_n}^{(n)}} \log p(x_{i_n,j}^{(n)} | U_{i_n}^{(n,t-1)}, V_j^{(t-1)}) + \nabla_{U_{i_n}^{(n)}} \log p(U_{i_n}^{(n,t-1)}) \Big) \tag{4.12}$$

$$\Delta \tilde{U}^{(n,t)} = \Delta U^{(n,t)} / \max\left(1, \frac{\| \Delta U^{(n,t)} \|_2}{G}\right) \tag{4.13}$$

$$U_{i_n}{}^{(n,t)} = U_{i_n}{}^{n,t-1} + \left(\eta_t \Delta \tilde{U}^{(n,t)} + \xi^{(n,t)}\right) \tag{4.14}$$

Here, the stepsize η_t and the noise matrix $\xi^{(n,t)}$ is chosen in the same way in the update rule for shared factors which is given in Equation (4.9). Differentially private CMF method that ensures CDP is fully described in Algorithm 1.

4.1.4. Local Differential Privacy

Local Differential Privacy (LDP) definition naturally fits our scenario where N distinct parties hold sensitive data from several individuals, and they wish to jointly estimate parameters of a statistical model conditioned on all data. Formally, there are N data sites each owning a dataset $X^{(n)} \in \mathcal{X}$ where $\mathcal{X} = \{X^{(n)}\}_{n=1}^{N}$ and each $X^{(n)}$ is independently sampled from some distribution P. A randomized response method $\mathcal{R}$ maps the original random variables $X^{(n)} \in \mathcal{X}$ to the privatized (perturbed) observations $Z^{(n)} \in \mathcal{Z}$ where $\mathcal{Z} = \{Z^{(n)}\}_{n=1}^{N}$ and this process is called *privatization*. In this way, the algorithm only sees the perturbed observations and the data remains private for even the algorithm. Our aim is to develop a coupled matrix factorization (CMF) framework for data sharing under LDP constraints. Most recently, Schein *et al* [57] introduced Bayesian inference under LDP. We use the privatization mechanism introduced in [57] to prove that our framework is locally differentially private.

In our local privacy setting, data matrices $\{X^{(n)}\}_{n=1}^{N}$ are independently sampled from a Poisson distribution where each instance $x_{i,j}$ in this data set is an independent Poisson random variable. Here, we target the Poisson matrix factorization [109], where X holds the count data and each element of this matrix $x_{i,j}$ is drawn as follows:

$$\lambda_{i,j} = U_i V_j \qquad x_{i,j} \sim \mathcal{PO}(\lambda_{i,j})$$

At this point, we have to choose a privatizing noise model to generate the perturbed observation matrices $Z^{(n)}$'s from $X^{(n)}$'s. In privacy literature, the standard noise adding mechanisms are Laplace and Gaussian which generate noise from real-valued distributions. However, they are not natural choices for count values. In [57] they showed that adding noise from a two-sided Geometric distribution to each of the observations guarantees ϵ-DP [119]. A two-sided geometric random variable $\tau \sim \mathcal{GE}(\epsilon)$ is an integer where $\tau \in \mathbb{Z}$. The probability mass function of the two-sided geometric distribution is: $2\mathcal{GE}(\tau, \epsilon) = \frac{1-\epsilon}{1+\epsilon}\epsilon^{|\tau|}$. Then, each

observation $x_{i,j}$ is privatized as follows:

$$\tau_{i,j} \sim 2\mathcal{GE}(\epsilon) \quad , \qquad z_{i,j} = x_{i,j} + \tau_{i,j} \tag{4.15}$$

In this section we define an MCMC algorithm that samples differentially privately from the distribution P defined in the following based on access to obscured views $Z^{(n)}$s of the original data $X^{(n)}$. We apply the privatization method in [57] to our coupled matrix factorization models and propose two locally differentially private CMF models: LDP1 and LDP2.

Local DP Model 1 **(LDP1)** . In this model, information is shared among a common factor V between the data sites. To protect ϵ-DP, each site with data $X^{(n)}$ sends a perturbed version of the data $Z^{(n)}$, via the ϵ-locally differentially privatization mechanism $\mathcal{R}$. Given the perturbed views $\{Z^{(n)}\}_{n=1}^{N}$, all the data sites make inferences based on the induced posterior distributions. According to this model definition, a data site has to use the privatized view of its own data to infer the 'site specific' private components ($U^{(n)}$ is inferred by using $Z^{(n)}$ for $n = 1, \cdots, N$). The common component V that is shared between all of the data sites is inferred by using the privatized views from all of the data sites, i.e., V is inferred by using $\{Z^{(n)}\}_{n=1}^{N}$.

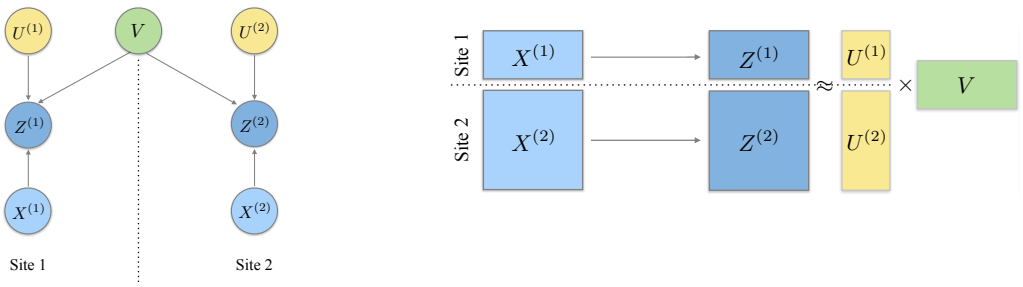


Figure 4.4. *LDP1 model*. Graphical model and matrix factorization representation of locally differentially private learning by sharing a common factor. The common factor V is learned collectively by the privatized data $Z^{(1)}$ and $Z^{(2)}$.

Figure 4.4 provides an example model for two data sites. Here, site 1 privatizes $X^{(1)}$ with the method defined in Equation (4.15) and site 2 privatizes $X^{(2)}$ likewise. In

this case, each data site learns $U^{(1)}$ and $U^{(2)}$ by decomposing the perturbed data $Z^{(1)}$ and $Z^{(2)}$ respectively. However, the common component V is shared by the both sites and it is learned by simultaneous decomposition of $Z^{(1)}$ and $Z^{(2)}$. This gives us the following matrix factorization model:

$$Z^{(1)} \approx U^{(1)}\, V \qquad\qquad Z^{(2)} \approx U^{(2)}\, V$$

and the posterior distributions for the given model is written as follows:

$$P^{(1)} \propto\ p(U^{(1)}, V \mid X^{(1)}, \Theta^{(1)}) = p(Z^{(1)} \mid U^{(1)}, V)\ p(U^{(1)} \mid \theta_{U^{(1)}})\ p(V \mid \theta_V)$$
$$P^{(2)} \propto\ p(U^{(2)}, V \mid X^{(2)}, \Theta^{(2)}) = p(Z^{(2)} \mid U^{(2)}, V)\ p(U^{(2)} \mid \theta_{U^{(2)}})\ p(V \mid \theta_V)$$

where $\Theta^{(1)} = (\theta_{U^{(1)}}, \theta_V)$ and $\Theta^{(2)} = (\theta_{U^{(2)}}, \theta_V)$.

Local DP Model 2 **(LDP2)** . The prediction accuracy depends on the amount of information that each data site is willing to share with the system. The first local-DP model, *LDP1*, directly follows the definition of local differential privacy where each data site first privatizes its data; then shares the privatized view with the system. Hence, this model protects both *user-level* and *site-level* privacy for each data site.

Here, we design a more flexible, hybrid model *LDP2*, which still preserves *site-level* ϵ-LDP for each data site n, however it cannot protect the *user-level* privacy in data site n. This model is useful when the local data is fully accessible to its owner and sharing of preferences between the local users without privacy protection is acceptable. Contrary to *LDP1*, each data site n factorizes both the original data ($X^{(n)}$) in its own site without privacy protection and the privatized data views observed from the other sites ($\{Z^{(n')}\}_{n'=1}^{N}$ for $n' \neq n$) simultaneously. We will demonstrate the gain from access to a non-private view of the self-data in Section 7.1.

An illustrative example is shown in Figure 4.5 for a special case with two data sites. In this model, site 1 jointly decomposes its own data $X^{(1)}$ and the privatized data $Z^{(2)}$ to infer the *private* component $U^{(1)}$ and the *shared* component $V^{(1)}$. Here, $Z^{(2)}$ is used as the

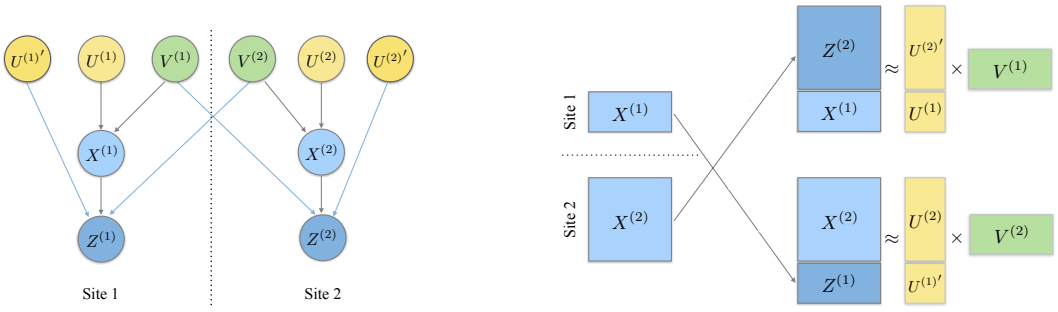


Figure 4.5. *LDP2 model*. Graphical model and matrix factorization representation of locally differentially private learning where each of the data sites factorizes collectively both the original data $X^{(1)}$ and $X^{(2)}$ and the privatized data views $Z^{(1)}$ and $Z^{(2)}$ that are observed from the other sites.

auxiliary data, so $U^{(2)\prime}$ is an auxiliary component that is not used to reconstruct $X^{(1)}$. (Site 2 learns $U^{(2)}$ and $V^{(2)}$ in the same manner.) Finally, we obtain the following model:

$$X^{(1)} \approx U^{(1)}\, V^{(1)} \qquad X^{(2)} \approx U^{(2)}\, V^{(2)}$$
$$Z^{(1)} \approx U^{(1)\prime}\, V^{(2)} \qquad Z^{(2)} \approx U^{(2)\prime}\, V^{(1)}$$

and the posterior distributions for site 1 and site 2 are defined as follows:

$$P^{(1)} \propto\ p(U^{(1)}, V^{(1)}, U^{(2)\prime} \mid X^{(1)}, Z^{(2)}, \Theta^{(1)})$$

$$P^{(2)} \propto\ p(U^{(2)}, V^{(2)}, U^{(1)\prime} \mid X^{(2)}, Z^{(1)}, \Theta^{(2)})$$

where $\Theta^{(1)} = (\theta_{U^{(1)}}, \theta_{U^{(2)\prime}}, \theta_{V^{(1)}})$ and $\Theta^{(2)} = (\theta_{U^{(2)}}, \theta_{U^{(1)\prime}}, \theta_{V^{(2)}})$.

One particularly effective model is Gibbs sampling for Bayesian matrix factorization models where predictions are averaged over samples from the posterior distribution [106]. In the general case of Gibbs sampling, a sequence of samples of U and V are drawn from their conditional posterior densities of the model parameters as given in (4.16) and (4.17) where $t = 1, \cdots, T$ denotes the iteration number. This procedure is the same as in non-private Poisson factorization and it converges to a sample from the joint posterior.

$$U^{(t+1)} \sim p(U \mid X, V^{(t)}, \theta_U) \qquad (4.16) \qquad V^{(t+1)} \sim p(V \mid X, U^{(t)}, \theta_V) \qquad (4.17)$$

For *LDP1* and *LDP2*, we first construct the privatized dataset Z from the original observation X. In this step, we assume that we have the privacy mechanism that generates two-sided geometric noise as given in Equation 4.15 so our MCMC algorithm guarantees ϵ -LDP. (In the Appendix D, we will explain two other privatization process.) Then, we generate samples for the model components that are defined above by using Gibbs sampling. The proposed model LDP1 preserves ϵ-LDP for each user and LDP2 preserves ϵ- LDP data site in the system.

Lastly, we summarize our methods presented in this section () in Table 4.1 regarding the privacy definitions they consider.

Table 4.1. Summary of the proposed DP methods in terms of the privacy contexts.

Methods	Privacy Definitions			
	Centralized DP	Local DP	User-level DP	Site-level DP
CDP (site)	✓			✓
CDP (user)	✓		✓	✓
LDP1		✓	✓	✓
LDP2		✓		✓

4.2. Coupled Tensor Factorization

This section considers a coupled tensor factorization setting, including two types of factors: *private* and *shared*. If an entity type e_d is specific to relation v, it implies that the corresponding factor Θ_d impacts only the relation $X^{(v)}$ and $\Theta_d^{(v)}$ is described as a *private*

factor for relation v. Besides, if more than two relations share an entity type e_d, it means that the factor Θ_d is *shared* factor for a group of relations rather than a single relation and the standard CTF can be obtained to model these multiple relations. Our main objective is to design a privacy-preserving tensor factorization scheme that only privately releases the shared factor(s) to public and allows the individual data sites to infer the private factor(s) locally. The private factors are inferred by two methods: *i)* to protect *site-level* privacy and *ii)* to protect *user-level* privacy. After that, each data site can predict its individual data entities by computing the product of both private and shared factors.

Let us consider the situation shown in Figure 4.2 where the users' ratings on an item set are located in N different data sites. Here, Θ_2 is a common item profile factor that is shared between all data sites (so we do not put a superscript to describe the relation); and the rest $\{\Theta_1^{(v)}\}_{v=1}^{N}$ are the user profile factors that are private to each data site $X^{(v)}$.

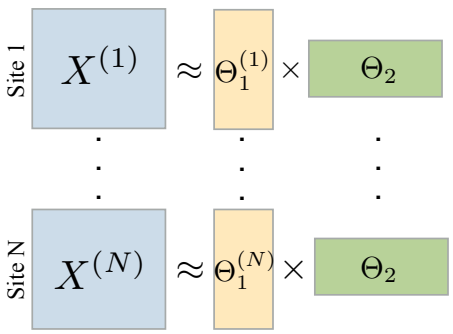
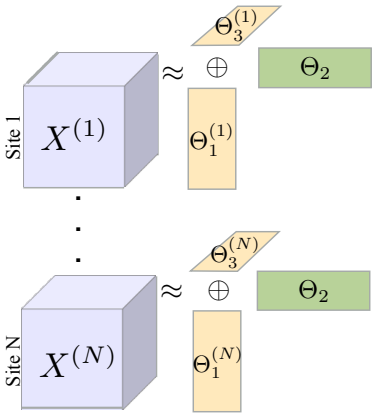


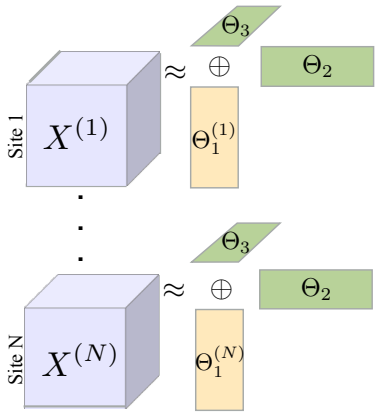
Figure 4.6. Example of matrix factorization setup: $X^{(v)} \approx \Theta_1^{(v)}\Theta_2^T$ for $v = 1, \ldots, N$; mode e_2 is shared between matrices.

We first consider the protection of *site-level* privacy, where the local data is accessible to its owner site. In this scenario, the user's data site is considered to be trusted and never disclose users' privacy. The final item profile matrix Θ_2 is published to satisfy *site-level* ϵ-DP, while each user profile matrix $\Theta_1^{(v)}$ is kept in the local of a data site. To protect *user-level* privacy, besides that the user profile vectors are kept private on the data site side, they should be protected against the other users in the same data site. So long as these item profile vectors satisfying ϵ-DP, the user profile vector derived from them must satisfy ϵ-DP as well, which means no user can learn private ratings of others in the same data site. In both settings, once Θ_2 is learned and published privately, a data site v can infer its user profile matrix $\Theta_1^{(v)}$ by

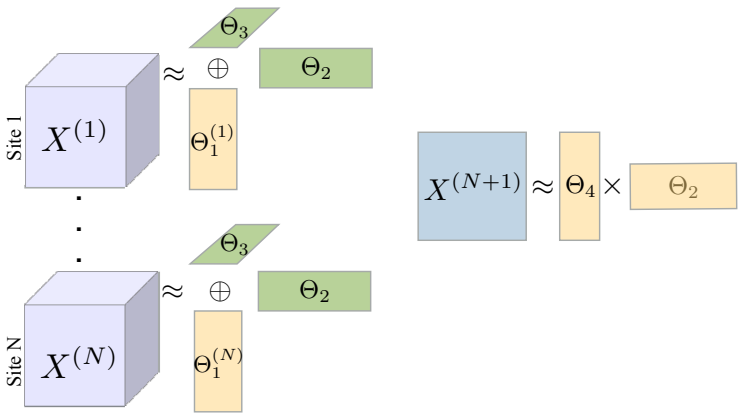
using either method, then with both $\Theta_1^{(v)}$ and Θ_2, it can further predict its ratings on all the items locally.



(a) Model-1



(b) Model-2



(c) Model-3

Figure 4.7. Examples of CTF setups: (a) shares one mode e_2, (b) shares two modes e_2 and e_3 and (c) shares two modes e_2, e_3 between tensors $\{X^{(v)}\}_{v=1}^{N}$ and shares e_2 with auxiliary item-feature matrix $X^{(N+1)}$.

4.2.1. Probabilistic Model

First, we describe the probabilistic model to jointly factorize the multiple relations represented by tensors. In order to be able to compare our methods with the state-of-the-art methods by [1, 2], we consider the following probabilistic model [106] where the conditional distribution of the observed dataset $X^{(v)}$ (the likelihood term) and the prior distributions over

Θ_d s are given as:

$$\begin{aligned} p(X^{(v)} \mid \{\Theta_d\}_{d \in S_v}) &= \mathcal{N}(X^{(v)} \mid \{\Theta_d\}_{d \in S_v}, \tau_v^{-1}) \\ p(\Theta_d \mid \mu_{\Theta_d}, \Lambda_{\Theta_d}) &= \mathcal{N}(\Theta_d \mid \mu_{\Theta_d}, \Lambda_{\Theta_d}^{-1}) \end{aligned}$$

Here, $\mathcal{N}$ denotes the Gaussian distribution with mean μ_{Θ_d} and precision Λ_{Θ_d} where Λ_{Θ_d} is a K-dimensional diagonal matrix whose k-th diagonal element is $\lambda_{\Theta_d k}$. We choose the following hyper-prior: $\lambda_{\Theta_d k} \sim \mathcal{G}(a_0, b_0)$ where $\mathcal{G}(a_0, b_0)$ denotes the Gamma distribution with hyperparameters a_0 and b_0 which reflect our prior knowledge about the specific problem. We assume that τ_v, a_0 and b_0 can be treated as constant during training. Then, we define the posterior distributions for each data site v as P_v:

$$P_v \propto p(\{\Theta_d\}_{d \in S_v} \mid X^{(v)}, \{\tau_d\}_{d \in S_v}) = p(X^{(v)} \mid \{\Theta_d\}_{d \in S_v}) \prod_{d \in S_v} p(\Theta_d \mid \tau_{\Theta_d})$$

Even more general setups with arbitrary collections of tensors and matrices that share some sets of entities can be modeled easily with proposed method. Some prototypical examples of CTF are illustrated in Figure 4.7. In these setups, the problem of interest is tag prediction, that is, to predict tags that users will assign to items. Here, the target tensors $\{X^{(v)}\}_{v=1}^{N}$ are constructed by the relation $S_v = (e_1, e_2, e_3)$ that indicates that user e_1 tags item e_2 with tag e_3. Additionally, auxiliary information is provided by matrix associating the items' features and it is stored in a public data site where all other data sites can reach the data without privacy restrictions.

In the first scenario in Figure 4.7(a), the common item factor Θ_2 is shared between the parties while the user and tag factors $\Theta_1^{(v)}$ and $\Theta_3^{(v)}$ are *private* to each data site v for $v = \{1, \ldots, N\}$. In our method, it is possible to share more than one factor between the parties in order to increase the prediction quality by sharing more information privately. Figure 4.7(b) illustrates the model where the parties share both item and tag factors Θ_2 and Θ_3 while user profile factors $\Theta_1^{(v)}$ are inferred locally by each party v. Figure 4.7(c) shows the last scenario: we integrate a public data site that has the item-feature matrix $X^{(N+1)}$. The relation $S_{N+1} = (e_2, e_4)$ that characterizes items with a 1024-dimension feature vector

constructs $X^{(N+1)}$, and the target tensors are complemented with $X^{(N+1)}$.

4.2.2. Differential Privacy for Coupled Tensor Factorization

Here, we use the connection between the Bayesian posterior sampling and differential privacy which has been studied in [27]. Wang *et al* have proved that SGLD [29] is (ϵ, δ)-differentially private with no algorithmic changes due to the use of Gaussian noise in the updates. We begin with an overview of the specialized version of the SGLD algorithm of Wang *et al* that guarantees (ϵ, δ)-DP by using the Advanced Composition theorem. Then, we propose our theorem that guarantees differential privacy for CTF models and we propose an improved version of DP-SGLD by using the zCDP. In Section 7.2, we compare the performances of CTF models with these two composition methods.

As in the previous section, we choose the abstract randomized algorithm $\mathcal{A}$ that is defined in Equation (2.5) as an SGLD sampler and we adjust the *stepsize* and *Gaussian noise* to produce multiple samples from a posterior distribution differentially privately. This time, we use the Theorem 4.4 [27] that restricts the selection of the SGLD parameters to provide (ϵ, δ)-DP. They call the specialization of the SGLD procedure where the parameters are chosen according to Theorem 4.4: *DP-minibatch SGLD*.

Theorem 4.4 (*DP-minibatch SGLD*)**.** *Assume the initial model parameter $\theta^{(0)}$ is chosen independent of the data, also assume loss function $\ell(X|\theta)$ is L-smooth with $\|\cdot\|_2$ for any $X \in \mathcal{X}$ where $\mathcal{X} \subseteq \mathbb{R}^n$ for some $n \geq 1$. Besides, ϵ, δ, B and T should be chosen to provide $T \geq \frac{\epsilon^2 C}{32B\log(2/\delta)}$ where C is the data size. Then, the update algorithm for θ provides (ϵ, δ)-differential privacy.*

This theorem implies that one can approximate the posterior mean using SGLD privately. θ can be the finite dimensional parameter of a single exponential family model or a collection of these in a graphical model. We adopt this theorem and make a small modification to derive a novel differentially private coupled tensor factorization model that protects each site's privacy

by guaranteeing to protect every individual's privacy.

In this setting, we have parameters $\{\Theta_d\}_{d=1}^D$. The loss function $\mathcal{L} = \sum_{v=1}^V \ell(X^{(v)}|\{\Theta_d\}_{d\in S_v})$ corresponds to $-\sum_{v=1}^V \sum_{i_v\in\Omega_v} \log p(X_{i_v}^{(v)}|\sum_{k=1}^K \prod_{d\in S_v} \theta_{i_d k}^{(d)})$ and the regularizer $r(\Theta_d)$ corresponds to $-\sum_{i_d=1}^{n_d} \log p(\theta_{i_d})$ under the empirical risk minimization. In *Theorem 4.5*, we claim that $\{\Theta_d\}_{d=1}^D$ are updated in each iteration differentially privately and we prove this theorem in Appendix C.

Theorem 4.5. *Suppose the initial components $\{\Theta_d\}_{d=1}^D$ are chosen independently from the data, and loss function $\ell(X^{(v)} \mid \{\Theta_d\}_{d\in S_v})$ is L-smooth in $\|\cdot\|_2$ for any $X^{(v)}$ and Θ_d. Besides, ϵ, δ, B, T should be chosen to provide $T \geq \frac{\epsilon^2 C_v}{32B\log(2/\delta)}$ where the data size is C_v, minibatch size is B and the number of iterations is T. Then we obtain (ϵ, δ)-differentially private Algorithm 2.*

For the proposed CTF models, we will apply SGLD [29] to derive the update rules iteratively for the two kinds of factors: *private* and *shared*. We obtain the samples of the private factors $\Theta_{d(t)}^{(v)}$:

$$\Theta_{d(t)}^{(v)} = \Theta_{d(t-1)}^{(v)} + \nabla\Theta_{d(t)}^{(v)}$$

and the shared factors $\Theta_{d(t)}$ as:

$$\Theta_{d(t)} = \Theta_{d(t-1)} + \sum_{v\in\mathcal{V}} \nabla\Theta_{d(t)}^{(v)}$$

where v and t denote the site number and iteration number respectively. The site-specific *private* factors $\Theta_{d(t)}^{(v)}$s are computed by using each data site's own data $X^{(v)}$. The *shared* factor $\Theta_{d(t)}$ is computed with data of the all sites that share this common factor where $\mathcal{V}$ is the set of the data sites that $\Theta_{d(t)}$ participates in. In the following section, we show the detailed computations of the factors.

Sampling factors using SGLD. First, we derive the SGLD update rule for *shared* factors. It must be noted that only data sites that share a common factor Θ_d need to compute the gradients; then all these local gradients are summed to update Θ_d. In our specific model, the common latent structure Θ_d participates in all data partitions, so $\mathcal{V} = \{1, \ldots, N\}$. We first compute the gradients $\Delta\Theta_{d_{(t)}}^{(v)}$ for each $v \in \mathcal{V}$, then clip the gradient norm to ensure that the computed gradients satisfy the Lipschitz condition in Theorem 4.5 and finally update Θ_d as follows:

$$\begin{aligned}
\Delta\Theta_{d_{(t)}}^{(v)} &= \tfrac{C_v}{B} \textstyle\sum_{i_v \in \mathcal{B}_{(t)}^{(v)}} \nabla_{\theta_{i_d k}^{(v)}} \log p(X_{i_v}^{(v)} \mid \{\theta_{i_d k}{}_{(t-1)}^{(v)}\}_{d \in S_v}) \\
&+ \quad \nabla_{\theta_{i_d k}^{(v)}} \log p(\theta_{i_d k}{}_{(t-1)}^{(v)}) \Big) \qquad (4.18) \\
\Delta\tilde{\Theta}_d^{(v)}{}_{(t)} &= \Delta\Theta_{d_{(t)}}^{(v)} / \max\left(1, \tfrac{\|\Delta\Theta_{d_{(t)}}^{(v)}\|_2}{G}\right) \qquad (4.19) \\
\theta_{i_d k_{(t)}} &= \theta_{i_d k_{(t-1)}} + \textstyle\sum_{v \in \mathcal{V}} \left(\eta_{(t)}^{(v)} \Delta\tilde{\Theta}_d^{(v)}{}_{(t)} + \xi_{(t)}^{(v)}\right) \qquad (4.20)
\end{aligned}$$

where C_v is the number of elements in $X^{(v)}$, i_v is the row indices of $X^{(v)}$, B is the minibatch size, ∇ denotes the gradients and G denotes the gradient norm bound. $\mathcal{B}_{(t)}^{(v)} \subset n_1 \times \ldots \times n_{|S_v|}$ is the set of entries that are subsampled from $X^{(v)}$ at iteration t. A privacy-preserving CTF scheme should guarantee that the execution of the algorithm exposes only the shared factor by providing (ϵ, δ)-DP. The privacy is protected by adjusting the noise matrix $\xi_{(t)}^{(v)}$ that is of the same size as Θ_d and its entries are independently Gaussian distributed. We give the details how we adjust the noise in the following section (Section 4.2.2).

Note that the indices (i_v's) correspond to the common modes should be same for all relations during the computation of the *shared* factor. In this method, each data site stores its private factor in its local memory. For the *shared* factors, there are two options that are equivalent from a privacy perspective: *i)* each data site keeps copies of the *shared* factors and updates them after collecting the gradients from the *other* data sites or *ii)* an independent data site keeps the *shared* factors and updates them after collecting the gradients from *all* the data sites. We use the second option in our experiments.

Next, we derive the update rules in order to obtain the samples of the *private* factor $\Theta_{d_{(t)}}^{(v)}$ specific for the data site v. The update rule differs for *site-level* and *user-level* DP. To

protect *site-level* DP, the algorithm publish the shared factor Θ_d to satisfy (ϵ, δ)-DP while each private factor $\Theta_{d_{(t)}^{(v)}}$ is kept in the local of a data site and is updated without privacy constraints. The update rule that is used to protect site-level DP is:

$$\begin{aligned} \Delta\Theta_{d_{(t)}^{(v)}} &= \kappa_t \Big(\tfrac{C_v}{B} \textstyle\sum_{i_v \in \mathcal{B}_{(t)}^{(v)}} \nabla_{\theta_{i_d k}^{(v)}} \log p(X_{i_v}^{(v)} \mid \{\theta_{i_d k_{(t-1)}^{(v)}}\}_{d \in S_v}) \\ &+ \nabla_{\theta_{i_d k}^{(v)}} \log p(\theta_{i_d k_{(t-1)}^{(v)}}) \Big) + \psi_{(t)}^{(v)} \end{aligned} \tag{4.21}$$

$$\theta_{i_d k_{(t)}^{(v)}} = \theta_{i_d k_{(t-1)}^{(v)}} + \Delta\Theta_{d_{(t)}^{(v)}} \tag{4.22}$$

The elements of the noise matrix $\psi_{(t)}^{(v)}$ are independently Gaussian distributed: $\psi_{(t)}^{(v)} \sim \mathcal{N}(0, 2\kappa_t \mathbb{I})$. For convergence, the step size κ_t must satisfy: $\sum_{t=0}^{\infty} \kappa_t = \infty$ and $\sum_{t=0}^{\infty} \kappa_t^2 < \infty$. If entity type e_d is specific to relation v, $\Theta_d^{(v)}$s are computed by Equation (4.22).

To protect *user-level* privacy, in addition to the shared factors, all the private factors $\Theta_{d_{(t)}^{(v)}}$ should satisfy (ϵ, δ)-DP. This means no user can learn private ratings of others even in the same data site. The following update rule is applied for this purpose.

$$\begin{aligned} \Delta\Theta_{d_{(t)}^{(v)}} &= \tfrac{C_v}{B} \textstyle\sum_{i_v \in \mathcal{B}_{(t)}^{(v)}} \nabla_{\theta_{i_d k}^{(v)}} \log p(X_{i_v}^{(v)} \mid \{\theta_{i_d k_{(t-1)}^{(v)}}\}_{d \in S_v}) \\ &+ \nabla_{\theta_{i_d k}^{(v)}} \log p(\theta_{i_d k_{(t-1)}^{(v)}}) \end{aligned} \tag{4.23}$$

$$\Delta\tilde{\Theta}_{d}^{(v)}{}_{(t)} = \Delta\Theta_{d_{(t)}^{(v)}} / \max\left(1, \tfrac{\|\Delta\Theta_{d_{(t)}^{(v)}}\|_2}{G}\right) \tag{4.24}$$

$$\theta_{i_d k_{(t)}^{(v)}} = \theta_{i_d k_{(t-1)}^{(v)}} + \left(\eta_{(t)}^{(v)} \Delta\tilde{\Theta}_{d}^{(v)}{}_{(t)} + \xi_{(t)}^{(v)}\right) \tag{4.25}$$

Here, the noise matrix $\xi_{(t)}^{(v)}$ is chosen in the same way in the update rule for shared factors (Equation (4.18)).

