Multi-Party Privacy-Preserving Machine Learning and Its Applications

---

A Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment
of the requirements for the degree

Doctor of Philosophy

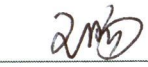
by

Yang Wang

August 2018

APPROVAL SHEET

This Dissertation
is submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Author Signature: 

This Dissertation has been read and approved by the examining committee:

Advisor: Donald Brown

Committee Member: Laura Barnes

Committee Member: Peter Beling

Committee Member: Steven Boker

Committee Member: QuanQuan Gu

Committee Member: _____

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, School of Engineering and Applied Science

August 2018

Doctoral Dissertation

Multi-Party Privacy-Preserving Machine Learning and Its Applications

Author:

Yang Wang

Advisor:

Dr. Donald Brown

A dissertation submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

in the

Department of Systems and Information Engineering

University of Virginia

August, 2018

Dissertation Committee Members

- Prof. Laura Barnes (Committee Member, Systems and Information Engineering)
- Prof. Peter Beling (Committee Chair, Systems and Information Engineering)
- Prof. Steven Boker (Co-advisor, Psychology)
- Prof. Donald Brown (Advisor, Systems and Information Engineering)
- Prof. Quanquan Gu (Committee Member, Computer Science, University of California, Los Angeles)

Abstract

Conducting machine learning and statistical analytics in a distributed manner while maintaining data privacy can be beneficial to a wide variety of scientific investigations involving human subject data. As there is usually a tension between privacy protection and aspiration for high quality data analytics, developing and adapting machine learning algorithms under a rigorous and customizable framework for privacy is highly urgent and desirable. Moreover, the advance of personal mobile devices and modern communication technology has foster a rapid growth of distributed collection and storage of data. How to perform distributed data analysis tasks (classifier learning, hypothesis testing, etc.) without access to raw personal data becomes a challenging yet intriguing problem. The two research areas involved in addressing this problem, multi-party machine learning and privacy-preserving techniques, are separately well established. However, an interdisciplinary integration of the research efforts from both areas has been lacking until recent years. In this dissertation, our primary goal is to bridge the gap by designing different privacy-preserving machine learning models (logistic regression, feedforward neural network and transfer learning) using different privacy protection techniques (differential privacy and traditional cryptographic techniques). The model performance, especially the utility-privacy trade-off is further evaluated on data sets from a variety of domains. Our work provides new perspectives and solutions to current privacy concerns, and hopefully directions for future research.

Acknowledgements

First and foremost I would like to express the deepest appreciation to my supervisor Professor Donald Brown, and my co-supervisor Professor Steven Boker. Without their constant guidance and persistent help in the development of research ideas throughout my study at University of Virginia, this dissertation would not have been possible.

I would also like to thank my committee members, Professor Laura Barnes, Professor Peter Beling and Professor Quanquan Gu for their valuable feedback and advice on my qualifying exams and dissertation proposal. I highly enjoy the academic collaboration with Professor Beling and Professor Gu on two of my publications and deeply appreciate their active engagement and valuable inspiration.

In addition, my sincere thanks go to Professor Timo von Oertzen at Universität der Bundeswehr München and Dr. Joshua Pritikin at Virginia Commonwealth University for their financial support, academic advice and personal encouragement on the MIDDLE project. I am very thankful to Dr. Steve Greenspan and his team at CA Technologies for providing funding and expertise to my research projects. I also appreciate and acknowledge the stimulating academic discussions with Dr. Wei Xie at Vanderbilt University. Moreover, I would like to thank all my friends and basketball group members at University of Virginia for all the fun we had. Without their precious support it would not be possible to conduct the research work in this dissertation while having such enjoyable times at Charlottesville.

Lastly, I would like to thank my family for all their unconditional love. My parents and my wife Ying Xiong always support me in all my pursuits and I am deeply indebted to them for all their understanding and encouragement. This dissertation is also dedicated to my dear little daughter Joanne, who has made me a stronger and more fulfilled person than I could have ever imagined.

To my beloved family

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	**A**rtificial **N**eural **N**etwork
CIPSEA	**C**onfidential **I**nformation **P**rotection and **S**tatistical **E**fficiency **A**ct
CNN	**C**onvolutional **N**eural **N**etwork
DNN	**D**eep **N**eural **N**etwork
DP	**D**ifferential **P**rivacy
ERM	**E**mpirical **R**isk **M**inimization
FCRA	**F**air **C**redit **R**eporting **A**ct
FERPA	**F**amily **E**ducational **R**ights and **P**rivacy **A**ct
FNN	**F**eedforward **N**eural **N**etwork
GAN	**G**enerative **A**dversarial **N**etwork
HIPAA	**H**ealth **I**nsurance **P**ortability and **A**ccountability **A**ct
IRB	**I**nstitutional **R**eview **B**oard
MIDDLE	**M**aintained **I**ndividual **D**ata **D**istributed **L**ikelihood **E**stimation
MLE	**M**aximum **L**ikelihood **E**stimate
MLP	**M**ulti**L**ayer **P**erceptrons
RNN	**R**ecurrent **N**eural **N**etwork
SGD	**S**tochastic **G**radient **D**escent
SMC	**S**ecure **M**ulti-party **C**omputation
SVM	**S**upport **V**ector **M**achine

Chapter 1

Introduction

1.1 Motivation

In various research fields (e.g., sociology, quantitative psychology, healthcare), sensitive identifiable human subjects data are often involved and exploration of such data must comply with a large body of privacy laws (e.g., HIPAA, FERPA, FCRA and CIPSEA). Failure to satisfy these legal requirements and mishandling of personally identifiable information in recent years have caused serious damage to public trust in the government and/or business and raised awareness about privacy protection globally. For example, in 2018 it was reported that around 87 million Facebook users' personal information were abused for illegal data mining by an outside company Cambridge Analytica without users' or Facebook's consent. The Facebook privacy scandal[1] stimulated a new round of discussion about user privacy and security, following several heated debates on similar issues such as the FBI-Apple encryption dispute[2].

Moreover, increasing concern over data privacy may impose restrictions or barriers to data sharing and make it difficult to coordinate large-scale collaborative studies. In modern data mining and machine learning, it is prevalent that data sets are collected by multiple parties (individuals or data centers) and the traditional practice of centralizing data before performing data analysis is both inefficient and privacy-breaching. As a concrete example, consider the problem faced by a healthcare research team, who designs

[1] https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal
[2] https://en.wikipedia.org/wiki/FBI-Apple_encryption_dispute

an experiment to analyze the sensitive personal data collected by different hospitals or personal wearable computing devices such as cell phones. It is crucial for the research team to have access to massive amounts of data in order to learn valuable information of the population and make reliable predictions about emerging diseases. However, the data owners (individuals or hospitals) are reluctant to participate in the study due to privacy concerns. Note that the privacy concerns here have multifold meanings. First, the data owners may be unwilling to contribute their sensitive data simply because they don't trust the research team. Second, even if they voluntarily consent to participate and allow access to their data, the participants still don't want their involvement to be revealed because the study itself is sensitive (e.g., HIV/AIDS research, sexual orientation study). Therefore, to facilitate such a collaborative data analysis task among multiple parties, designing privacy-preserving distributed machine learning algorithms under a rigorous privacy framework is highly desirable.

There have been relentless efforts to address this prominent problem in numerous research areas. In particular, Boker et al. [19] proposed a paradigm-shifting framework named "Maintained Individual Data Distributed Estimated Likelihood" (MIDDLE) which revolutionizes the existing research work flow in the area of psychology and behavioral study. Our work in this dissertation is mainly inspired by the idea of MIDDLE.

1.2 The Maintained Individual Data Distributed Estimated Likelihood Framework

The traditional practices of behavioral and psychological experimental design share a common sequence of events (see Figure 1.1). First, a testable hypothesis is proposed and experimental methodology is designed. Next, experiment participants are recruited and data are collected into a centralized repository to fit candidate statistical models (commonly by obtaining maximum likelihood estimates (MLE) or Bayesian estimates of model parameters). The data are generally considered to belong to the research group

that collects the observational data. Experiment results are then disseminated through journal articles and/or conference talks. Finally, other research groups may replicate the experiments and generate new hypotheses based on the previous results.

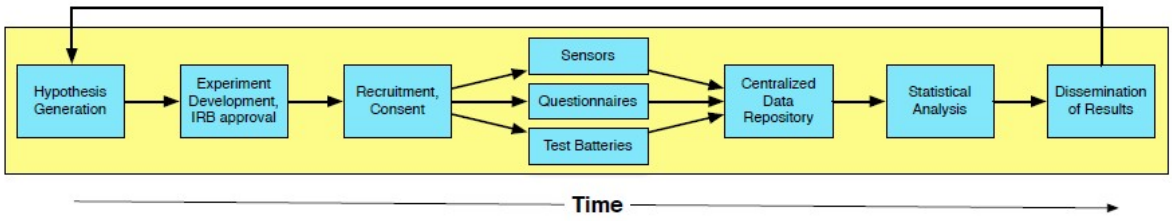


FIGURE 1.1: Traditional experiment flowchart, from [19]

However, there are several inherent problems with this workflow:

1. There are conflicting views on who owns the data which cause the legitimate concern despite the existence of various privacy protection policies.

2. Privacy protection creates barriers to data sharing and longitudinal data linking.

3. There can be a very long interval between the generation of a hypothesis and the next opportunity for the hypothesis to be revised or the experiment replicated.

4. Failure to take good care of the centralized data repository may lead to the leakage of sensitive personal information, therefore ruining the reputation of research groups and discouraging prospective participant engagement.

What is the MIDDLE framework?

Fortunately, the upsurge of smart phones and personal wearable devices in the past decade has provided a new perspective in addressing these long-existing limitations. In the MIDDLE framework, data stay where they are originally collected – on each participant's smart phone, computer, tablet, or any wearable computing device, and are never revealed to outside parties. Candidate statistical models are fit by requesting intermediate information (model parameters, gradients, Hessian, likelihood function values, etc.) instead

of the raw data from each participant. In [19], each personal computing device will calculate the likelihood function value based on its own private data and the current model parameters during each model-fitting iteration and no raw data will ever be transmitted back to the research group. A central coordinator in the research lab may be deployed to manage the process. The aim is to find a maximally likely set of parameter values for the model without seeing personal data. Also, participants can choose to opt in or out during the experiment and model-fitting process. When sufficient data are collected to reach a pre-selected statistical power, the study can automatically terminate or switch to a cross validation regime. This new paradigm would ease a variety of new study designs that involve bigger data than previously practical. For example, it would become feasible to study patient history across hospitals even when data is legally prohibited from leaving the hospital. MIDDLE would also facilitate longitudinal linking of participant data and sharing of data between studies. In general, there is great potential for substantially reducing the time between hypothesis generation and dissemination of results while simultaneously reducing participant burden. The flowchart of the MIDDLE framework is shown in Figure 1.2.

What are the benefits of MIDDLE?

Imagine the MIDDLE framework is fully implemented and smoothly coordinating research. Suppose one research team has a hypothesis about diet and exercise that involves a self-report questionnaire instrument and smart phone accelerometer sensor data. Using the MIDDLE experiment creation web page, the team creates a detailed description of the experiment and draws the statistical models that represent the hypotheses. After Institutional Review Board (IRB) approval, the MIDDLE service publishes the experiment in a web-accessible central repository – its "app store for science". The MIDDLE host manages opt-in and opt-out consent documents and facilitates the download of the MIDDLE experiment apps (including the model likelihood calculator) to each participant's smart phone. As participants opt into the experiment, the researcher's MIDDLE optimizer

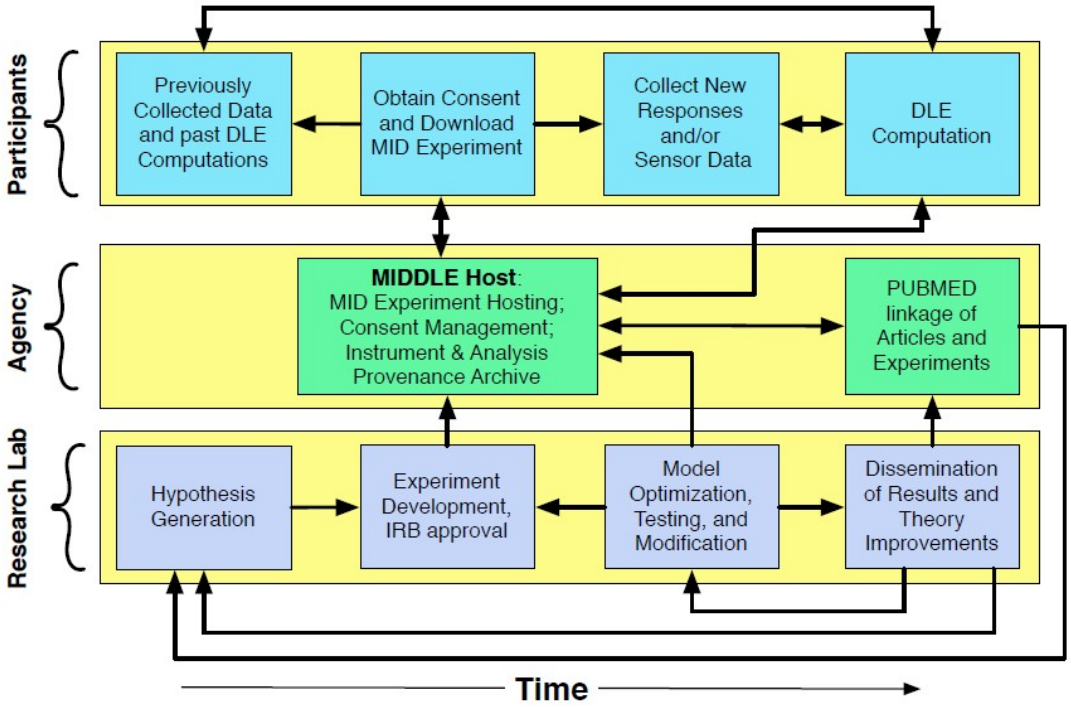


FIGURE 1.2: MIDDLE framework flowchart, from [19]

software begins the optimization of statistical models. Privacy protection techniques are carefully designed so that participant's sensitive information doesn't leave his/her devices and intermediate results won't leak information even if being captured and analyzed by attackers. IRB-approved experimental modifications can be re-disseminated for participant consent and update. When participants consent to the use of previously-collected data, each new experiment starts optimization with a large set of data automatically shared from previously-run MIDDLE experiments. Longitudinal data collection is thus automatically enabled and linked at the individual level at zero cost to a new project. Participants could even choose to collect personal data outside the context of an experiment using wearable activity monitors, health monitors, GPS tracking, or questionnaires. If these participants then opt into an experimental analysis, they can choose to allow access of these previously collected data in the new experiment.