Privacy Analysis for DP-CTF. Algorithm 2 illustrates the application of SGLD algorithm with DP for CTF. The difference of the DP-CTF algorithm from the original CTF is the amount of noise added and the gradient norm clipping. In the previous section we explained that clipping the norm is necessary to satisfy the Lipschitz condition and prevent the gradient explosion [120]. It is also a general assumption on loss functions in a DP setting [27, 116, 121]

Algorithm 2 Differentially Private SGLD for CTF (DP-CTF)

Inputs: Number of parties N, number of iterations T, minibatch size B, ϵ, δ, Lipschitz constant L, gradient clipping norm G, $\mathcal{X} = \{X^{(1)}, \ldots, X^{(N)}\}$, $\boldsymbol{\theta} = \{\Theta_{1_{(0)}}^{(v)}, \ldots, \Theta_{D_{(0)}}^{(v)}\}_{v=1}^{N}$

Outputs: $\boldsymbol{\theta} = \{\Theta_{1_{(t)}}^{(v)}, \ldots, \Theta_{D_{(t)}}^{(v)}\}_{t,v}$ where $t = \{1, \ldots, T\}$ and $v = \{1, \ldots, N\}$

for $t \leftarrow 1$ to T **do**
  for Each data site $v = 1 \ldots N$ **do**
    Set the step size κ_t for *private* factors and $\eta_{(t)}^{(v)}$ for *shared* factors using Eqn. (4.27)
    Sample each coord. of $\psi_{(t)}^{(v)}$ i.i.d from $\mathcal{N}(0, 2\kappa_t)$ and $\xi_{(t)}^{(v)}$ i.i.d. from $\mathcal{N}(0, \sigma^2 \vee \eta_{(t)}^{(v)})$
    Sample a minibatch $\mathcal{B}_{(t)}^{(v)}$ from $X^{(v)}$ of size B
    for each i_v in $\mathcal{B}_{(t)}^{(v)}$ **do**
      if site-level privacy **then**
        Update *private* factors $\Theta_{d_{(t)}}^{(v)}$ using Eqn. (4.22)
      else if user-level privacy **then**
        Compute (Eqn. (4.23)) and clip gradients (Eqn. (4.24)) for *private* factors $\Theta_{d_{(t)}}^{(v)}$
        Update *private* factors $\Theta_{d_{(t)}}^{(v)}$ using Eqn. (4.25)
      end if
      Compute gradients for *shared* factors using Eqn. (4.18)
      Clip gradients for *shared* factors using Eqn. (4.19)
    end for
  end for
  Update *shared* factors $\Theta_{d(t)}$ using Eqn. (4.20)
  Return all $\Theta_{d(t)}$s as posterior samples
end for

where the output of a target function should not be sensitive to any changes on an arbitrary data point. Therefore, it is essential to bound the impact of a single data point to that function. In addition, the convergence rate of the SGLD is not affected by the norm clipping since the clipping process is equal to choosing an adaptive step size in SGLD [122].

In this section we first show that Algorithm 2 preserves (ϵ, δ)-DP and under a certain

condition. Our analysis on the differential privacy of DP-CTF algorithm combines theorems and lemmas from the differential privacy literature. Since this algorithm is iterative and each iteration involves adding Gaussian noise, we need the results for the Gaussian mechanism [89] and a composition of differentially private mechanisms. For the first DP definition, we examine the Advanced Composition (AC) [26] method. Then, we make use of another definition of differential privacy, namely zero concentrated differential privacy (zCDP) [84]. We show that DP-CTF preserves (ϵ, δ)-zCDP with less amount of noise by using the relation between DP and zCDP.

DP-CTF with Advanced Composition. In order to study the differential privacy of the DP-CTF algorithm, it is useful to view its iterations as a query that is used as a function to be applied to a dataset. In iterative approaches, the following queries can be chosen adaptively based on the observed responses to the previous queries. We need to efficiently model the composition by considering that our dataset can be used as input to succeeding mechanisms. In our algorithm, as a result of multiple (say k in order to be consistent with the literature) iterations, we obtain a bound for the expected cumulative privacy loss where each iteration provides ϵ_{iter}-DP. We have to decide on a small amount of δ_{iter} to get a much better bound for the privacy loss in consideration of the composition. We begin with the analysis of the advanced composition, later we apply the recently proposed zCDP composition analysis. Here, we recall the Advanced Composition [89] and Gaussian Mechanism [89] that are given formerly in Chapter 2.

Theorem 4.6. ***(Advanced composition)*** *For all ϵ_{iter}, δ_{iter}, $\delta' \geq 0$, the class of $(\epsilon_{iter}, \delta_{iter})$-DP mechanisms satisfy $(\epsilon_{tot}, \delta_{tot})$-DP under k-fold adaptive composition for*

$$\epsilon_{tot} = \sqrt{2k \log(1/\delta')}\epsilon_{iter} + k\epsilon_{iter}(e^{\epsilon_{iter}} - 1) \qquad \delta_{tot} = k\delta_{iter} + \delta' \tag{4.26}$$

Given the desired amount of privacy ϵ, the privacy loss for the current iteration ϵ_{iter} is computed by using the Gaussian mechanism.

Theorem 4.7. ***(Gaussian Mechanism (GM)*** *Given a function f with L_2 sensitivity of $\triangle_2 f$, releasing $f(X) + \zeta$ where $\zeta \sim \mathcal{N}(0, \sigma^2)$ is (ϵ, δ)-DP when $\sigma \geq \triangle_2 f \sqrt{2\log(1.25/\delta)}/\epsilon$.*

Due to the minibatch setting, we will also make use of the following lemma of *privacy for subsampled data*.

Theorem 4.8. ***(Privacy for subsampled data, Theorem 1 in [123])*** *Any $(\epsilon_{iter}, \delta_{iter})$-DP mechanism running on a subsample of any dataset $X \in \mathcal{X}^S$, where each data point is uniformly and independently sampled with probability γ and where $\gamma > \delta_{iter}$, guarantees $(\log(1 + \gamma(\exp(\epsilon_{iter}) - 1)), \gamma\delta_{iter})$-DP.*

Note that we will use minibatches with fixed size B in each iteration and the data instances are included independently with probability $\gamma = B/C_v$. So we perturb the gradient $\Delta\Theta_{d_{(t)}^{(v)}}$ at iteration t by adding Gaussian noise $\xi_{(t)}^{(v)} \sim \mathcal{N}(0, \sigma^2)$ where $\sigma \geq \triangle_2 f \sqrt{2\log(1.25/\delta)}/\epsilon_{iter}$. We use the Theorem 4.6, 4.7 and 4.8 to compute the amount of additional noise in each iteration. The computation of the value of σ in terms of the DP-CTF algorithm's parameters is given in the Appendix C.

Choice of step size $\eta_{(t)}^{(v)}$. In [27], the authors proved that the other way to preserve (ϵ, δ)-DP is to choose the value of the step size $\eta_{(t)}^{(v)}$ to be small enough to satisfy:

$$\eta_{(t)}^{(v)} = \frac{\alpha\epsilon^2}{128L^2\log(2.5C_vT/(B\delta))\log(2/\delta)t} \tag{4.27}$$

Here, the α-*phase transition* is used to determine the step size: whenever $t > \alpha C_v T$ and the algorithm is run for the last $(1-\alpha)C_vT/B$ iterations, the approximate posterior samples can be collected differential privately. As mentioned earlier, the absolute value of the privacy loss will be bounded by ϵ with probability at least $1 - \delta$ [87]. In the experiments, we choose a small value for δ except for zero to maximize the probability while avoiding division by zero.

DP-CTF with zCDP composition. Recently [84] has demonstrated that zCDP composition allows a tighter analysis of the per-iteration privacy budget with the Gaussian mechanism. This is stated by the following proposition that shows the required amount of noise for Gaussian mechanism with zCDP.

Proposition 4.9. ***(Proposition 1.6 in [84])*** *The Gaussian Mechanism with some noise variance τ and a sensitivity $\triangle$ satisfies $\triangle^2/(2\tau)$-zCDP.*

In the following, we cite the crucial relation between zCDP and DP that is needed for the conversion between them and enables us to have tighter bounds on the DP of each iteration of DP-CTF algorithm hence less amount of noise added in each step.

Proposition 4.10. ***(Proposition 1.3 in [84])*** *If an algorithm $\mathcal{A}$ is ρ-zCDP, then it is (ϵ, δ)-DP for all (ϵ, δ) jointly satisfying $\delta > 0$ and $\epsilon = \rho + 2\sqrt{\rho \log(1/\delta)}$, δ-DP for any $\rho > 0$.*

Our final result on zCDP holds that zCDP admits the following basic composition property.

Lemma 4.11. ***(Lemma 1.7 in [84])*** *If two mechanisms $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy ρ_1-zCDP and ρ_2-zCDP respectively, then the composition $\mathcal{A}(\mathcal{X}) = (\mathcal{A}_1(\mathcal{X}), \mathcal{A}_2(\mathcal{X}))$ (with the same domain $\mathcal{X}$) is $(\rho_1 + \rho_2)$-zCDP.*

First we use Proposition 4.9 and Lemma 4.11 to compute the T composition of the Gaussian mechanism. DP-CTF algorithm with zCDP after T iteration provides $T \triangle^2/(2\tau)$-zCDP. Then we convert the privacy loss in zCDP to DP for the comparison purpose. Using Proposition 4.10, we map $T \triangle^2/(2\tau)$-zCDP to $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$-DP where $\rho = T \triangle^2/(2\tau)$. Herein we have the amount of the total privacy budget both in terms of DP and zCDP. After this step the per-iteration privacy budget is computed in the same way with the advanced composition case. (The complete proof is given in Appendix C). The same amount of privacy ϵ is defined as:

$$\epsilon = T \triangle^2/(2\tau) + 2\sqrt{\rho \log(1/\delta)} \quad \text{where} \quad \tau \geq 2\log(1.25/\delta') \triangle^2/\epsilon'^2 \tag{4.28}$$

Here the (ϵ', δ') are defined as intermediate privacy budget. Given that the total privacy budget (ϵ, δ) and number of iterations T, Equation 5.6 is solved to compute (ϵ', δ'). Lastly,

the intermediate privacy budget is mapped to the per iteration privacy budget $(\epsilon_{iter}, \delta_{iter})$ by using the Theorem 4.8 and solving the following equation:

$$\epsilon' = \log(1 + \gamma(\exp(\epsilon_{iter}) - 1)) \quad \text{and} \quad \delta' = \gamma\delta_{iter} \tag{4.29}$$

Since the value of ϵ_{iter} in this case is bigger than the ϵ_{iter} in the advanced composition, the amount of the added Gaussian noise is smaller. In the following section, the experimental results demonstrate the difference between DP and zCDP.

5. DIFFERENTIALLY PRIVATE BAYESIAN NEURAL NETWORKS

Deep neural networks (DNN) have recently generated significant interest, largely due to their successes in several important learning applications, including image classification, language modeling and many more (e.g., [124–126]). The success of neural networks is directly related to the availability of big and illustrative training datasets. However, most of the time these datasets are collected from individuals, such as their tastes and behavior as well as medical health records, and present obvious privacy issues. In these cases, applying deep learning approaches to the training data are restricted by privacy requirements or legislative regulations. Therefore, sharing and usage of the data about individuals require methods that provide precise privacy guarantees while meeting the demands of the applications.

Representing model uncertainty in deep neural networks is another issue of crucial importance. Recently the deterministic deep learning models have been replaced by Bayesian models [127, 128] in order to be able to capture parameter uncertainty and its effects over predictions. It is significant to use uncertainty information in medical applications [129], life sciences [130], automated decision making in the autonomous control and self driving cars [131]. Bayesian deep learning [132–135] -a field at the intersection between Bayesian probability theory and deep learning- offers us uncertainty estimates. Bayesian Neural Networks are DNNs with prior probability distributions placed over their weights. (Figure 2.4 in Section 5.1 illustrates the standard DNN and Bayesian DNN models.) They are effectively used for inferring complex multi-modal posterior distributions. Given observed data, inference is performed to find what are the apparent and improbable weights to describe the data. For the deep learning models, two ways are typically used to build uncertainty estimates which are establishing distributions over model weights and learning a direct alignment to probabilistic outputs.

One last challenge we handle in this study is the overfitting problem in training deep neural networks; since DNNs can model highly complex prediction functions using a large

number of parameters. It is often difficult to optimize these functions due to the potentially large number of local minimas in the space of parameters, and standard optimization techniques are prone to getting stuck in a local minimum which might be far from the global optimum. A popular regularization technique to avoid such local minima is dropout [136–138] which introduces noise into a model and optimizes loss function. Recently, it was shown that dropout can be treated as a Bayesian regularization method [139, 140] and it can be used to tune each weight's individual dropout rates. Gal *et al* [128] has proved that optimization of any neural network with dropout corresponds to a kind of approximate inference in a probabilistic interpretation of the model. Further, this implies that if a neural network is trained with dropout becomes a BNN; and it is able to capture the uncertainty [140]. Besides its primary objective of regularization and uncertainty representation, dropout can be used to hide the details of the training data for achieving privacy. The key purpose of this paper is to analyze dropout in order to provide a theoretical guarantee for the privacy protection of the deep neural networks. As a consequence, we propose an approach that represents uncertainty while providing a solution to overfitting and the privacy problems.

In this chapter, we develop a differentially private algorithm by exploiting the inherent randomization of the dropout. We study Gaussian dropout in the case we tune individual dropout rates for each weight of neural network to provide measurable privacy guarantee. In order to use the privacy budget more efficiently over many iterations, our approach uses the zCDP composition combined with the privacy amplification effect due to subsampling of data, which significantly decreases the amount of additive noise for the same expected privacy guarantee compared to the standard DP analysis.

5.1. Notations and Background

In this section, we first describe the notation that we use throughout this chapter and briefly explain the dropout in DNNs in order to provide a background for the following section.

5.1.1. Notation

Throughout this paper, we show the data with $\mathcal{D} = \{d_i\}_{i=1}^{N}$ where d_i = (x_i, y_i), with input object/feature $x_i \in \mathcal{R}^D$ and output label $y_i \in \mathcal{Y}$ where $\mathcal{Y}$ stands for the discrete output space. The relationship from x to y is defined by a model with parameters (or weights) θ. Here, the main purpose is to tune the parameters θ of a model $p(y|x, \theta)$ that predicts the output y given the input x and the parameter θ. Bayesian inference in such a model consists of updating some initial belief over parameters θ in the form of a prior distribution $p(\theta)$, after observing data $\mathcal{D}$, into an updated belief over these parameters in the form of the posterior distribution $p(\theta|\mathcal{D})$. The posterior distribution of a set of N items is $p(\theta|\mathcal{D}) \propto p(\theta)\ p(\mathcal{D}|\theta)$ where the corresponding data likelihood is $p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(d_i|\theta)$.

Computing the posterior is often difficult in practice as it requires the computation of analytically intractable integrals, so we need to use approximation techniques. One of such techniques is Stochastic Gradient Langevin dynamics (SGLD) [29] that is used to scale up Bayesian learning by combining a popular class of methods called stochastic optimization [141] and Markov chain Monte Carlo (MCMC) [94] that generates a sequence of samples from a Markov chain.

5.1.2. Dropout

Dropout is one of the most popular regularization techniques for neural networks which injects multiplicative random noise to the input of each layer during the training procedure. For a fully connected neural network, the formalization of dropout is denoted as:

$$h_2 = w((\xi \circ \theta)h_1) \quad \text{with} \quad \xi_{i,j} \sim p(\xi_{i,j}) \tag{5.1}$$

where h_1 and h_2 are the consecutive layers, θ is the weight matrix for the current layer and $w(\cdot)$ is the nonlinear function. The $\odot$ symbol denotes the elementwise (Hadamard) product of the input matrix with a matrix of independent noise variables ξ. The previous publications [136, 138, 142] show that the weight parameters θ are less likely to overfit to the training data by adding noise to the input or weights during optimization. At first, Hinton *et*

al. [136] proposed the Binary Dropout where the elements of ξ are drawn from a Bernoulli distribution with parameter $1-p$. It means each element of the input matrix is equal to one with probability $1-p$ where p is called as *dropout rate*. Afterwards, the same authors proposed the Gaussian Dropout using continuous noise $\xi_{i,j} \sim \mathcal{N}(1, \alpha = \frac{p}{1-p})$ with same relative mean and variance works as well or better [138].

5.2. Methodology

In this section, we can describe our approach toward differentially private training of neural networks and introduce the proposed differentially private dropout algorithm. First, we define the connection between dropout and SGLD by replacing the multiplicative noise term of dropout with an additive noise term. Afterwards, we present our dropout algorithm and compute the per-iteration privacy budget for both advanced composition and zCDP composition.

5.2.1. Connection between dropout and SGLD

Dropout has been proposed to improve model generalization in neural networks by adding noise to global weights or hidden units during training. In [85], Molchanov *et al* proved that the Stochastic Gradient MCMC brings a Bayesian interpretation to dropout [142]. In this interpretation, the SGD updates are combined with Gaussian dropout and the resulting update rule has the same form as SGLD:

$$\theta_{t+1} = (\xi \circ \theta_t) + \frac{\eta_t}{2}\nabla_\theta g \tag{5.2}$$

$$= \theta_t + \frac{\eta_t}{2}\nabla_\theta g + \xi' \tag{5.3}$$

where $\xi' \sim \mathcal{N}(0, \eta_t V)$ and $V = \frac{\alpha}{\eta_t} diag(\theta_t^2)$. At each step, the gradient $\nabla_\theta g = \nabla \mathcal{L}(\theta)$ is computed for a random subset of examples as:

$$\nabla_\theta g = \nabla_\theta \log p(\theta_t) + \frac{N}{S}\sum_{i \in \mathcal{S}_t} \nabla_\theta \log p(d_{t_i}|\theta_t) \tag{5.4}$$

In this way, we replace the multiplicative noise term ξ in Eq.(5.2) with an additive noise term ξ' in Eq.(5.3) and it corresponds to the additive noise from the Brownian motion of Langevin dynamics. The injected noise also helps us to propose a dropout algorithm that satisfies the differential privacy definition by adding independent Gaussian noise to the updates of each weight.

5.2.2. Differentially Private Dropout (DPD)

Eq.(5.3) is used to learn and regularize the model by adding random noise ξ'. Our idea is to tune the noise term to provide privacy protection while keeping an acceptable α (which is equal to $\frac{p}{1-p}$ where p is the dropout rate) to improve the model. To protect the privacy of training data, we need to perturb the gradients with a Gaussian noise in each iteration. We use the existing random noise $\xi' \sim \mathcal{N}(0, \eta_t V)$ that is already being used for regularization where $V = \frac{\alpha}{\eta_t} diag(\theta_t^2)$ and eventually it equals to $\xi' \sim \mathcal{N}(0, \alpha\ diag(\theta_t^2))$.

In the original method [85], the authors considered the case when there is a single α for the model and the noise is controlled by it. In our case, at each iteration of the training scheme, *DPD* takes a minibatch of data and computes the gradient, clips the L_2 norm of each gradient and adds noise to the gradient to protect privacy. The noisy stochastic gradient is then used to update model parameters θ via SGLD method by iteratively applying the update equation (5.3). The algorithm requires several parameters to determine the privacy budget such as sampling frequency $\nu = S/N$ for subsampling within the dataset, a total number of iterations T and clipping threshold G. Clipping the gradients using the threshold G will lead L_2 sensitivity of gradient sum to be $2G$. We have chosen to perturb parameter updates with zero mean multivariate normal noise with covariance matrix $(2G)^2\sigma^2\mathbb{I}$. Parameter σ in noise level determines our total ϵ and the amount of noise is chosen to be equal to the $\xi' \sim \mathcal{N}(0, 4G^2\sigma^2\mathbb{I})$. This amount is equal to the $\xi' \sim \mathcal{N}(0, \alpha\ diag(\theta_t^2))$, hence α can be obtained by using this equality after determining the noise level that preserves ϵ-DP. Here, α is not fixed for each weight and is controlled by the noise level that protects ϵ-DP.

Algorithm 3 outlines our method for training a model with parameters θ. In the next subsection, we describe in detail how privacy design parameters are chosen and privacy budget

is calculated.

Algorithm 3 Differentially Private Dropout (DPD)

1: **Inputs**: Input data $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$, number of data passes T, minibatch size B, learning rate η_t, noise scale σ, gradient norm bound G.
2: Initialize θ_0 randomly
3: **for** $t \leftarrow 1$ to T **do**
4: Take a random sample $\mathcal{B}_t$ of size B with probability $\nu = B/N$
5: Compute gradient $\nabla_\theta g$ from $(x_i, y_i) \in \mathcal{B}_t$
6: Clip gradient: $\nabla_\theta \bar{g} = \nabla_\theta g / \max\left(1, \| \nabla_\theta g \|_2 / G\right)$
7: Compute noise: $\xi' \sim \mathcal{N}(0, 4G^2\sigma^2\mathbb{I})$
8: Update parameter: $\theta_{t+1} = \theta_t + \frac{\eta_t}{2}\left(\nabla_\theta \bar{g} + \xi'\right)$
9: **end for**
10: **Output**: θ_T.

We note that all the parameters are represented by a single parameter θ of the loss function $L(.)$ in Algorithm 3. We consider each layer independently for multi-layer neural networks. This allows setting different noise scales σ or different clipping thresholds G for each layer. The privacy parameter ϵ of each layer is summed up to compute the total privacy of the neural network. However, we use constant settings for G and σ in the results presented in Section 8.1.

5.2.3. Privacy Analysis for DPD

We encounter the common challange *cumulative privacy loss* that acuumulates the privacy loss at each access to the given training data to the neural network. We calculate the privacy loss by using the zCDP to use the privacy budget of DPD method more effectively. In this section, we first calculate the per-iteration privacy budget using the key properties of advanced composition theorem and call this method *DPD-AC* in the experiments. Then, we use a relaxed notion of differential privacy, called zCDP [84] that bounds the moments of the privacy loss random variable and call this method *DPD-zCDP*. As we mentioned in the previous section, the moments bound yields a tighter tail bound, and consequently, it allows

for a higher per-iteration budget than standard DP-methods for a given total privacy budget.

DPD-AC:. In order to study the differential privacy of the DP-AC algorithm, it is useful to view its iterations as a query that is used as a function to be applied to a dataset. In iterative approaches, the queries can be chosen adaptively based on the observed responses to the previous queries. We need to efficiently model the composition by considering that our dataset can be used as input to succeeding mechanisms. In our algorithm, we obtain a bound on the expected cumulative privacy loss as a result of a number of (say k in order to be consistent with the literature) iterations, each providing ϵ_{iter}-DP. We have to decide on a small amount of δ_{iter} to get a much better bound for the privacy loss in consideration of the composition. We begin with the analysis of the advanced composition, later we apply the recently proposed zCDP composition analysis. One more time, we use the advanced composition theorem. Moreover, we present it here to make this section easier to follow.

Theorem 5.1. ***(Advanced composition)*** *For all ϵ_{iter}, δ_{iter}, $\delta' \geq 0$, the class of $(\epsilon_{iter}, \delta_{iter})$-DP mechanisms satisfy $(\epsilon_{tot}, \delta_{tot})$-DP under k-fold adaptive composition for*

$$\epsilon_{tot} = \sqrt{2k\log(1/\delta')}\epsilon_{iter} + k\epsilon_{iter}(e^{\epsilon_{iter}} - 1) \qquad \delta_{tot} = k\delta_{iter} + \delta' \tag{5.5}$$

Remark 5.2. ***(Remark 1 in [27]):*** *When $\epsilon = \frac{c}{\sqrt{2k\log(1/\delta')}} < 1$ for some constant $c < \sqrt{\log(1/\delta')}$, the equation of ϵ' can be simplified into $\epsilon' \leq 2c$ by applying the inequality $e^{\epsilon} - 1 \leq 2\epsilon$.*

The theorem states that with small ϵ and small loss in δ_{tot}, more strict ϵ_{tot} is obtained than just summing the ϵ. This is clear by looking at the first order expansion for small ϵ (Taylor Theorem is used with assumption that $\epsilon \leq 1$) of $\epsilon_{tot} = \sqrt{2k\log(1/\delta')}\epsilon + k\epsilon^2$.

In this section, we add differential privacy into Gaussian dropout by clipping and perturbing the gradients. As our method for perturbation, we use *Gaussian Mechanism* (GM) where Gaussian noise calibrated to the global sensitivity is added.

Theorem 5.3. ***(Gaussian Mechanism (GM):*** *Let $\epsilon \in (0, 1)$ be arbitrary.* Gaussian Mechanism *states that given function f with L_2 sensitivity of $\triangle_2 f$, releasing $f(X) + Z$ where $Z \sim \mathcal{N}(0, \sigma^2)$ is (ϵ, δ)-DP when $\sigma \geq \triangle_2 f \sqrt{2\log(1.25/\delta)}/\epsilon$.*

Given two adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$, the important L_2-sensitivity of a function f is defined as:

$$\triangle_2 f = \sup_{\mathcal{D},\mathcal{D}',\|\mathcal{D}-\mathcal{D}'\|=1} \| f(\mathcal{D}) - f(\mathcal{D}') \|_2$$

We use a stochastic gradient algorithm that uses minibatches of data while learning. Due to the minibatch setting, we can make use of the *amplifying effect of the subsampling* on privacy. The version of the privacy amplification theorem we use is as follows:

Theorem 5.4. ***(Privacy for subsampled data:*** *Any (ϵ_{iter}, δ_{iter})-DP mechanism running on a sampled subset of the data, where each data point is included independently with probability ν, and where $\nu > \delta_{iter}$, guarantees ($\log(1 + \nu(\exp(\epsilon_{iter}) - 1))$, $\nu\delta_{iter}$)-DP.*

We will assume that the instances are included independently with probability $\nu = B/N$; and for ease of implementation, we will use minibatches with fixed size B in our experiments. As we mentioned before, parameter σ in noise level determines our total ϵ and depends on the total δ in privacy budget. Given the desired amount of privacy ϵ, the privacy loss for the current iteration ϵ_{iter} is computed by using the Gaussian mechanism. We calculate the total privacy budget ϵ_{tot} by setting $\sigma = \sqrt{2\log(1.25/\delta_{iter})}/\epsilon_{iter}$. Clipping will lead L_2 sensitivity of gradient sum to be $2G$, so perturbing each sum with aforementioned noise will lead each iteration to be (ϵ_{iter},δ_{iter})-DP w.r.t the subset. Now if we set $\delta_{iter} = (\delta_{tot} - \delta')/T\nu$, where δ' comes from advanced composition, we can provide δ_{tot} as δ parameter in total privacy cost. Using Theorem 5.1 and Theorem 5.3, the ϵ parameter in our total privacy cost for *DPD-AC* will be:

$$\epsilon_{tot} = \sqrt{2T\log(1/\delta')}/\sigma' + T(\sigma')^2 \quad \text{where}$$
$$\sigma' = \log\left(1 + \nu(\exp(\sqrt{2\log(1.25/\delta_{iter})}/\sigma) - 1)\right)$$

DPD-zCDP:. For *DPD-zCDP* algorithm, we calculate the per-iteration budget using *the zCDP composition* [84] that is used for tracking privacy loss of the composite mechanisms. It permits a tighter analysis of the per-iteration privacy budget with the Gaussian mechanism. This is stated by the following proposition that shows the required amount of noise for Gaussian mechanism with zCDP.

Proposition 5.5. ***(Proposition 1.6 in [84]):*** *The Gaussian mechanism with some noise variance τ and a sensitivity $\triangle$ satisfies $\triangle^2/(2\tau)$-zCDP.*

In the following, we cite the crucial relation between zCDP and DP that is needed for the conversion between them and enables us to have tighter bounds on the DP of each iteration of DP-CTF algorithm hence less amount of noise added in each step.

Proposition 5.6. ***(Proposition 1.3 in [84])*** *If an algorithm $\mathcal{A}$ is ρ-zCDP, then it is (ϵ, δ)-DP for all (ϵ, δ) jointly satisfying $\delta > 0$ and $\epsilon = \rho + 2\sqrt{\rho \log(1/\delta)}$, δ-DP for any $\rho > 0$.*

Our final result on zCDP holds that zCDP admits the following basic composition property.

Lemma 5.7. ***(Lemma 1.7 in [84])*** *If two mechanisms $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy ρ_1-zCDP and ρ_2-zCDP respectively, then the composition $\mathcal{A}(\mathcal{X}) = (\mathcal{A}_1(\mathcal{X}), \mathcal{A}_2(\mathcal{X}))$ (with the same domain $\mathcal{X}$) is $(\rho_1 + \rho_2)$-zCDP.*

First we use Proposition 5.5 and Lemma 5.7 to compute the T composition of the Gaussian mechanism. DP-CTF algorithm with zCDP after T iteration provides $T \triangle^2/(2\tau)$-zCDP. Then we convert the privacy loss in zCDP to DP for the comparison purpose. Using Proposition 5.6, we map $T \triangle^2/(2\tau)$-zCDP to $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$-DP where $\rho = T \triangle^2/(2\tau)$. Herein we have the amount of the total privacy budget both in terms of DP and zCDP. After this step the per-iteration privacy budget is computed in the same way with the advanced composition case. The same amount of privacy ϵ is defined as:

$$\epsilon = T \triangle^2/(2\tau) + 2\sqrt{\rho \log(1/\delta)} \quad \text{where} \quad \tau \geq 2\log(1.25/\delta') \triangle^2/\epsilon'^2 \tag{5.6}$$

Here the (ϵ', δ') are defined as intermediate privacy budget. Given that the total privacy budget (ϵ, δ) and number of iterations T, Equation 5.6 is solved to compute (ϵ', δ'). Lastly, the intermediate privacy budget is mapped to the per iteration privacy budget $(\epsilon_{iter}, \delta_{iter})$ by using the Theorem 4.8 and solving the following equation:

$$\epsilon' = \log(1 + \gamma(\exp(\epsilon_{iter}) - 1)) \quad \text{and} \quad \delta' = \gamma\delta_{iter} \tag{5.7}$$

In the following section, we demonstrate the experimental results that proves the amount of the Gaussian noise added by DPD-AC is bigger than the amount of noise added by DPD-zCDP at each iteration.