A second research group runs a study on a hypothesis related to the first study. They look up the first study's results in a publication archive and follow the link to the associated models and instruments in the MIDDLE archive. However, their new hypothesis

requires an experiment with an in-lab component as well as a self-report questionnaire and in-home sensor data. The second research group modifies the first group's instrument and statistical models and advertises their IRB-approved study, offering additional compensation to participants from the first study. Participants opt-in and most of those from the first study opt to allow data sharing. Within a week, the study has a relatively large data sample. Participants who consent to the in-lab followup are randomly selected to a treatment or control condition. For participants who opt-in, the MIDDLE service transmits contact information to the research group, which arranges appointments for the in-lab study. Participants bring their personal devices to the lab and the in-lab data are uploaded into the personal devices for the participants to take home. Participants are given the choice of whether to allow the lab to archive a copy of their data. The analysis and write up proceed in the same manner as in the epidemiological experiment. Note that the in-lab data are always uploaded to the participants' devices. Thus, these data are available, with participant consent, for sharing and longitudinal linking in other experiments. As more data are accumulated into participants' personal devices, their data become more and more valuable to future researchers, and thus of greater value to the participant.

In summary, the benefits MIDDLE brings to the research community include, but are not limited to the following:

- Accelerated Translation of Research into Practice
- Faster and Easier Data Sharing
- Improved Longitudinal Data for Person-Specific Medicine
- Reduced Burden and Mitigated Risk for Participants
- More Reliable Methods, Instruments, and Statistical Tests

What are the challenges of implementing MIDDLE?

Technically there are optimization and privacy challenges involved in building the MIDDLE framework. The first challenge is to develop a distributed optimization algorithm to train a machine learning model that leverages data sets from multiple parties. The objective functions of most distributed machine learning models can be summarized in the form:

$$\begin{aligned} \text{minimize} \quad & F(\boldsymbol{\theta}) = \sum_{i=1}^{M} f_i(\boldsymbol{\theta}; \boldsymbol{X}_i) \\ \text{subject to} \quad & \boldsymbol{\theta} \in \boldsymbol{\Theta}, \end{aligned} \tag{1.1}$$

where M is the number of involved parties, f_i is the component loss function, $\boldsymbol{X}_i$ is the data set of the i-th party, and $\boldsymbol{\Theta}$ is a nonempty, closed and convex subset of parameter space $\mathcal{R}^n$. The optimal solution $\boldsymbol{\theta}^*$ is called the M-estimator ("M" for "maximum likelihood-type"). In statistics, M-estimator represents a broad class of estimators. Numerous common machine learning models can be classified as M-estimators, for instance, linear regression, logistic regression and support vector machine (SVM) [49]. Most of the time no closed form solution exists for an M-estimator and an iterative optimization approach is required to solve the optimization problem (gradient descent method, Newton's method, iteratively re-weighted least squares method, etc.).

The second challenge in building the MIDDLE framework is to ensure the privacy of participants. Integrating privacy-preserving techniques with multi-party machine learning is the focus of the work in this dissertation. One great advantage of MIDDLE compared to traditional experiment design paradigm is that no raw data will be collected into a central repository, therefore mitigating the risk of centralized database being hacked and reducing the payoff of potential attackers. However, attackers may capture all the incoming and outgoing messages of a specific participant in the model-fitting process and learn its raw data by scrutinizing the gathered information. As revealed by recent research [153, 37], even the release of intermediate information or summary statistics is potentially privacy-breaching and gains the attackers edge in guessing the participant's personal data with remarkable confidence. Moreover, the introduction of privacy protection mechanism

usually results in loss of data utility [89, 4]. Therefore, we need a provably working privacy notion implemented in MIDDLE to investigate the privacy-utility trade-off and provide a reasonable bound of how much utility is possible for a given level of privacy. Among others, the framework of differential privacy (DP) proposed by Dwork et al. [43] seems a suitable choice.

1.3 Overview of the Dissertation

The rest of the dissertation is organized as follows. Chapter 2 provides a review of background knowledge and discusses related work in distributed optimization and privacy-protection techniques. In Chapter 3, we develop a differentially private distributed deep learning algorithm and evaluate its performance on a real-world credit card fraud detection data set. We show that our algorithm is effective in selecting the important gradients in the stochastic gradient descent method while providing better differential privacy guarantee than the state-of-the-art. The experiment results further implied that privacy can be achieved without adversely affecting the data utility. In Chapter 4, we introduce differential privacy into the increasingly popular multi-source transfer learning problem. We adapt a differential private importance weighting mechanism to quantify the relationship between the source domains and the target domain, which improves the Bayesian model aggregation for logistic regression on the target domain where labeled samples are extremely scarce. In Chapter 5, we propose an approximate Newton's method which significantly speeds up the secure logistic regression of distributed databases when combined with cryptographic techniques. Chapter 6 concludes the dissertation with a summary of our contributions and a discussion of future work.

Chapter 2

Background, Definitions and Related Work

In this chapter, we first review the distributed optimization algorithms and machine learning models that we explore throughout this dissertation. Then we provide background information on differential privacy and secure multi-party computation (SMC) which form the foundation of our privacy notion.

2.1 Distributed Optimization Methods

Optimization problems are the central pieces of a wide range of machine learning models [9]. Training a machine learning model usually reduces to finding a set of model parameters which optimizes the loss function on the available training data set. As discussed in Chapter 1, the decentralized collection or storage of data sets makes the development of distributed optimization algorithms highly desirable. Under such a collaborative multi-party machine learning setting, the goal of each party is to cooperatively optimize the global objective function $F(\boldsymbol{\theta})$ with respect to the model parameters $\boldsymbol{\theta}$ using a combination of local data sets (see Equation 1.1).

Distributed optimization problem has been explored in depth in many applications, such as wireless sensor networks [126, 127, 132] that are characterized by the lack of centralized access to data set, unpredictable link/node failures and time-varying topology. Due to these defining characteristics, the operating algorithms need to be simple

yet efficient and robust against ever-changing networks dynamics. In general, such distributed optimization algorithms can be classified according to the information diffusion and communication mechanism (incremental, consensus, gossip, broadcast, etc.), the synchronization of the system (synchronous or asynchronous) and the iterative updating methods (gradient-based, Hessian-based, dual averaging, alternating direction method of multipliers, etc.). In this section, we are mainly focusing on the primal domain methods. In particular, we will survey some of the state-of-the-art gradient-based and Hessian-based distributed optimization algorithms.

2.1.1 Gradient-based Methods

Stochastic gradient descent (SGD) method, or in some literature known as incremental gradient descent method, is a variant of the standard gradient descent method for minimizing a convex objective function that is written as a sum of component functions. In a standard gradient descent method, the model parameter estimate $\boldsymbol{\theta}_k$ at iteration k is updated as following:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \alpha_k \sum_{i=1}^{M} g_i(\boldsymbol{\theta}_{k-1}; \boldsymbol{X}_i), \tag{2.1}$$

where $g_i(\boldsymbol{\theta}_{k-1}; \boldsymbol{X}_i)$ is the gradient of the component function f_i at $\boldsymbol{\theta}_{k-1}$ and α_k is the step size for the update. The calculations of gradients are performed at each local data set $\boldsymbol{X}_i$ in parallel. The aggregation of local gradients and the update of the model parameters usually happen on a central server. However, when $\boldsymbol{X}_i$'s are large, evaluating $g_i(\boldsymbol{\theta}_{k-1}; \boldsymbol{X}_i)$ at each iteration just for one parameter update becomes extremely inefficient and redundant as it recomputes the gradients for similar samples in $\boldsymbol{X}_i$'s. In such a situation it is more appealing and practical to compute the gradient using only a single or a few training samples from $\boldsymbol{X}_i$'s at each iteration. This is a stochastic approximation of the standard gradient descent underlying SGD [151, 14].

SGD is widely adopted to solve a variety of machine learning models (e.g., SVM, linear regression and logistic regression) in the context of large-scale learning [170]. Notably, it is also used for the training of deep artificial neural networks under the name "back

propagation algorithm" [135]. The theories behind SGD have been studied since the 1960s to solve the least mean squares problem, or more generally, the M-estimation problems [11, 10]. Compared with the standard gradient descent method, it performs more frequent updates with a higher variance and can also be used to learn online with new samples coming in real time. SGD can be viewed as a standard gradient descent method with error in the calculation of gradients. The convergence analysis and empirical performance of such gradient methods with error have been studied in [12, 113, 112]. When using a diminishing step size α_k satisfying $\lim_{k\to\infty} \alpha_k = 0$ and $\sum_{k=0}^{\infty} \alpha_k = \infty$ at the k-th iteration, SGD converges almost surely to a global minimum under regularity conditions (convexity, bounded gradient, etc.), and otherwise converges almost surely to a local minimum [22]. When using a constant step size, the method will converge to a neighborhood of the optimal solution. On the other hand, SGD's fluctuation enables it to jump out of local minimum and converge to a potentially better estimate, especially when the objective function is complicated and non-convex [5, 115].

An extension of SGD is proposed by Nedic and Ozdaglar [111, 114], which combines the gradient-based method and a consensus algorithm for cooperative distributed multi-agent optimization. In their multi-agent setting, no parameter estimate will be circulated within the network; instead each agent will initialize and maintain its own local parameter estimate. During each iteration, the agent combines its own estimate with the estimates received from its neighbors, and updates the weighted average of estimates using the gradients of its own loss function at the current estimate:

$$\boldsymbol{\theta}_{k,i} = \sum_{j=1}^{M} w_{k,i\to j}\boldsymbol{\theta}_{k-1,j} - \alpha_{k,i} g_i(\boldsymbol{\theta}_{k-1,i}; \boldsymbol{X}_i), \tag{2.2}$$

where $\boldsymbol{\theta}_{k,i}$ is the parameter estimate of the i-th agent at iteration k and $w_{k,i\to j}$ is the nonnegative weight that agent i gives to the estimate $\boldsymbol{\theta}_{k-1,j}$ received from agent j at iteration k. The weight matrix $\boldsymbol{W} \in \mathcal{R}^{M\times M}$ at each iteration is required to be doubly stochastic. The intuition behind this requirement is that each agent will impose equally important impact on the optimization problem.

The network structure of such a multi-agent consensus system is represented by a directed graph $G = (N, E)$, with the vertex set N and the edge set E. The graph records the relation between different agents in the network. The network works as a medium to diffuse information over time. One key assumption is that the network needs to be sufficiently strongly connected over a certain amount of time. Intuitively, it means the local information of an agent can be circulated either directly or indirectly among the whole network given sufficient time. In addition, the network is required to be frequently strongly connected. Note that the topology is allowed to change with time, as long as every agent shows up in the network and talk to some others from time to time, which mimics the dynamics in the MIDDLE framework. This multi-agent consensus algorithm is in general robust to topology changes and incur small overhead per iteration. Another variant of this communication pattern is called randomized gossip algorithm [25, 139]. This type of algorithms is mostly used to deal with average consensus problems [15, 28]. Eventually every agent will reach a consensus under certain regularity conditions due to the constant information flow in the network and this common parameter vector turns out to be the optimal solution of the optimization problem.

The distributed multi-agent gradient method is extended under a variety of settings after Nedic [114, 131, 149, 33, 69]. Notably, Shi et al. [142] develop a synchronous distributed consensus gradient method which converges at a rate of $O(1/k)$ if F is strongly convex. In particular, it updates the parameter estimate using the gradient information from the previous two iterations. The major contribution of their work is the introduction of a method which converges to the exact solution using fixed step size. For strongly convex objective functions, this method guarantees linear convergence rate.

In the MIDDLE framework, the participants can choose to opt in or opt out and the mobile devices can be turned on and off at any time. These dynamics can be reflected in the ever-changing topology. For example, if one participant happens to be off network at a certain iteration, we can simply set its neighborhood to be null and the preceding algorithms will apply. Sometimes even the private data of a certain participant can change, which changes its local objective function accordingly. In this case, Equation 1.1 can be

more precisely described as a dynamic optimization with time-varying objective.

$$\begin{aligned} &\text{minimize} \quad F(\boldsymbol{\theta}) = \sum_{i=1}^{M} f_i(\boldsymbol{\theta}; \boldsymbol{X}_{k,i}) \\ &\text{subject to} \quad \boldsymbol{\theta} \in \boldsymbol{\Theta} \end{aligned} \tag{2.3}$$

In particular, Simonetto and Leus [145] and Simonetto, Kester, and Leus [144] address this problem by proposing an asynchronous distribute gradient-based algorithm similar to the distributed consensus algorithm and proved its convergence under the assumption that the distance between the optimizers at subsequent time steps $||\boldsymbol{\theta}_k^* - \boldsymbol{\theta}_{k-1}^*||$ is bounded. Other researchers also provide convergence analysis of similar problem formulations using different methodologies [70, 94].

2.1.2 Hessian-based Method

The de facto approach in practice to solving some of the most common (regularized) M-estimation problems, such as logistic regression, is the Newton's method [59]. This is also the method of choice for most existing privacy-preserving solutions [159, 162, 72]. To optimize the objective function $F(\boldsymbol{\theta})$ in Equation 1.1, the parameter estimate is updated as follows during each iteration in the Newton's method:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{H}^{-1}(\boldsymbol{\theta}_k) g(\boldsymbol{\theta}_k), \tag{2.4}$$

where $\boldsymbol{H}(\boldsymbol{\theta}_k)$ is the Hessian matrix of F at $\boldsymbol{\theta}_k$ and $g(\boldsymbol{\theta}_k)$ is the gradient of F at $\boldsymbol{\theta}_k$. Note that both the gradient and the Hessian are decomposable in terms of the component functions.

$$g(\boldsymbol{\theta}_k) = \sum_{i=1}^{M} g_i(\boldsymbol{\theta}_k), \quad \boldsymbol{H}(\boldsymbol{\theta}_k) = \sum_{i=1}^{M} \boldsymbol{H}_i(\boldsymbol{\theta}_k) \tag{2.5}$$

Therefore, the computation can be distributed to each party in the network and only the gradients and Hessian matrices are passed back to a central place for further aggregation and update. Compared with gradient-based methods, Newton's method converges much faster when starting values are chosen appropriately. However, for high dimensional

problems, the evaluation of the Hessian matrix will be quite costly and the storage and circulation of Hessian matrix will be a problem for small computing nodes and unstable, low-bandwidth networks. Nonetheless, Newton's method and its variant (quasi-Newton method) are still a viable choice for some of the existing distributed frameworks when computation power is not a restriction [77, 172, 96, 21]. The logic behind the quasi-Newton methods is to find a proper approximation to the Hessian matrix or its product with gradients (for example, conjugate gradient method) in order to reduce the computation workload and storage requirement at each iteration.

2.2 Machine Learning Models

2.2.1 Logistic Regression

Logistic regression is a statistical model developed by Cox [36] to estimate the probability of a categorical response variable based on one or more predictor variables. It is a widely used classification model in machine learning, especially in the fields of medical study and social sciences [6, 65, 80]. Example use cases include: predicting disease affection in biomedicine [97], detecting genetic variants associated with disease in genetics (i.e., genome-wide association study, or GWAS) [87], finding factors attributing to certain outcomes in economics [64], predicting click-through-rate (CTR) in computational advertising on the internet [104], etc.

The response variable in logistic regression can be binomial, ordinal or multinomial and we focus on the binary logistic regression model here. The goal of logistic regression is to model the probability $\Pr(y = 1|\boldsymbol{x})$ of a response variable y being 1 given predictive features $\boldsymbol{x}$. Mathematically, the probability is modeled as a logistic function of $\boldsymbol{x}$:

$$\Pr(y = 1|\boldsymbol{x}) = \frac{1}{1 + \exp^{-\beta^\intercal \boldsymbol{x}}}, \quad \Pr(y = 0|\boldsymbol{x}) = 1 - \Pr(y = 1|\boldsymbol{x}). \tag{2.6}$$

Equivalently, we can model the log-odds by taking the logarithm of both sides in Equation 2.6

$$\log\left(\frac{\Pr(y=1|\boldsymbol{x})}{1-\Pr(y=1|\boldsymbol{x})}\right)=\boldsymbol{\beta}^{\intercal}\boldsymbol{x}. \tag{2.7}$$

The regression parameters $\boldsymbol{\beta}$ are usually estimated using maximum likelihood estimation given a labeled training set $(\boldsymbol{X},\boldsymbol{Y})$. The intuition is to find the set of parameters such that the predicted probability $Pr(y=1|\boldsymbol{x})$ is close to 1 when $y=1$, and otherwise close to 0. The likelihood function $L(\boldsymbol{\beta};\boldsymbol{X},\boldsymbol{Y})$ on the training set $(\boldsymbol{X},\boldsymbol{Y})$ with N samples is defined as

$$L(\boldsymbol{\beta};\boldsymbol{X},\boldsymbol{Y})=\prod_{i=1}^{N}\Pr(y_i|\boldsymbol{x}_i)=\prod_{i=1}^{N}\Pr(y_i=1|\boldsymbol{x}_i)^{y_i}(1-\Pr(y_i=1|\boldsymbol{x}_i))^{1-y_i}. \tag{2.8}$$

Usually, it will be easier to maximize the log likelihood function $l(\boldsymbol{\beta};\boldsymbol{X},\boldsymbol{Y})$:

$$\begin{aligned} l(\boldsymbol{\beta};\boldsymbol{X},\boldsymbol{Y}) &= \log(L(\boldsymbol{\beta};\boldsymbol{X},\boldsymbol{Y})) \\ &= \sum_{i=1}^{N} y_i\log(\Pr(y_i=1|\boldsymbol{x}_i))+(1-y_i)\log(1-\Pr(y_i=1|\boldsymbol{x}_i)). \end{aligned} \tag{2.9}$$

Real-world studies often apply a regularization term on logistic regression to aid model selection and prevent over-fitting by penalizing extreme parameters [84, 90]. For example, the ℓ_2-regularized logistic regression imposes an additional regularization term $-\frac{\lambda}{2}\boldsymbol{\beta}^{\intercal}\boldsymbol{\beta}$ to the objective function during model estimation, where λ is the predefined regularization parameter. The ℓ_2-regularized log likelihood function becomes

$$l_2(\boldsymbol{\beta};\boldsymbol{X},\boldsymbol{Y})=l(\boldsymbol{\beta};\boldsymbol{X},\boldsymbol{Y})-\frac{\lambda}{2}\boldsymbol{\beta}^{\intercal}\boldsymbol{\beta} \tag{2.10}$$

Equation 2.10 is usually optimized using Newton's method or Quasi-Newton method (e.g. iteratively reweighted least squares [27]). Here, we briefly describe solving ℓ_2-regularized logistic regression with Newton's method. At each iteration k, the regression

parameters $\boldsymbol{\beta}$ is updated by:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k - \boldsymbol{H}^{-1}(\boldsymbol{\beta}_k)\ g(\boldsymbol{\beta}_k)\ , \tag{2.11}$$

where $\boldsymbol{H}(\boldsymbol{\beta}_k)$ and $g(\boldsymbol{\beta}_k)$ denote the Hessian and gradient of the objective $l_2(\boldsymbol{\beta}; \boldsymbol{X}, \boldsymbol{Y})$ evaluated at the current regression parameters estimate $\boldsymbol{\beta}_k$. To be more specific,

$$g(\boldsymbol{\beta}) = \boldsymbol{X}^\intercal(\boldsymbol{Y} - \boldsymbol{P}) - \lambda\boldsymbol{\beta}, \tag{2.12}$$

$$\boldsymbol{H}(\boldsymbol{\beta}) = -\boldsymbol{X}^\intercal\boldsymbol{A}\boldsymbol{X} - \lambda\boldsymbol{I}, \tag{2.13}$$

where $\boldsymbol{P}$ is an N-dimensional vector with each element being the predicted probability for $(\boldsymbol{x}_i, y_i)$: $p_i = \Pr(y_i = 1|\boldsymbol{x}_i)$; $\boldsymbol{A} \in \mathcal{R}^{N\times N}$ is a diagonal matrix with elements on the main diagonal being $\boldsymbol{A}(i,i) = p_i(1-p_i)$; $\boldsymbol{I} \in \mathcal{R}^{N\times N}$ is the identity matrix. This updating process iterates until model convergence, as determined by the relative change of log-likelihood.

2.2.2 Deep Neural Network

A deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers between the input and output layers [138]. The ANN model is inspired by the biological neural networks in animal brains. The early concept of perceptron as an algorithm for supervised learning was developed in the 1950s by Rosenblatt [133]. There are multiple ANN classes that are applicable to a variety of tasks (e.g., recurrent neural network (RNN) [67], convolutional neural network (CNN) [50], generative adversarial network (GAN) [58]). Here we focus on the simplest type of ANN – feedforward neural network (FNN) or multilayer perceptrons (MLPs).

As shown in the Figure 2.1, FNN consists of multiple layers of computing units called neurons interconnected in a feed-forward way. The leftmost layer in the network is called the input layer, and the rightmost layer is the output layer. The values of input and output neurons come from the original training data set. The layers in the middle are called the

hidden layers. Each neuron uses the outputs from the neurons in the previous layer as inputs and feeds them into an activation function such as a sigmoid function, which maps the weighted inputs to its own output. The information flows in one direction, from the input layer, through the hidden layers (if any), and to the output layers.

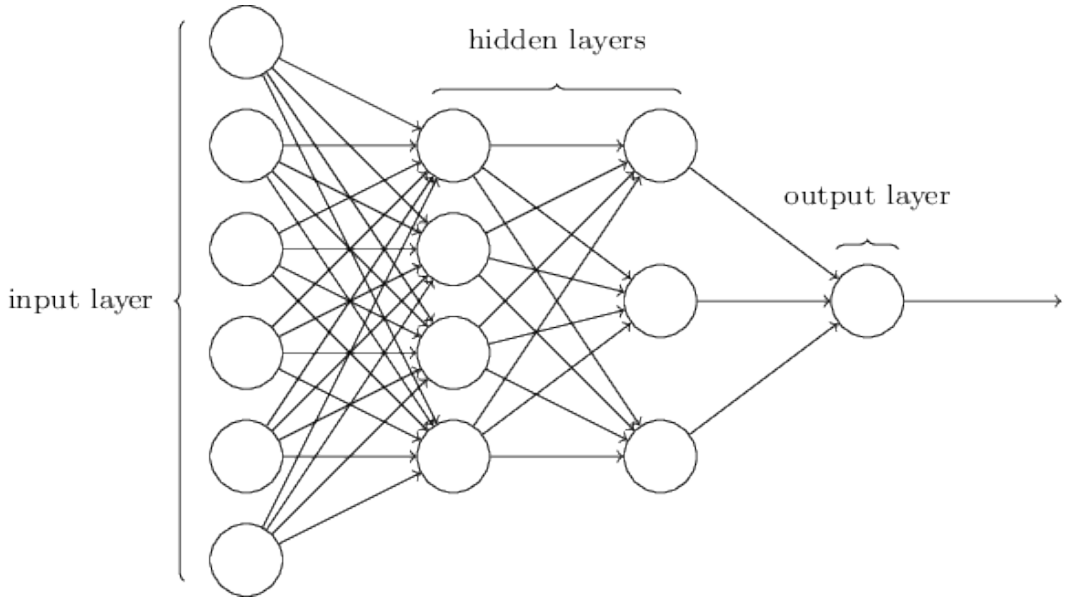


FIGURE 2.1: Illustration of the architecture of a 4-layer FNN, from [116]

The universal approximation theorem for neural networks [38] states that every continuous real function can be approximated arbitrarily closely by a MLP with just one hidden layer. Therefore ANN is a powerful tool to model complex non-linear relationships, especially for complicated machine learning tasks in different fields, including computer vision, speech recognition, natural language processing, audio recognition and machine translation. An efficient gradient descent method called backpropagation was developed in the 1960s and 1970s, and applied to train FNN in the 1980s [135]. The output values by the neural network are compared with the correct answer to compute the value of some predefined loss function. The error of the loss function is calculated at the output layer and propagated backwards through the network layers.

2.3 Differential Privacy

2.3.1 What does Privacy Mean?

Privacy can mean vastly different things in different scenarios, such as the right to be let alone, secrecy or the option to limit the access to personal information. To be clear, the major privacy notion we are focusing on is data privacy for statistical database. A statistical database is a data repository used for statistical analysis purposes by allowing queries/computation from data analysts. Some real-world examples are patients' medical records in a hospital, transaction information on credit cards issued by a financial institution, census data collected by government agencies, internet users' search history through a specific search engine and personal health data maintained by individuals in a ubiquitous environment.

The raw data in the statistical database are usually not revealed to unauthorized entities and queries targeted on certain individual records are forbidden. Researchers are required to undergo confidentiality training and sign data use statements in order to gain access to the database. However, these simple administrative and technical measures turn out to be insufficient to protect the privacy of individual records especially when the way in which data privacy is violated evolves along with technology development. It is an extremely hard and computationally challenging task for the database administrator to recognize potentially privacy-breaching queries. Intelligent yet malicious data analysts who are entitled to legitimate statistical queries may possess auxiliary knowledge and design a series of seemingly innocuous queries to derive private information about a single individual.

The history of data privacy research is rife with failed practices as attested by recent studies. The first infamous example is data anonymization – scrub the personally identifiable information (e.g. names, ages, addresses, date of birth) and release a sanitized version of the data. In practice, however, it can be difficult to determine what particular combination of attributes constitute personally identifiable information, especially in the newly emerging fields of technology. Successful linkage attacks on de-identified data have shown

that data anonymization are vulnerable to privacy breaches because anonymous data can be cross-referenced with other auxiliary data sets to de-anonymize the data set. For instance, the medical records of the governor of Massachusetts were identified by matching anonymized medical data with the public Massachusetts voter registration records [150]. Another example is that Netflix subscribers whose viewing histories were published by Netflix with personal identifiable attributes removed as training data for a competition on recommendation were identified by cross referencing with the Internet Movie Database (IMDb) [109]. Even the more robust versions of the anonymization practices through reducing the granularity of a data representation (e.g., k-anonymization, l-diversity, t-closeness) are susceptible to many privacy attacks especially when prior knowledge is available [137, 99, 88]. Secondly, forcing queries to be on summary statistics might compromise privacy due to differencing attack, where the differences between multiple queries, together with auxiliary information, reveal personal information. For example, some recent research [68, 153] show that releasing summary statistics such as allele frequency or genotype counts, do not mask identity within genome-wide association studies (GWAS).

In general, privacy fails to be retained because database administrator neglect or cannot determine the impact of the presence of auxiliary information. Therefore, it is highly desirable to provide a robust privacy guarantee against unanticipated auxiliary knowledge an attacker may possess or even potential attack models currently unseen. Moreover, privacy protection usually comes at a cost: data utility will eventually be compromised by the introduction of privacy preservation mechanism. It is therefore appealing to design a mechanism providing a quantifiable trade-off between privacy and utility.

2.3.2 What is Differential Privacy?

The concept of differential privacy (DP) was first proposed by Cynthia Dwork more that a decade ago [43]. It is a rigorous definition of privacy motivated by one of our inherent understandings of statistical database privacy – the presence or the absence of an individual in a database, or its particular value, cannot be determined by examining

the query output, even with the help of auxiliary information. In other words, algorithms or queries on a statistical database satisfy the definition of differential privacy when their outputs is not strongly correlated to any particular data point. Here the queries are mainly aggregate statistical estimations, ranging from simple average to machine learning models.