6. EXPERIMENTS ON COUPLED TENSOR FACTORIZATION

In this chapter, we will evaluate our estimation methods that are described in Chapter 3. In the subsequent chapter, we will evaluate the differentially private methods that we have presented in Chapter 4. Our experiments are conducted on link prediction problem. In Section 6.1, we will apply our MAP estimation method for learning the latent factors and in Section 6.2 we will apply the variational Bayesian inference method. We first demonstrate the advantages of the coupled factorization models in the setting where we assume that the dispersion parameters are given and compare the loss functions. Then, we will present our results on link prediction experiments where we evaluate the Varitional Inference on coupled tensor factorization. We left the evaluation of MCMC based inference to the next chapter.

6.1. Experiments with ML and MAP Estimation

This section reports our experimental study on two real world datasets: UCLAF and Digg. For both datasets, we first demonstrate that coupled tensor factorizations outperform low-rank approximations of a single tensor in terms of missing link prediction. Then, within the context of coupled tensor factorizations, we compare different tensor models and loss functions and show that selection of the tensor model and loss function is significant in terms of link prediction performance, especially when the data is sparse. Our experiments demonstrate that loss functions that have not been studied for link prediction before, such as IS-divergence, outperform the commonly-used loss functions. For each dataset, we begin with describing the datasets and the tensor factorization models on these datasets. Then, we present the results. As evaluation metrics, we use Area Under the Receiver Operating Characteristic Curve (AUC) and P@K for link prediction results and Root Mean Square Error (RMSE) for tensor completion results.

RMSE: RMSE is a measure of the "average" error, weighted according to the square of the error. In our experiments, we use RMSE to measure the tensor reconstruction performance.

AUC: Link prediction datasets are characterized by extreme imbalance, i.e., the number of

links known to be present is often significantly less than the number of edges known to be absent. This issue motivates the use of AUC as a performance measure since AUC is viewed as a robust measure in the presence of imbalance [143].

P@K: Precision at k (P@K) measures the precision at a fixed number of retrieved items (i.e., top K) of the ordered list $r\prime$ and the unordered list r [144]. Assume $TopK$ and $TopK\prime$ are the retrieved items of r and $r\prime$ respectively, then the P@K is defined as $P@K = \frac{|TopK \cap TopK\prime|}{K}$

6.1.1. Results on UCLAF Dataset

UCLAF dataset [145] is extracted from the GPS data that include information of three types of entities: user, location and activity (see Figure 6.1 for an illustration of the data). The relations between user-location-activity triplets are used to construct a three-way tensor $X^{(1)}$. In tensor $X^{(1)}$, an entry indicates the frequency of a user i_1 visiting location i_2 and doing activity i_3 there; otherwise, it is 0. Since we address the link prediction problem in this section, we define the user-location-activity tensor $X^{(1)}$ as:

$$X^{(1)}_{i_1,i_2,i_3} = \begin{cases} 1 & \text{if user } i_1 \text{ visits location } i_2 \text{ and performs activity } i_3 \text{ there} \\ 0 & \text{otherwise} \end{cases}$$

To construct the dataset, raw GPS points were clustered into 168 meaningful locations and the user comments attached to the GPS data were manually parsed into activity annotations for the 168 locations. Consequently, the data consists of 164 users, 168 locations and 5 different types of activities, i.e., 'Food and Drink', 'Shopping', 'Movies and Shows', 'Sports and Exercise', and 'Tourism and Amusement' [145]. The collected data also includes additional side information: the user-location preferences from the GPS trajectory data and the location features from the POI (points of interest) database, represented as the matrix X_2 and X_3 respectively. Using the location features, we could gain information about location similarities.

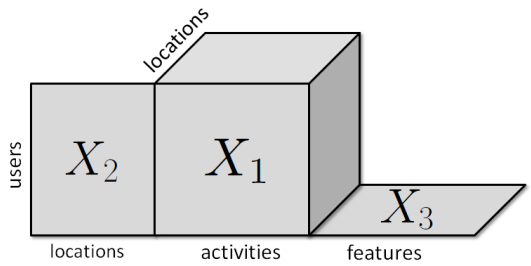


Figure 6.1. UCLAF dataset represented in the form of a third-order tensor coupled with two matrices in two different modes.

We have a three-way observation tensor $X^{(1)}$ with elements 0 and 1, where 0 denotes a known absent link and 1 denotes a known present link, and two auxiliary matrices $X^{(2)}$ and $X^{(3)}$ that provide side information. Our aim is to restore the missing links in $X^{(1)}$. This is a difficult link prediction problem since $X^{(1)}$ contains less than 1% of all possible links or an entire slice of $X^{(1)}$ may be missing. In order to fill in the missing links in tensor $X^{(1)}$, we form two different coupled models changing in the way tensor $X^{(1)}$ is factorized using a CP [34] and Tucker [41] models. For all models, we use KL divergence and Euclidean as cost functions in our non-negative decomposition problems.

Model 1 (CP): In the first model, we applied the coupled approach to a CP-style tensor factorization model by analyzing the tensor $X^{(1)}$ jointly with the additional matrices $X^{(2)}$ and $X^{(3)}$ in order to solve the sparsity problem in $X^{(1)}$ effectively. This gives us the following model:

$$\hat{X}^{(1)}_{i_1,i_2,i_3} = \sum_k \Theta_{1i_1,k}\Theta_{2i_2,k}\Theta_{3i_3,k} \tag{6.1}$$

$$\hat{X}^{(2)}_{i_1,i_4} = \sum_k \Theta_{1i_1,k}\Theta_{4i_4,k} \tag{6.2}$$

$$\hat{X}^{(3)}_{i_2,i_5} = \sum_k \Theta_{2i_2,k}\Theta_{5i_5,k} \tag{6.3}$$

Here, we have three observed tensors, that share common factors; therefore, we have a coupled tensor factorization problem. The coupling matrix R with $|v|= 3$ and $|d|= 5$ for this model is

defined as follows:

$$R = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{6.4}$$

Model 2 (Tucker): Following the same line of thought, we apply the coupled approach using a Tucker factorization to form our second model, which is as follows:

$$\hat{X}^{(1)}_{i_1,i_2,i_3} = \sum_{k,l,m} \Theta_{1i_1,k}\Theta_{2i_2,l}\Theta_{3i_3,m}\Theta_{6k,l,m} \tag{6.5}$$

$$\hat{X}^{(2)}_{i_1,i_4} = \sum_{k} \Theta_{1i_1,k}\Theta_{4i_4,k} \tag{6.6}$$

$$\hat{X}^{(3)}_{i_2,i_5} = \sum_{l} \Theta_{2i_2,l}\Theta_{5i_5,l} \tag{6.7}$$

In this model, once again, the factor Θ_1 is shared by $X^{(1)}$ and $X^{(2)}$, while the factor Θ_2 is shared by $X^{(1)}$ and $X^{(3)}$. Here, a latent preference space interpretation is less intuitive but the model has more freedom to represent the dependence.

We assess the performance of the coupled models, which are defined above, in terms of tensor completion and/or missing link prediction. By setting different amounts of data to missing in user-location-activity tensor $X^{(1)}$, we compare the proposed models using both KL-divergence and the Euclidean as cost functions. In all experiments, *number of components* k (number of columns in each factor matrix) is set to 2 and the pattern of missing data is chosen randomly.

Tensor Completion . Table 6.1 shows tensor completion performances of standard CP and Tucker models, coupled models and PCLAF [145]. PCLAF is a personalized collaborative location and activity filtering algorithm, which uses a collective tensor and matrix factorization. In addition to the data that we have used in our models, PCLAF uses user-user

and activity-activity similarity matrices in UCLAF dataset. Also, PCLAF uses CP tensor factorization model and Euclidean distance as cost function. For PCLAF algorithm, they run the experiments five times, and report the average RMSE scores. Specifically, at each trial, they randomly split some percentage (30% and 50%) of the existing tensor entries for training and hold out the other for testing. We also set the same amount of missing entities randomly and report the average RMSE scores of ten independent runs. Hence, our results are comparable to PCLAF algorithm's results. Eventually, we observe that our models outperform the PCLAF approach, which has outperformed many collaborative filtering methods in [38], especially when we use KL divergence which is a lot more natural than a Euclidean cost for this data.

Table 6.1. RMSE for different models with different percentages of training data.

	EUC		KL	
Model	30%	50%	30%	50%
CP	0.27 ∓ 0.03	0.28 ∓ 0.04	0.24 ∓ 0.03	0.23 ∓ 0.03
TUCKER	0.26 ∓ 0.02	0.26 ∓ 0.04	0.22 ∓ 0.02	0.22 ∓ 0.02
Coupled(CP)	0.24 ∓ 0.01	0.23 ∓ 0.02	0.19 ∓ 0.02	0.18 ∓ 0.02
Coupled(TUCKER)	0.22 ∓ 0.01	0.22 ∓ 0.02	0.18 ∓ 0.01	0.18 ∓ 0.01
PCLAF [145]	0.30 ∓ 0.01	0.29 ∓ 0.01	-	-

Link Prediction . In order to demonstrate the power of coupled analysis, we compared the link prediction performance of standard CP and Tucker models with coupled ones using EUC and KL cost functions at different amounts, i.e., $\{40, 60, 80, 90, 95\}$, of randomly unobserved elements. For all cases, coupled models outperform the standard models clearly. We presented the results of this set of experiments in Table 6.2. First, we compare the CP and coupled CP models with different cost functions and different amount of missing data. As we can see, the coupled models that try to use as much additional information as possible to help alleviate the data sparsity issue perform better than the standard models; in particular, when

the percentage of missing data is high. When the fraction of missing data was more than 80%, the standard models could not find a solution. In order to demonstrate the effect of the cost function modeling the data, we have also carried out experiments on both coupled CP and Tucker models at different missing data fractions. For all models and missing data fractions, the KL cost function seems to perform better than EUC. Especially when the fraction of missing entries is high, KL outperforms EUC with a significant difference. Finally, we focus

Table 6.2. Link prediction results on UCLAF with different experimental settings.

	40%		80%		90%	
	EUC	KL	EUC	KL	EUC	KL
CP	0.920	0.946	0.808	0.867	-	-
Tucker	0.943	0.960	0.896	0.917	-	-
CP (Coupled)	0.951	0.968	0.915	0.937	0.813	0.869
Tucker (Coupled)	0.965	0.983	0.934	0.948	0.871	0.908

on the comparison of coupled CP and Tucker models in order to indicate the tensor model which models the data best. Table 6.2 shows that Tucker model outperforms the CP model; because Tucker model is more flexible due to the full core tensor which is helpful for us to explore the structural information embedded in the data.

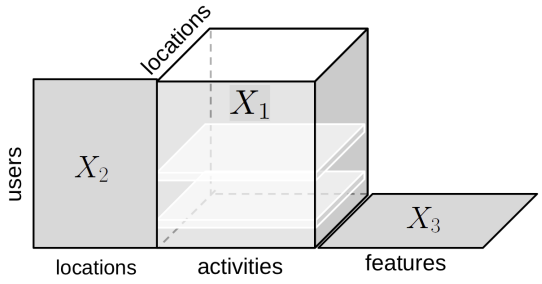
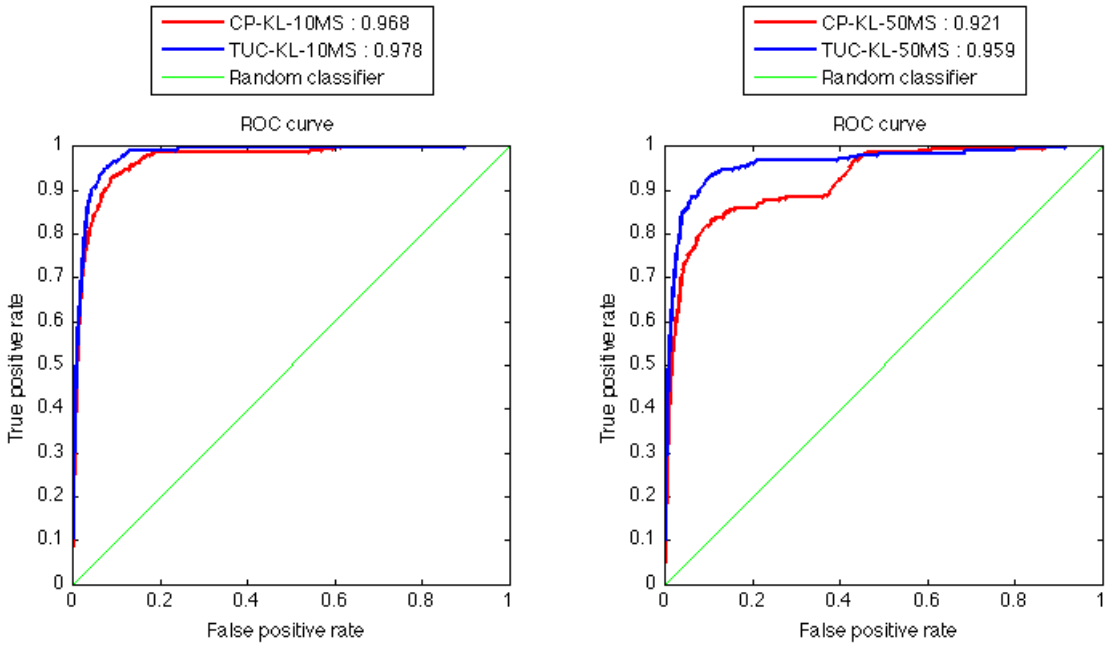


Figure 6.2. Cold-start problem.

Cold Start Problem . We study the case with completely missing slices, which corresponds to the *cold-start problem* in our link prediction setting and demonstrate that it is



(a) CP and Tucker - 10MS

(b) CP and Tucker - 50MS

Figure 6.3. Link prediction result with missing slices and KL cost.

still possible to predict missing links using the proposed coupled models whereas low-rank approximations of a single tensor would fail to do so. This problem is particularly important in link prediction because we may often have new users starting to use an application, e.g., a location-activity recommender system. Since they are new users, they will have no entry in $X^{(1)}$, i.e., a completely missing slice (See Figure 6.2 for illustration of the problem). It is not possible to reconstruct a missing slice of a tensor using its low-rank approximation. A similar argument is valid in the case of matrices for completely missing rows/columns [146]. In such cases, additional sources of information will be useful [147] to make recommendations to new users. We observe that our coupled models could predict the links when there is no information about a user in tensor $X^{(1)}$, by utilizing the additional sources of information. We test this case by setting randomly missing slices in $X^{(1)}$.

Figure 6.3 demonstrates the performance of coupled models with KL divergence when 10 users' data and 50 users' data are missing. Also note that Tucker is superior to CP as the amount of missing data increases.

6.1.2. Results on Digg Dataset

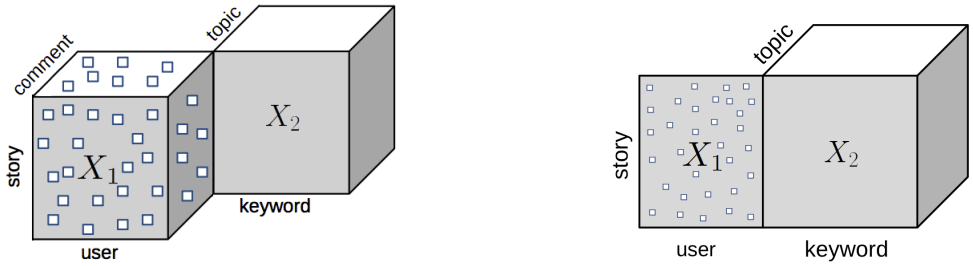
We address link prediction problem also on a large-scale dataset collected from Digg in order to show the scalability of the proposed approach. Digg is a social news resource that allows users to submit, digg and comment on news stories. Lin *et al.* [39] have collected data from a large set of user actions from Digg. The dataset is a subset of data scrapped from Digg by Choudhury *et al.* during January 2009 [148]. It includes stories, users and their actions (submit, digg, comment and reply) with respect to the stories, as well as the explicit friendship (contact) relation among these users. It also includes the topics of the stories and keywords extracted from the titles of stories. There are five types of entities: user, story, comment, keyword and topic and six relationships among them (see [39] for a comprehensive illustration of relations).

We will use three relationships in this study: user-story-comment, story-keyword-topic and user-story. Lin *et al.* [39] extract tuples with timestamps ranging from August 1 to August 27, 2008, segment the data duration into nine time slots (i.e. every three days), and construct a sequence of data tensors for each dynamic relation in order to study the data evolution. Except for the contact relation, all relations in this dataset have timestamps. However, in our work, since we are not modeling the evolution in time, we integrate the nine segments together and evaluate missing link prediction tasks on this integrated data. The total number of tuples in each integrated data tensor per relation is 151.779, 1.157.529 and 94.551 respectively. The prediction results are compared with the actual diggs and comments as ground truth. Based on the Digg scenario, we design two prediction tasks on Digg dataset:

(i) comment prediction - what stories a user will comment on,

(ii) digg prediction - what stories a user will digg.

Comment Prediction . For comment prediction, the relation between the user-story-comment is used to construct tensor $X^{(1)}$ of size $I_1 \times I_2 \times I_2$ where the number of users is $I_1 = 9583$, the number of stories is $I_2 = 44005$ and the number of comments is $I_3 = 241800$.



(a) Comment Prediction (b) Digg prediction

Figure 6.4. Comment and Digg prediction on Digg dataset.

$X^{(1)}$ is defined as:

$$X^{(1)}_{i_1,i_2,i_3} = \begin{cases} 1 & \text{if user } i_1 \text{ comments on story } i_2 \text{ with comment } i_3 \\ 0 & \text{otherwise} \end{cases}$$

Additionally, the data includes the topics of the stories and extracted keywords from the stories' titles. We represent this data as the three-way tensor $X^{(2)}$. In our model the story-keyword-topic tensor has entries $X^{(2)}(i_2, i_4, i_5)$ of size $I_2 \times I_4 \times I_5$, where the number of stories is $I_2 = 44005$, the number of keywords is $I_4 = 13714$ and the number of topics is $I_5 = 51$.

Our aim is to restore the missing links in $X^{(1)}$ where X_1 contains less than 0.07% of all possible links (see Figure 6.4(a) for an illustration of the modeled data). We form two coupled models in order to fill in the missing links in tensor $X^{(1)}$ through joint analysis of $X^{(1)}$ and $X^{(2)}$. For both models, we use Euclidean distance, KL divergence and IS divergence. We also explore the behaviour of the models using various cost functions, i.e., $p \in [0, 2]$, based on β-divergences.

Model 1 (CP): In the first model, we applied the coupled approach to a CP-style tensor factorization model by analyzing the tensor $X^{(1)}$ jointly with the additional tensor $X^{(2)}$ as

follows:

$$\hat{X}^{(1)}_{i_1,i_2,i_3} = \sum_k \Theta_{1i_1,k}\Theta_{2i_2,k}\Theta_{3i_3,k} \tag{6.8}$$

$$\hat{X}^{(2)}_{i_2,i_4,i_5} = \sum_k \Theta_{2i_2,k}\Theta_{4i_4,k}\Theta_{5i_5,k} \tag{6.9}$$

Model 2 (Tucker): We also apply the coupled approach using a Tucker factorization as follows:

$$\hat{X}^{(1)}_{i_1,i_2,i_3} = \sum_{k,l,m} \Theta_{1i_1,k}\Theta_{2i_2,l}\Theta_{3i_3,m}\Theta_{6k,l,m} \tag{6.10}$$

$$\hat{X}^{(2)}_{i_2,i_4,i_5} = \sum_l \Theta_{2i_2,l}\Theta_{4i_4,l}\Theta_{5i_5,l} \tag{6.11}$$

In both models, we have two observed tensors $X^{(1)}$ and $X^{(2)}$ that share factor matrix Θ_2.

Digg Prediction . For digg prediction, the relation between users and stories is used to construct matrix $X^{(1)}$ of size $I_1 \times I_2$ where the number of users is $I_1 = 9583$ and the number of stories is $I_2 = 44005$. The user-story matrix $X^{(1)}$ is defined as:

$$X^{(1)}_{i_1,i_2} = \begin{cases} 1 & \text{if user } i_1 \text{ votes (i.e. digg) on news stories } i_2 \\ 0 & \text{otherwise} \end{cases} .$$

Additionally, the data includes the topics of the stories and extracted keywords from titles of stories. We represent this data as a three-way tensor $X^{(2)}$. In our model the story-keyword-topic tensor has entries $X^{(2)}_{i_2,i_3,i_4}$ of size $I_2 \times I_3 \times I_4$, where the number of stories $I_2 = 44005$, the number of keywords is $I_3 = 13714$ and the number of topics is $I_4 = 51$ (see Figure 6.4(b) for an illustration of the modeled data).

Here, our aim is to restore the missing links in $X^{(1)}$. This is also a difficult link

prediction problem since $X^{(1)}$ contains less than 0.008% of all possible links. Once again, we form coupled models based on CP and Tucker models in order to fill in the missing links in matrix $X^{(1)}$. For both models, as cost functions, we use Euclidean distance, KL divergence, IS divergence as well as various cost functions, i.e., $p \in [0, 2]$, based on β-divergences.

Model 1 (CP): We applied the coupled approach based on a CP-style tensor model by analyzing matrix X_1 jointly with tensor X_2 as follows:

$$\hat{X}^{(1)}_{i_1,i_2} = \sum_k \Theta_{1i_1,k}\Theta_{2i_2,k} \tag{6.12}$$

$$\hat{X}^{(2)}_{i_2,i_3,i_4} = \sum_k \Theta_{2i_2,k}\Theta_{3i_3,k}\Theta_{4i_4,k} \tag{6.13}$$

Model 2 (Tucker): We also use a Tucker model for the coupled approach as follows:

$$\hat{X}^{(1)}_{i_1,i_2,i_3} = \sum_k \Theta_{1i_1,k}\Theta_{2i_2,k} \tag{6.14}$$

$$\hat{X}^{(2)}_{i_2,i_4,i_5} = \sum_{k,l,m} \Theta_{2i_2,k}\Theta_{3i_3,l}\Theta_{4i_4,m}\Theta_{5k,l,m} \tag{6.15}$$

In order to demonstrate the power of coupled analysis, we compared the link prediction performance of standard CP and Tucker models with coupled ones using EUC, KL and IS cost functions at different amounts, i.e., $\{40, 80, 90\}$, of randomly unobserved elements. In Table 6.3, we show results of the experiments on *both comment and digg prediction tasks*. For all cases, coupled models outperform the standard models clearly. We first discern the comparison of CP and coupled CP models with different cost functions when 40% and 80% of the data are missing. As we can see, coupled models perform better than the standard models; in particular, when the percentage of missing data is high. When the fraction of missing data was more than 80%, the standard models could not find a solution. In order to demonstrate the effect of the cost function modeling the data, we have also carried out experiments on both coupled CP and Tucker models at different missing data fractions. For all cases, the IS cost function seems to perform better than EUC and KL for both prediction tasks, especially

when the fraction of missing entries is high. When the missing data rate becomes higher, the difference between performances of cost functions become clearer. We lastly observe that CP model outperforms the Tucker model in terms of capturing the structural information embedded in the data.

Table 6.3. Link prediction results on Digg with different experimental settings in terms of AUC.

	Digg Prediction				Comment Prediction			
	40%		90%		40%		90%	
	CP	Tucker	CP	Tucker	CP	Tucker	CP	Tucker
EUC	0.855	0.831	0.817	0.801	0.845	0.831	0.810	0.780
KL	0.921	0.882	0.853	0.827	0.871	0.845	0.824	0.810
IS	0.939	0.923	0.895	0.869	0.901	0.885	0.882	0.859

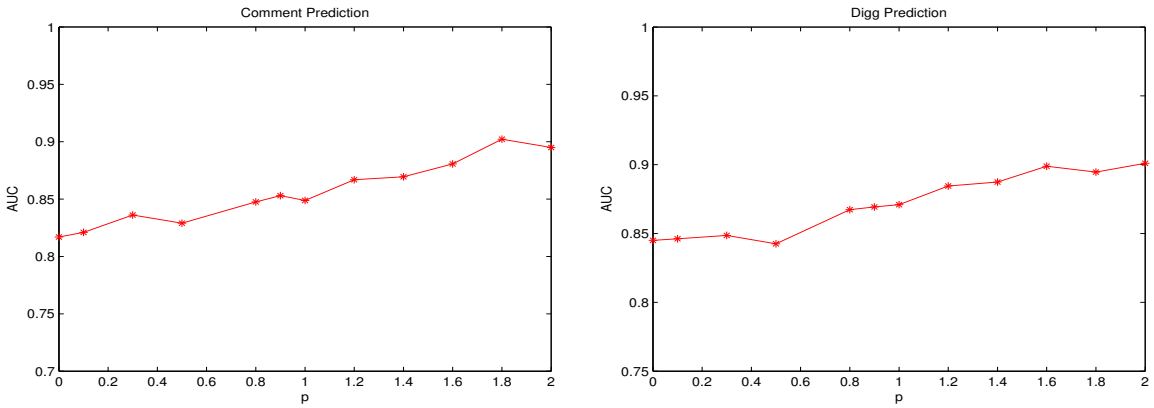
Digg and comment prediction have also been studied in [39] using the MFT (Metafac factorization with time evolving data) approach. The overall comment and digg prediction performances of MFT algorithm were obtained as 0.135 ± 0.001 and 0.543 ± 0.007 in terms of P@10, respectively in [39]. While our prediction results are clearly higher than those of MFT in terms of P@10, we cannot directly compare them since we use the integrated Digg dataset instead of the segmented data and do not deal with the temporal aspect of the data. However, in Table 6.4, we observe that if we use the loss function and the tensor model used in MFT, i.e., CP model based on KL-divergence, then it performs worse than modeling the data using IS-divergence. This clearly shows that GCTF framework is useful in terms of making use of better loss functions for modeling datasets.

In addition, in order to demonstrate the effect of the cost function modeling the data, we have also carried out experiments on both coupled CP and Tucker models at different missing data fractions using different p values. Figure 6.5 illustrates the performance of arbitrary cost functions (with different p values) for the coupled CP model for both comment and digg prediction when 90% of the data is unobserved. These results also confirm that IS-divergence,

Table 6.4. The average prediction performance for digg and comment prediction, evaluated by $P@10$ values. Modeling the data using IS-divergence gives the best results.

	Digg Prediction		Comment Prediction	
	40%	90%	40%	90%
GCTF-EUC	0.7154	0.6615	0.3632	0.3241
GCTF-KL	0.7414	0.6865	0.3751	0.3383
GCTF-IS	**0.7858**	**0.7479**	**0.4075**	**0.3846**

i.e., $p = 2$, performs better than KL-divergence, i.e., $p = 1$, which performs better than Euclidean distance, i.e., $p = 0$.



(a) CP - Comment Prediction (b) CP - Digg prediction

Figure 6.5. Comparison of different cost functions.

6.2. Experiments with Variational Inference

In this section, we demonstrate the use of the proposed variational Bayesian coupled tensor factorization method (CTF-VB) for missing link prediction problem in order to show that joint analysis of data from multiple sources via coupled factorization significantly improves the link prediction performance. We evaluate the performance of CTF-VB on the datasets: UCLAF and Digg. First, we demonstrate that coupled tensor factorizations (CTF-VB) outperform low-rank approximations of a single tensor (we call it TF-VB), then we

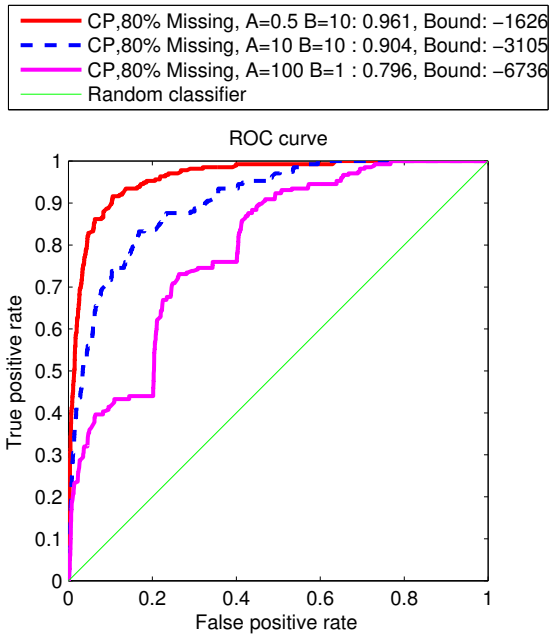


Figure 6.6. Effect of hyperparameter selection for CP model when R=2 and the missing data fraction is 80%.

compare the performance of the proposed variational Bayesian approach (CTF-VB) with the standard approach (we call it CTF-ML in this section) defined in Section 3.1.1 in terms of missing link prediction recovery.

Hyperparameter Selection . We observe that hyperparameter adaptation is crucial for obtaining good prediction performance. In our simulations, results for TF-VB without hyperparameter adaptation were occasionally poorer than the TF-ML estimates. We set both shape A and scale B hyperparameters same for all components $\Theta_{1:3}$. We tried several number of different values for hyperparameters to obtain the best prediction results under missing data case. Figure 6.6 shows the comparison of three different hyperparameter settings; $A = 0.5, B = 10$, $A = 10, B = 10$ and $A = 100, B = 1$ in terms of link prediction performance. As we can see, we obtain best result when initialising the shape hyperparameter $A = 0.5$ and scale hyperparameter $B = 10$ for all settings of missing data. So, we use these values of hyperparameter A and B for the following experiments. In addition, we obtain that when we set $A < 1$ and $B > 10$, we get better results.

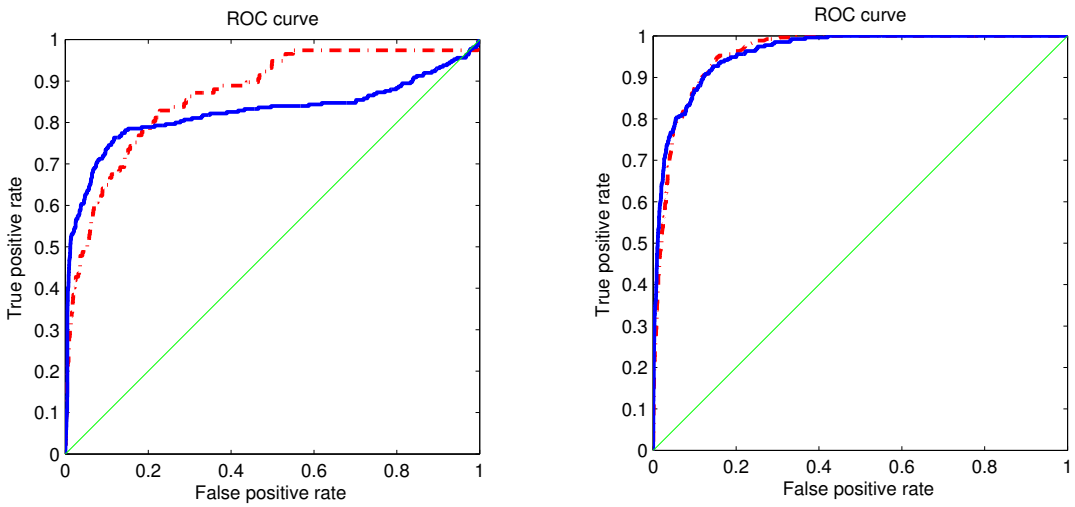
Link Prediction. We compare the performance of ML-MAP estimation and variational approaches of TF-VB and CTF-VB on both CP and Tucker tensor factorization models at different amounts of randomly unobserved elements. In these experiments, the incomplete tensor is factorized using either a CP or a Tucker model and the extracted factor matrices are used to construct the full tensor and estimate scores for missing links. We use AUC to measure the link prediction performance. The following results show the average link prediction performance of 10 independent runs in terms of AUC.