This is an important and appealing privacy notion because intuitively it soothes the concern of individuals who are considering contributing their data to a private database yet worried that their privacy is leaked from the use of the data. For example, Alice is considering giving her medical data to a renowned lab for research purposes, but she fears that the subsequent statistical analysis on the database might hurt her in some way the researchers are not capable of detecting or preventing. This is a practical concern given the increasingly sophisticated efforts of outside attackers. However, if the data is only analyzed in a differentially private manner, Alice can be assured that any risk of data analysis to her due to her contribution of the data is no more likely to occur, as compared to if she does not provide her data. Potential attackers would not know significantly more about Alice no matter how carefully they scrutinize the algorithm outputs or what prior knowledge they possess. The definition of differential privacy below converts the intuition into a precise mathematical statement [41].

Definition 1 (Differential Privacy)**.** *A randomized algorithm K gives (ϵ, δ)-differential privacy if for all data sets D_1 and D_2 differing on at most one element, and all $S \in Range(K)$,*

$$Pr[K(D_1) \in S] \leq \exp(\epsilon) \times Pr[K(D_2) \in S] + \delta \tag{2.14}$$

The parameter ϵ is typically called the privacy budget which represents the level of privacy guarantee and quantifies the privacy loss whenever an algorithm or query is applied on the data. When we think about ϵ, we should understand it as a "budget" rather than a purely statistical upper bound of the probability ratio in Definition 1. A differentially private algorithm with smaller ϵ means that it is more private. Every time the statistical database is queried by a ϵ-differentially private algorithm, the amount of the privacy

budget ϵ will be consumed. When the private database is queried by a composition of algorithms, the consumed privacy budget will sum up following the composition theorems for differential privacy [75]. In practice, each participant sets a total privacy budget based on the desired level of privacy guarantee. When the consumed privacy budget exceeds the total budget, algorithms or queries can no longer be applied to the dataset. The other privacy parameter δ quantifies the probability of the algorithm failing to satisfy differential privacy and is usually set to a very small number or zero.

2.3.3 How is Differential Privacy Achieved?

Essentially differential privacy is achieved by introducing calibrated randomness into the queries or algorithms. An early example of this idea was first adopted in the survey method "Randomized Responses" in the social science area [156]. It allows respondents to answer sensitive questions (illegal behavior, sexual orientation, etc.) without worrying their confidentiality being disclosed or used against them. Specifically, the respondent flips a coin in private before answering the survey question. If the coin comes up tails, he/she will answer "yes"; otherwise, he/she will answer truthfully. The key implication behind randomized response is that it provides "plausible deniability". A response of "yes" may be because of either coin flip or the respondent's true answer to the survey questions. No firm conclusion can be reached about a specific respondent by simply examining the responses.

Inspired by the methodology of "Randomized Responses", there are two basic mechanisms to achieve differential privacy – the Laplace mechanism for numerical queries and the exponential mechanism for non-numeric valued queries.

The Laplace mechanism is basically a noise adding mechanism making use of the Laplace distribution (a symmetric exponential distribution, see Definition 2) and the concept of l_1-sensitivity of a function (see Definition 3). For example, the sensitivity of a count query on a statistical database is 1. The sensitivity of a function gives an estimation on how much we must perturb its output to hide the contribution of any single individual

and low sensitivity queries can be answered with little noise.

Definition 2 (The Laplace Distribution). *A random variable has a Lap(μ, b) distribution if its probability density function is*

$$f(x|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right) \tag{2.15}$$

Definition 3 (l_1 Sensitivity). *The l_1 sensitivity of a function $f : D^n \to R^d$ is the smallest number $S(f)$ such that for all x, $x' \in D^n$ which differ in a single entry,*

$$||f(x) - f(x')||_1 \leq S(f). \tag{2.16}$$

Essentially the Laplace mechanism works by adding random Laplace noise to the output of numeric queries with variance depending on the privacy parameter ϵ and the sensitivity of the queries (see Definition 4). It has been proven to be $(\epsilon, 0)$-differential private and a bound on the accuracy of the Laplace mechanism is given in [42]. Similarly adding Gaussian noise with variance calibrated to $S(f)\ln(1/\delta)/\epsilon$ will provide (ϵ, δ)-differential privacy.

Definition 4 (Laplace mechanism). *Given any function $f : D^n \to R^d$, the Laplace mechanism is defined as:*

$$M_L(x, f(.), \epsilon) = f(x) + (Y_1, ..., Y_d), \tag{2.17}$$

where Y_i are i.i.d. random variables drawn from Lap$(0, S(f)/\epsilon)$.

A variant of Laplace mechanism is often applied to the type of "select the best" queries. For example, for the query "what is the most common medical condition among a set of candidate conditions in a hospital?", the differential private approach called Report-Noisy-Max is to add Laplace noise to each count of candidate conditions and return the condition with the largest noisy count as the query output. Report-Noisy-Max is also $(\epsilon, 0)$-differential private with a reasonable accuracy guarantee.

For general non-numerical queries, however, the Laplace mechanism or Report-Noisy-Max don't apply. Even worse, adding noise directly to the query output can completely destroy its value in some cases (e.g., setting a price in auction). The exponential mechanism [105] is a technique for designing differentially private algorithms to answer queries with arbitrary utilities. Intuitively the exponential mechanism begins with defining a utility function $u : \mathcal{D}^n \times \mathcal{R} \to \mathbb{R}$ to map database/output pairs to utility scores, which measures how good an output $r \in \mathcal{R}$ is for a database $x \in \mathcal{D}^n$. The output with higher utility score is preferred.

Definition 5 (The Exponential Mechanism)**.** *For any utility function $u : \mathcal{D}^n \times \mathcal{R} \to \mathbb{R}$, the exponential mechanism selects and outputs an answer $r \in \mathcal{R}$ for a query and a database $x \in \mathcal{D}^n$ with probability proportional to* $\exp\left(\frac{\epsilon u(x,r)}{2S(u)}\right)$.

The exponential mechanism preserves $(\epsilon, 0)$-differential privacy and the output with higher utility is exponentially more likely to be selected.

With the basic techniques for designing a differentially private algorithm, it is also important to investigate how the privacy risk accumulates through a composition of differentially private algorithms. The fact that privacy degrades after repeated computations/queries on the same database applies to all privacy preserving approaches. Differential privacy provides a framework for quantifying and bounding the cumulative privacy risk with the privacy budget parameter ϵ. Reducing the consumption of the privacy budget while achieving satisfactory model performance remains an interesting and challenging question [75]. The following theorems are the basic and advanced composition theorems in differential privacy.

Theorem 2.3.1 (Basic composition theorem)**.** *Let M_i be an (ϵ_i, δ_i)-differentially private algorithm ($i = 1, 2, ..., K$). Then the composition of all the K algorithms $M_{[K]}(D) = (M_1(D), ..., M_K(D))$ is $(\sum_i \epsilon_i, \sum_i \delta_i)$-differentially private.*

Theorem 2.3.2 (Advanced composition theorem)**.** *For all $\epsilon, \delta, \delta' \geq 0$, the class of (ϵ, δ)-differentially private mechanisms satisfies $(\epsilon', k\delta + \delta')$-differential privacy under k-fold*

adaptive composition for:

$$\epsilon' = \sqrt{2k \ln(1/\delta')}\epsilon + k\epsilon(e^{\epsilon} - 1). \tag{2.18}$$

Another important characteristics of differential privacy is that it is immune to post-processing and auxiliary information: a data analyst cannot increase privacy loss by examining the outputs of differential private queries/algorithms, no matter what auxiliary information is available. More formally,

Theorem 2.3.3 (Post-processing)**.** *Let M be a (ϵ, δ)-differentially private randomized algorithm. Let f be an arbitrary randomized mapping. Then $f \circ M$ is (ϵ, δ)-differentially private.*

2.3.4 Differential Privacy and Machine Learning

As one of the most prevalent data analysis tasks, machine learning has received a significant amount of research attention. In particular, developing algorithms for machine learning under the constraint of differential privacy becomes increasingly important especially when a huge amount of personal sensitive information (e.g., medical, social network, financial) are being analyzed with or without notice. In fact, the goal of machine learning aligns with that of privacy preservation – extract informative pattern about the population while abstracting away individual records. In other words, there is a strong connection between differential privacy and stability and generalization capability of machine learning models [23, 108, 140].

Some impactful research in recent years provided promising guidance and direction. The work of Chaudhuri et al. [30, 31] addressed the regularized empirical risk minimization (ERM) framework for classification while guaranteeing ϵ-differential privacy. ERM problems can also be seen as a standard M-estimation problem with the component function being the empirical loss of each sample. The general framework is similar to Equation 1.1, with an extra regularization term $\lambda N(\theta)$. They proposed two privacy-preserving approaches to the problem: output perturbation and objective perturbation.

Output perturbation is derived from the Laplace mechanism. Random noise is added to the output of the standard ERM algorithm. In contrast, objective perturbation adds noise to the objective function prior to the optimization process. Both procedures are proven to preserve differential privacy when the regularized loss function satisfies certain regularity properties (convexity, bounded derivatives, etc.). Accuracy guarantee and the corresponding upper bound of the training sample size are also derived.

The work of Lei [85] addressed the problem of centralized differentially private M-estimators using perturbed histograms. He demonstrated how histograms can be used as a tool for statistical parameter estimation under strong privacy constraints. In particular, the database publishes a perturbed histogram of its samples with privacy parameter ϵ and the bandwidth parameter h. Based on this perturbed histogram, the approximation error of the M-estimator is shown to be bounded. The differentially private objective function based on the perturbed histogram is

$$F_{PH}(\theta) = \frac{1}{M} \sum_{r} \hat{n}_r f(c_r, \boldsymbol{\theta}), \tag{2.19}$$

where r is the nubmer of cells in the histogram, c_r is the center of each cell and $\hat{n}_r$ is the perturbed number of data points in the corresponding cell. $\hat{n}_r$ is perturbed based on the true value n_r as follows

$$\hat{n}_r = n_r + z_r, \tag{2.20}$$

where z_r follows the Laplace distribution with density $p(z_r) \sim \epsilon \exp(-\epsilon|z_r|/2)/4$. Under certain common regularity conditions (bounder gradient of component functions, convexity, etc), the distance between the optimizers of $F(\theta)$ and $F_{PH}(\theta)$ is shown to be bounded

$$|\theta^*_{PH} - \theta^*| = O_P(1/\sqrt{m} + (\sqrt{\log n}/n)^{2/(2+d)}), \tag{2.21}$$

where d is the number of features and n is the number of samples. Their method works well for low-dimensional problems. But adapting the method to a high-dimensional problem remains an open question and yet to be developed.

The previous two studies we just mentioned inspire numerous relevant work and extensions [78, 155]. However, very few of them focus on the distributed multi-party setting. Rajkumar and Agarwal [130] considered the problems of developing differentially private stochastic gradient method in a distributed multi-agent setting and achieved a relatively weak form of differential privacy. Their procedures amount to performing Gaussian objective perturbation on the overall objective. However, the method is not truly stochastic as claimed in the paper since all the agents are looping exactly once sequentially within every iteration and no changing topology or communication failure is considered. Despite the incompleteness, it provides some inspiration for future extensions.

Besides, Hale and Egerstedty [61] actively published in the area of cloud-enabled multi-agent differential private optimization. As always, agents in their framework aim to optimize a global objective which is the sum of local objectives. Agents communicate with each other through a trusted cloud and send/receive information from it. The cloud perturbs the information from the agents in a way to guarantee privacy and sends them back. They have shown that under mild conditions the parameter estimate of each agent will converge in the sense of mean-square to the unique solution.

Moreover, researchers in Google have published its open-source framework "RAPPOR" which allows inferring statistics about populations while preserving the privacy of individual users in the sense of differential privacy [45]. These tools all provide basic building blocks for further development of differentially private machine learning algorithms.

2.4 Secure Multi-party Computation (SMC)

Besides differential privacy, privacy-preserving data mining leveraging cryptographic techniques (SMC in particular) is a classical and reviving solution for addressing the problem [3, 110, 117]. SMC is a generic framework to support multiple parties towards the goal of jointly computing a function on their input without revealing one party's information to the other parties [168, 8, 32, 57]. The first work in SMC is Yao's garbled circuit [168, 93]

for securely supporting two-party evaluation of a function $f(x_1, x_2)$ on two sources of secret inputs: x_1 (held by Party P_1) and x_2 (held by Party P_2). In brief, function $f(x_1, x_2)$ is first represented in a binary circuit form. Then Party P_1 translates ("garbles") the function circuit into a secure version ("garbled" versions of circuit and computation table based on its private input x_1). The resulting garbled circuit and computation table are later sent to the opponent P_2, who initiates a 1-out-of-2 oblivious transfer (OT) protocol (assisted by P_1) to obliviously compute garbled values corresponding to his input x_2 and outputs the result to prescribed parties. The whole protocol reveals nothing to any parties other than the final result. A more formal description of the protocol can be found in [168, 93], and example applications based on garbled circuit include [95, 165, 39]. The two party case was followed by a generalization to the multi-party by Goldreich, Micali, and Wigderson [57].

Compared with differential privacy, SMC focuses on computational privacy: the computation is performed with no single party being allowed to see the data from other parties on which the analysis is run. One of the drawbacks of SMC is that it becomes inefficient for complex functions or large data sizes. Therefore, the goal of research in this area has moved to study practical improvements of the secure protocols. The SMC model has been optimized significantly in terms of theory and engineering in recent years, as partially summarized in [95]. In addition, there is research work to combine differential privacy and SMC, further enhancing the privacy guarantee of the multi-party system [74, 125, 167, 160].

Chapter 3

Privacy Preserving Distributed Deep Learning and its Application in Credit Card Fraud Detection

In practice, most existing privacy-preserving machine learning solutions suffer from several critical limitations, such as significantly reduced utility under privacy constraints or excessive communication burden between the information fusion center and local data providers. In this chapter, we propose and implement a new distributed deep learning framework that addresses these shortcomings and preserves privacy more efficiently than previous methods. During the stochastic gradient descent training of a deep neural network, we focus on the parameters with large absolute gradients in order to save privacy budget consumption. We adopt a generalization of the Report-Noisy-Max algorithm in differential privacy to select these gradients and prove its privacy guarantee rigorously. Inspired by the recent novel idea of Terngrad [158], we also quantize the released gradients to ternary levels $\{-B, 0, B\}$, where B is the bound of gradient clipping. Applying Terngrad can significantly reduce the communication cost without incurring severe accuracy loss. Furthermore, we evaluate the performance of our method on a real-world credit card fraud detection data set consisting of millions of transactions.

3.1 Introduction

Detecting fraudulent credit card transactions is one of the biggest challenges in the banking industry. Modern technology, especially innovations in machine learning, has been applied to both analyzing the spending patterns of customers and blocking transactions that are irregular or possibly fraudulent [2, 20, 169, 52]. Among various machine learning approaches, artificial neural network models have demonstrated exceptional performance, provided an abundant supply of labeled training data is available [92, 136]. Financial transaction information, however, is considered sensitive in both its relationship to customer privacy and its importance as a source of proprietary information for banks. Machine learning models for fraud detection (e.g. deep learning) are typically trained using only the in-house data collected by each bank individually because of these privacy issues. The increasing concern over data privacy imposes restrictions and barriers to data sharing and makes it difficult to coordinate large-scale collaborative studies. Privacy-preserving multi-party machine learning can take advantage of data sets from different banks and thereby construct a model superior to stand-alone in-house efforts.

Consider the real-world example where credit card companies and banks would jointly benefit from an improved fraud detection model. The best performing model would need to be trained using a data set that considers the dynamics of each customer base because of the varying spending patterns of card holders over different customer bases. Due to the competitive nature of the financial industry, credit card companies and banks are reluctant to share their proprietary data with each other or a central repository. To resolve this impasse, it is crucial to design a multi-party machine learning mechanism to facilitate the collaboration among different financial institutions without violating customer privacy or leaking business secret. It should be noted that similar opportunities are present in other fields [19]. In medical or psychological studies, central aggregation of data sets may not be possible or permitted due to logistic complexities or privacy concerns associated with patient records or sensitive human-subject data.

There have been various attempts to address the privacy challenge, such as anonymization and encryption [164, 51, 53]. However, some of the existing solutions either have high communication and computation burden or do not have mathematically rigorous privacy guarantees [150, 109]. Much of the previous work along these lines has been proven to not be private at all [68, 153]. Here, we follow a prevalent theoretical framework for differential privacy [41] and propose a differentially private method to train a deep neural network. There are numerous efforts focused on training differentially private machine learning models from multi-party data sets. One line of research develops algorithms to combine classifiers from different participants. Pathak, Rane, and Raj [124] leverage cryptography to securely average over local classifiers and release a perturbed version of the averaged model. Their work lacks theoretical justification regarding accuracy optimality of the averaged model, as critiqued in Rajkumar and Agarwal [130], and performs poorly when irregular or imbalanced data sets are present across different organizations. Another recent paper [62] proposes an ensemble learning and pseudo-labeling solution that is less restricted to averaging compared to previous proposal. It designates a *trusted authority* to compose an ensemble model of local models and use this model to label privacy-free (public) auxiliary data. The newly-labeled data are then used to train a global differentially private classifier, which can be released safely. The difficulty with this approach is the assumption that a trusted authority exists; the primary motivation for multi-party learning derives from the impossibility of establishing such an entity in practice.

Another line of work attempts to improve over Song, Chaudhuri, and Sarwate [148] by proposing differentially private versions of iterative numerical optimizers, such as private (stochastic) gradient descent. These solutions often work by adding noise to the gradients in each iteration. However, it is possible that the model performance will be severely affected for practical deployment in multi-party learning. Moreover, in a large model like a deep neural network, there are often hundreds of thousands parameters to train and the excessive interactions/iterations results in fairly quick privacy budget consumption.

This research aims to provide a privacy-preserving method for training deep neural networks from multiple sources of data. The performance of the proposed method is

evaluated using a real-world credit card fraud detection data set. Our work bridges the gap between collaborative machine learning and privacy-preserving machine learning, with the aim of providing a framework that can be the basis for banks and other entities to share their data without revealing sensitive information.

In summary, our principal contributions are:

- We provide end-to-end differential privacy guarantee on distributed deep learning, which improves recent proposals in terms of utility and privacy budget consumption.
- We propose a novel differentially private algorithm for selecting the most important gradients in each iteration and prove its privacy properties rigorously.
- We apply the idea of Terngrad to implement an efficient learning process with less communication bandwidth requirement.
- We evaluate our method on a real-world fraud detection data set with multi-million transactions and achieve performances comparable to the previous work Rushin et al. [136] even under privacy constraints.

In the remaining sections of this chapter, Section 3.2 and Section 3.3 present our method and provide a theoretical analysis of its privacy-preserving property. Section 3.4 describes the data set and the experimental results. Finally, in Section 3.5, we conclude our work.

3.2 Methods

In this section, we present our privacy-preserving distributed deep learning system in detail, which enables multiple participants to jointly train a neural network model without sharing raw data. There are in total K participants (banks), each owning a private data set $D^k = \{\mathbf{X}_i^k, y_i^k\}|_{i=1}^{n_k}$ $(k = 1, 2, 3, ..., K)$, where $\mathbf{X}_i^k$ is the feature vector, y_i^k is the corresponding label and n_k is the size of the data set associated with participant k. In addition, there is a parameter server that oversees information integration and dissemination. All the participants will communicate directly with the server (fetching parameters, uploading gradients, etc). The server is responsible for updating the parameters iteratively and

making them available to all participants. Here we assume both the participants and the server are honest but curious. They will follow the protocols honestly, but also try to learn as much information as possible about the other participants from their views of the protocols. Potentially, this can be privacy-leaking as shown in Hitaj, Ateniese, and Pérez-Cruz [66]. Therefore, all the information coming out of a participant has to be differentially private in order to protect its private data set. The structure and basic workflow of the distributed deep learning framework is illustrated in Figure 3.1.

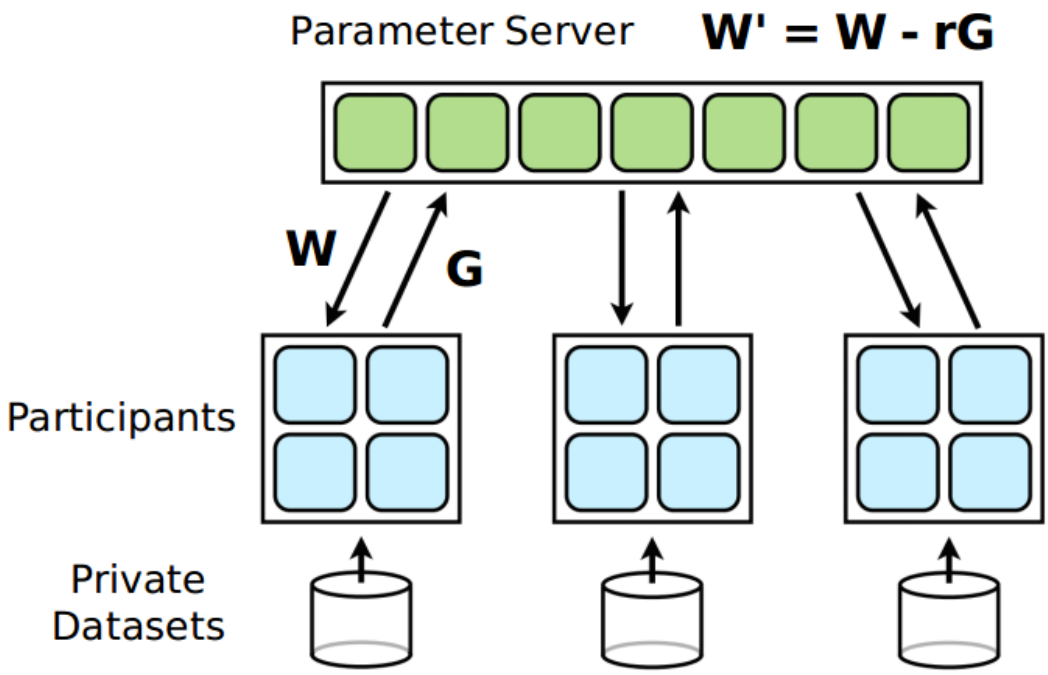


FIGURE 3.1: Diagram of the distributed deep learning system. W represents the parameters, G represents the gradient information and r is the step-size for gradient descent update.

In our privacy-preserving multi-bank credit card fraud detection problem, the goal is to protect the privacy of each bank's customers, or more exactly, obscure which transactions belong to each bank, by enforcing differential privacy. Before jointly training the models, all participating banks will first agree on a common model (the architecture of the neural network, features, activation function in each hidden layer, loss functions etc). The parameter server will initialize the parameters randomly and pass them to the first participant. The first participant trains the model on its own private data set and uploads

the differentially private gradients to the server, who updates the parameters accordingly. Then the server will contact the next participant and pass the new parameters to it for more training, and so on. The whole training process will go on for T iterations. During each iteration, all the participants will be contacted once in a round-robin manner. Following the setting of most existing works [143, 1], each participant will fix its total privacy budget and number of iterations ahead of time and avoid adaptive choice of privacy parameters.

During each iteration, after downloading the current parameters from the server, the participant will sample a small subset from its own private data set and compute the average gradient on this subset. To deal with the exploding gradient problem, the gradient will first be clipped element-wise if it exceeds in absolute value a fix bound B, as is usually done in previous work (e.g., Pascanu, Mikolov, and Bengio [123] and Mikolov [107]). Then we apply the newly proposed differentially private Report-Noisy-Top-N (**RNTN**) algorithm (Algorithm 3) repeatedly in order to select the parameters with large absolute gradients. By focusing on these large gradients, we can save on the privacy budget because the amount spent from the privacy budget is linearly related to the number of gradients uploaded to the server. A similar idea was explored in the work of Shokri and Shmatikov [143], which used a sparse vector technique for this purpose, but the privacy and accuracy analysis was incomplete. Moreover, instead of uploading the noisy gradients to the server, we upload the "ternary gradient" $\{-B, 0, B\}$ based on the sign of the selected noisy gradients in order to reduce the communication burden. This idea is proposed by the recent work of Wen et al. [158].

The details of the model training process are described in Algorithms 1 through 3. Algorithm 1 is the overall parameter update process controlled by the parameter server. Algorithm 2 is the noisy gradients calculation and uploading process going on in a participant during each iteration. Algorithm 3 describes how to pick the top noisy gradients in a differentially private way.

Algorithm 1 Server Side

Require: neural network model M (neural network structure, activation function, loss function $L(\mathbf{W},\ \mathbf{X},\ y)$, features schema, etc), total iteration T, number of participants K, learning rate r_t at iteration t.

Ensure: final differentially private parameter $\mathbf{W} \in R^d$.

1: Initialize the parameter $\mathbf{W}$ (by default initialized to 0) and disseminate the model M to all the participants.
2: **for** $t = 1$ to T **do**
3: **for** $k = 1$ to K **do**
4: Communicate with the k-th participant and pass the current parameters $\mathbf{W}$ to it.
5: Receive the differentially private ternary gradient vector $\tilde{\mathbf{G}} = \mathbf{DPSTGD}()$ from the participant k.
6: Update the current parameters

$$\mathbf{W} := \mathbf{W} - r_t\tilde{\mathbf{G}}. \tag{3.1}$$

7: **end for**
8: **end for**
9: Return $\mathbf{W}$

Algorithm 2 Participant Side: Differentially Private Stochastic Ternary Gradient Descent (**DPSTGD**)

Require: parameters $\mathbf{W} \in R^d$, loss function $L(\mathbf{W},\ \mathbf{X},\ y)$, privacy budget ϵ, number of gradients to upload n, mini-batch sampling ratio q, private data set D, gradient bound $B > 0$.

Ensure: ternary gradient vector $\tilde{\mathbf{G}} \in \{-B, 0, B\}^d$.

1: Sample a subset D_q from the private data set D with sampling probability q.

2: For each point $(\mathbf{X}_i,\ y_i)$ in the subset D_q $(i = 1, 2, ..., |D_q|)$:

calculate gradient:

$$\mathbf{g}_i = \nabla L(\mathbf{W},\ \mathbf{X}_i,\ y_i), \tag{3.2}$$

clip gradient:

$$\hat{\mathbf{g}}_i = \mathbf{g}_i / \max(1, \frac{|\mathbf{g}_i|}{B}). \tag{3.3}$$

3: Take the average of the clipped gradients of all the points in D_q:

$$\mathbf{G} = \frac{1}{|D_q|} \sum_i \hat{\mathbf{g}}_i. \tag{3.4}$$

4: Calculate sensitivity λ of the absolute value of each gradient j $(j = 1, 2, ..., d)$

$$\lambda = \max_{D,D'}(|\mathbf{G}^j(D)| - |\mathbf{G}^j(D')|) = \max_{D,D'} \frac{1}{|D_q|}(|\sum_i \hat{\mathbf{g}}_i^j(D)| - |\sum_i \hat{\mathbf{g}}_i^j(D')|) = \frac{2B}{|D_q|}. \tag{3.5}$$

5: Run **Report-Noisy-Top-N()** with privacy budget ϵ and sensitivity λ to pick the index set $\mathbf{I}$ of the n gradients with the largest noisy absolute values.

6: For each j in the index set $\mathbf{I}$, set

$$\tilde{\mathbf{G}}^j = \left(\mathbf{G}^j + \mathbf{Lap}\left(\frac{\lambda}{\epsilon}\right)\right) \times B; \tag{3.6}$$

for other gradients, set $\tilde{\mathbf{G}}^j = 0$.

7: Return $\tilde{\mathbf{G}}$.