For all cases, variational approach outperforms the standard approach clearly. Table 6.5 shows the time and accuracy performances of TF-ML, TF-VB, CTF-ML and CTF-VB methods for the CP model, when $\{60, 80, 90\}$ of the data is missing. Based on these results, we can conclude two results: (i) the variational methods due to implicit self-regularization effect [149], perform better than the standard methods; (ii) coupled models outperform the single models in particular, when the percentage of missing data is high.

Moreover, we study the performance of CTF-ML and CTF-VB in terms of robustness to model order selection. As model order increases, the prediction performance of CTF-ML drops. This is as expected since CTF-ML is prone to overfitting and the increase in model order causes an increase in the number of free parameters that, in turn, enlarges penalty term in CTF-ML. On the other hand, the prediction performance of the variational approach is not very sensitive to the model order and is immune to overfitting since Bayesian approach alleviates over-fitting by integrating out all model parameters [109]. We compare the prediction performances of CTF-ML and CTF-VB methods for the CP tensor model when the component number R is equal to 2 and 20 and for different amounts of missing data, i.e., $\{40, 60, 80\}$ of the data is missing. Figure 6.7(a) and Figure 6.7(b) demonstrate that when the model order increases, the prediction performance of TF-VB approach stays almost same; however, the prediction performance of TF-ML approach declines as expected.

Table 6.5. AUC score (by *'mean* $\pm$ *std'*) and time comparison of ML-MAP approaches and the proposed VB algorithms on various data sets with CP-tensor factorization model and different proportion of missing data. Results are averaged over 10 runs.

Dataset	Algorithm		60% missing		80% missing		90% missing	
			AUC	Time(sec)	AUC	Time(sec)	AUC	Time(sec)
UCLAF	TF	ML	0.940 ± 0.004	1.69	0.867 ± 0.005	1.57	0.844 ± 0.005	1.43
		VB	$\mathbf{0.973 \pm 0.002}$	2.12	$\mathbf{0.959 \pm 0.003}$	2.04	$\mathbf{0.917 \pm 0.003}$	1.93
	CTF	ML	0.917 ± 0.003	5.14	0.892 ± 0.004	5.08	0.869 ± 0.004	4.98
		VB	$\mathbf{0.981 \pm 0.001}$	6.19	$\mathbf{0.962 \pm 0.001}$	6.16	$\mathbf{0.939 \pm 0.002}$	6.01
Digg - (Comment Prediction)	TF	ML	0.848 ± 0.003	582.54	0.829 ± 0.004	326.16	0.813 ± 0.004	167.98
		VB	$\mathbf{0.917 \pm 0.002}$	787.94	$\mathbf{0.897 \pm 0.003}$	460.24	$\mathbf{0.879 \pm 0.003}$	243.75
	CTF	ML	0.856 ± 0.002	1019.78	0.837 ± 0.003	598.22	0.824 ± 0.003	312.53
		VB	$\mathbf{0.928 \pm 0.001}$	1251.40	$\mathbf{0.913 \pm 0.002}$	735.47	$\mathbf{0.891 \pm 0.002}$	452.99
Digg - (Digg Prediction)	TF	ML	0.864 ± 0.005	277.11	0.845 ± 0.006	159.51	0.829 ± 0.006	89.54
		VB	$\mathbf{0.935 \pm 0.003}$	340.20	$\mathbf{0.917 \pm 0.003}$	221.78	$\mathbf{0.898 \pm 0.004}$	127.35
	CTF	ML	0.892 ± 0.003	473.42	0.870 ± 0.004	290.34	0.853 ± 0.004	168.51
		VB	$\mathbf{0.961 \pm 0.001}$	537.54	$\mathbf{0.947 \pm 0.001}$	354.18	$\mathbf{0.923 \pm 0.002}$	211.15



(a) CTF, 80%

(b) CTF-VB, 80%

Figure 6.7. Effect of model order on the performance of a) CTF-ML and b) CTF-VB approaches for CP model when R=2.

7. EXPERIMENTS ON DIFFERENTIALLY PRIVATE COUPLED MATRIX AND TENSOR FACTORIZATION

The aim of this chapter is to demonstrate our differentially private methods presented in Chapter 4 on different models and datasets. In particular, we begin with the differentially private CMF models with different privacy settings. Then, we show the differentially private CTF results on three different models with different notions of privacy on both synthetic and real world datasets.

7.1. Experiments on Differentially Private Coupled Matrix Factorization

In order to evaluate how our proposed methods work in practice, we conducted experiments on both synthetic and real data in two settings: we compare the performances of the methods *i)* by varying the DP budget ϵ and *ii)* by varying the number of data sites *N*.

In Bayesian estimation, the log-likelihood function can be interpreted as a measure of how well the parameter values fit the training example [111]. For the synthetic data experiments, we considered computations using the *log-likelihood (LL)* (the log of the likelihood function given in Equation (4.1)) evaluated at the usual posterior simulations of the parameters. In our recommendation tasks, we employed *Root Mean Square Error (RMSE)* as our measurement. For each partition $\{X^{(n)}\}_{n=1}^{N}$, we randomly chose 80% of the entries for training and the remaining 20% for testing. We updated the learning rate per round as $\kappa_t = \kappa_0/t^{\gamma}$ where the initial learning rate κ_0 is set specific to each dataset by searching over the set $\{0.1, 0.2, \cdots, 1\}$ that maximizes the LL results on training set in the synthetic data experiments and the RMSE results on the training set in the real data experiments. We fixed the decay rate γ to 1 for all datasets, a typical value used in the literature [2, 27, 29, 150]. The learning rate $\eta_{(t)}^{(n)}$ that provides the DP for the factor updates was computed under the constraints of Theorem 4.3 when the stepsize decreases.

Besides, the prediction accuracy depends on a number of factors that must be carefully

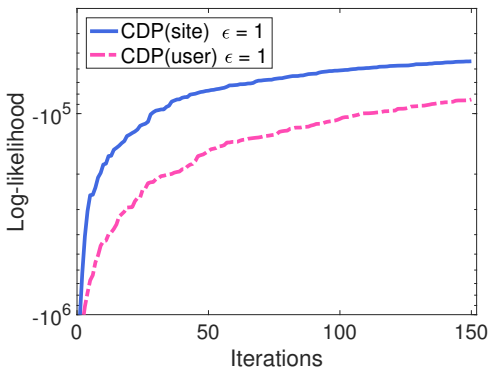
tuned to optimize the performance. For our DP methods, these factors include the number of latent factors D, the number of data passes T, the minibatch size B, the gradient clipping norm G and the privacy parameter δ. In order to choose the best parameter combination of the prediction results, we tried several values for these parameters and reported the results for the best performing ones. For all experiments, we set the rank D=5, clipping norm G=2 and δ = 10^{-3}. T and B are chosen differently for each dataset and we will specify these values in the following. All of the results in this paper are the average of 10 runs of the proposed models.

7.1.1. Synthetic Data

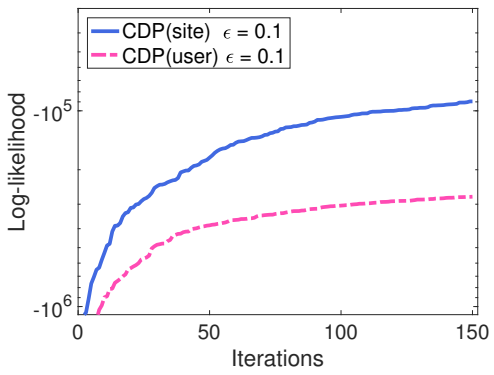
In our first set of experiments, we generated two data matrices $X^{(1)} \in \mathbb{R}^{150\times 50}$ and $X^{(2)} \in \mathbb{R}^{100\times 50}$ by using the generative model that is described in Section 4.1.2 with data sparsity = 0.2. Once the data is partitioned, we ran our algorithms CDP (site), CDP (user), LDP1 and LDP2 on this dataset for different ϵ values with mini-batch size B=1. Throughout the experiments section, for the CDP we will show the results of *site-level* privacy since it's the main proposal of this paper. However, we first compare the site-level privacy with the user-level privacy in Figure 7.1 to demonstrate the improvement by site-level privacy.

The number of data passes T is the most difficult element to determine in the algorithm as it needs to be sufficient but not too large. We tried several values in the range of $[50, 1000]$ and observed that we obtained the best results when T is between 100 and 300. Figure 7.2 shows the effect of the number of data passes to the prediction accuracy when $\epsilon = 0.1$. We ran all the methods in a fixed number of data passes. That is 150 passes for CDP (site) and 120 passes for LDP1 and LDP2 because we observe that the prediction accuracy peaks at these values.

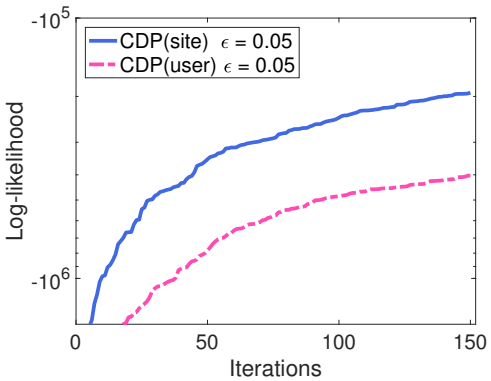
In order to choose the best hyperparameter combination for the prediction results, we tried several values and chose the best performing α and β. We specifically chose α and β values over the set $\{0.01, 0.1, 1, 10\}$ and present the comparison results of four sample settings in Table 7.4. Evidently, we obtain the best result when we set the shape hyperparameter α = 0.1



(a) $\epsilon = 1$



(b) $\epsilon = 0.1$



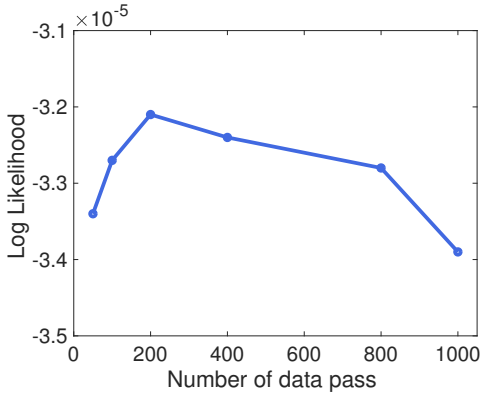
(c) $\epsilon = 0.05$

Figure 7.1. Comparison of *CDP (site)* and *CDP (user)* in terms of log-likelihood computed on $X^{(1)}$ when a) $\epsilon = 1$, b) $\epsilon = 0.1$ and c) $\epsilon = 0.05$.

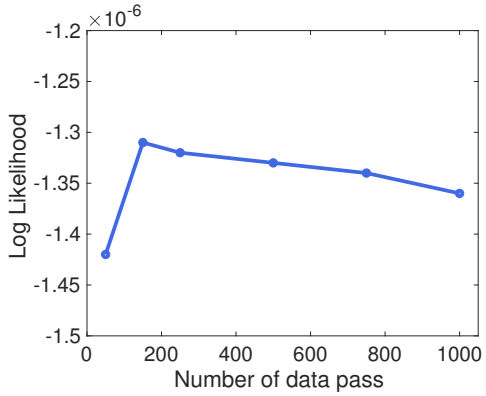
and the scale hyperparameter $\beta = 0.01$; so we use these values in the following experiments. We observed that the performance is fairly robust to the changes in the hyperparameters.

We learned the estimates $\hat{X}^{(1)}$ and $\hat{X}^{(2)}$ of the original matrices $X^{(1)}$ and $X^{(2)}$, then computed the predictive log-likelihood on the recovery of $X^{(1)}$ and $X^{(2)}$. Figure 7.3 shows the log-likelihood values of our methods on the recovery of $X^{(1)}$, under varying levels of differential privacy protection and non-private (NP) case.

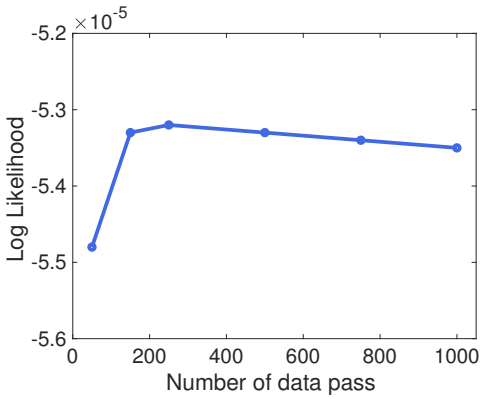
We draw two conclusions from these results: i) smaller ϵ values indicate stronger privacy guarantee, and we obtain lower prediction accuracy in this case. Figure 7.3(a), 7.3(b) and 7.3(c) clearly show that as the value of ϵ increases so does the prediction accuracy of the system for each of the three methods. ii) LDP provides a stronger form of privacy than CDP



(a) CDP (site)



(b) LDP1



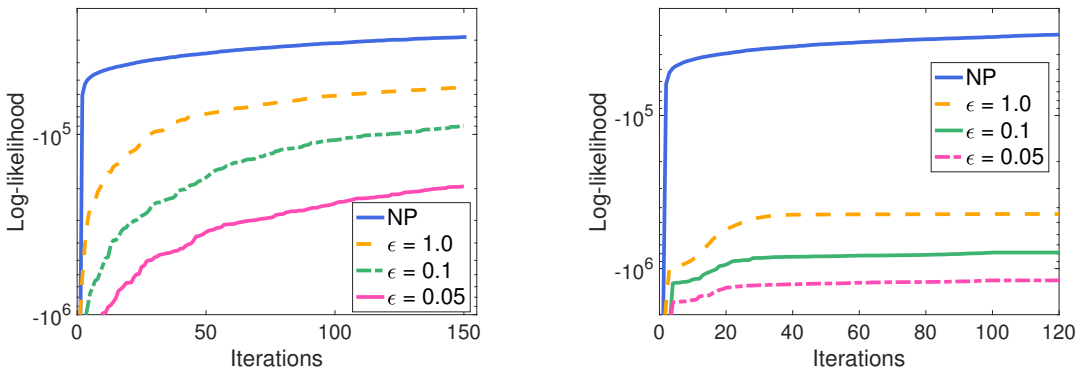
(c) LDP2

Figure 7.2. Predictive log-likelihood results computed for $X^{(1)}$ when $\epsilon = 0.1$ for different number of data passes for (a) CDP, (b) LDP1 and (c) LDP2.

Table 7.1. Log-likelihood scores of the proposed methods for different ϵ values and different hyperparameter settings when there are 2 data sites.

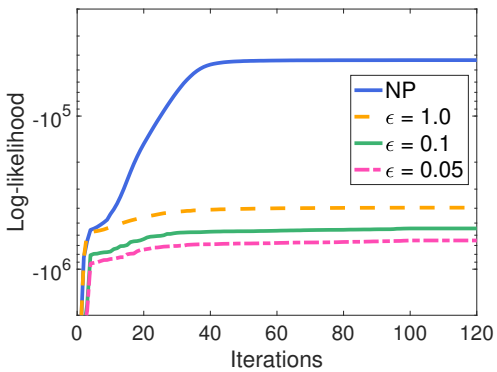
	CDP (site)			LDP1		
Hyperparameters	$\epsilon = 0.05$	$\epsilon = 0.1$	NP	$\epsilon = 0.05$	$\epsilon = 0.1$	NP
α **= 0.1,** β **= 0.01**	**-5.12** $\times 10^{-5}$	**-3.21** $\times 10^{-5}$	**-5.40** $\times 10^{-4}$	**-2.30** $\times 10^{-6}$	**-1.33** $\times 10^{-6}$	**-5.40** $\times 10^{-4}$
$\alpha = 10$, $\beta = 0.01$	-5.18 $\times 10^{-5}$	-3.25 $\times 10^{-5}$	-5.45 $\times 10^{-4}$	-2.37 $\times 10^{-6}$	-1.38 $\times 10^{-6}$	-5.45 $\times 10^{-4}$
$\alpha = 0.1$, $\beta = 10$	-5.23 $\times 10^{-5}$	-3.29 $\times 10^{-5}$	-5.48 $\times 10^{-4}$	-2.39 $\times 10^{-6}$	-1.40 $\times 10^{-6}$	-5.48 $\times 10^{-4}$
$\alpha = 10$, $\beta = 10$	-5.15 $\times 10^{-5}$	-3.24 $\times 10^{-5}$	-5.41 $\times 10^{-4}$	-2.35 $\times 10^{-6}$	-1.36 $\times 10^{-6}$	-5.41 $\times 10^{-4}$

so CDP performs better. However, our second locally private model *LDP2* is more flexible than LDP1. In the LDP2 model, each data site uses the original data on its own site (without

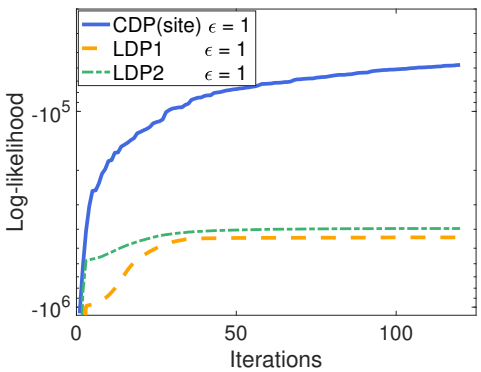


(a) CDP (site)

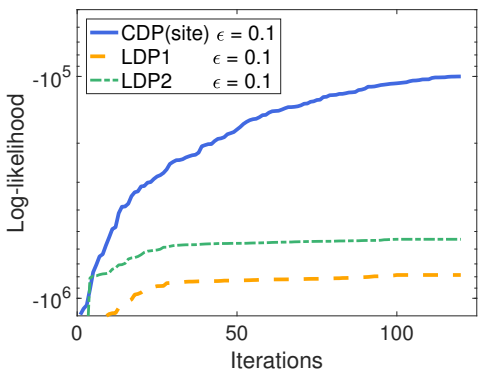
(b) LDP1



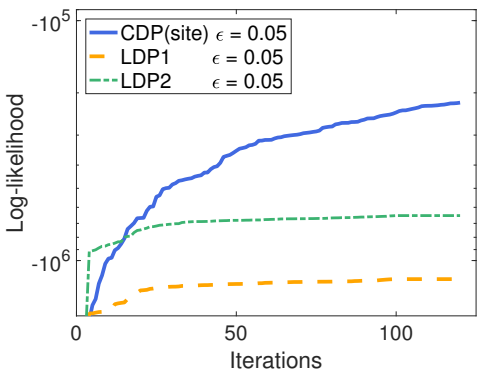
(c) LDP2



(d) $\epsilon = 1$



(e) $\epsilon = 0.1$

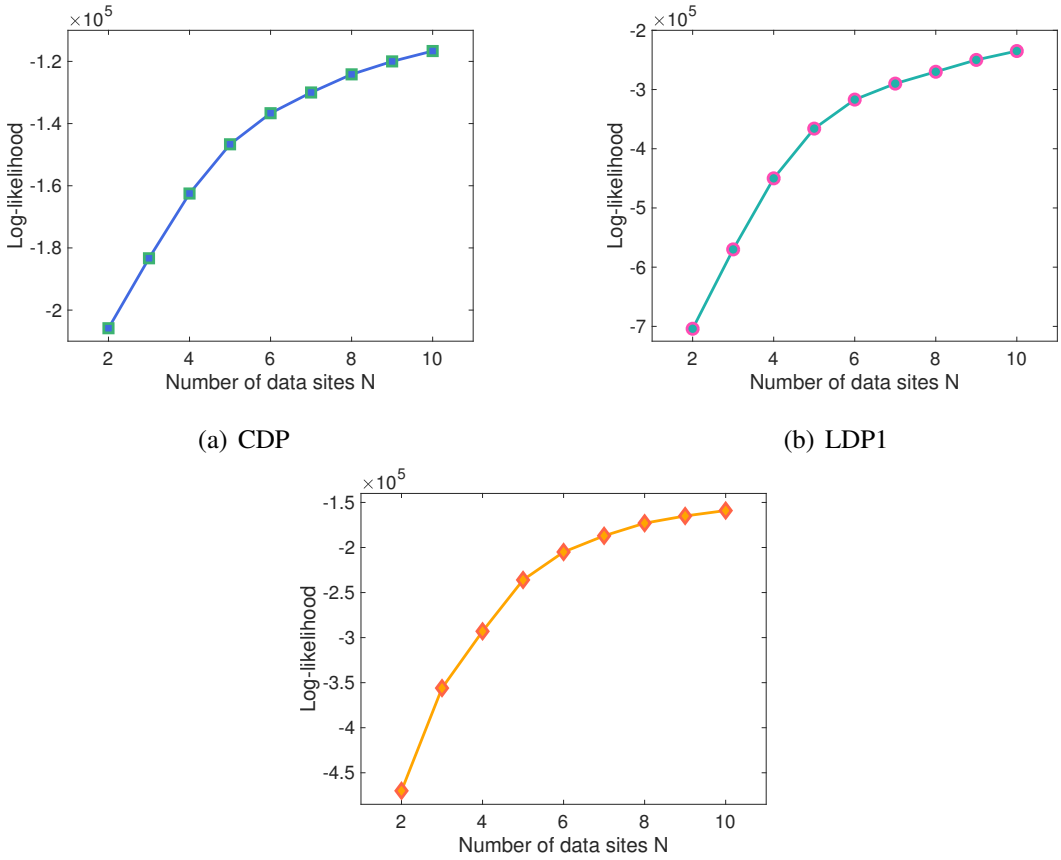


(f) $\epsilon = 0.05$

Figure 7.3. Comparison of the log-likelihood results computed for $X^{(1)}$ of a) CDP, b) LDP1 and c) LDP2 on the synthetic data generated with the Poisson-NMF model for $\epsilon = \{0.05, 0.1, 1\}$. CDP, LDP1 and LDP2 comparison when d) $\epsilon = 1$, e) $\epsilon = 0.1$ and f) $\epsilon = 0.05$.

privacy protection) and the privatized data views from the other sites; then decomposes them collectively. Hence, LDP2 compensates the performance loss that is based on the definition of local privacy. Figure 7.3(d), 7.3(e) and 7.3(f) demonstrate our claims. We first treat CDP

and LDP2 models and see that there is a little performance loss for LDP2, compared to the CDP model. Then, we compare the performances of two locally differentially private models LDP1 and LDP2 and see that LDP2 outperforms LDP1 especially when ϵ is smaller.



(a) CDP

(b) LDP1

(c) LDP2

Figure 7.4. Predictive log-likelihood results computed for $X^{(1)}$ when $\epsilon = 0.1$ for different number of data sites N=$\{2, \cdots, 10\}$ for (a) CDP, (b) LDP1 and (c) LDP2.

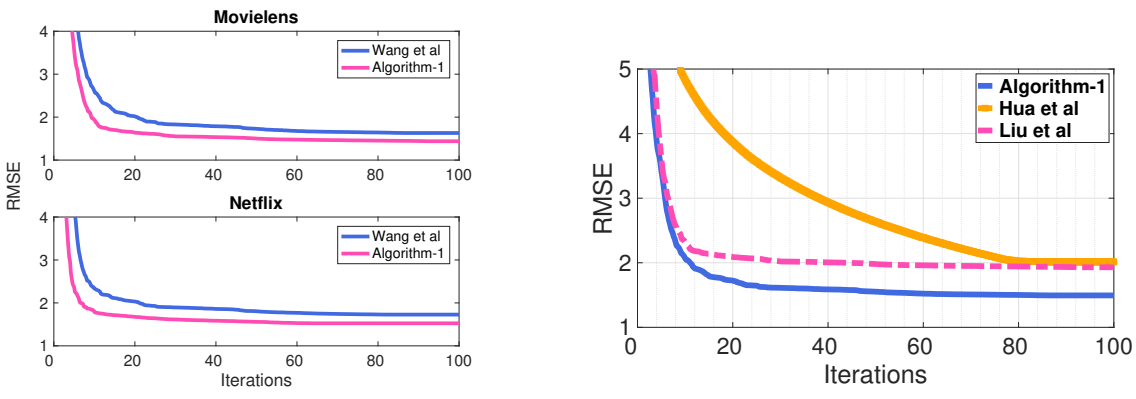
In our second set of experiments, we generated a dataset $\{X^{(i)}\}_{i=1}^{10} \in \mathbb{R}^{100\times 50}$ by using the same generative model, and distributed $\{X^{(i)}\}_{i=1}^{10}$ to different data sites. Then, we observed the predictive log-likelihood value on the recovery of $X^{(1)}$ as the number of data sites increases from 2 to 10. It means we report the results for one data partition, which is $X^{(1)}$, as we increase the number of auxiliary sites for collective factorization. Figure 7.4 demonstrates that when the number of data sites integrated to the system increases, we obtain higher predictive log-likelihood results for all three methods. The reason for the increase in the log-likelihood is the increase in the information sharing. When one data site receives more

information from the other data sites, the prediction accuracy improves.

7.1.2. Real Data

We conducted matrix completion experiments for recommendation on MovieLens [1] and Netflix datasets [2]. Movielens data contains approximately 1M movie ratings from 3952 movies made by 6040 MovieLens users and the ratings in this dataset are integer scores ranging from 1 to 5. Netflix data consists of 99M ratings at a scale of 1 to 5 made by ∼480K users for ∼18K movies. Both datasets contain less than 1% of all possible entries. In the real data experiments, data instances, corresponding to the rows of a matrix, were distributed across different sites. For each experiment in this section, we used $B = 100$, $\kappa_0 = 0.1$ for Movielens and $B = 1000$, $\kappa_0 = 0.3$ for Netflix. We ran the algorithm for 100 passes and the algorithm took under 60 passes for Movielens and 80 passes for Netflix to converge.

Comparison with existing method . First of all, we compared our proposed algorithm with the DP-SGLD method proposed in [27] (We call it *Wang et al* in the experiments). As we mentioned in Section 4.1.3, Li *et al* increases the upper bound for the stepsize by a factor of $T^{1/3}\log(T/\delta)$. The practical improvement can be observed in Figure 7.5(a).



(a) (b)

Figure 7.5. a) DP-SGLD comparison results in terms of RMSE on Movielens and Netflix datasets when $\epsilon = 0.1$. b) Comparison results of [1] and [2] with Algorithm-1 (*CDP(user)*) on Movielens dataset when $\epsilon = 0.1$.

[1] www.grouplens.org/ node/73
[2] https://github.com/BIDData/BIDMach/wiki/Benchmarks

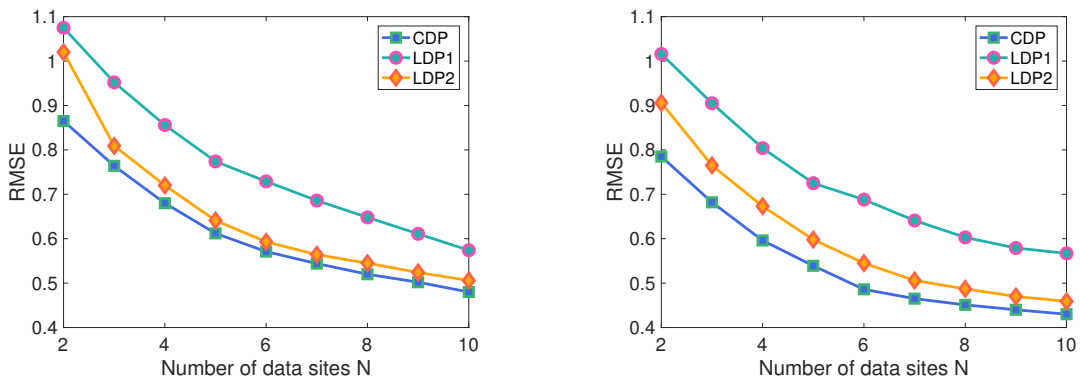
Then, we compared our method to the most related algorithms proposed by Hua *et al* [1] and Liu *et al* [2]. Note that these algorithms can only model data in *one single data site* and considers *user-level* privacy, so we use the *whole* data that is stored in one data site and chose *CDP (user)* as our privacy notion to compare. We set B=1, δ = 0.01 and ran all the methods on Movielens dataset with varying ϵ. Figure 7.5(b) shows that there is a slight difference between [1] and [2] when ϵ = 0.1 and *Algorithm-1* is noticeably performs better than the others. The main reason of the difference is the composition methods of the algorithms. Both [1] and [2] use Advanced Composition (Theorem 3.20 in [89]) and we use the moments accountant based SGLD. In addition to the accuracy, we compared the computation times. Both our approach and [2] use the SGLD based approach, so each iteration almost takes the same time: $\sim$30 sec. The algorithm of [1] is a bit faster than the Bayesian approaches since it is based on deterministic SGD; it takes $\sim$26 sec per iteration.

Method Comparisons . In the first set of real data experiments, we stored the Movielens dataset in two matrices as $X^{(1)} \in \mathbb{R}^{2000\times6040}$, $X^{(2)} \in \mathbb{R}^{1952\times6040}$, and Netflix dataset in two matrices as $X^{(1)} \in \mathbb{R}^{240K\times17770}$, $X^{(2)} \in \mathbb{R}^{240K\times17770}$. Then, we generated the latent factors by using the generative model described in Section 4.1.2. Table 7.2 reports the performances of the proposed methods CDP (site), LDP1 and LDP2 in terms of RMSE between the real data ($X^{(1)}$, $X^{(2)}$) and the estimated data ($\hat{X}^{(1)}$, $\hat{X}^{(2)}$). These results justify the theoretical claims and the empirical results on synthetic data: when the privacy protection of the system is increased by decreasing ϵ, we obtain lower prediction accuracy.

Furthermore, we can obtain from these results that CDP model outperforms the LDP models since local DP provides a stronger privacy protection. However, our main goal was to increase the prediction accuracy; that is why we proposed the LDP2 model. The results given in Table 7.2 demonstrate that LDP2 displays very close prediction performance to CDP. LDP2 also assures better prediction accuracy than LDP1 while preserving the same amount of *site-level* privacy with LDP1. In the *non-private* (NP) case of the algorithms, CDP and LDP1 methods correspond to the same model since we do not use the privatization mechanism in *NP* case in Figure 4.5.