Algorithm 3 Report-Noisy-Top-N (**RNTN**)

Require: privacy budget ϵ, gradient functions g_i $(i = 1, 2, ..., d)$ each with sensitivity at most λ, number of gradients to select N, data set D.

Ensure: index and noisy values of the top N gradients with largest noisy values.

1: Compute the noisy gradients of D:

$$\hat{g}_i(D) = g_i(D) + \nu_i, \tag{3.7}$$

where $\nu_i \sim \mathbf{Lap}(\lambda/\epsilon)$.

2: Return the top N indices with the largest noisy values, and also their noisy values $\hat{g}_i(D)$ $(i \in I)$.

3.3 Privacy Analysis

In this section, we analyze the privacy-preserving property of the proposed algorithms. First we will demonstrate that the mini-batch stochastic gradient descent method in each iteration will greatly save on privacy budget consumption.

Lemma 3.3.1. *Let D be a data set and M be a ϵ-differentially private algorithm on the domain of D. Define the algorithm M_q as follows:*

1. *Sample a random subset D_q from D with sampling probability q.*
2. *Run M on D_q and return the results.*

Then M_q is a $\ln(1 + (e^\epsilon - 1)q)$-differentially private algorithm.

Proof. Let D and D' be two neighboring data sets such that $D' = D \cup \{x\}$. For any event O in the output space of the algorithm M, we consider the following two cases:

1. x is not in D'_q (probability: $1 - q$)
2. x is in D'_q (probability: q)

When x is not in D'_q, the output distribution of running M_q on D is the same as that of running M_q on D'. Therefore,

$$Pr(M_q(D') \in O) = Pr(M_q(D) \in O). \tag{3.8}$$

When x is in D'_q, by the definition of $(\epsilon, 0)$-differential mechanism, we have

$$Pr(M_q(D') \in O) \leq e^{\epsilon} Pr(M_q(D) \in O). \tag{3.9}$$

Taking the probability of each case into account, we have

$$\begin{aligned} Pr(M_q(D') \in O) &= (1-q) \times Pr(M_q(D') \in O | x \notin D'_q) + q \times Pr(M_q(D') \in O | x \in D'_q) \\ &\leq (1-q) Pr(M_q(D) \in O) + q e^{\epsilon} Pr(M_q(D) \in O) \\ &= (1 + (e^{\epsilon} - 1)q) Pr(M_q(D) \in O). \end{aligned} \tag{3.10}$$

Therefore, by the definition of differential privacy, M_q is $(\ln(1+(e^{\epsilon}-1)q), 0)$-differentially private on the domain of D. □

Note that $\ln(1 + (e^{\epsilon} - 1)q) < \epsilon$ when $0 < q < 1$. Lemma 3.3.1 confirms that by sampling from the whole data set for the gradient calculation, which is a common method in stochastic gradient descent and deep neural network, we can save on the privacy budget. The savings are illustrated in Figure 3.2. When ϵ is small, the savings are almost linear with q. However, when ϵ is large, the sampling does not have a significant impact on saving the privacy budget because the savings vanish quickly with respect to q. Luckily, in our situation, the privacy budget ϵ for each iteration is small because we are expecting many iterations and many parameters. Therefore, mini-batch sampling is indeed a great way to save on the privacy budget.

The following theorem provides a rigorous proof of the privacy-preserving property of our proposed method.

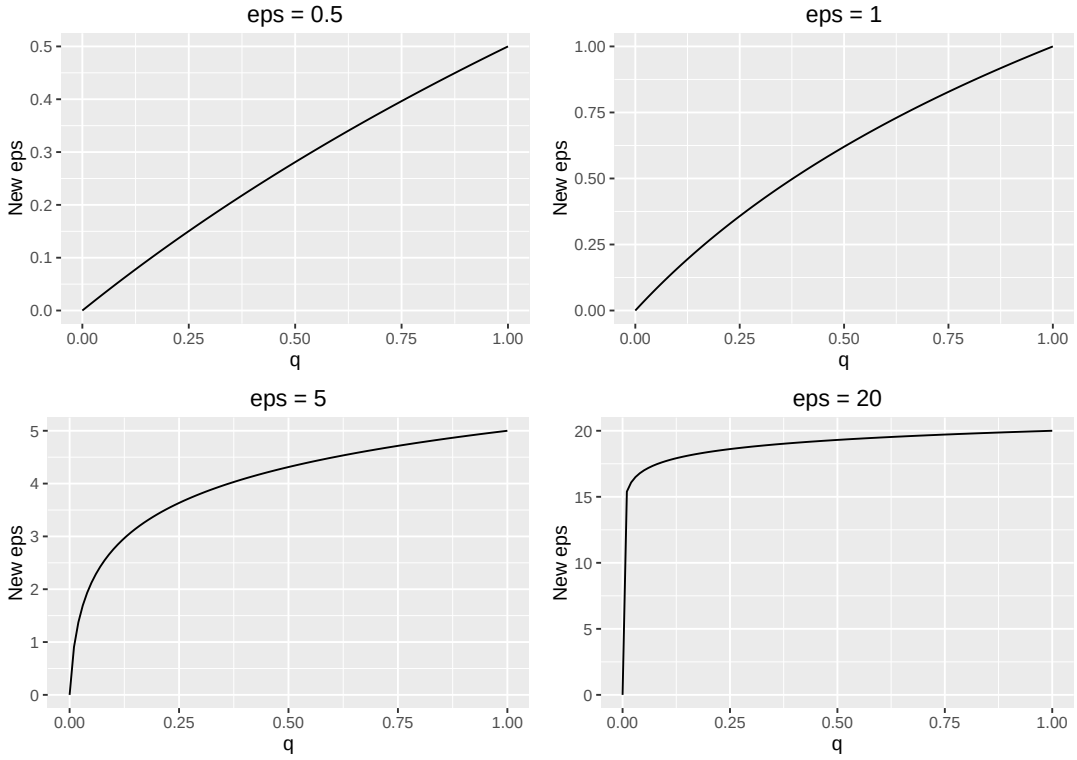


FIGURE 3.2: Visual illustration of Lemma 3.3.1: "eps" is the privacy budget consumed without mini-batch sampling and "new eps" is the privacy budget consumed under varying mini-batch sampling ratio q.

Theorem 3.3.2. *Algorithm 2 is* $(2n\epsilon_g, 0)$*-differentially private, where*

$$\epsilon_g = \ln(1 + (e^{\epsilon} - 1)q). \tag{3.11}$$

Alternatively, for any cryptographically small privacy parameter δ *(e.g.* 2^{-30}*), Algorithm 2 is* $(\sqrt{4n\ln(1/\delta)}\epsilon_g + 2n\epsilon_g(e^{\epsilon_g} - 1), \delta)$*-differentially private.*

Proof. First, we claim that algorithm **Report-Noisy-Top-N** is $2n\epsilon$-differentially private. This is a generalization of the **Report-Noisy-Max** algorithm in Dwork and Roth [42]. For two neighboring data sets D and $D' = D \cup \{x\}$, let g_i, g_i' denote the values of the gradient functions on D and D', and ν_i, ν_i' denote the noises respectively. Fix any output event E with index set I and the corresponding noisy gradient values $\hat{g}_i (i \in I)$, we want to bound the ratio of the probabilities of $Pr[E|D]$ and $Pr[E|D']$.

First we state a useful fact of Laplace distribution. If $\nu \sim \mathbf{Lap}(b)$, then for $t > 0$,

$$Pr[|\nu| \geq tb] = \exp(-t). \tag{3.12}$$

Next, we fix all the noises ν_j for $j \notin I$. For $i \in I$, in order to be selected by the algorithm, it suffices to satisfy that for every $j \notin I$,

$$g_i + \nu_i \geq g_j + \nu_j, \tag{3.13}$$

$$g'_i + \nu'_i \geq g'_j + \nu_j \tag{3.14}$$

Since the sensitivity of the gradient functions is at most λ, we have $g'_i - \lambda \leq g_i \leq g'_i + \lambda$ for all i. Therefore,

$$g_i + \nu_i \geq g_j + \nu_j \Rightarrow g'_i + \nu_i + 2\lambda \geq g'_j + \nu_j, \ \forall j \notin I \tag{3.15}$$

The probability of event E being output on D is

$$Pr[E|D] = \prod_{i \in I} Pr[i, \hat{g}_i|D] = \prod_{i \in I} Pr[v_i > \max_j (g_j + \nu_j) - g_i)] \tag{3.16}$$

$$Pr[E|D'] = \prod_{i \in I} Pr[i, \hat{g}_i|D'] = \prod_{i \in I} Pr[v'_i > \max_j (g'_j + \nu_j) - g'_i] \tag{3.17}$$

To bound $Pr[E|D]/Pr[E|D']$, we look at the ratio of each index $i \in I$ independently.

$$\frac{Pr[\nu_i > \max_j (g_j + \nu_j) - g_i]}{Pr[\nu'_i > \max_j (g'_j + \nu_j) - g'_i]} \leq \frac{Pr[\nu_i > \max_j (g'_j + \nu_j) - g'_i - 2\lambda]}{Pr[\nu'_i > \max_j (g'_j + \nu_j) - g'_i]}, \tag{3.18}$$

where $\nu_i, \nu'_i \sim (\lambda/\epsilon)$. Let $\nu^* = \max_j (g'_j + \nu_j) - g'_i$. We consider the following cases.

1. $\nu^* > 2\lambda$

$$\frac{Pr[\nu_i > \max_j (g'_j + \nu_j) - g'_i - 2\lambda]}{Pr[\nu'_i > \max_j (g'_j + \nu_j) - g'_i]} = \frac{\frac{1}{2}\exp(-(\nu^* - 2\lambda)\epsilon/\lambda)}{\frac{1}{2}\exp(-\nu^*\epsilon/\lambda)} = \exp(2\epsilon) \tag{3.19}$$

2. $0 < \nu^* < 2\lambda$

$$\frac{Pr[\nu_i > \max_j (g'_j + \nu_j) - g'_i - 2\lambda]}{Pr[\nu'_i > \max_j (g'_j + \nu_j) - g'_i]} = \frac{1 - \frac{1}{2}\exp((\nu^* - 2\lambda)\epsilon/\lambda)}{\frac{1}{2}\exp(-\nu^*\epsilon/\lambda)} \leq \exp(2\epsilon) \quad (3.20)$$

3. $\nu^* < 0$

$$\frac{Pr[\nu_i > \max_j (g'_j + \nu_j) - g'_i - 2\lambda]}{Pr[\nu'_i > \max_j (g'_j + \nu_j) - g'_i]} = \frac{1 - \frac{1}{2}\exp((\nu^* - 2\lambda)\epsilon/\lambda)}{1 - \frac{1}{2}\exp(\nu^*\epsilon/\lambda)} \leq \exp(2\epsilon) \quad (3.21)$$

Therefore, for all $i \in I$

$$\frac{Pr[\nu_i > \max_j (g_j + \nu_j) - g_i]}{Pr[\nu'_i > \max_j (g'_j + \nu_j) - g'_i]} \leq \exp(2\epsilon) \quad (3.22)$$

Similarly, we can also bound the probability ratio $Pr[E|D]/Pr[E|D']$ and conclude that

$$Pr[E|D]/Pr[E|D'] \leq \prod_{i \in I} \exp(2\epsilon) = \exp(2N\epsilon) \quad (3.23)$$

and

$$Pr[E|D']/Pr[E|D] \leq \prod_{i \in I} \exp(2\epsilon) = \exp(2N\epsilon), \quad (3.24)$$

which by the definition of differential privacy proves that **Report-Noisy-Top-N** algorithm is $2N\epsilon$-differentially private.

By Lemma 3.3.1, if only a mini-batch q of the private data set is sampled during each iteration, then the privacy budget spent by **Report-Noisy-Top-N** per uploaded gradient is

$$\epsilon_g = \ln(1 + (e^\epsilon - 1)q). \quad (3.25)$$

From basic composition theorem 2.3.1, the total privacy budget spent on uploading N noisy gradients during each iteration is $2n\epsilon_g$. Alternatively, from advanced composition theorem 2.3.2, for any tiny privacy parameter δ, algorithm 2 is $(\sqrt{4n\ln(1/\delta)}\epsilon_g + 2n\epsilon_g(e^{\epsilon_g} - 1), \delta)$-differentially private. □

Next, we will present the utility analysis of the **Report-Noisy-Top-N** algorithm in

selecting the set of largest absolute gradients. Since it is a generalization of **Report-Noisy-Max** [42], we will focus on bounding the probability ratio of selecting the largest value and second largest value in the **Report-Noisy-Max** algorithm.

Theorem 3.3.3 (Accuracy analysis of **Report-Noisy-Max**)**.** *Let ϵ be the privacy parameter in the **Report-Noisy-Max** algorithm, and let λ be the sensitivity of the absolute gradient. The probability of selecting the gradient with the largest absolute value is at least $e^{\Delta\epsilon/\lambda}/\left(\Delta\epsilon/4\lambda + 0.5\right) - 1$ larger than that of selecting the gradient with the second largest absolute value, where Δ is the difference between the largest absolute gradient and the second largest absolute gradient.*

Proof. First, given N absolute gradients $|g_1|$, $|g_2|$, ... , $|g_N|$, we derive the probability P_i of selecting an arbitrary i-th gradient $|g_i|$ by **Report-Noisy-Max** algorithm.

$$P_i = \prod_{j=1,..,N,j\neq i} Pr(|g_i| + n_i > |g_j| + n_j) = \prod_{j=1,..,N,j\neq i} Pr(n_i > n_j + \Delta_{j,i}), \tag{3.26}$$

where n_i, $n_j \sim \text{Lap}(\lambda/\epsilon)$ and $\Delta_{j,i} = |g_j| - |g_i|$. Then we have

$$\begin{aligned}
&Pr(n_i > n_j + \Delta_{j,i}) \\
=&\int_{-\infty}^{\infty} Pr(n_i > n_j + \Delta_{j,i} | n_i = t) Pr(n_i = t)\,\mathrm{d}t \\
=&\int_{-\infty}^{\infty} Pr(n_j < t - \Delta_{j,i}) Pr(n_i = t)\,\mathrm{d}t \\
=&\frac{\epsilon}{4\lambda}\int_{-\infty}^{\Delta_{j,i}} \exp(\frac{\epsilon t - \epsilon|t| - \Delta_{j,i}\epsilon}{\lambda})\,\mathrm{d}t + \frac{\epsilon}{2\lambda}\int_{\Delta_{j,i}}^{\infty} \exp(-\frac{\epsilon|t|}{\lambda})\left[1 - \frac{1}{2}\exp(\frac{\Delta_{j,i}\epsilon - \epsilon t}{\lambda})\right]\,\mathrm{d}t.
\end{aligned} \tag{3.27}$$

We consider the following two cases:

1. When $\Delta_{j,i} > 0$, Equation 3.27 becomes

$$\begin{aligned}
&Pr(n_i > n_j + \Delta_{j,i}) \\
=&\frac{\epsilon}{4\lambda}\exp(-\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{-\infty}^{0}\exp(\frac{2\epsilon t}{\lambda})\,\mathrm{d}t + \frac{\epsilon}{4\lambda}\exp(-\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{0}^{\Delta_{j,i}}\mathrm{d}t + \frac{\epsilon}{2\lambda}\int_{\Delta_{j,i}}^{\infty}\exp(-\frac{\epsilon t}{\lambda})\,\mathrm{d}t \\
&-\frac{\epsilon}{4\lambda}\exp(\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{\Delta_{j,i}}^{\infty}\exp(-\frac{2\epsilon t}{\lambda})\,\mathrm{d}t \\
=&\exp(-\frac{\Delta_{j,i}\epsilon}{\lambda})\left(\frac{\Delta_{j,i}\epsilon}{4\lambda}+\frac{1}{2}\right).
\end{aligned} \tag{3.28}$$

2. When $\Delta_{j,i} < 0$, Equation 3.27 becomes

$$\begin{aligned}
&Pr(n_i > n_j + \Delta_{j,i}) \\
=&\frac{\epsilon}{4\lambda}\exp(-\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{-\infty}^{\Delta_{j,i}}\exp(\frac{2\epsilon t}{\lambda})\,\mathrm{d}t + \frac{\epsilon}{2\lambda}\int_{\Delta_{j,i}}^{0}\exp(\frac{\epsilon t}{\lambda})\,\mathrm{d}t + \frac{\epsilon}{2\lambda}\int_{0}^{\infty}\exp(-\frac{\epsilon t}{\lambda})\,\mathrm{d}t \\
&-\frac{\epsilon}{4\lambda}\exp(\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{\Delta_{j,i}}^{0}\mathrm{d}t - \frac{\epsilon}{4\lambda}\exp(\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{0}^{\infty}\exp(-\frac{2\epsilon t}{\lambda})\,\mathrm{d}t \\
=&1-\frac{1}{2}\exp(\frac{\Delta_{j,i}\epsilon}{\lambda})+\frac{\Delta_{j,i}\epsilon}{4\lambda}\exp(\frac{\Delta_{j,i}\epsilon}{\lambda}).
\end{aligned} \tag{3.29}$$

Note that both Equation 3.28 and Equation 3.29 are decreasing functions of $\Delta_{j,i}$, meaning that it is less likely to pick i over j when $\Delta_{j,i}$ increases.

Next, consider the gradient with the largest absolute value and the gradient with the second largest absolute value. Assume the index of these two gradients are m_1 and m_2 respectively. Then by Equation 3.28 and Equation 3.29, we have the probability of selecting m_1 and m_2 are

$$P_{m_1} = \prod_{\substack{j=1,..,N,\\ j\neq m_1}} Pr(|g_{m_1}| + n_{m_1} > |g_j| + n_j) = \prod_{\substack{j=1,..,N,\\ j\neq m_1}} \left[1-\frac{1}{2}\exp(\frac{\Delta_{j,m_1}\epsilon}{\lambda})+\frac{\Delta_{j,m_1}\epsilon}{4\lambda}\exp(\frac{\Delta_{j,m_1}\epsilon}{\lambda})\right], \tag{3.30}$$

$$P_{m_2} = \prod_{\substack{j=1,..,N,\\ j\neq m_2}} Pr(|g_{m_2}| + n_{m_2} > |g_j| + n_j)$$
$$= \exp(-\frac{\Delta_{m_1,m_2}\epsilon}{\lambda}) \left(\frac{\Delta_{m_1,m_2}\epsilon}{4\lambda} + \frac{1}{2}\right) \prod_{\substack{j=1,..,N,\\ j\neq m_2}} \left[1 - \frac{1}{2}\exp(\frac{\Delta_{j,m_2}\epsilon}{\lambda}) + \frac{\Delta_{j,m_2}\epsilon}{4\lambda}\exp(\frac{\Delta_{j,m_2}\epsilon}{\lambda})\right]. \tag{3.31}$$

Therefore,

$$\frac{P_{m_1}}{P_{m_2}} = \frac{\left[1 - \frac{1}{2}\exp(\frac{\Delta_{m_2,m_1}\epsilon}{\lambda}) + \frac{\Delta_{m_2,m_1}\epsilon}{4\lambda}\exp(\frac{\Delta_{m_2,m_1}\epsilon}{\lambda})\right]}{\exp(-\frac{\Delta_{m_1,m_2}\epsilon}{\lambda})\left(\frac{\Delta_{m_1,m_2}\epsilon}{4\lambda} + \frac{1}{2}\right)} \times \underbrace{\prod_{\substack{j=1,..,N,\\ j\neq m_1,m_2}} \frac{\left[1 - \frac{1}{2}\exp(\frac{\Delta_{j,m_1}\epsilon}{\lambda}) + \frac{\Delta_{j,m_1}\epsilon}{4\lambda}\exp(\frac{\Delta_{j,m_1}\epsilon}{\lambda})\right]}{\left[1 - \frac{1}{2}\exp(\frac{\Delta_{j,m_2}\epsilon}{\lambda}) + \frac{\Delta_{j,m_2}\epsilon}{4\lambda}\exp(\frac{\Delta_{j,m_2}\epsilon}{\lambda})\right]}}_{A}. \tag{3.32}$$

Note that each component in the product A is larger than 1 since $\Delta_{j,m_1} < \Delta_{j,m_2}$ and Equation 3.29 is a decreasing function. Therefore,

$$\frac{P_{m_1}}{P_{m_2}} \geq \frac{\left[1 - \frac{1}{2}\exp(\frac{\Delta_{m_2,m_1}\epsilon}{\lambda}) + \frac{\Delta_{m_2,m_1}\epsilon}{4\lambda}\exp(\frac{\Delta_{m_2,m_1}\epsilon}{\lambda})\right]}{\exp(-\frac{\Delta_{m_1,m_2}\epsilon}{\lambda})\left(\frac{\Delta_{m_1,m_2}\epsilon}{4\lambda} + \frac{1}{2}\right)} = \frac{\exp(\frac{\Delta_{m_1,m_2}\epsilon}{\lambda})}{\frac{\Delta_{m_1,m_2}\epsilon}{4\lambda} + \frac{1}{2}} - 1. \tag{3.33}$$

□

The theorem above gives a utility guarantee in the sense that small gradients are exponentially less likely to be selected by the **Report-Noisy-Max** algorithm after noise is added.

3.4 Experiments

The main goal of the experiment is to implement our method and evaluate its utility-privacy trade-off on a real-world credit card fraud data set. This experiment reflects the previously described motivating example where several independent banking institutions would like to improve the performance of fraud detection models while maintaining customer privacy and safeguarding business secrets. Imposing privacy requirements will

generally degrade the model performance due to the random noise added to the gradients. We build and train neural networks on the preprocessed data set under varying settings of privacy parameters, mini-batch ratio and ratio of uploaded gradients. In particular, we are interested in how the performance of the neural network varies with the privacy budget. The results demonstrate that the proposed method achieves performance close to the non-private baseline even under relatively strict privacy requirement.

3.4.1 Data preprocessing

In the experiment, a real-world credit card transaction data set contributed by a US bank is studied. All the 78 million transactions from over 2 million accounts happened in the first eight months of 2013 and were labeled as either non-fraudulent or fraudulent. There are 69 categorical and numeric features in the original data set, including transaction amount, transaction date and time, merchant code, account-level information, card present or not, distance to merchant, etc. In addition, we derived new features reflecting the spending pattern of customers with the help of domain expertise, such as amount difference and distance difference from last transaction, time duration since last transaction, velocity variables, standard deviation of transaction amount for each account over time, etc. We apply effect coding to all the categorical variables and standardize all the numerical variables. There are in total 264 features in the processed data set.

The percentage of fraudulent transaction in the entire data set is 0.136%. Since it is extremely unbalanced, we decide to under-sample the non-fraudulent transactions at the account-level. To be more specific, we kept all the accounts with fraudulent transactions and randomly sample the same amount of accounts from the remaining non-fraudulent accounts. All the fraudulent transactions and a random 20% of all the non-fraudulent transactions from our selected accounts are used as our sample for building and testing the neural network model. After preprocessing, there are around 0.7 million transactions with a fraud rate of 14% in the sample.

3.4.2 Results

The model used in the numerical experiments is a generic neural network with 2 hidden layers of neurons. The number of neurons in each hidden layer are 50 and 20 respectively. Therefore, the total number of weights and bias parameters in the neural network is 14,312. To simulate the multi-party situation, the accounts are divided evenly among each participating bank and a holdout test set. The data from all the participants is used to train the neural network, and the test set is used to evaluate the algorithms.

First, we present the amount of the privacy budget consumed at each iteration, and how the privacy budget is affected by the number of uploaded gradients n and the mini-batch sampling ratio q. The results are displayed in Table 3.1. We can see that without mini-batch sampling and gradients selection, the privacy budget consumed at each iteration will explode, which makes the differentially private method impractical. Sampling a mini-batch and uploading a proportion of all the gradients will save a lot privacy budget without hurting the model performance (see the experiments below). Moreover, if we can tolerate a tiny probability for the algorithm failing to preserve privacy (e.g. $\delta = 2^{-30}$), the advanced composition theorem will greatly save the privacy budget compared to the basic composition theorem.

To illustrate the utility-privacy trade-off, we compare the area under the receiver operating characteristics curve (AUC) of the model on the test set of our approach to a non-private neural network baseline under different values of ϵ. In addition, we also test the impact of the number of participants in the systems, the mini-batch sampling ratio, and the number of uploaded gradients. The whole analysis was repeated 10 times with different seeds for random number generator. The results are very similar for different trials. For sake of clarity only a single result is presented.

ϵ	q	n	ϵ_{iter}^{basic}	ϵ_{iter}^{adv}
0.1	0.01	1431	3.01	0.37
0.1	0.01	2862	6.02	0.52
0.1	0.05	1431	15.01	1.88
0.1	1	14312	2862.4	410.1
0.5	0.01	1431	18.50	2.35
0.5	0.01	2862	37.01	3.39
0.5	0.05	1431	91.35	13.97
0.5	1	14312	14312.4	9830.1

TABLE 3.1: Privacy budget consumption at each iteration. ϵ is the privacy parameter in Algorithm 2, q is the mini-batch sampling ratio and n is the number of uploaded gradients. ϵ_{iter}^{basic} is the privacy budget consumed by basic composition theorem; ϵ_{iter}^{adv} is the privacy budget consumed by advanced composition theorem when $\delta = 2^{-30}$.

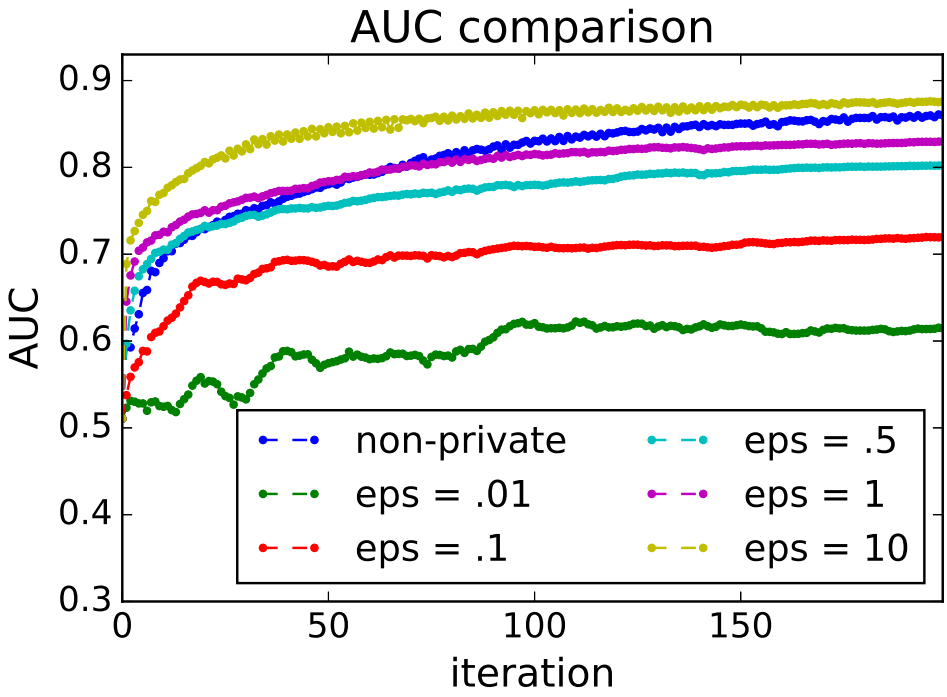


FIGURE 3.3: AUC comparison under different privacy budget (5 participants, mini-batch ratio = 0.01, ratio of uploaded gradient = 10%).