Table 7.2. RMSE scores of the proposed methods for different ϵ values when there are 2 data sites.

ϵ	Movielens						Netflix					
	CDP (user)		LDP1		LDP2		CDP (site)		LDP1		LDP2	
	$X^{(1)}$	$X^{(2)}$	$X^{(1)}$	$X^{(2)}$	$X^{(1)}$	$X^{(2)}$	$X^{(1)}$	$X^{(2)}$	$X^{(1)}$	$X^{(2)}$	$X^{(1)}$	$X^{(2)}$
NP	0.831	0.719	0.831	0.719	0.861	0.717	0.775	0.802	0.775	0.802	0.799	0.827
1	0.930	0.737	1.032	0.914	1.026	0.882	0.902	0.933	0.945	0.973	0.950	0.985
0.1	1.419	1.224	1.826	1.511	1.695	1.362	1.354	1.419	1.598	1.654	1.405	1.468
0.05	1.866	1.631	2.164	1.992	1.992	1.831	1.683	1.812	1.905	1.972	1.751	1.835



(a) Movielens (b) Netflix

Figure 7.6. The performance comparison of CDP (site), LDP1 and LDP2 on a) Movielens b) Netflix when ϵ = 1 and $N = \{2, \cdots, 10\}$.

Effect of number of data sites . As the final set of experiments, we divided the datasets into 10 partitions: $\{X^{(i)}\}_{i=1}^{9} \in \mathbb{R}^{400\times 6040}$, $X^{(10)} \in \mathbb{R}^{352\times 6040}$ for Movielens and $\{X^{(i)}\}_{i=1}^{10} \in \mathbb{R}^{24K\times 17770}$, for Netflix; then distributed them to different data sites. We monitored the RMSE between $X^{(1)}$ and $\hat{X}^{(1)}$ when we increase the number of data sites N from 2 to 10. Table 7.3 illustrates the behavior of our methods when we increase the number of data sites in the system. The performances of all three methods improve, as the number of additional data sources increases. In this table, bold numbers emphasize that the difference in the prediction accuracy of LDP1 and LDP2 increases with the number of data sites integrated to the system. Figure 7.6(a) and 7.6(b) show this behavior clearly.

Table 7.3. RMSE scores for $X^{(1)}$ when $\epsilon = 1$ and the number of data sites N in the system is varied from 2 to 10.

	Method	Number of data sites (N)								
		2	3	4	5	6	7	8	9	10
MovieLens	CDP (site)	0.865	0.764	0.680	0.612	0.571	0.544	0.520	0.502	0.480
	LDP1	1.075	0.952	0.856	0.774	0.729	0.686	0.648	0.611	0.574
	LDP2	1.020	0.809	0.720	0.641	0.593	0.564	0.545	0.524	0.506
Netflix	CDP (site)	0.785	0.682	0.596	0.539	0.486	0.465	0.451	0.440	0.430
	LDP1	1.016	0.905	0.804	0.725	0.688	0.641	0.603	0.579	0.567
	LDP2	0.906	0.765	0.673	0.598	0.545	0.506	0.487	0.470	0.459

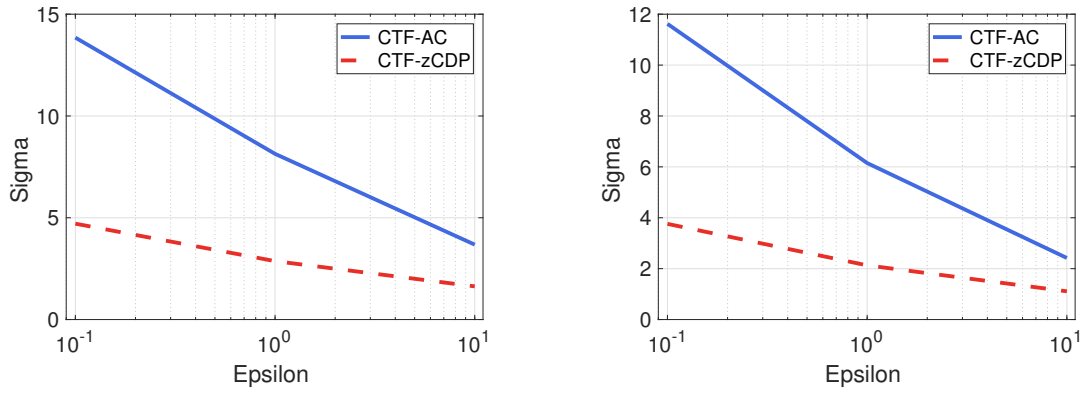
7.2. Experiments on Differentially Private Coupled Tensor Factorization

This section presents the experiment results on both synthetic and real data in three settings: we compared the prediction accuracy by altering *i)* the DP budget ϵ, *ii)* the number of data sites *N* incorporated to the system and *ii)* the number of shared factors between data sites. For the synthetic data experiments, we considered computations using the log-likelihood (LL) evaluated at the usual posterior simulations of the parameters. In our recommendation tasks, we employed Root Mean Square Error (RMSE) as our metric. For each partition $\{X^{(v)}\}_{v=1}^{N}$, we used 80% of the entries for training and the 20% for testing. We updated learning rate per round as $\kappa_t = \kappa_0/t^{\varrho}$ where the decay rate $\varrho \in (0.5, 1]$ and the initial learning rate $\kappa_0 \in (0, 1]$. The learning rate η_t that provides the DP for the factor updates was computed by Equation (4.27) and the parameter α was set to 0.9 to determine it. For each dataset, we tried several values for these parameters and reported the results for the best performing ones. All of the results are the average of 10 runs of the proposed models.

7.2.1. Synthetic Data

First, we generated two matrices $X^{(1)} \in \mathbb{R}^{100\times 50}$ and $X^{(2)} \in \mathbb{R}^{100\times 50}$ by using the model described in Section 4.2.1. Then, we ran our algorithm for different ϵ values and learned the estimates of $\hat{X}^{(1)}$ and $\hat{X}^{(2)}$ of the original matrices $X^{(1)}$ and $X^{(2)}$, then computed the predictive log-likelihood on the recovery of them. During this section, we use *user-level* privacy (for comparison purposes) and *zCDP* (to obtain better prediction results) as our baseline DP notion.

We start with the comparison of the composition methods. Figure 7.7 compares the usage of the advanced composition (AC) and the zCDP composition analysis on synthetic data and Movielens respectively. This figure shows that when we use the zCDP, we get a much tighter estimation of the privacy loss. For example, when ϵ = 1, the σ = 8.14 for AC and σ = 2.87 for zCDP on synthetic data. We can obtain from this figure that we get a much lower noise by using the zCDP for a fixed the privacy loss ϵ. Therefore, for our CTF models with a total privacy budget fixed to ϵ, the amount of noise added is smaller for zCDP, and the prediction accuracy is higher. We have already known that zCDP is supeior in theory, but



(a) Synthetic (b) Movielens

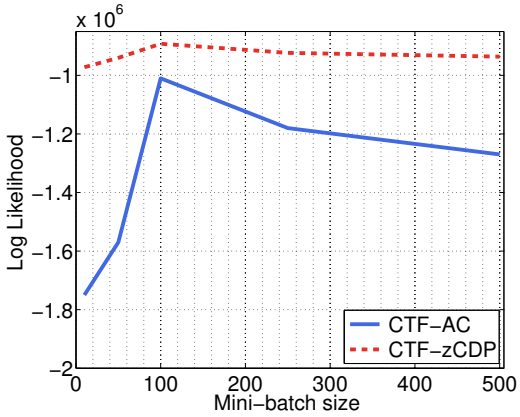
Figure 7.7. The σ value as a function of ϵ.

these results demonstrate that it also works well in practice.

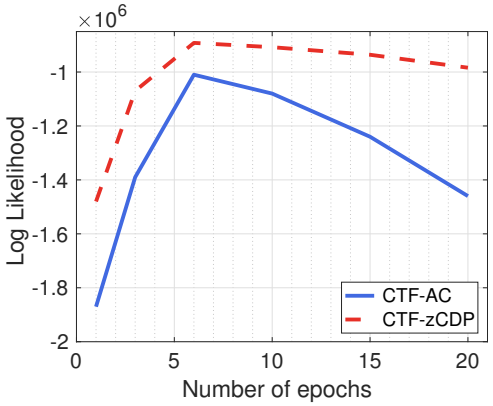
Afterwards, we compare performances of our algorithms' parameters for different values to choose the best parameter and hyperparameter combinations for CTF-AC and CTF-zCDP methods.

Effect of the parameters. The prediction accuracy depends on a number of factors that must be carefully tuned to optimize the performance. For our DP methods, these factors consist of the latent factor dimension K, the number of data passes T, the minibatch size B, the gradient clipping norm G and the privacy parameter δ. To select the best parameter combination of the prediction results, we tried several values for these parameters. We control a parameter individually by fixing the rest as constant. For the synthetic data experiments, we set the parameters as follows: K = 5, B = 100, G = 2 and δ = 10^{-3} when the privacy budget ϵ is set to 1. As we defined in Section 4.2.2, the sampling ratio of each minibatch is $\gamma = B/C_v$. The number of iterations T is equal to E/γ where E is the number of epochs. We choose E = 6 for the synthetic data experiments. The complete results are presented in Figure 7.8.

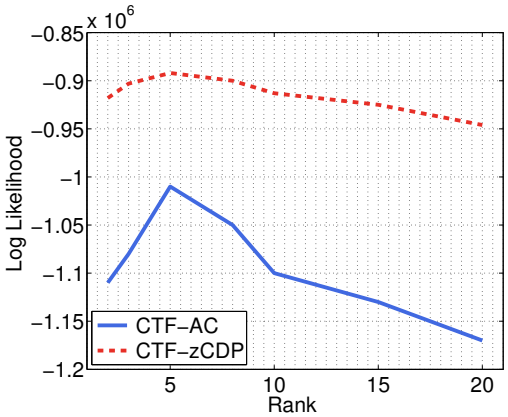
We first monitor the effect of the minibatch size. In DP settings, choosing smaller minibatch size leads running more epochs, however, the added noise has a smaller relative effect for a larger minibatch. Figure 7.8(a) shows that relatively larger minibatch sizes give better results. Empirically, we obtain the best accuracy when B is around 100 (so the γ =



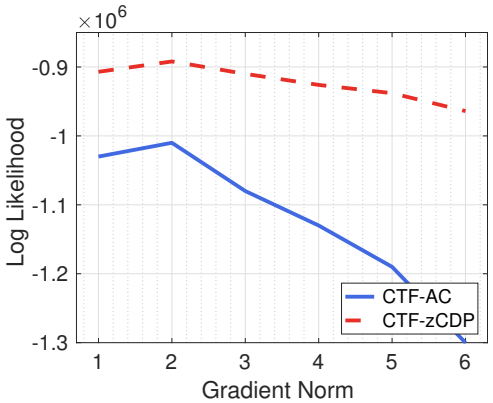
(a) Minibatch size



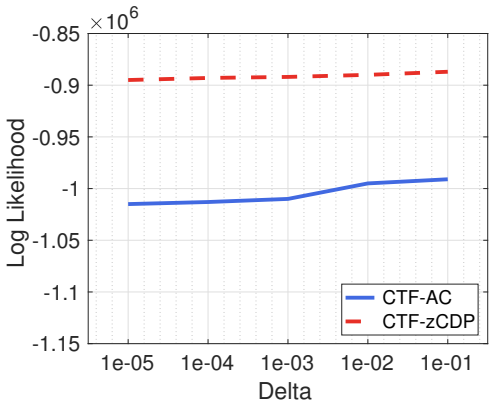
(b) Number of epochs



(c) Number of latent factors



(d) Gradient clipping norm (G)



(e) Privacy parameter (δ)

Figure 7.8. The effect of the model parameters on synthetic dataset when $\epsilon = 1$ under user-level DP.

100/500 = 0.02).

The number of iterations T needs to be sufficient, but not too large. The privacy cost

in zCDP increases in proportion to $\sqrt{T}$ and it is more tolerable than DP. For the number of epochs E, we tried several values in the range of $[1, 20]$ and observed that we obtained the best results when E is between 5 and 10. We choose E = 6, so the number of iterations T becomes equal to 6/0.02 = 300. (For the rest of the experiments, we will directly report the number of iterations T.)

In matrix factorization models, using more latent factors is often preferable and increases the prediction accuracy of the predictive model. Due to the increase in the sensitivity of the gradient, more noise is added at each update when we use more latent factors in differentially private models. However, increasing the number of factors does not always decrease accuracy since some tensors have many underlying structures and represented by several latent factors. So we run experiments with various rank values and present the results in Figure 7.8(c). We can see that accuracy is very close to a latent factor number in the range of $[2, 10]$ and peaks at 5.

Tuning the gradient clipping threshold depends on the details of the model. When the threshold is quite small, the clipped gradient could be too different from the correct gradient and points to a contrasting direction. Besides, when we increase the threshold, we add a large amount of noise to the gradients. In our experiments, we tried G values in the range of $\{1, 6\}$. Figure 7.8(d) shows that our model peaks at G = 2 and tolerable up to G = 4, then the accuracy decreases marginally.

Moreover, we monitor the accuracy for various δ values for a fixed ϵ in Figure 7.8(e). We draw the most accurate result for each δ where $\delta = \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and $\epsilon = 1$. Figure 7.8(e) shows that there is only a slight difference for different δ. However, different values of ϵ for a fixed δ may dramatically influence the accuracy (See Figure 7.9(b)).

Finally, we determine the gamma distribution hyperparameters a_0 and b_0. We specifically chose a_0 and b_0 values over the set $\{0.01, 0.1, 1, 10\}$ and present the comparison results of four sample settings in Table 7.4. Evidently, we obtain the best result when we set the shape hyperparameter a_0 = 1 and scale hyperparameter b_0 = 0.01. We observed that the performance is fairly robust to the changes in the hyperparameters for all privacy levels.

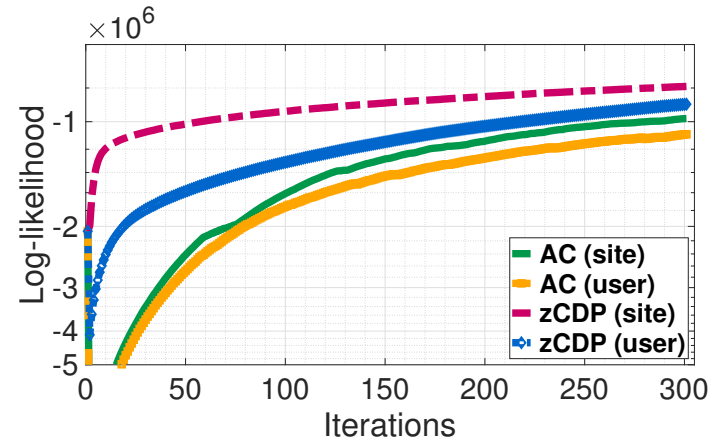
Table 7.4. Log-likelihood scores for $X^{(1)}$ of the proposed method for different ϵ values (for *CTF-zCDP (user-level)* method) and different hyperparameter settings when N = 2.

Hyperparameters	NP	ϵ = 10	ϵ = 1	ϵ = 0.1
$\mathbf{a_0 = 1,\ b_0 = 0.01}$	$\mathbf{-3.21 \times 10^5}$	$\mathbf{-5.12 \times 10^5}$	$\mathbf{-8.92 \times 10^5}$	$\mathbf{-1.18 \times 10^6}$
$a_0 = 10, b_0 = 0.01$	-3.25×10^5	-5.18×10^5	-9.05×10^5	-1.35×10^6
$a_0 = 0.1, b_0 = 10$	-3.29×10^5	-5.23×10^5	-9.11×10^5	-1.32×10^6
$a_0 = 10, b_0 = 10$	-3.24×10^5	-5.15×10^5	-8.97×10^5	-1.24×10^6

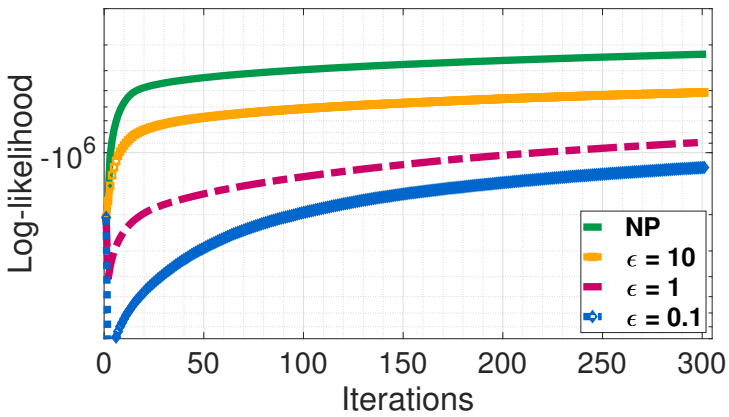
Prediction Accuracy. After tuning these parameters to optimize prediction performance, we compared the prediction accuracy of various combination of privacy notions defined in Section 4.2.2. These are the site-level DP with AC and zCDP compositions; and the user-level DP with AC and zCDP compositions. First, we compare the performances of our methods in Figure 7.9(a). Then we demonstrate the performance of our baseline method, *CTF-zCDP (user-level)* for different level of privacy protection in Figure 7.9(b).

We obtain two outcomes from these experiments: i) as mentioned in Section 4.2.2, zCDP composition provides a compact bound on the differential privacy budget compared to the AC theorem. Therefore, for our CTF models with a total privacy budget fixed to ϵ, the amount of noise added is smaller for zCDP, and the test accuracy is higher. ii) The smaller ϵ values indicates stronger privacy guarantee and we get lower prediction accuracy in this case. We can clearly see that when the value of ϵ increases, the prediction accuracy of the system increases too.

Up to this point, we showed the effect of privacy parameters on accuracy. Now, we measure the effect of the number of data sites *N* that is incorporated into the system. So, we generated a dataset $\{X^{(v)}\}_{v=1}^{10} \in \mathbb{R}^{100\times 50}$ by using the same generative model and distribute them to different data-sites. We monitored the LL score for *the data of site 1* which is $X^{(1)}$ as the number of data sites increase from 1 to 10 (note that the LL score is measured on a fixed data for all values of N) and report the results for $N = \{1, 2, 5, 10\}$. We illustrate the results on synthetic dataset for ϵ = 1 in Figure 7.10 and report the results for all privacy levels



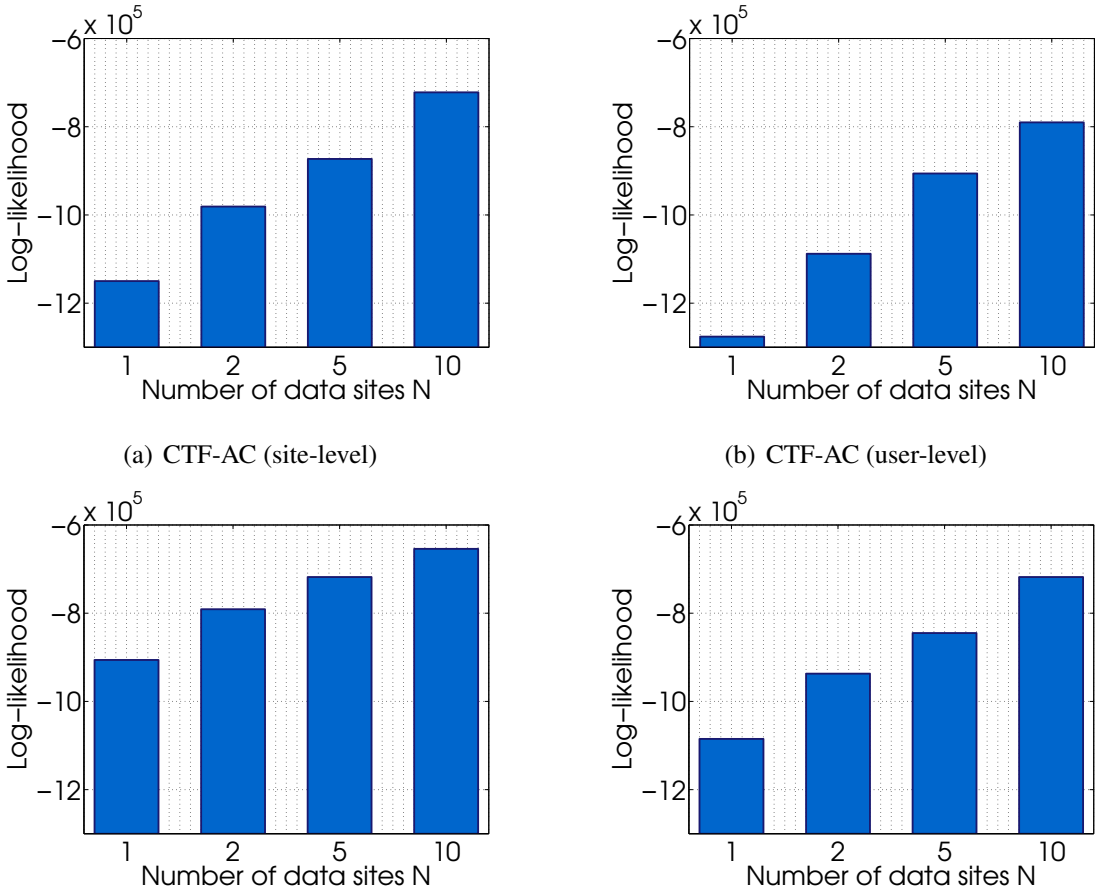
(a) Accuracy of various methods



(b) Accuracy of various ϵ values

Figure 7.9. Comparison of the prediction results (in terms of LL) for (a) different privacy notions for $\epsilon = 1$ and (b) different privacy budgets and non-private (NP) case for *CTF-zCDP (user-level)*.

in Table 7.5. They demonstrate that when the number of data sites integrated to the system increases, we obtain higher log-likelihood scores. The reason of the increase in the LL is the information sharing. When one data site gets more information from the other data sites, the prediction accuracy improves. The results that are obtained for all privacy levels and all types of privacy notions are presented in Table 7.5 and they clearly support this claim. These results also compares the performances of our methods and different privacy levels.



(a) CTF-AC (site-level)

(b) CTF-AC (user-level)

(c) CTF-zCDP (site-level)

(d) CTF-zCDP (user-level)

Figure 7.10. LL scores for $X^{(1)}$ when the number of data sites for $N = \{1, 2, 5, 10\}$ and $\epsilon = 1$.

7.2.2. Real Data

We designed tensor completion experiments for recommendation on MovieLens [3] and UCLAF datasets; and for tag prediction on Flickr dataset. Movielens data contains approximately 1M movie ratings from 3952 movies made by 6040 MovieLens users and the ratings in this dataset are integer scores ranging from 1 to 5. UCLAF includes 164 users, 168 locations, 14 location features and 5 types of activities. The scores in user-activity-location data indicate the frequency of a user i_1, visiting location i_2 and doing activity i_3 there, and ranges from 1 to 5. The collected data also includes locations' features from the points of interest database. The third dataset we consider is crawled from *Flickr* by using the social network connectors (Flickr API) [4] . This dataset consists of 2866 users, 60339 tags, 46733 images and feature vector of 1024 dimensional visual features. The

[3] www.grouplens.org/node/73

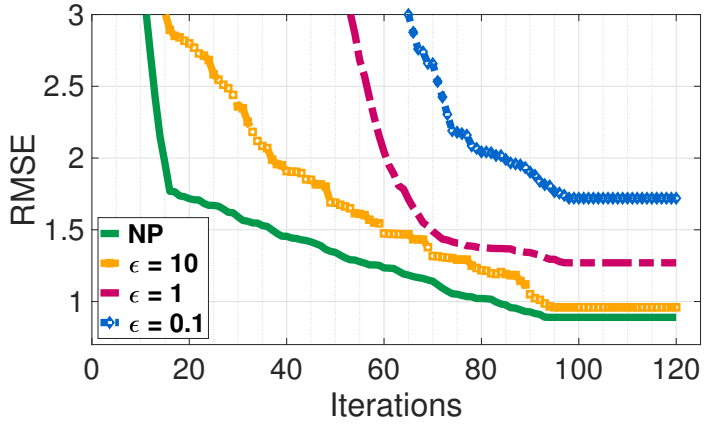
[4] www.flickr.com/services/api/

Table 7.5. LL scores for $X^{(1)}$ when the number of data sites for $N = \{1, 2, 5, 10\}$ and $\epsilon = \{10, 1, 0.1\}$ for various privacy notions. (All results in the table are divided to 10^5 for the sake of clarity.)

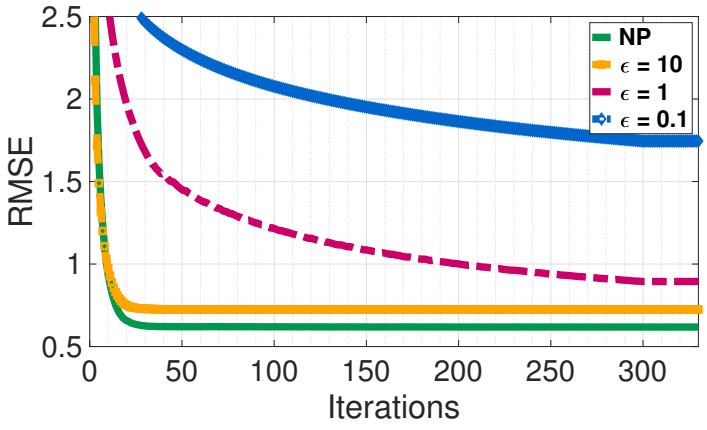
		N = 1	N = 2	N = 5	N = 10
Method	NP	-3.37	-3.21	-3.06	-2.85
CTF-AC (site-level)	ϵ = 10	-6.65	-5.62	-4.86	-4.02
	ϵ = 1	-11.50	-9.81	-8.73	-7.22
	ϵ = 0.1	-14.42	-12.38	-11.04	-9.60
CTF-AC (user-level)	ϵ = 10	-7.98	-6.53	-5.74	-4.65
	ϵ = 1	-12.76	-10.88	-9.06	-7.90
	ϵ = 0.1	-15.31	-13.24	-11.87	-10.13
CTF-zCDP (site-level)	ϵ = 10	-5.37	-4.79	-4.31	-3.62
	ϵ = 1	-9.06	-7.91	-7.18	-6.54
	ϵ = 0.1	-12.05	-10.48	-9.14	-8.30
CTF-zCDP (user-level)	ϵ = 10	-6.13	-5.12	-4.59	-3.95
	ϵ = 1	-10.32	-8.92	-8.04	-6.83
	ϵ = 0.1	-13.25	-11.80	-10.26	-9.09

scores indicate the user-item-tag relations that result in binary tensor. First, we divided the datasets into 10 partitions: $\{X^{(v)}\}_{v=1}^{9} \in \mathbb{R}^{400\times6040}$, $X^{(10)} \in \mathbb{R}^{352\times6040}$ for Movielens, $\{X^{(v)}\}_{v=1}^{9} \in \mathbb{R}^{16\times168\times5}$, $X^{(10)} \in \mathbb{R}^{20\times168\times5}$ for UCLAF and $\{X^{(v)}\}_{v=1}^{10} \in \mathbb{R}^{286\times60339\times46733}$for Flickr; then distributed them to different data-sites. Moreover, we used an additional data-site that stores *location-feature* matrix for UCLAF and *location-item* matrix for Flickr. We used this additional matrix $X^{(11)}$ for the *Model-3*. Finally, we generated the latent factors $\{\Theta_d\}_{d=1}^{D}$ by using the generative model described in Section 5.2.

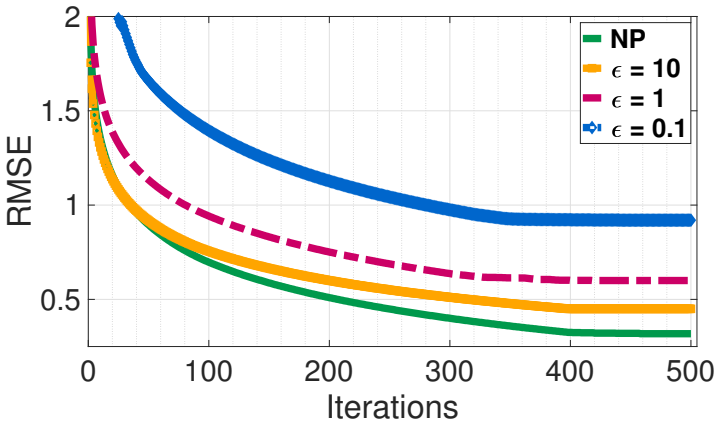
In the first set of experiments, we investigate the *influence of privacy loss on accuracy*. To begin with, we determine the number of latent factors K, the number of data passes



(a) Movielens



(b) UCLAF



(c) Flickr

Figure 7.11. Comparison of the prediction results (in terms of RMSE) on (a) Movielens, (b) UCLAF (Model-2) and (c) Flickr (Model-2) for $\epsilon = \{10, 1, 0.1\}$ and $N = 2$.

T, the minibatch size B, the gradient clipping norm G, the privacy parameter δ and the hyperparameters for all datasets to get the best results. The initial learning rates were set to $\kappa_0 = \{0.1, 0.3, 0.08\}$, $\varrho = \{1, 0.9, 0.9\}$, the latent factors were set to $K = \{5, 3, 5\}$, the

minibatch sizes were set to $B = \{80, 4, 40\}$ and the gradient clipping norm is chosen to be $G = \{3, 3, 2\}$ for Movielens, UCLAF and Flickr respectively. We ran the algorithm for 120 passes for Movielens, 300 passes for UCLAF and 400 passes for Flickr. Finally, we choose $a_0 = 1$, $b_0 = 0.01$ and $\delta = 10^{-4}$ for all datasets.

Figure 7.11(a), 7.11(b) and 7.11(c) report the performances of the proposed algorithm between the original data $X^{(1)}$ and the estimated data $\hat{X}^{(1)}$ in terms of RMSE on all datasets. These results justify the theoretical claims and the empirical results on synthetic data that *lower* prediction accuracy is obtained when the privacy protection is increased by decreasing ϵ.

Comparison with existing method. We are able to compare our baseline method *CTF-zCDP (user-level)* to the most related algorithms proposed by Hua *et al* [1] and Liu *et al* [2]. Hua *et al* propose a differentially private SGD algorithm for matrix factorization (MF) where each user is accepted as a party; so the rows of the user factor are kept as private. Then, the data aggregator (recommender) uses the item factor to share preferences between users using a Laplace mechanism. The recommender has access to the intermediate results from all users after each iteration and it can easily eliminate the noise vectors' effect. So, the noise vector is needed to be decomposed and determined after each iteration using a fairly complicated procedure. Liu *et al* exploit the fact which is proved in [27] and propose a DP-collaborative filtering approach. They focus on the *single-party* setting where *no information sharing between data sites* is considered. Here, we propose a simple MCMC algorithm for multiparty prediction systems where each party can hold data of a different modality. Our method allows information sharing while preserving both the *user-level* DP and the *site-level* DP. Note that these algorithms can only model data in *one single data site*, so we use the *whole* data that is stored in one data site. We consider the *user-level* DP for a fair comparison in same setting.

We set B=1, δ = 0.01 and ran all the methods on Movielens and Flickr datasets with varying ϵ. Figure 7.12 shows that there is a slight difference between Hua *et al* [1] and Liu *et al* [2] on Movielens dataset when ϵ = 0.1 and *CTF-zCDP (user-level)* is noticeably performs better than the others. Table 7.6 demonstrates that same behaviour is observed for all privacy

levels on both datasets. The main reason of the difference is the different privacy notions of the methods. Both [1] and [2] use Advanced Composition. Besides, we compared the

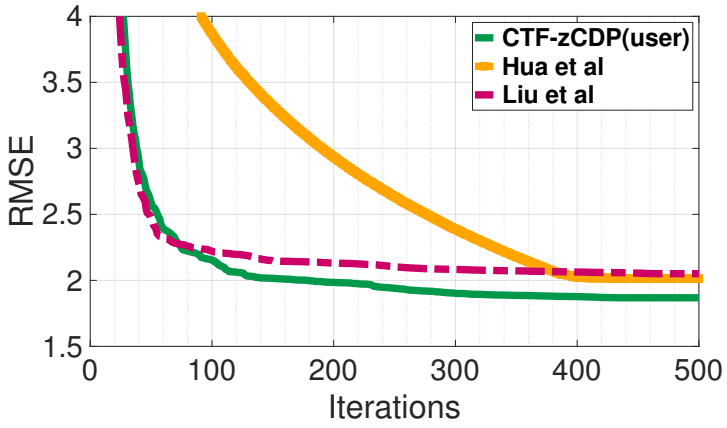


Figure 7.12. Comparison results of Hua *et al* [1] and Liu *et al* [2] with CTF-zCDP (user-level) on Movielens dataset when $\epsilon = 0.1$.

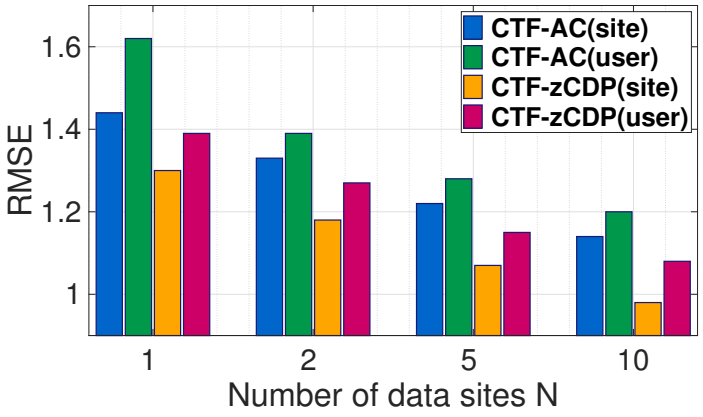
computation times. Both CTF-zCDP and Liu *et al* [2] use the SGLD based approach, so each iteration almost takes the same time: ∼30 sec. The algorithm of Hua *et al* [1] is a bit faster than the Bayesian approaches since it is based on deterministic SGD; it takes ∼26 sec per iteration.

Table 7.6. Comparison of methods in terms of RMSE for $\epsilon = \{10, 1, 0.1\}$ and NP-case.

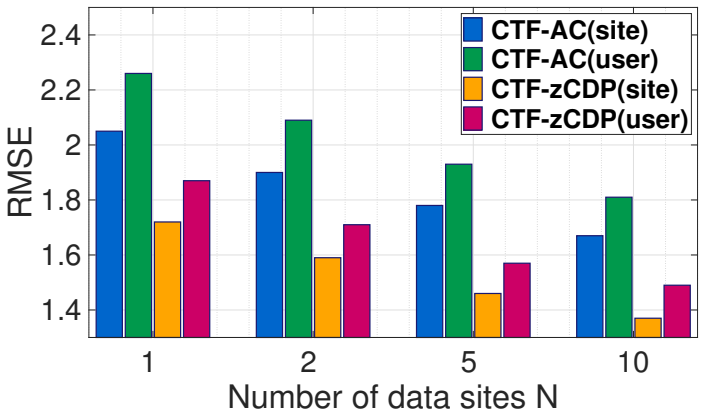
		NP	ϵ = 10	ϵ = 1	ϵ = 0.1
Movielens	CTF-zCDP (user-level)	0.96	1.05	1.39	1.87
	Hua *et al* [1]	0.93	1.25	1.48	2.01
	Liu *et al* [2]	0.93	1.18	1.46	2.05
Flickr	CTF-zCDP (user-level)	0.37	0.51	0.74	1.02
	Hua *et al* [1]	0.39	0.55	0.80	1.10
	Liu *et al* [2]	0.37	0.52	0.75	1.02

Finally, we monitored the RMSE between $X^{(1)}$ and $\hat{X}^{(1)}$ when we increase the number of data sites N from 1 to 10 on real datasets. We first used the matrix factorization model for

Movielens data that is illustrated in Figure 4.2 where the item factor Θ_2 is shared between the data sites. Figure 7.13(a) and Figure 7.13(b) illustrates the prediction performances of our methods for different number of data sites when $\epsilon = 1$ and $\epsilon = 0.1$ respectively. Moreover, the results for all privacy levels and methods are reported in terms of *RMSE* in Table 7.7.



(a) $\epsilon = 1$



(b) $\epsilon = 0.1$

Figure 7.13. Comparison of our methods on Movielens for $N = \{1, 2, 5, 10\}$.

It is clear that the performance of all methods improves as the quantity of additional data sources increase. However, when the privacy loss ϵ is small, CTF-zCDP (especially with the site-level privacy) performs significantly better than the other methods.

Varying the number of shared factors between data sites. We conducted experiments on modeling relations with several combinations of *private* and *shared* factors in order to answer the question: does sharing more information via the *shared* factors lead to improved

Table 7.7. Prediction results for $X^{(1)}$ when the number of data sites for $N = \{1, 2, 5, 10\}$ on Movielens data and $\epsilon = \{10, 1, 0.1\}$ for various privacy notions.

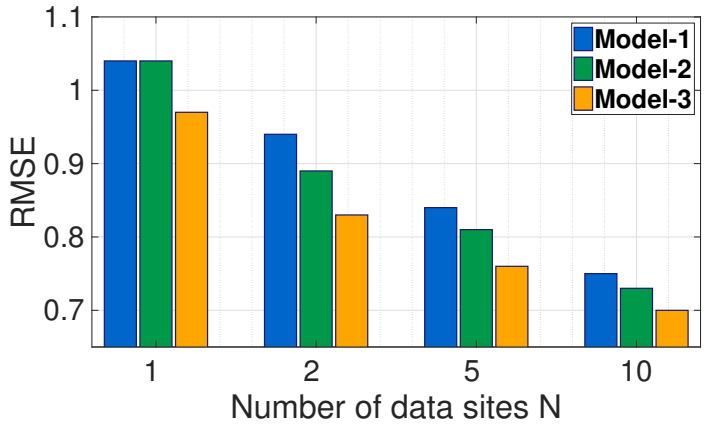
		N = 1	N = 2	N = 5	N = 10
Method	NP	0.96	0.89	0.81	0.74
CTF-AC (site-level)	ϵ = 10	1.11	1.02	0.94	0.88
	ϵ = 1	1.44	1.33	1.22	1.14
	ϵ = 0.1	2.05	1.90	1.78	1.67
CTF-AC (user-level)	ϵ = 10	1.19	1.10	1.01	0.95
	ϵ = 1	1.62	1.39	1.28	1.20
	ϵ = 0.1	2.26	2.09	1.93	1.81
CTF-zCDP (site-level)	ϵ = 10	1.01	0.92	0.83	0.77
	ϵ = 1	1.30	1.18	1.07	0.98
	ϵ = 0.1	1.72	1.59	1.46	1.37
CTF-zCDP (user-level)	ϵ = 10	1.05	0.96	0.87	0.81
	ϵ = 1	1.39	1.27	1.15	1.08
	ϵ = 0.1	1.87	1.72	1.57	1.49

prediction performance? We compared the performances of the models that are visualized in Figure 4.7(a), 4.7(b) and 4.7(c) on UCLAF and Flickr datasets. In *Model-1*, only Θ_2 (*location* factor for UCLAF and *item* factor for Flickr) is shared and in *Model-2*, Θ_3 (*activity* factor for UCLAF and *tag* factor for Flickr) is shared additional to Θ_2 between the data sites. *Model-3* integrates the public *item-feature* matrix to *Model-2* and this matrix provides auxiliary information to the inference of *shared* item feature matrix Θ_2. We use the auxiliary matrix $X^{(11)}$ for $N = \{1, 2, 5, 10\}$ in *Model-3* to see the effect of non-private additional information. In this set of experiments, we choose $\epsilon = \{10, 1, 0.1\}$ and compare the results with the non-private (NP) case. We give the results for *CTF-zCDP (user-level)* method and the number of data sites $N = \{1, 2, 5, 10\}$. The complete results are provided in Table 7.8.

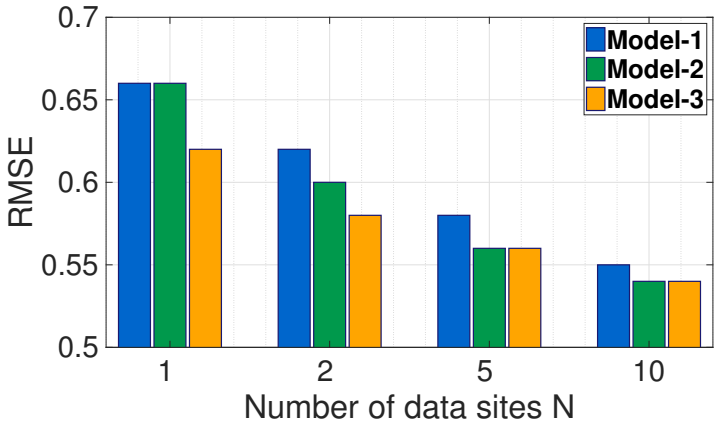
Table 7.8. Model comparison results for CTF-zCDP (user-level) when the number of data sites in the system increases. The results are reported for $\epsilon = \{10, 1, 0.1\}$ and NP case on UCLAF and Flickr.

		UCLAF				Flickr			
		$N = 1$	$N = 2$	$N = 5$	$N = 10$	$N = 1$	$N = 2$	$N = 5$	$N = 10$
NP	Model-1	0.77	0.65	0.55	0.47	0.42	0.36	0.31	0.26
	Model-2	0.77	0.62	0.51	0.43	0.42	0.33	0.27	0.23
	Model-3	0.68	0.54	0.44	0.37	0.36	0.29	0.23	0.20
$\epsilon = 10$	Model-1	0.89	0.77	0.66	0.57	0.54	0.48	0.43	0.38
	Model-2	0.89	0.72	0.60	0.51	0.54	0.45	0.38	0.33
	Model-3	0.80	0.64	0.53	0.46	0.46	0.38	0.32	0.27
$\epsilon = 1$	Model-1	1.04	0.94	0.84	0.75	0.66	0.62	0.58	0.55
	Model-2	1.04	0.89	0.81	0.73	0.66	0.60	0.56	0.54
	Model-3	0.97	0.83	0.76	0.70	0.62	0.58	0.56	0.54
$\epsilon = 0.1$	Model-1	1.83	1.66	1.52	1.40	0.98	0.93	0.88	0.83
	Model-2	1.83	1.74	1.68	1.64	0.98	0.92	0.89	0.88
	Model-3	1.64	1.60	1.57	1.57	0.91	0.86	0.85	0.87

Figure 7.14(a) and Figure 7.14(b) compares the *Model-1*, *Model-2* and *Model-3* for different number of data sites when $\epsilon = 1$. Similar to the synthetic data and Movielens results, the performance of all models improves as the number of additional data sources integrated into the system increase. When N gets larger, more information is provided to update the shared factors; so the difference of the performances of *Model-1* and *Model-2* also becomes larger. However for $N = 1$, there is no data sharing for *Model-1* and *Model-2*, therefore all models give the same result. It is also clear that the non-private additional information in *Model-3* improves the prediction accuracy.



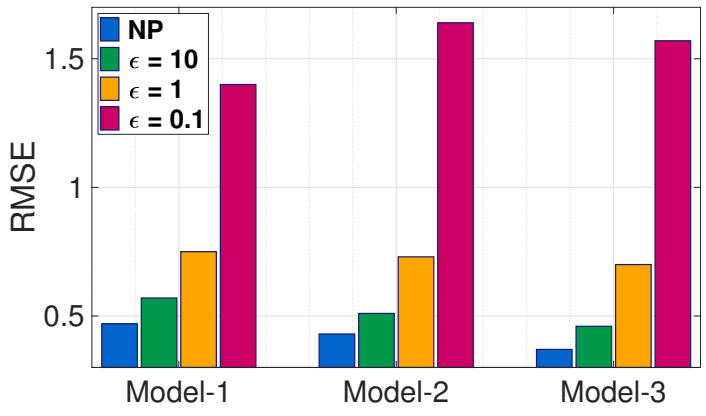
(a) UCLAF, ϵ = 1



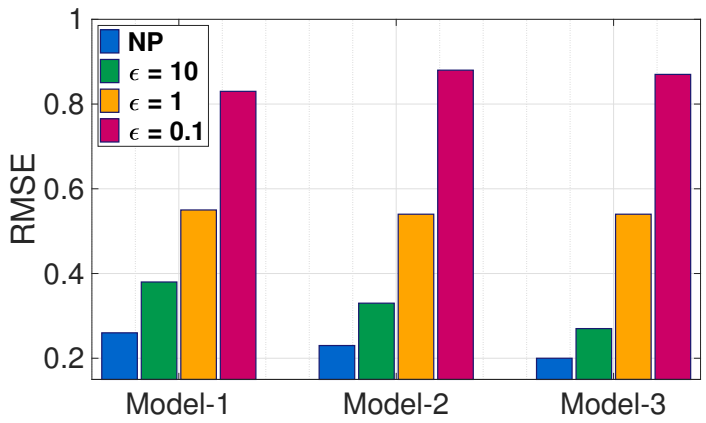
(b) Flickr, ϵ = 1

Figure 7.14. Prediction results of *Model-1*, *Model-2* and *Model-3* when $\epsilon = 1$ for $N = \{1, 2, 5, 10\}$ with CTF-zCDP (user-level).

Besides, we compare the performances of the models when $N = 10$ for different privacy levels in Figure 7.15(a) and Figure 7.15(b). When the number of data sites increase for high privacy loss (when $\epsilon < 1$), the cumulative noise added to the shared factors reaches to large values. In these figures, we can see that the prediction quality decreases when the number of shared factors increases and the ϵ decreases. For NP and ϵ =1, the prediction accuracy is higher for *Model-2* and *Model-3*. For ϵ = 1 and ϵ = 0.1 *Model-1* and *Model-2* performs nearly the same and *Model-3* is slightly better because of the non-private additional information.



(a) UCLAF



(b) Flickr

Figure 7.15. Prediction results of *Model-1*, *Model-2* and *Model-3* when $N = 10$ for NP and $\epsilon = \{0.1, 1, 10\}$ with CTF-zCDP (user-level).

8. EXPERIMENTS ON DIFFERENTIALLY PRIVATE BAYESIAN NEURAL NETWORKS

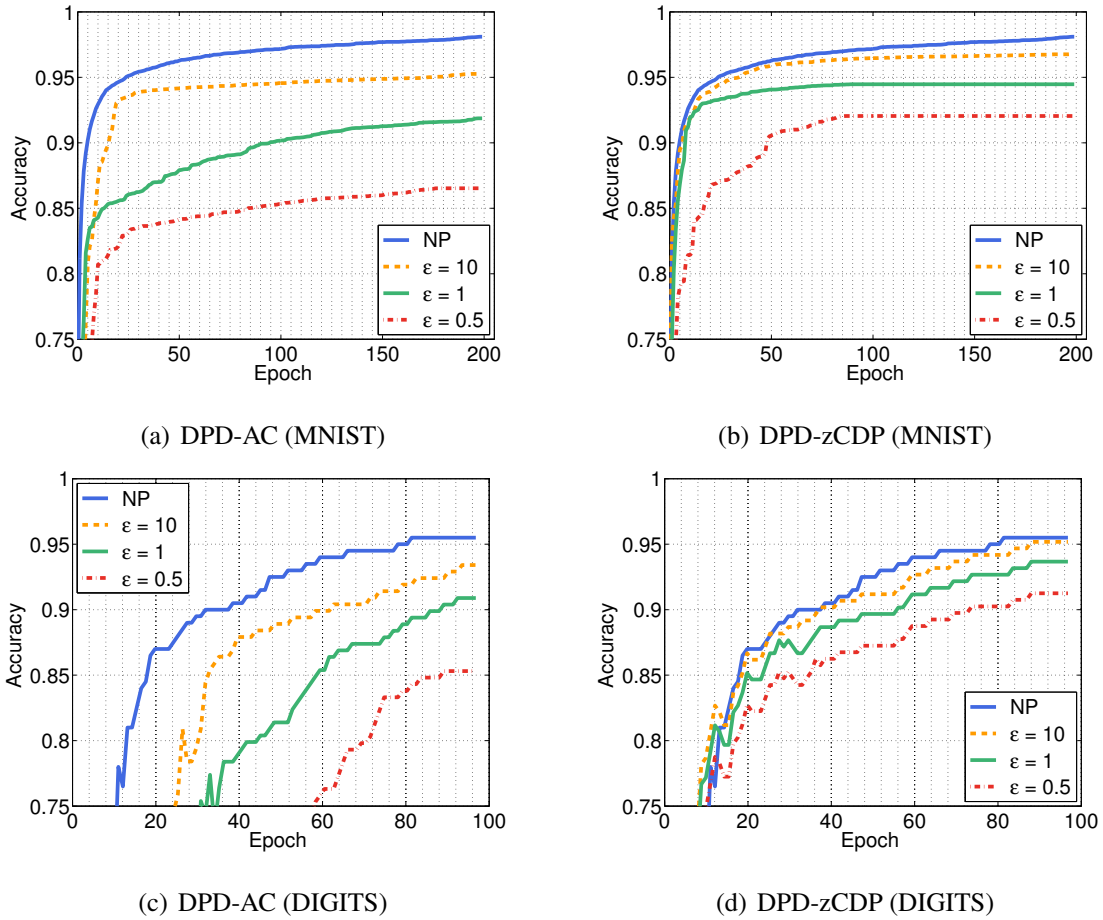
The aim of this chapter is to evaluate the performance our differentially private Bayesian neural networks presented in Chapter 5 on real world datasets.

8.1. Experiments of Differentially Private Bayesian Neural Networks

We use two standard benchmarking datasets MNIST [124] and DIGITS [151] to evaluate our method. MNIST dataset contains 70K 28×28 handwritten digits (60K for training and 10K for testing) and DIGITS dataset consists of 1797 8×8 grayscale images (1439 for training and 360 for testing) of handwritten digits. We generate and use a single layer feed-forward neural network for the experiments in this section. We use rectifier linear units (ReLU) as the activation function and softmax operation for the classification of MNIST and DIGITS where both have 10 classes. We choose the non-private (NP) model as our baseline. For MNIST, we use the minibatch of size $S = 600$ with 1000 hidden units and reach an accuracy of 98.20% in about 200 epochs. For DIGITS, we use the minibatch of size $S = 100$ with 500 hidden units and reach an accuracy of 95.50% in about 100 epochs. All of the results are implemented in Theano [152] and are the average of 10 runs.

8.1.1. Differentially private models

We use the architecture that is defined above for the differentially private version as well. We applied gradient clipping to the hidden layer of the network in order to limit the sensitivity and compute the amount of noise to be added to protect the privacy. We choose the gradient clipping norm $G = 3$ for MNIST and $G = 2$ for DIGITS. For three different noise levels, $\epsilon = \{10, 1, 0.5\}$, we run the experiments and report the results. For any fixed ϵ, δ is varied between 10^{-2} and 10^{-5}. There is a slight difference with different δ values (less than 10^{-3}), but still we choose the best performing $\delta = 10^{-4}$ for both datasets. We set the initial learning rate $\eta_0 = 0.1$ for MNIST and $\eta_0 = 0.05$ for DIGITS and update it per round



(a) DPD-AC (MNIST)

(b) DPD-zCDP (MNIST)

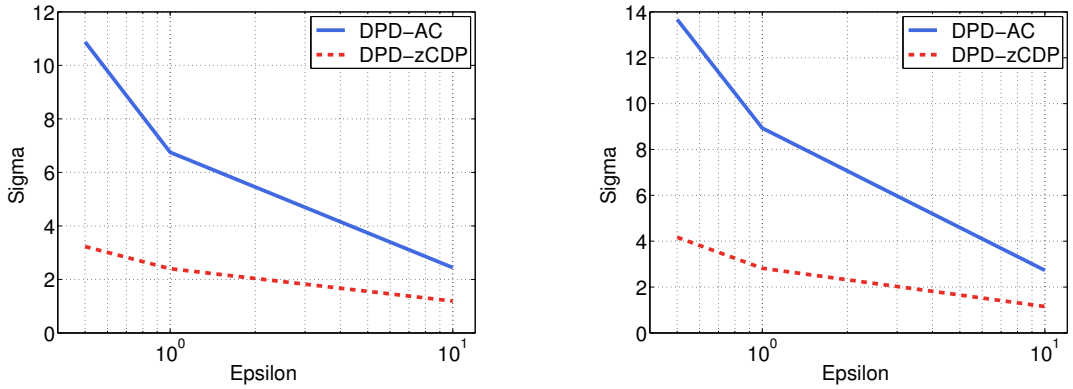
(c) DPD-AC (DIGITS)

(d) DPD-zCDP (DIGITS)

Figure 8.1. Comparison of the test accuracies for $\epsilon = \{10, 1, 0.5\}$ and the NP case for (a) AC, b) zCDP compositions on MNIST and (c) AC, d) zCDP compositions on DIGITS.

as $\eta_t = \eta_0/t^\gamma$. We fixed the decay rate γ to 1 for both of the datasets.

In the first set of experiments, we investigate *the influence of privacy loss on accuracy*. The mini-batch sizes were set to $S = 600$ and $S = 100$ for MNIST and DIGITS respectively. We ran the algorithm for 200 passes for MNIST and 100 passes for DIGITS. Figure 8.1(a), 8.1(c) report the performance of the DPD-AC method and Figure 8.1(b), 8.1(d) report the performance of the DPD-zCDP method for different noise levels. These results justify the theoretical claims that lower prediction accuracy is obtained when the privacy protection is increased by decreasing ϵ. One more deduction from these results is that dropout with the zCDP composition (*DPD-zCDP* method) can reach an accuracy very close to the non-private level especially under reasonably strong privacy guarantees (when $\epsilon > 0.5$).



(a) σ vs ϵ (MNIST)

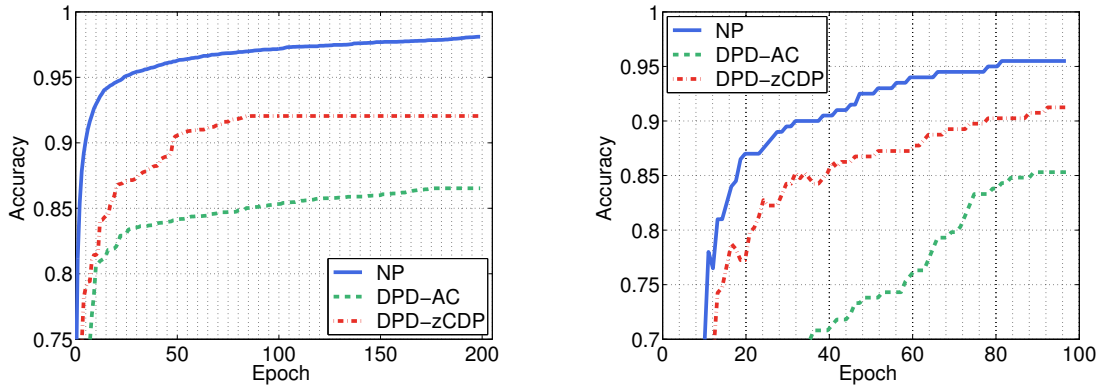
(b) σ vs ϵ (DIGITS)

Figure 8.2. The σ value as a function of ϵ on MNIST and DIGITS respectively.

Then, we compare the classification accuracy of models learned using two variants of our algorithm: DPD-AC and DPD-zCDP. As mentioned in Section 5.2.3, zCDP composition provides a compact bound on the differential privacy budget compared to the AC theorem. In order to test it empirically, we run experiments with concrete ϵ values. The noise level can be computed from the overall privacy loss ϵ, the sampling ratio of each minibatch $\nu = S/N$ and the number of epochs E (where the number of iterations is computed as $T = E/\nu$). For our MNIST and DIGITS experiments, we set $\nu = 0.01$, $E = 200$ and $\nu = 0.05$, $E = 100$, respectively. Then, we compute the value of σ. For example, when $\epsilon = 0.5$, the σ values are 10.88 for DPD-AC and 3.23 for DPD-zCDP on MNIST. We can see from Figure 8.2(a) and Figure 8.2(b) that we get a much lower noise by using the zCDP for a fixed the privacy loss ϵ.

Therefore, for our models with a total privacy budget fixed to ϵ, the amount of noise added is smaller for zCDP, and the test accuracy is higher. Figure 8.3(a) and Figure 8.3(b) show the comparison results of DPD-AC and DPD-zCDP methods when $\epsilon = 0.5$ with NP model. Both results clearly show that using the zCDP composition further helps in obtaining even more accurate results at a comparable level of privacy.

Lastly, we compare our methods to the most related algorithm proposed by Abadi *et al.* [81] and the case when no dropout is used. For the algorithm with no dropout, we use SGLD to update the weights of the neural network. We ran all the methods on MNIST and DIGITS with varying ϵ. Table 8.1 reports the test accuracies of all methods. The previous



(a) MNIST (b) DIGITS

Figure 8.3. (a) and (b) are the test accuracy results of DPD-AC and DPD-zCDP for $\epsilon = 0.5$ on MNIST and DIGITS datasets.

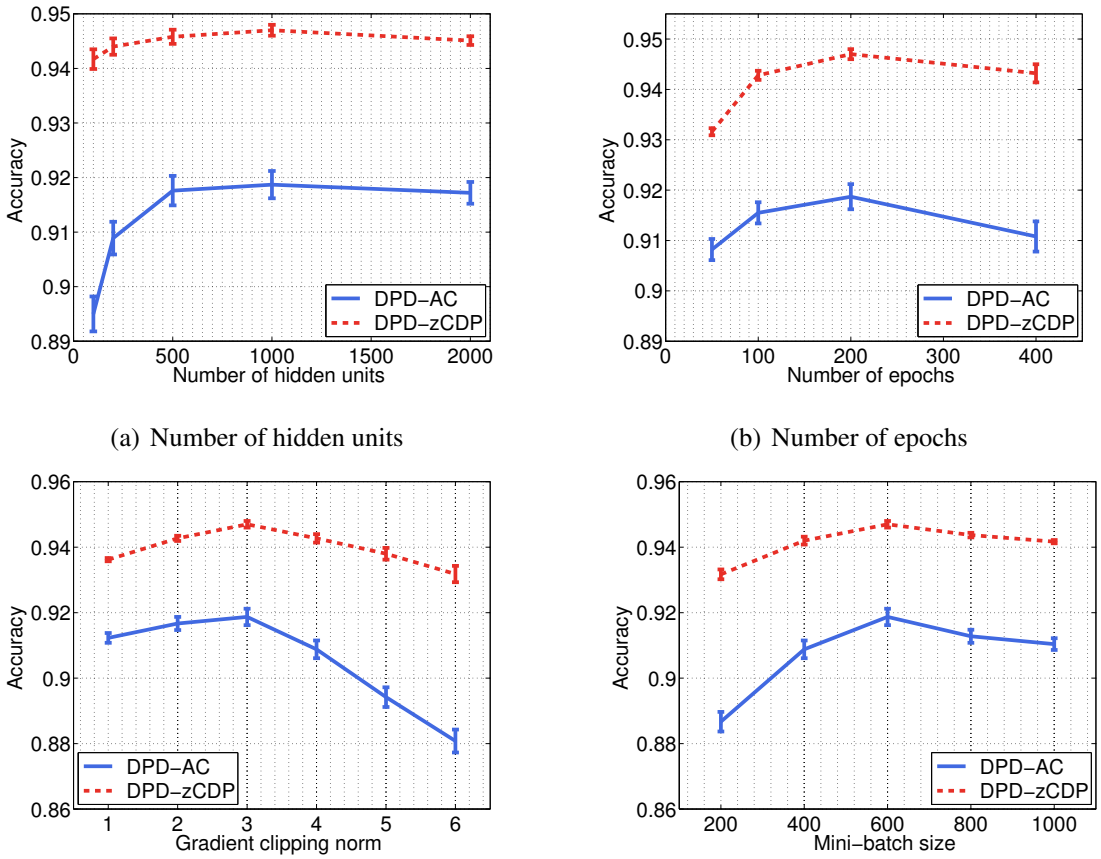
Table 8.1. Comparison of the methods for $\epsilon = \{10, 1, 0.5\}$. Bold values indicate the best results.

	MNIST			DIGITS		
	$\epsilon = 10$	$\epsilon = 1$	$\epsilon = 0.5$	$\epsilon = 10$	$\epsilon = 1$	$\epsilon = 0.5$
DPD-AC	0.9526	0.9187	0.8658	0.9341	0.9089	0.8521
DPD-zCDP	0.9718	**0.9470**	**0.9205**	**0.9518**	**0.9367**	**0.9125**
SGLD-zCDP (no dropout)	0.9581	0.9216	0.8952	0.9403	0.9187	0.8821
Abadi *et al.* [81]	**0.9720**	0.9305	0.8965	0.9480	0.9265	0.9003

experiments have already demonstrated that DPD-zCDP significantly improves the prediction accuracy. These results also support it and show that *dropout* improves the prediction accuracy especially when the privacy budget is low.

8.1.2. Effect of the parameters

The accuracy in neural networks depends on a number of factors and parameters that are needed to be tuned to optimize the performance. For our DP methods, these factors consist the number of hidden units, the number of epochs/iterations, the gradient clipping threshold, the minibatch size and the noise level. In the previous section, we compared the effect of



(a) Number of hidden units

(b) Number of epochs

(c) Gradient clipping norm

(d) Minibatch size

Figure 8.4. The effect of the model parameters on MNIST dataset.

different noise levels to the classification accuracy. Here, we demonstrate the effects of the remaining parameters. We control a parameter individually by fixing the rest as constant. For MNIST experiments, we set the parameters as follows: 1000 hidden units, minibatch size of 600, gradient clipping norm of 3, initial learning rate of 0.1, 200 epochs and the privacy budget ϵ to 1. The results are presented in Figure 8.4.

In non-private neural networks, using more hidden units is often preferable and increases the prediction accuracy of the trained model. For differentially private training, using more hidden units may lead too much additional noise due to the increase in the sensitivity of the gradient. However, this doesn't mean that when the number of hidden units increases, the accuracy will decrease. The large networks are tolerant to noise, so the optimum number of hidden units should be tuned. Figure 8.4(a) shows that accuracy is very close for a hidden unit number in the range of $[500, 2000]$ and peaks at 1000.

The number of iterations T (that can be computed as $T = E/\nu$) needs to be sufficient but not too large. The privacy cost in zCDP increases in proportion to $\sqrt{T}$ and it is more tolerable than DP. We tried several values in the range of $[50, 400]$ and observed that we obtained the best results when E is between 100 and 200 for MNIST.

Tuning the gradient clipping threshold depends on the details of the model. When the threshold is quite small, the clipped gradient could be too different from the correct gradient and points to a contrasting direction. Besides, when we increase the threshold, we add a large amount of noise to the gradients. Abadi *et al.* [81] proposed that a good way to choose a value for C is taking the median of the norms of the unclipped gradients. In our experiments, we tried G values in the range of $[1, 6]$. Figure 8.4(c) shows that our model is tolerable to the noise up to $G = 4$, then the accuracy decreases marginally.

Finally, we monitor the effect of the minibatch size. In DP settings, choosing smaller minibatch size leads running more epochs, however, the amount of the noise added has a less effect comparatively for a larger minibatch. Figure 8.4(d) shows that relatively larger minibatch sizes give better results. Empirically, we obtain the best accuracy when S is around 600 (so $\nu = S/N = 0.01$). The behavior of the results on MNIST and DIGITS are quite similar, so we only demonstrate the results on MNIST data.

Multi-layer BNNs: . In the multi-layer neural networks, each layer can be analyzed independently. This feature of the neural networks grants us to set different gradient norm bounds G or use different privacy budgets ϵ (that ends up with different noise scales σ) for each layer. Here, we demonstrate the results of our experiments with multi-layer neural networks. Throughout this chapter, we designed experiments with single layer neural networks. Because, the multi-layer neural networks accumulates the privacy loss at each hidden layer and the amount of the added noise can become very large. The result in Figure 8.5 shows that the single layer network performs best on the MNIST data. Here, we didn't optimize the other network parameters (number of hidden units, iteration, clipping norm, mini-batch size) per layer. We fixed them to the same values for each layer and these values are equal to the choosen values for MNIST experiments.

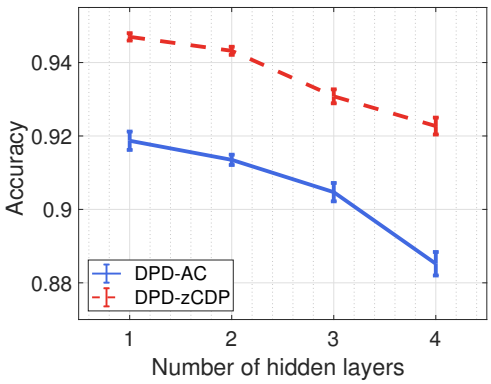


Figure 8.5. The effect of the number of hidden layers on MNIST dataset.

Connection between dropout rate and the privacy budget: . As we show in Section 5.2.2, we perturb parameter updates with a multivariate normal noise $\xi' \sim \mathcal{N}(0, 4G^2\sigma^2\mathbb{I})$. Here, σ determines the total privacy budget ϵ and this amount is equal to the $\xi' \sim \mathcal{N}(0, \alpha\, diag(\theta_t^2))$. Here, α is not fixed for each weight and is controlled by the noise level that protects ϵ-DP. We cannot directly compute the value of α because it depends on the value of the θ. However,we run an experiment to give an intuition about the connection between total privacy budget ϵ and α.

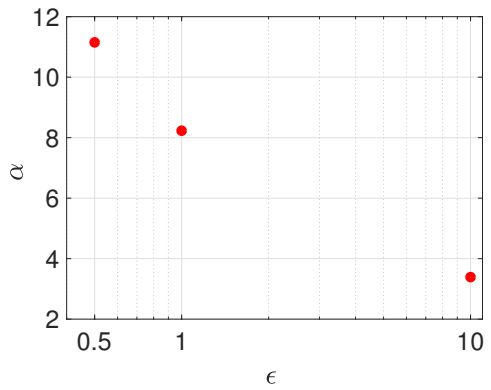


Figure 8.6. ϵ vs α on MNIST dataset.

We observe from Figure 8.6 that $\alpha = \frac{p}{1-p}$ is getting smaller (it means p is getting larger) when the privacy budget ϵ is relatively large. We may conclude that smaller privacy budget causes more sparse neural networks. However, the value of the network parameter θ is an important factor here and the connection between θ-α and θ-ϵ are also needed to be examined. We will work on these connections in detail in the future.

9. CONCLUSION AND FUTURE WORK

This dissertation addressed two major problems: i) data fusion and ii) privacy protection. We evaluated the privacy protection problem in two parts as: privacy protection in coupled tensor factorization and privacy protection of deep neural networks. For the *data fusion* problem, we proposed coupled matrix and tensor factorization models that can handle the heterogeneous data jointly which is collected from different sources and stored in the form of matrices and higher-order tensors. We particularly studied the link prediction problem by formulating it as simultaneous factorization of matrices/tensors and extracting common latent factors from the shared modes. While most existing studies on coupled analysis have been developed to fit a specific type of a tensor model using a particular loss function, we have developed coupled models for joint analysis of multiple datasets in a compact way using various tensor models and cost functions. In our coupled analysis for the link prediction problem, in addition to the commonly used KL-divergence and EUC distance based loss functions, we have also studied various other cost functions based on β-divergences. Through extensive experiments on real datasets, we assessed the performance of various alternative tensor models and loss functions. Numerical experiments clearly demonstrate that not only joint analysis of data from multiple sources via coupled factorization improves the link prediction performance but also the selection of right loss function and tensor model is crucial for accurately predicting missing links.

Furthermore, we developed a variational Bayesian inference for coupled tensor factorization with KL cost from a full Bayesian perspective that also handles the missing data naturally. This method provided a practical way without incurring much additional computational cost to EM approach for computing the approximation distribution and full conditionals. Our experiments demonstrate that the variational approach alleviates the overfitting better than the standard tensor factorization approaches and leads to the improved performance.

Sharing information across matrices and/or tensors brings the privacy protection requirement. For the *privacy protection* in coupled tensor factorization, we proposed a privacy-aware protocol to extract profitable knowledge from broad and distributed data sources via proba-

bilistic modeling. Our model complies to the restrictions and constraints that are caused by the data ownership or privacy issues on the models that are to be shared. We optimize our approach to make effective use of the efficient privacy techniques to leak little information. We initially developed the coupled matrix factorization algorithms to learn privacy preserving shared representations, which fuse data sources from different providers and extract actionable information for estimation. We then extended our method for coupled tensor factorization models. For these methods, we used several composition methods and privacy notions; also proposed novel definitions of privacy and analyzed the privacy theoretically on the basis of the *differential privacy* which is our privacy metric. In conclusion, we empirically showed that the performance of our methods significantly increases when more data providers are integrated into the system. Empirical results also suggested that our algorithms are robust and work very well in practice.

Finally, we introduced differentially private dropout (DPD) for the *privacy protection* in coupled tensor factorization deep neural networks. DPD outputs privatized results in BNNs with accuracy close to the non-private model learning results, especially under reasonably strong privacy guarantees. To make effective use of the privacy budget over multiple iterations, we proposed to calculate the cumulative privacy cost by using zCDP. Then, we showed how to perform DPD in private neural network training settings and illustrated the effectiveness of our algorithm on several benchmark datasets for DNNs.

There are several open leads for future directions, especially for the *privacy protection* problem. We can tackle this directions in two categories: i) theoretical directions and ii) application-based directions. For the first category, a useful and interesting future step is providing a formal utility guarantee for the proposed approaches, since the privacy-utility tradeoff for both tensor factorization and neural network training is still an open problem. In addition to the utility analysis, using novel privacy protection mechanisms and analyze the privacy with them is an important future direction. Most recently, Balle and Wang [153] proposed an *improved Gaussian mechanism* that significantly increases the accuracy of standard Gaussian mechanism especially for high-dimensional datasets. Once again, Balle *et al* [154] introduced a novel privacy amplification approach by using subsampling. This approach leads to the tighter analysis of differential privacy and makes the design of new

differentially private algorithms easier. We are planning to use these two techniques to improve the utility of our proposed algorithms. One substantial next step is to extend the CMF and CTF approaches to a fully distributed scenario where a public dataset can be improved by additional information provided by the analyses of other datasets in a differentially private way. In that setting, we assume N sites that hold private data and M sites that hold public data that has no privacy constraints. Besides, each data site may have a set of two types of users: *public* and *private*. The public users are the people who are volunteers to share their data, such as their scores in recommender systems, openly. On the other hand the private users demands brutal privacy guarantees, so their data is required to be protected. We aim to develop a mechanism for every participant to control the desired level of privacy when participating in the system. As a last theoretical direction, we approach the Bayesian Neural Networks (BNNs). We presented a method for differentially private training of a single BNN. However, the sensitive data that is used for the training of the neural networks are generally distributed across different parties. Besides, knowledge aggregation and transfer in deep learning recently become very popular and they increases the requirement of privacy protection of the data. For these reasons, we consider to work on a differentially private training of distributed deep neural networks and differentially private transfer learning.

Besides, we can increase the number of applications for our differentially private algorithms. First of all, we can extend the approach to more sophisticated stochastic gradient algorithms [155, 156] in place of SGLD to further improve the mixing rate and then apply our proposed generic algorithm to any latent factor model. For example, we left its application to other probabilistic models such as logistic regression and Latent Dirichlet Allocation (LDA) to future work. Finally, the DPD algorithm proposed in this thesis are generic and it can be applied to any neural network model. We left its application to other variants of neural networks such as convolutional and recurrent neural networks for future work.

APPENDIX A: Gradient Computation for Exponential Familiy Distributions

For all differentially private coupled matrix and tensor factorization methods that we use SGLD, we have to derive the gradient of the log likelihood w.r.t. Θ_d. We give the update equations in Section 4.2.2 as:

$$\Theta_{d_{(t)}}^{(v)} = \Theta_{d_{(t-1)}}^{(v)} + \nabla\Theta_{d_{(t)}}^{(v)}$$
$$\Theta_{d(t)} = \Theta_{d(t-1)} + \sum_{v\in\mathcal{V}} \nabla\Theta_{d_{(t)}}^{(v)}$$

for private and shared factors respectively. In this section, we will derive the gradients for DP-CTF in detail. Before deriving the equations w.r.t. Θ_d, we will write down the general form the derivative of the β-divergence that is described in Equation 2.4 with respect to $\hat{X}$ as follows:

$$\frac{\partial d_p(X;\hat{X})}{\partial \hat{X}} = -X\hat{X}^{-p} + \hat{X}^{1-p} = \frac{\hat{X} - X}{\hat{X}^p}$$

In our case X is the tensor, and the derivative of the β-divergence $d_{p_v}(X^{(v)};\hat{X}^{(v)})$ w.r.t. an element of the model output tensor $\hat{X}^{(v)}$ similarly becomes:

$$\frac{\partial d_{p_v}(X^{(v)};\hat{X}^{(v)}))}{\partial \hat{X}^{(v)}} = -X^{(v)}(\hat{X}^{(v)})^{-p_v} + (\hat{X}^{(v)})^{1-p_v} = \frac{\hat{X}^{(v)} - X^{(v)}}{(\hat{X}^{(v)})^{p_v}} \tag{A.1}$$

We can obtain the partial derivatives by using Equation A.1 as follows:

$$\frac{\partial\mathcal{L}}{\partial\Theta_d} = \sum_v \frac{1}{\phi_v} \sum_{i_v\in\Omega_v} \frac{\partial d_{p_v}(X^{(v)};\hat{X}^{(v)}))}{\partial\hat{X}^{(v)}} \frac{\partial\hat{X}^{(v)}}{\partial\Theta_d} \tag{A.2}$$

$$= \sum_v \left[R^{v,d}\phi_v^{-1} \sum_{i_v\cap\bar{i}_d} \left(\frac{\hat{X}^{(v)} - X^{(v)}}{(\hat{X}^{(v)})^{p_v}}\right) \sum_k \prod_{d\notin S_v} \theta_{i_d k}^{(d)} \right] \tag{A.3}$$

where $\mathcal{L}$ is the loss function that corresponds to $-\sum_{v=1}^{V}\sum_{i_v\in\Omega_v} \log p(X_{i_v}^{(v)} | \sum_{k=1}^{K}\prod_{d\in S_v}\theta_{i_d k}^{(d)})$.

Finally, we can rewrite the equation above in the following form:

$$\Delta_{\Theta_d}\mathcal{L} = \sum_v \left[R^{v,d}\phi_v^{-1}\Delta_{d,v}((\hat{X}^{(v)})^{(1-p_v)})\right] - \sum_v \left[R^{v,d}\phi_v^{-1}\Delta_{d,v}((\hat{X}^{(v)})^{(-p_v)} \circ X^{(v)})\right]$$

where $\Delta_{d,v}(A) = \left[\sum_{i_v \cap \bar{i}_d} A^{(v)}_{i_v} \sum_k \prod_{d \notin S_v} \theta^{(d)}_{i_d k}\right]$ as we mentioned in Section 3.1.1.

Here, we compute the gradients under the assumption of each observation tensor is modeled by the same type of distribution having the same dispersion parameter. This results in the same cost function (p_v) for all the observed tensors $X^{(v)}$ and we can cancel out the dispersion parameters from the update equations. To compute the gradients for DP-CMF model (the Poisson-Gamma model) we described in Section 4.1.2 we take p_v=1 and for the DP-CTF model (the Normal-Normal model) that is described in Section 4.2.1 we take p_v=0.

APPENDIX B: Noise Computation for DP-CMF

We begin with the definitions of the relevant ingredients than compute the amount of noise that is needed for each iteration of Algorithm 1.

Using the moments accountant features given in Section 2.2.3, Abadi *et al* [81] proved the following lemma.

Lemma B.1. *Suppose that* $f : \mathcal{X} \to \mathbb{R}^p$ *with* $\| f(\cdot) \| \leq 1$. *Let* $\sigma \geq 1$ *and* $\mathcal{B}$ *is a minibatch sample with sampling probability* q, *i.e.,* $q = B/S$ *with minibatch size of* B. *If* $q < \frac{1}{16\sigma}$, *for any positive integer* $\lambda \leq \sigma^2 \log \frac{1}{q\sigma}$, *the mechanism* $\mathcal{A}(Y) = \sum_{i \in \mathcal{B}} f(x_i) + \mathcal{N}(0, \sigma^2 \mathbb{I})$ *satisfies:*

$$\alpha_{\mathcal{A}}(\lambda) \leq \frac{q^2 \lambda(\lambda+1)}{(1-q)\sigma^2} + O(q^3 \lambda^3 / \sigma^2)$$

Li *et al* [28] used Lemma B.1 for their privacy analysis. We make use of their analysis to prove Theorem 4.3. There are two options that are needed to be satisfied to prove this theorem. We can use a decreasing or fixed stepsize and we will show that (ϵ, δ)-DP is guaranteed in both cases.

Proof when the stepsize decreases . We first prove that if the variance of $\xi^{(n,t)}$ is set to $\sigma_t^2 = \frac{c_2^2 G^2 T^{2/3} t^{1/3} \log(1/\delta)}{\epsilon^2 S^2} \eta_t^2 \mathbb{I}$, Algorithm 2 satisfies (ϵ, δ)-DP.

Since Algorithm 2 sequentially updates the model parameters U (could be multiple $U^{(n}$ for each site n) and V, each iteration corresponds to a randomized mechanism $\mathcal{A}_i$ that is defined in Theorem 2.8. So, the first step is to derive moments accountant for each update.

- The only data access in each iteration is $\sum_{(i_n,j) \in \mathcal{B}^{(n,t)}} \Delta \tilde{U}^{(n,t)}$ and $\sum_{(i_n,j) \in \mathcal{B}^{(n,t)}} \Delta \tilde{V}^{(n,t)}$. Thus, we focus on the interaction between noise $\xi^{(n,t)}$ and the gradients. Given that the

updates of U and V are similar, we focus on the interaction between $\sum_{(i_n,j)\in\mathcal{B}^{(n,t)}} \Delta\tilde{V}^{(n,t)}$ that is $\frac{\eta_t}{q}\sum_{(i_n,j)\in\mathcal{B}^{(n,t)}} \Delta\tilde{V}^{(n,t)} + \xi^{(n,t)}$ where $q = \frac{B}{S^{(n)}}$.

- To simplify the notation, we let $\tilde{\eta}^2 = \frac{\sigma_t^2 q^2}{G^2\eta_t^2 t^{1/3}}$, and the variance of $\xi^{(n,t)}$ can be rewritten as $\sigma_t^2 = \frac{\tilde{\eta}^2 G^2 \eta_t^2 t^{1/3}}{q^2}\mathbb{I}$. Then we have:

$$\begin{aligned}\frac{\eta_t}{q}\sum_{(i_n,j)\in\mathcal{B}^{(n,t)}} \Delta\tilde{V}^{(n,t)} + \xi^{(n,t)} &= \frac{\eta_t}{q}\left(\sum_{(i_n,j)\in\mathcal{B}^{(n,t)}} \Delta\tilde{V}^{(n,t)} + \mathcal{N}(0,\sigma_t^2 q^2/\eta_t^2)\mathbb{I}\right)\\ &= \frac{\eta_t G}{q}\left(\frac{1}{G}\sum_{(i_n,j)\in\mathcal{B}^{(n,t)}} \Delta\tilde{V}^{(n,t)} + \mathcal{N}(0,\tilde{\eta}^2 t^{1/3}\mathbb{I})\right)\end{aligned}$$

We can apply Lemma B.1 to compute the upper bound for the log moment of privacy loss random variable. When we set $f(x_i) = \frac{1}{G}\Delta\tilde{V}^{(n,t)}$ and $\sigma^2 = \tilde{\eta}t^{1/3}$, the upper bound for the t^{th} iteration:

$$\alpha(\lambda) \leq t^{-1/3} q^2\lambda^2/\tilde{\eta}^2$$

- When Lemma B.1 is satisfied, $\tilde{\eta}^2 t^{1/3} \geq 1$ and $q < \frac{1}{16\tilde{\eta}t^{1/6}}$. By Theorem 2.8 the log moment of the privacy loss random variable over T iterations is bounded by:

$$\alpha(\lambda) \leq \sum_{t=1}^{T} t^{-1/3} q^2\lambda^2/\tilde{\eta}^2$$

- Since $\sum_{t=1}^{T} t^{-1/3} = O(T^{2/3})$, when $\epsilon = c_1 q^2 T^{2/3}$ and $\tilde{\eta} = c_2\frac{q\sqrt{T^{2/3}\log(1/\delta)}}{\epsilon}$ the conditions (2) and (3) of Theorem 4.1 is satisfied. When we plug in $\tilde{\eta}$ and q, we obtain the required σ_t^2 that proves Algorithm 2 is (ϵ,δ)-DP.
- In addition to conditions (2) and (3), condition (1) also needs to be satisfied. In practice it is easy to satisfy that $\eta_t \leq \frac{S}{G^2}$ and most of the time it is satisfied. In all our experiments this condition is satisfied. In the worst case scenario, $S = 7500$ and $G = 2$, and η_t is much smaller than 1875. □

Proof when the stepsize is fixed: . The difference for the fixed stepsize is the variance of $\xi^{(n,t)}$ which is set to $\sigma_t^2 = \frac{\eta_0^2 G^2 \eta_t^2}{q^2}\mathbb{I}$. Then we apply composability theorem and Lemma B.1 to show Algorithm 2 satisfies (ϵ, δ)-DP:

$$Tq^2\lambda^2/\eta_0^2 \leq \lambda\epsilon/2$$
$$\exp(-\lambda\epsilon/2) \leq \eta_0^2 \log(1/q\eta_0)$$

We can find constants c_3 and c_4 using the properties defined in the previous section. All the above conditions are hold when $\epsilon = c_3 q^2 T$ and $\eta_0 = c_4 \frac{q\sqrt{\log(1/\delta)}}{\epsilon}$. When we plug in q and η_0 and compare it to $\frac{\eta}{S}$, we see that Algorithm 2 is (ϵ, δ)-DP. $\square$

APPENDIX C: Noise Computation for DP-CTF

We first provide the proof for Theorem 4.5 to demonstrate that the Algorithm 2 preserves (ϵ, δ) -DP. Then, we compute the amount of noise that is needed to be added at each iteration for AC and zCDP cases. We begin with the definitions of the relevant ingredients that are used for the proof of Theorem 4.5

Lipschitz continuity:. A function $f(x) : \mathbb{R} \to \mathbb{R}$ is Lipschitz continuous (L-Lipschitz), if for all pairs (x_1, x_2) we have $|f(x_1) - f(x_2)| \leq L|x_1 - x_2|$.

Lipschitz smoothness: . If a function $f(x)$ is differentiable, with $|f'(x)| \leq L$ for all x, then $f(x)$ is also Lipschitz smooth (L-smooth) [117, 118].

In many DP studies [116, 117, 121], the boundedness of the loss-function is a typical assumption. When the parameter space is unbounded, the loss function is generally unbounded too. And, if the loss function is unbounded, the algorithm does not meet $(\epsilon, 0)$-DP in general. In this study, the loss function is the log-likelihood and under the boundedness assumption of the loss function, we get

$$\log p(x'^{(v)}_{i_v} \mid \{\theta_{i_d k}{}^{(v)}\}_{d \in S_v}) - \log p(x_{i_v}{}^{(v)} \mid \{\theta_{i_d k}{}^{(v)}\}_{d \in S_v}) \leq L \parallel x'^{(v)}_{i_v} - x_{i_v}{}^{(v)} \parallel_\star$$
$$\leq 2L$$

When $\hat{X^{(v)}} = \sum_{k=1}^{K} \prod_{d \in S_v} \Theta_d$ is bounded, the Lipschitz constant L is usually small for continuous distributions [27]. L will be a bound on $\log p(x^{(v)}_{i_v} \mid \{\theta_{i_d k}{}^{(v)}\}_{d \in S_v})$ and will be a small constant unless $p(x^{(v)}_{i_v} \mid \{\theta_{i_d k}{}^{(v)}\}_{d \in S_v})$ increase or decrease exponentially fast at any point. We use gradient clipping to satisfy the constraint of Lipschitz continuity. Here is the

simple prrof that how gradient clipping supports the Lipschitz conditions:

$$L \parallel \Delta f(x) \parallel = L \parallel x_1 - x_2 \parallel \geq \mid f(x_2) - f(x_1) \mid \geq \mid \langle \Delta f(x_1), \Delta f(x_2) \rangle \mid$$
$$= \parallel \Delta f(x) \parallel^2$$

Finally, we will show a remark that is used for our noise computation:

Remark C.1. ***(Remark 1 in [27])*** *Given target privacy parameters $\epsilon_{tot}, \delta_{tot} > 0$, to ensure ($\epsilon_{tot}$, $k\delta + \delta_{tot}$) cumulative privacy loss over k mechanisms, it suffices that each mechanism is (ϵ_{iter}, δ_{iter})-DP, where*

$$\epsilon_{iter} = \frac{c}{\sqrt{2k \log(1/\delta_{tot})}} < 1 \tag{C.1}$$

for some constant $c < \sqrt{\log(1/\delta_{tot})}$. Equation (5.5) (Advanced Composition Theorem) can be simplified into $\epsilon_{tot} \leq 2c$ by applying the inequality $e^{\epsilon_{iter}} - 1 \leq 2\epsilon_{iter}$. (Taylor's Theorem is used for the proof.)

Now, we continue with the noise computation that is needed to calculate the privacy loss of the iterative approaches. In DP methods, a query is a function to be applied to a dataset. An iterative approach permits the data analyst to use queries adaptively, deciding which query to pose next based on the observed responses to previous queries. Here, we aim to model the composition to formalize the privacy of our algorithm. We need the composition because each party or each iteration may adaptively influence the data. In our algorithm, as a result of multiple (say T) iterations, we obtain a bound for the expected cumulative privacy loss where each iteration provides ϵ_{iter}-DP. We can trade off a small amount of δ to get a much better bound for the privacy loss due to the composition. The privacy loss for the current iteration is computed by using Privacy for subsampled data, Gaussian Mechanism and Advanced Composition that are described in Section 4.2.2. By the Gaussian Mechanism, the privacy

loss for the current iteration is in fact bounded from above by:

$$\epsilon_{iter} = \triangle_2 f \sqrt{2\log(1.25/\delta_{iter})}/\sigma' \tag{C.2}$$

In Algorithm 2, we set σ to $\sqrt{\frac{128C_vTL^2}{B\epsilon^2}\log\left(\frac{2.5C_vT}{B\delta}\right)2\log(2/\delta)\left(\eta_{(t)}^{(v)}\right)^2}$. The gradient of the log-likelihood (e.g. $\sum_{i_v\in\mathcal{B}_v^{(t)}}\nabla_{\theta_{i_d}}\log p(X_{i_v}^{(v)}|\{\Theta_d{}^{(t)}\}_{d\in S_v})$) is multiplied by the learning rate $\eta_{(t)}^{(v)}$ and the number of minibatches $n_B = C_v/B$. So, we divide the standard deviation σ by $(\eta_{(t)}^{(v)}n_B)$ to obtain the required amount of noise that is added to $\sum_{i_v\in\mathcal{B}_v^{(t)}}\nabla_{\theta_{i_d}}\log p(X_{i_v}^{(v)}|\{\Theta_d{}^{(t)}\}_{d\in S_v})$ and obtain σ' in Equation (C.2):

$$\begin{aligned}\sigma' &= \frac{\sqrt{128n_BTL^2/\epsilon^2\log\left(2.5n_BT/\delta\right)2\log(2/\delta)\left(\eta_{(t)}^{(v)}\right)^2}}{\eta_{(t)}^{(v)}n_B}\\ &= \frac{L\sqrt{256TB\log(2.5n_BT/\delta)\log(2/\delta)}}{\epsilon\sqrt{C_v}}\end{aligned}$$

After computing σ', we compute ϵ_{iter} where $\delta_{iter} = \frac{\delta}{2n_BT}$ by *Advanced Composition*:

$$\begin{aligned}\epsilon_{iter} &= \frac{\triangle_2 f\sqrt{2\log(1.25/\delta_{iter})}}{\sigma'} = \frac{2L\sqrt{2\log(2.5n_BT/\delta)}\,\epsilon\sqrt{C_v}}{L\sqrt{256TB\log(2.5n_BT/\delta)\log(2/\delta)}}\\ &= \frac{\epsilon\sqrt{C_v}}{\sqrt{32\,TB\log(2/\delta)}}\end{aligned}$$

By privacy for subsampled data, the privacy loss for the current iteration is:

$$\frac{\epsilon\sqrt{C_v}}{\sqrt{32BT\log(2/\delta)}}\frac{2B}{C_v} = \frac{\epsilon/2}{\sqrt{2(C_vT/B)\log(2/\delta)}}$$

Verify that we can indeed do that as $\frac{\epsilon\sqrt{C_v}}{\sqrt{32BT\log(2/\delta)}} < 1$ from the assumption on T. Note that to get T data passes with minibatch size B, we need to go through at most $\lfloor n_BT\rfloor$ iterations. Apply the advanced composition theorem, we get an upper bound of the total privacy loss ϵ and failure probability $\delta = \frac{\delta}{2} + \frac{\delta}{2n_BT}n_BT$ accordingly.□

We compute the amount of noise for Advanced composition. This noise can be converted to zCDP noise by using the Proposition 4.9, Proposition 4.10 and Lemma 4.11 given in Section 4.2.2. For the conversion from DP to zCDP, we first convert the epsilon used for AC case to a value which is used for zCDP case as:

$$\epsilon_{zCDP} = \rho + 2 * \sqrt{\rho \log(1/\delta)}$$

Then, for $A = 1$, $B = 2\sqrt{-\log(\delta)}$ and $C = -\epsilon$ we solve the following differential equation:

$$\rho = ((-B + \sqrt{B.^2 - 4 * A * C})/(2 * A))^2$$

Finally, we compute the new σ value based on the new ϵ.

APPENDIX D: Local Differential Privacy: CMF

We reserve this appendix to prove that the proposed methods LDP1 and LDP2 are *locally ϵ-differentially* private. Following the definition of local differential privacy provided in [60], we prove that the privatization mechanism $\mathcal{R}$ used in our methods preserves ϵ-DP.

As we explained in Section 2.2.4 and Section 4.1.4, we chose a privatization method $\mathcal{R}$ that allows us to form the marginal likelihood $P_{\mathcal{R}}(z_{i,j}|\lambda_{i,j}, \epsilon)$ and sample from $P_{\mathcal{R}}(x_{i,j}|z_{i,j}, \lambda_{i,j}\epsilon)$. We added noise from a two-sided Geometric distribution to each of the observations in order to guarantee ϵ-local-DP. In [57] they explained and proved that there are three ways to produce the perturbed data. First we show how two-sided Geometric random variable can be generated in a different way:

$$t_1 \sim \mathcal{EXP}(\frac{\epsilon}{1-\epsilon}), \qquad t_2 \sim \mathcal{EXP}(\frac{\epsilon}{1-\epsilon}), \qquad \tau \sim \text{Skellam}(t_1, t_2)$$

where the Skellam distribution is defined [157] as the marginal distribution over the difference $\tau = g_1 - g2$ of two independent Poisson random variables $g_1 \sim \mathcal{PO}(\mu_1)$ and $g_2 \sim \mathcal{PO}(\mu_2)$. Then, we show the three methods in Table D.1.

Method 1 shows that our algorithm guarantees privacy, because two-sided geometric noise is an existing privacy mechanism. We used this method in our experiments. *Method 2* represents the two-sided geometric noise in terms of a pair of Poisson random variables with exponentially distributed rates. *Method 3* marginalizes out all three Poisson random variables including $x_{i,j}$. In so doing, $z_{i,j}$ is directly drawn from Skellam distribution.

Table D.1. Randomized privatization mechanisms $\mathcal{R}$.

Method 1	$\tau_{i,j} \sim 2\mathcal{GE}(\epsilon)$ $x_{i,j} \sim \mathcal{PO}(\lambda_{i,j})$ where $\lambda_{i,j} = U_i V_j$ $z_{i,j} = x_{i,j} + \tau_{i,j}$
Method 2	$t_{i,j,n} \sim \mathcal{EXP}(\frac{\epsilon}{1-\epsilon})$ for $n \in \{1,2\}$ $g_{i,j,n} \sim \mathcal{PO}(t_{i,j,n})$ for $n \in \{1,2\}$ $x_{i,j} \sim \mathcal{PO}(\lambda_{i,j})$ $z_{i,j} = x_{i,j} + g_{i,j,1} - g_{i,j,2}$
Method 3	$t_{i,j,n} \sim \mathcal{EXP}(\frac{\epsilon}{1-\epsilon})$ for $n \in \{1,2\}$ $z_{i,j} = \text{Skellam}(t_{i,j,1} + \lambda_{i,j}, t_{i,j,2})$

APPENDIX E: Noise Computation for DPD

In this section, we compute the noise for DPD algorithm by using both AC and ZCDP. For the DP-CTF (AC), we use Advanced Composition (Theorem 4.6), Gaussian Mechanism (Theorem 4.7) and Privacy for subsampled data (Theorem 4.8) to compute the noise for each step for DP-CTF (and DP-CMF) algorithms. However, we need to solve $\delta_{iter} = (\delta_{tot} - \delta')/T\nu$ in order to use δ_{iter} in the following equations.

$$\epsilon_{tot} = \sqrt{2T\log(1/\delta')}/\sigma' + T(\sigma')^2 \qquad \text{(Advanced composition)}$$

$$\sigma' = \log\left(1 + \nu(\exp(\sqrt{2\log(1.25/\delta_{iter})}/\sigma) - 1)\right) \qquad \text{(Privacy for subsampled data)}$$

$$\epsilon_{iter} = \sqrt{2\log(1.25/\delta_{iter})}/\sigma \qquad \text{(Gaussian mechanism)}$$

We can compute the amount of noise/privacy budget per iteration in three steps:

- Compute σ' by setting $A = 2T$, $B = \sqrt{2T\log(1/\delta')}$, $C = -\epsilon$ and solving the equation $\sigma' = (-B + \sqrt{B^2 - 4AC})/(2A)$;
- Compute $\epsilon_{iter} = \log(((\exp(\sigma') - 1)/\nu) + 1)$;
- Compute $\sigma = \sqrt{2\log(1.25/\delta_{iter})}/\epsilon_{iter}$.

For the DP-CTF (zCDP) we also use the Proposition 4.9, Proposition 4.10 and Lemma 4.11 to convert DP noise to zCDP. First we need to convert the ϵ_{tot} of DP to new total amount of noise for zCDP case by applying the Equation 4.28. The same amount of privacy ϵ_{tot} is defined as $\epsilon_{tot} = T/(2\tau) + 2\sqrt{T/(2\tau)\log(1/\delta)}$ and δ' equals to $\nu\delta_{iter}$. Now, we can compute the amount of noise/privacy budget per iteration in these three steps:

- Compute $\epsilon' = (-B + \sqrt{B^2 - 4AC})/(2A)$ from given ϵ_{tot} where $A = T/(4\log(1.25/\delta'))$, $B = \sqrt{T\log(1/\delta)}/\log(1.25/\delta')$, $C = -\epsilon_{tot}$ and $\delta' = \nu\delta_{iter}$;
- Compute $\epsilon_{iter} = \log(((\exp(\epsilon') - 1)/\nu) + 1)$;
- Compute $\sigma = \sqrt{2\log(1.25/\delta_{iter})}/\epsilon_{iter}$.

Eventually, the Gaussian noise ξ' is computed as $\xi' \sim \mathcal{N}(0, 4G^2\sigma^2\mathbb{I})$.

REFERENCES

1. Hua, J., C. Xia and S. Zhong, "Differentially Private Matrix Factorization", *IJCAI 2015*, pp. 1763–1770, 2015.

2. Liu, Z., Y.-X. Wang and A. Smola, "Fast differentially private matrix factorization", *Proceedings of the 9th ACM Conference on Recommender Systems*, pp. 171–178, ACM, 2015.

3. Caruana, R., "Multitask learning", *Machine learning*, Vol. 28, No. 1, pp. 41–75, 1997.

4. Argyriou, A., T. Evgeniou and M. Pontil, "Multi-task feature learning", *Advances in neural information processing systems*, pp. 41–48, 2007.

5. Yang, Y. and T. Hospedales, "Deep multi-task representation learning: A tensor factorisation approach", *arXiv preprint arXiv:1605.06391*, 2016.

6. Alonso, H. M. and B. Plank, "When is multitask learning effective? Semantic sequence prediction under varying data conditions", *arXiv preprint arXiv:1612.02251*, 2016.

7. Liu, S., S. J. Pan and Q. Ho, "Distributed multi-task relationship learning", *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 937–946, ACM, 2017.

8. Bach, F. R., G. R. Lanckriet and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm", *Proceedings of the twenty-first international conference on Machine learning*, p. 6, ACM, 2004.

9. Sonnenburg, S., G. Rätsch, C. Schäfer and B. Schölkopf, "Large scale multiple kernel learning", *Journal of Machine Learning Research*, Vol. 7, No. Jul, pp. 1531–1565, 2006.

10. Gönen, M. and E. Alpaydin, "Localized multiple kernel learning", *Proceedings of the*

25th international conference on Machine learning, pp. 352–359, ACM, 2008.

11. Gönen, M. and E. Alpaydın, "Multiple kernel learning algorithms", *Journal of machine learning research*, Vol. 12, No. Jul, pp. 2211–2268, 2011.

12. Singh, A. P. and G. J. Gordon, "Relational Learning via Collective Matrix Factorization", *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pp. 650–658, 2008.

13. Bouchard, G., D. Yin and S. Guo, "Convex collective matrix factorization", *Artificial Intelligence and Statistics*, pp. 144–152, 2013.

14. Klami, A., G. Bouchard and A. Tripathi, "Group-sparse embeddings in collective matrix factorization", *arXiv preprint arXiv:1312.5921*, 2013.

15. Yılmaz, K. Y., A. T. Cemgil and U. Simsekli, "Generalised coupled tensor factorisation", *Advances in neural information processing systems*, pp. 2151–2159, 2011.

16. Acar, E., T. G. Kolda and D. M. Dunlavy, "All-at-once Optimization for Coupled Matrix and Tensor Factorizations", *MLG'11: Proceedings of Mining and Learning with Graphs*, August 2011.

17. Şimşekli, U., B. Ermiş, A. T. Cemgil and E. Acar, "Optimal weight learning for Coupled Tensor Factorization with mixed divergences", *21st European Signal Processing Conference (EUSIPCO 2013)*, pp. 1–5, 2013.

18. Ermiş, B., E. Acar and A. T. Cemgil, "Link prediction in heterogeneous data via generalized coupled tensor factorization", *Data Mining and Knowledge Discovery*, Vol. 29, No. 1, pp. 203–236, 2015.

19. Zheng, V. W., B. Cao, Y. Zheng, X. Xie and Q. Yang, "Collaborative Filtering Meets Mobile Recommendation: A User-Centered Approach.", *AAAI*, Vol. 10, pp. 236–241, 2010.

20. Lee, C. M., M. A. Mudaliar, D. Haggart, C. R. Wolf, G. Miele, J. K. Vass, D. J. Higham and D. Crowther, "Simultaneous non-negative matrix factorization for multiple large scale gene expression datasets in toxicology", *PloS one*, Vol. 7, No. 12, p. e48238, 2012.

21. Acar, E., A. J. Lawaetz, M. A. Rasmussen and R. Bro, "Structure-revealing data fusion model with applications in metabolomics", *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, pp. 6023–6026, IEEE, 2013.

22. Wu, Q., L.-Q. Zhang and G.-C. Shi, "Robust feature extraction for speaker recognition based on constrained nonnegative tensor factorization", *Journal of computer science and technology*, Vol. 25, No. 4, pp. 783–792, 2010.

23. Beckmann, C. F. and S. M. Smith, "Tensorial extensions of independent component analysis for multisubject FMRI analysis", *Neuroimage*, Vol. 25, No. 1, pp. 294–311, 2005.

24. Smilde, A., R. Bro and P. Geladi, *Multi-way analysis: applications in the chemical sciences*, John Wiley & Sons, 2005.

25. Dwork, C., F. McSherry, K. Nissim and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis", *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pp. 265–284, Springer-Verlag, Berlin, Heidelberg, 2006.

26. Dwork, C. and A. Smith, "Differential privacy for statistics: What we know and what we want to learn", *Journal of Privacy and Confidentiality*, Vol. 1, No. 2, p. 2, 2010.

27. Wang, Y., S. E. Fienberg and A. J. Smola, "Privacy for Free: Posterior Sampling and Stochastic Gradient Monte Carlo", *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 2493–2502, 2015.

28. Li, B., C. Chen, H. Liu and L. Carin, "On Connecting Stochastic Gradient MCMC and Differential Privacy", *arXiv preprint arXiv:1712.09097*, 2017.

29. Welling, M. and Y. W. Teh, "Bayesian learning via Stochastic Gradient Langevin Dynamics", *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.

30. Chien, J.-T. and Y.-T. Bao, "Tensor-factorized neural networks", *IEEE transactions on neural networks and learning systems*, Vol. 29, No. 5, pp. 1998–2011, 2018.

31. Wu, X., B. Shi, Y. Dong, C. Huang and N. Chawla, "Neural Tensor Factorization", *arXiv preprint arXiv:1802.04416*, 2018.

32. Mondelli, M. and A. Montanari, "On the Connection Between Learning Two-Layers Neural Networks and Tensor Decomposition", *arXiv preprint arXiv:1802.07301*, 2018.

33. Khrulkov, V., O. Hrinchuk and I. Oseledets, "Generalized Tensor Models for Recurrent Neural Networks", *arXiv preprint arXiv:1901.10801*, 2019.

34. Harshman, R., "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis", *UCLA Working Papers in Phonetics*, Vol. 16, 1970.

35. Long, B., Z. M. Zhang, X. Wu and P. S. Yu, "Spectral Clustering for Multi-type Relational Data", *ICML'06*, pp. 585–592, 2006.

36. Banerjee, A., S. Basu and S. Merugu, "Multi-way Clustering on Relation Graphs", *SDM'07*, pp. 145–156, 2007.

37. Smilde, A., J. A. Westerhuis and R. Boque, "Multiway multiblock component and covariates regression models", *Journal of Chemometrics*, Vol. 14, pp. 301–331, 2000.

38. Zheng, V. W., Y. Zheng, X. Xie and Q. Yang, "Towards mobile intelligence: Learning from GPS history data for collaborative recommendation", *Artificial Intelligence*, Vol. 184-185, pp. 17–37, 2012.

39. Lin, Y.-R., J. Sun, P. Castro, R. Konuru, H. Sundaram and A. Kelliher, "Metafac:

community discovery via relational hypergraph factorization", *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 527–536, ACM, 2009.

40. Choi, D., J.-G. Jang and U. Kang, "Fast, accurate, and scalable method for sparse coupled matrix-tensor factorization", *arXiv preprint arXiv:1708.08640*, 2017.

41. Tucker, L. R., "Implications of factor analysis of three-way matrices for measurement of change", C. W. Harris (Editor), *Problems in measuring change.*, pp. 122–137, University of Wisconsin Press, 1963.

42. Takeuchi, K., R. Tomioka, K. Ishiguro, A. Kimura and H. Sawada, "Non-negative multiple tensor factorization", *2013 IEEE 13th International Conference on Data Mining*, pp. 1199–1204, IEEE, 2013.

43. Khan, S. A. and S. Kaski, "Bayesian multi-view tensor factorization", *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 656–671, Springer, 2014.

44. Ermis, B. and A. T. Cemgil, "A Bayesian tensor factorization model via variational inference for link prediction", *arXiv preprint arXiv:1409.8276*, 2014.

45. Khan, S. A., E. Leppäaho and S. Kaski, "Bayesian multi-tensor factorization", *Machine Learning*, Vol. 105, No. 2, pp. 233–253, 2016.

46. Kaya, K., F. Öztoprak, Ş. İ. Birbil, A. T. Cemgil, U. Şimşekli, N. Kuru, H. Koptagel and M. K. Öztürk, "HAMSI: A Parallel Incremental Optimization Algorithm Using Quadratic Approximations for Solving Partially Separable Problems", *arXiv preprint arXiv:1509.01698*, 2015.

47. Tang, Y., D. Chen, L. Wang, A. Y. Zomaya, J. Chen and H. Liu, "Bayesian tensor factorization for multi-way analysis of multi-dimensional EEG", *Neurocomputing*, Vol. 318, pp. 162–174, 2018.

48. Balasubramaniam, T., R. Nayak and C. Yuen, "Nonnegative coupled matrix tensor factorization for smart city spatiotemporal pattern mining", , 2018.

49. Wu, Q., J. Wang, J. Fan, G. Xu, J. Wu, B. Johnson, X. Li, Q. Do and R. Ge, "Improved Coupled Tensor Factorization with Its Applications in Health Data Analysis", *Complexity*, Vol. 2019, 2019.

50. Williams, O. and F. McSherry, "Probabilistic inference and differential privacy", *Advances in Neural Information Processing Systems*, pp. 2451–2459, 2010.

51. Dimitrakakis, C., B. Nelson, A. Mitrokotsa and B. I. Rubinstein, "Robust and private Bayesian inference", *Algorithmic Learning Theory*, pp. 291–305, Springer, 2014.

52. Zhang, Z., B. I. P. Rubinstein and C. Dimitrakakis, "On the Differential Privacy of Bayesian Inference", *AAAI 2016*, 2016.

53. Foulds, J. R., J. Geumlek, M. Welling and K. Chaudhuri, "On the Theory and Practice of Privacy-Preserving Bayesian Data Analysis", *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2016.

54. Park, M., J. Foulds, K. Chaudhuri and M. Welling, "Variational Bayes in private settings (VIPS)", *arXiv preprint arXiv:1611.00340*, 2016.

55. Park, M., J. R. Foulds, K. Choudhary and M. Welling, "DP-EM: Differentially Private Expectation Maximization", *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pp. 896–904, 2017.

56. Heikkilä, M., Y. Okimoto, S. Kaski, K. Shimizu and A. Honkela, "Differentially Private Bayesian Learning on Distributed Data", *arXiv preprint arXiv:1703.01106*, 2017.

57. Schein, A., Z. S. Wu, M. Zhou and H. Wallach, "Locally Private Bayesian Inference for Count Models", *arXiv preprint arXiv:1803.08471*, 2018.

58. Mir, D. J., "Information-theoretic foundations of differential privacy", *Foundations and Practice of Security*, pp. 374–381, Springer, 2013.

59. Kairouz, P., S. Oh and P. Viswanath, "Extremal mechanisms for local differential privacy", *Advances in neural information processing systems*, pp. 2879–2887, 2014.

60. Duchi, J. C., M. I. Jordan and M. J. Wainwright, "Local privacy, data processing inequalities, and minimax rates", *arXiv:1302.3203*, 2013.

61. Duchi, J. C., M. I. Jordan and M. J. Wainwright, "Local privacy and statistical minimax rates", *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pp. 429–438, IEEE, 2013.

62. Barber, R. F. and J. C. Duchi, "Privacy and Statistical Risk: Formalisms and Minimax Bounds", *arXiv:1412.4451*, 2014.

63. Joseph, M., A. Roth, J. Ullman and B. Waggoner, "Local Differential Privacy for Evolving Data", *arXiv preprint arXiv:1802.07128*, 2018.

64. Sarwate, A., S. Plis, J. Turner, M. Arbabshirani and V. Calhoun, "Sharing privacy-sensitive access to neuroimaging and genetics data: a review and preliminary validation", *Frontiers in Neuroinformatics*, Vol. 8, No. 35, 2014.

65. Rajkumar, A. and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification", *International Conference on Artificial Intelligence and Statistics*, pp. 933–941, 2012.

66. Imtiaz, H. and A. D. Sarwate, "Distributed Differentially-Private Algorithms for Matrix and Tensor Factorization", *arXiv preprint arXiv:1804.10299*, 2018.

67. Pathak, M., S. Rane and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers", *Advances in Neural Information Processing Systems*, pp. 1876–1884, 2010.

68. Hamm, J., P. Cao and M. Belkin, "Learning Privately from Multiparty Data", *CoRR*, Vol. abs/1602.03552, 2016, http://arxiv.org/abs/1602.03552.

69. Shokri, R. and V. Shmatikov, "Privacy-Preserving Deep Learning", *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pp. 1310–1321, 2015.

70. Papernot, N., M. Abadi, Ú. Erlingsson, I. J. Goodfellow and K. Talwar, "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data", *CoRR*, Vol. abs/1610.05755, 2016.

71. Gupta, S. K., S. Rana and S. Venkatesh, "Differentially Private Multi-task Learning", *Proceedings of the 11th Pacific Asia Workshop on Intelligence and Security Informatics - Volume 9650*, pp. 101–113, Springer-Verlag New York, Inc., New York, NY, USA, 2016.

72. Xie, L., I. M. Baytas, K. Lin and J. Zhou, "Privacy-Preserving Distributed Multi-Task Learning with Asynchronous Updates", *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pp. 1195–1204, ACM, New York, NY, USA, 2017.

73. Nikolaenko, V., S. Ioannidis, U. Weinsberg, M. Joye, N. Taft and D. Boneh, "Privacy-preserving matrix factorization", *Proceedings of the 2013 ACM SIGSAC*, pp. 801–812, ACM, 2013.

74. Balu, R. and T. Furon, "Differentially private matrix factorization using sketching techniques", *ACM Workshop on Information Hiding and Multimedia Security*, 2016.

75. Friedman, A., S. Berkovsky and M. A. Kaafar, "A differential privacy framework for matrix factorization recommender systems", *User Modeling and User-Adapted Interaction*, Vol. 26, No. 5, pp. 425–458, 2016.

76. Wang, Y. and A. Anandkumar, "Online and differentially-private tensor decomposition",

Advances in Neural Information Processing Systems, pp. 3531–3539, 2016.

77. Phan, N., Y. Wang, X. Wu and D. Dou, "Differential Privacy Preservation for Deep Auto-Encoders: an Application of Human Behavior Prediction", *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pp. 1309–1316, 2016.

78. Phan, N., X. Wu, H. Hu and D. Dou, "Adaptive Laplace Mechanism: Differential Privacy Preservation in Deep Learning", *arXiv preprint arXiv:1709.05750*, 2017.

79. Acs, G., L. Melis, C. Castelluccia and E. De Cristofaro, "Differentially Private Mixture of Generative Neural Networks", *arXiv preprint arXiv:1709.04514*, 2017.

80. Phan, N., X. Wu and D. Dou, "Preserving differential privacy in convolutional deep belief networks", *Machine Learning*, Vol. 106, No. 9-10, pp. 1681–1704, 2017.

81. Abadi, M., A. Chu, I. Goodfellow, B. McMahan, I. Mironov, K. Talwar and L. Zhang, "Deep Learning with Differential Privacy", *23rd ACM Conference on Computer and Communications Security (ACM CCS)*, pp. 308–318, 2016, https://arxiv.org/abs/1607.00133.

82. Neelakantan, A., L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach and J. Martens, "Adding gradient noise improves learning for very deep networks", *arXiv preprint arXiv:1511.06807*, 2015.

83. Dwork, C. and G. N. Rothblum, "Concentrated differential privacy", *arXiv:1603.01887*, 2016.

84. Bun, M. and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds", *Theory of Cryptography Conference*, pp. 635–658, Springer, 2016.

85. Li, C., A. Stevens, C. Chen, Y. Pu, Z. Gan and L. Carin, "Learning weight uncertainty with stochastic gradient mcmc for shape classification", *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pp. 5666–5675, 2016.

86. Cichocki, A., R. Zdunek, A. Phan and S. Amari, *Nonnegative Matrix and Tensor Factorization*, Wiley, 2009.

87. Dwork, C., "Differential privacy", *ICALP*, pp. 1–12, Springer, 2006.

88. Dwork, C. and J. Lei, "Differential privacy and robust statistics", *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 371–380, ACM, 2009.

89. Dwork, C. and A. Roth, "The algorithmic foundations of differential privacy", *Theoretical Computer Science*, Vol. 9, No. 3-4, pp. 211–407, 2013.

90. Holohan, N., D. J. Leith and O. Mason, "Extreme points of the local differential privacy polytope", *Linear Algebra and its Applications*, Vol. 534, pp. 78–96, 2017.

91. Song, S., K. Chaudhuri and A. D. Sarwate, "Learning from Data with Heterogeneous Noise using SGD", *arXiv:1412.5617*, 2014.

92. S., W., "Randomized response: a survey technique for eliminating evasive answer bias", *Journal of the American Statistical Association*, Vol. 60, No. 309, pp. 63–69, 1965.

93. Dwork, C. and A. Roth, "The Algorithmic Foundations of Differential Privacy", *Foundations and Trends in Theoretical Computer Science*, Vol. 9, No. 3-4, pp. 211–407, 2014.

94. Robert, C. P. and G. Casella, *Monte Carlo Statistical Methods*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

95. McSherry, F. and K. Talwar, "Mechanism design via differential privacy", *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pp. 94–103, IEEE, 2007.

96. Neal, R. M. *et al.*, "MCMC using Hamiltonian dynamics", *Handbook of Markov Chain*

Monte Carlo, Vol. 2, No. 11, p. 2, 2011.

97. Gilks, W. R., S. Richardson and D. Spiegelhalter, *Markov Chain Monte Carlo in practice*, CRC press, 1995.

98. Chib, S. and E. Greenberg, "Understanding the Metropolis-Hastings algorithm", *The american statistician*, Vol. 49, No. 4, pp. 327–335, 1995.

99. Kent, J., "Time-reversible diffusions", *Advances in Applied Probability*, Vol. 10, No. 4, pp. 819–835, 1978.

100. Glorot, X., A. Bordes and Y. Bengio, "Deep sparse rectifier neural networks", *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

101. Csáji, B. C., "Approximation with artificial neural networks", *Faculty of Sciences, Etvs Lornd University, Hungary*, Vol. 24, p. 48, 2001.

102. Hernández-Lobato, J. M. and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks", *International Conference on Machine Learning*, pp. 1861–1869, 2015.

103. Lee, D. D. and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization", *Nature*, Vol. 401, No. 6755, p. 788, 1999.

104. Hoyer, P. O., "Non-negative matrix factorization with sparseness constraints", *Journal of Machine Learning Research*, Vol. 5, No. Nov, pp. 1457–1469, 2004.

105. Koren, Y., R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems", *Computer*, Vol. 42, No. 8, pp. 30–37, 2009.

106. Salakhutdinov, R. and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo", *Proceedings of the 25th international conference on Machine learning*, pp. 880–887, ACM, 2008.

107. Porteous, I., A. U. Asuncion and M. Welling, "Bayesian Matrix Factorization with Side Information and Dirichlet Process Mixtures.", *AAAI*, 2010.

108. Şimçekli, U., R. Badeau, G. Richard and A. T. Cemgil, "Stochastic thermodynamic integration: efficient Bayesian model selection via stochastic gradient MCMC", *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 2574–2578, IEEE, 2016.

109. Cemgil, A. T., "Bayesian inference for nonnegative matrix factorisation models", *Computational Intelligence and Neuroscience*, 2009.

110. Yilmaz, Y. K. and A. T. Cemgil, "Probabilistic Latent Tensor Factorization", *LVA/ICA*, pp. 346–353, 2010.

111. Bishop, C. M., *Pattern Recognition and Machine Learning (Information Science and Statistics*, Springer, 2007.

112. Ghahramani, Z. and M. J. Beal, "Propagation algorithms for variational Bayesian learning", *Advances in neural information processing systems*, pp. 507–513, 2001.

113. Ahn, S., A. Korattikara, N. Liu, S. Rajan and M. Welling, "Large-scale distributed Bayesian matrix factorization using stochastic gradient MCMC", *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 9–18, ACM, 2015.

114. Geman, S. and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images", *Readings in Computer Vision*, pp. 564–584, Elsevier, 1987.

115. Carlo, C. M., "Markov Chain Monte Carlo and Gibbs sampling", *Lecture notes for EEB*, Vol. 581, 2004.

116. Bassily, R., A. Smith and A. Thakurta, "Private empirical risk minimization, revisited", *arXiv:1405.7085*, 2014.

117. Chaudhuri, K., C. Monteleoni and A. D. Sarwate, "Differentially private empirical risk minimization", *Journal of Machine Learning Research*, Vol. 12, pp. 1069–1109, 2011.

118. Tewari, A. and S. Chaudhuri, "On Lipschitz Continuity and Smoothness of Loss Functions in Learning to Rank", *arXiv:1405.0586*, 2014.

119. Ghosh, A., T. Roughgarden and M. Sundararajan, "Universally utility-maximizing privacy mechanisms", *SIAM Journal on Computing*, Vol. 41, No. 6, pp. 1673–1693, 2012.

120. Pascanu, R., T. Mikolov and Y. Bengio, "On the difficulty of training recurrent neural networks", *International Conference on Machine Learning*, pp. 1310–1318, 2013.

121. Song, S., K. Chaudhuri and A. D. Sarwate, "Stochastic gradient descent with differentially private updates", *IEEE Global Conference on Signal and Information Processing*, 2013.

122. Li, C., C. Chen, D. E. Carlson and L. Carin, "Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks.", *AAAI*, Vol. 2, p. 4, 2016.

123. Li, N., W. Qardaji and D. Su, "On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy", *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pp. 32–33, ACM, 2012.

124. LeCun, Y., 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, 1998.

125. Maddison, C. J., A. Huang, I. Sutskever and D. Silver, "Move Evaluation in Go Using Deep Convolutional Neural Networks", *CoRR*, Vol. abs/1412.6564, 2014, http://arxiv.org/abs/1412.6564.

126. Vinyals, O., L. u. Kaiser, T. Koo, S. Petrov, I. Sutskever and G. Hinton, "Grammar as a Foreign Language", C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett

(Editors), *Advances in Neural Information Processing Systems 28*, pp. 2773–2781, 2015.

127. Schmidhuber, J., "Deep learning in neural networks: An overview", *Neural networks*, Vol. 61, pp. 85–117, 2015.

128. Gal, Y., "Uncertainty in deep learning", *University of Cambridge*, 2016.

129. Yang, X., R. Kwitt and M. Niethammer, "Fast predictive image registration", *Deep Learning and Data Labeling for Medical Applications*, pp. 48–57, Springer, 2016.

130. Herzog, S. and D. Ostwald, "Experimental biology: sometimes Bayesian statistics are better", *Nature*, Vol. 494, No. 7435, p. 35, 2013.

131. Bojarski, M., D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars", *arXiv preprint arXiv:1604.07316*, 2016.

132. MacKay, D. J., "Probable networks and plausible predictions—a review of practical Bayesian methods for supervised neural networks", *Network: Computation in Neural Systems*, Vol. 6, No. 3, pp. 469–505, 1995.

133. Neal, R. M., *Bayesian learning for neural networks*, Vol. 118, Springer Science & Business Media, 2012.

134. Graves, A., "Practical variational inference for neural networks", *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

135. Blundell, C., J. Cornebise, K. Kavukcuoglu and D. Wierstra, "Weight uncertainty in neural networks", *arXiv preprint arXiv:1505.05424*, 2015.

136. Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", *arXiv preprint arXiv:1207.0580*, 2012.

137. Wang, S. and C. Manning, "Fast dropout training", S. Dasgupta and D. McAllester (Editors), *Proceedings of the 30th International Conference on Machine Learning*, Vol. 28 of *Proceedings of Machine Learning Research*, pp. 118–126, PMLR, 17–19 Jun 2013.

138. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, Vol. 15, No. 1, pp. 1929–1958, 2014.

139. Kingma, D. P., T. Salimans and M. Welling, "Variational dropout and the local reparameterization trick", *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

140. Gal, Y. and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning", *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, 2016.

141. Robbins, H. and S. Monro, "A Stochastic Approximation Method", *Ann. Math. Statist.*, Vol. 22, No. 3, pp. 400–407, 1951.

142. Wan, L., M. Zeiler, S. Zhang, Y. Le Cun and R. Fergus, "Regularization of neural networks using dropconnect", *International Conference on Machine Learning*, pp. 1058–1066, 2013.

143. Stager, M., P. Lukowicz and G. Troster, "Dealing with class skew in context recognition", *Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on*, pp. 58–58, IEEE, 2006.

144. Sanderson, M., "Test Collection Based Evaluation of Information Retrieval Systems.", *Foundations and Trends in Information Retrieval*, Vol. 4, No. 4, pp. 247–375, 2010.

145. Zheng, V. W., B. Cao, Y. Zheng, X. Xie and Q. Yang, "Collaborative Filtering Meets Mobile Recommendation: A User-centered Approach", *AAAI'10*, 2010.

146. Candès, E. J. and Y. Plan, "Matrix Completion With Noise", *Proceedings of the IEEE*, Vol. 98, pp. 925–936, 2010.

147. Narita, A., K. Hayashi, R. Tomioka and H. Kashima, "Tensor Factorization Using Auxiliary Information", *ECML PKDD '11*, pp. 501–516, 2011.

148. De Choudhury, M., H. Sundaram, A. John and D. D. Seligmann, "Social synchrony: Predicting mimicry of user actions in online social media", *2009 international conference on computational science and engineering*, pp. 151–158, IEEE, 2009.

149. Nakajima, S. and M. Sugiyama, "Implicit Regularization in Variational Bayesian Matrix Factorization", *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 815–822, USA, 2010.

150. Hoffman, M. D., D. M. Blei, C. Wang and J. W. Paisley, "Stochastic variational inference", *Journal of Machine Learning Research*, Vol. 14, No. 1, pp. 1303–1347, 2013.

151. Bache, K. and M. Lichman, "UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences", , 2013.

152. Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions", *arXiv e-prints*, Vol. abs/1605.02688, May 2016.

153. Balle, B. and Y.-X. Wang, "Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising", *arXiv preprint arXiv:1805.06530*, 2018.

154. Balle, B., G. Barthe and M. Gaboardi, "Privacy amplification by subsampling: Tight analyses via couplings and divergences", *arXiv preprint arXiv:1807.01647*, 2018.

155. Chen, T., E. B. Fox and C. Guestrin, "Stochastic Gradient Hamiltonian Monte Carlo", *Proceedings of the ICML 2014*, pp. 1683–1691, 2014.

156. Patterson, S. and Y. W. Teh, "Stochastic gradient Riemannian Langevin dynamics on the probability simplex", *Advances in Neural Information Processing Systems*, pp.

3102–3110, 2013.

157. Skellam, J. G., “The frequency distribution of the difference between two Poisson variates belonging to different populations.”, *Journal of the Royal Statistical Society. Series A (General)*, Vol. 109, No. Pt 3, pp. 296–296, 1946.