Figure 3.3 displays the AUC on the test set under different per-iteration privacy budgets iteration for 5 participants. We set the mini-batch ratio to be 0.01 and for our

DPSTGD algorithm, only 10% of all the gradients are uploaded to the sever in each iteration. As expected, a smaller privacy budget ($\epsilon = 0.01$ or 0.1) gives relatively weaker performance. For an intermediate differential privacy guarantee ($\epsilon = 0.5$ or 1), the performance is comparable to the non-private baseline. It is also worth noting that when the privacy budget is large ($\epsilon = 10$), the performance of our method even beats the non-private baseline. This experiment is a proof-of-concept that privacy preservation can give satisfactory performance even under an intermediate differential privacy guarantee on a real-world data set. Moreover, adding a small amount of noise and focusing on the top gradients will not only reduce the communication burden but also help learning the model faster and improving the accuracy. It is also consistent with the phenomenon studied in the paper [115]. Our understanding is that adding noise to gradients most likely helps the optimizer jump out of local minimum for a non-convex problem like deep neural network, whereas an exact gradient descent method will sometimes get the optimizer stuck. Figure 3.4 presents the results when there are 30 participants. The trend of AUC curves is similar to that of Figure 3.3.

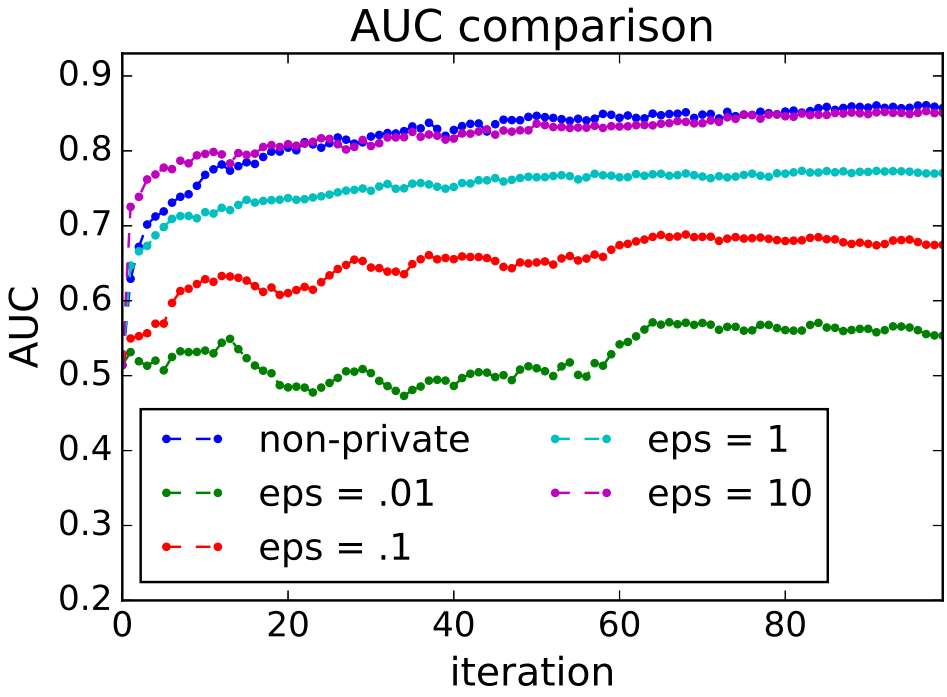


FIGURE 3.4: AUC comparison under different privacy budget (30 participants, mini-batch ratio = 0.01, ratio of uploaded gradient = 10%).

Lastly, we look at the impact of mini-batch ratios and ratio of uploaded gradients with $\epsilon = 1$. As expected, Figure 3.5 and 3.6 show that when the mini-batch ratio and the ratio of uploaded gradients are increased, the performance is also improved. Note that even if only 30% of all the noisy ternary gradients are uploaded to the server, the model performance is superior to that of the non-private baseline. It further verifies the impression that for a deep neural network, only a proportion of the gradients are large enough to change the parameter and improve the value of loss function, and our **Report-Noisy-Top-N** algorithm is effective in selecting these gradients in a differential private manner.

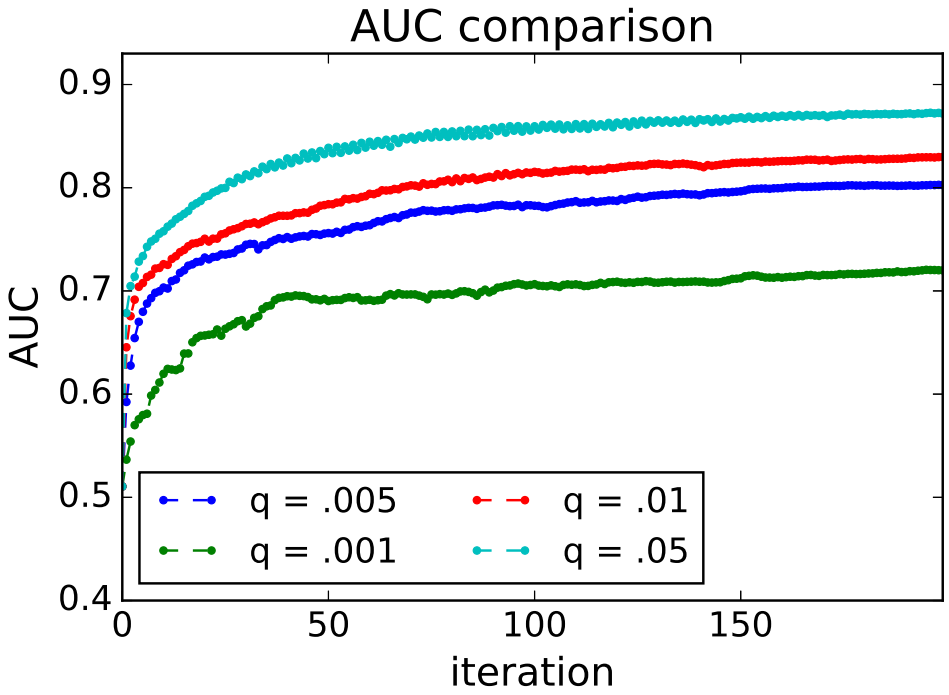


FIGURE 3.5: AUC comparison under mini-batch ratio (5 participants, $\epsilon = 1$, ratio of uploaded gradient = 10%).

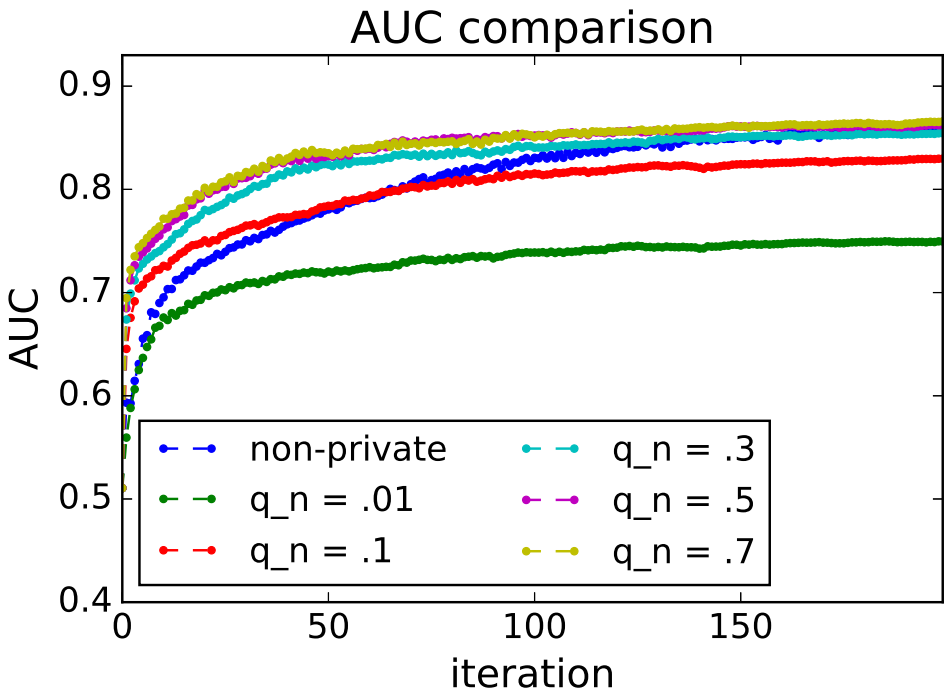


FIGURE 3.6: AUC comparison under different ratios of uploaded gradient q_n (5 participants, $\epsilon = 1$, mini-batch ratio = 1%).

3.5 Chapter Conclusion

In this chapter, we propose a privacy-preserving method to train deep neural networks in a distributed setting and evaluate its utility-privacy trade-off on a real-world credit card fraud detection data set. In order to save privacy budget consumption, we sample a mini-batch from the private data set of each participant and focus on the large absolute gradients. In addition, we upload ternary gradients instead of the exact gradients to reduce communication burden. Our privacy-preserving method achieves model performance (measured by AUC on test set) comparable to the non-private baseline. It provides a practical solution to the multi-party privacy-preserving deep learning, which is especially beneficial to financial institutions who are willing to jointly learn machine learning models, but are prohibited by privacy restrictions.

Chapter 4

Differentially Private Hypothesis Transfer Learning

In recent years, the focus of machine learning has been shifting to the paradigm of transfer learning where the data distribution in the target domain differs from that in the source domain. This is a prevalent setting in real-world classification problems and numerous well-established theoretical results in the classical supervised learning paradigm will break down under this setting. In addition, the increasing privacy protection awareness restricts access to source domain samples and poses new challenges for the development of privacy-preserving transfer learning algorithms. In this paper, we propose a novel differentially private multiple-source hypothesis transfer learning method for logistic regression. The target learner operates on differentially private hypotheses and importance weighting information from the sources to construct informative Gaussian priors for its logistic regression model. By leveraging a publicly available auxiliary data set, the importance weighting information can be used to determine the relationship between the source domain and the target domain without leaking source data privacy. Our approach provides a robust performance boost even when high quality labeled samples are extremely scarce in the target data set. The extensive experiments on two real-world data sets confirm the performance improvement of our approach over several baselines.

4.1 Introduction

In the era of big data, an abundant supply of high quality labeled data is crucial for the success of modern machine learning. But it is both impractical and unnecessary for a single entity, whether it is a tech giant or a powerful government agency, to collect the massive amounts of data single-handedly and perform the analysis in isolation. Collaboration among data centers or crowd-sourced data sets can be truly beneficial in numerous data-driven research areas and industries nowadays. As modern data sets get bigger and more complex, the traditional practice of centralizing data from multiple data owners turns out to be extremely inefficient. Increasing concerns over data privacy also create barriers to data sharing, making it difficult to coordinate large-scale collaborative studies especially when sensitive human subject data (e.g., medical records, financial information) are involved. Instead of moving raw data, a more efficient approach is to exchange the intermediate computation results obtained from distributed training data sets (e.g., gradients [114], likelihood values in MLE [19]) during the machine learning process. However, even the exchange of intermediate results or summary statistics is potentially privacy-breaching as revealed by recent evidence [68, 153]. There is a large body of research work addressing this problem in a rigorous privacy-preserving manner, especially under the notion of differential privacy [130, 63, 143, 154].

In this paper, we consider the differentially private distributed machine learning problem under a more realistic assumption – the multiple training data sets are drawn from different distributions and the learned machine learning hypothesis (i.e. classifier or predictive model) will be tested and employed on a data set drawn from another different yet related distribution. In this setting, the training sets are referred to as the "source domains" and the test set is referred to as the "target domain". High quality labeled data in the target domain is usually costly, if not impossible, to collect, making it necessary and desirable to take advantage of the sources in order to build a reliable hypothesis on the target. The task of "transferring" source knowledge to improve target "learning" is known as transfer learning, one of the fundamental problems in statistical learning that

is attracting increasing attention from both academia and industry. This phenomenon is prevalent in real-world applications (e.g., robot manipulation [134], visual object recognition [46], sentiment analysis [16]) and solving the problem is important because numerous solid theoretical justification, especially the generalization capability, in traditional supervised learning paradigm will break down under transfer learning. Plenty of research work has been done in this area, as surveyed in [120]. Meanwhile, the unprecedented emphasis on data privacy today brings up a natural question for the researchers:

How to improve the learning of the hypothesis on the target using knowledge of the sources while protecting the data privacy of the sources?

To our best knowledge, limited research work exists on differentially private multiple-source transfer learning. Two challenges emerge when addressing this problem – what to transfer and how to ensure differential privacy. In the transfer learning literature, there are four major approaches – instance re-weighting [91, 73, 54], feature representation transfer [129], hypothesis transfer [82, 152, 81] and relational knowledge transfer [106]. In particular, we focus on hypothesis transfer learning, where hypotheses trained on the source domains are utilized to improve the learning of target hypothesis and no access to source data is allowed [82]. To better incorporate the source hypotheses, we also adapt an importance weighting mechanism in [71] to measure the relationship between sources and target by leveraging a public auxiliary unlabeled data set. Based on the source-target relationship, we can determine how much weight to assign to each source hypothesis in the process of constructing informative Gaussian priors for the parameters of the target hypothesis. Furthermore, the enforcement of differential privacy requires an additional layer of privacy protection because unperturbed source domain knowledge may incur privacy leakage risk. In the following sections, we explicitly apply our methodology to binary logistic regression. Our approach will provide end-to-end differential privacy guarantee by perturbing the importance weighting information and the source hypotheses before transferring them to the target. Extensive empirical evaluations on real-world data demonstrate its utility and significant performance lift over several important baselines.

The main contributions of this work can be summarized as follows.

- We propose a novel multiple-source differentially private hypothesis transfer learning algorithm and focus specifically on its application in binary logistic regression. It addresses the notorious problem of insufficient labeled target data by making use of unlabeled data and provides rigorous differential privacy guarantee.
- Compared with previous work, only one round of direct communication between sources and target is required in our computationally efficient one-shot model aggregation, therefore overcoming some known drawbacks (e.g. avoiding the complicated iterative hypothesis training process and privacy budget composition in [60, 163], eliminating the need for a trusted third party in [62]).
- The negative impact of unrelated source domains (i.e. negative transfer) is alleviated as the weight assigned to each source hypothesis will be determined by its relationship with the target.
- The non-private version of our method achieves performance comparable to that of the target hypothesis trained with an abundant supply of labeled data. It provides a new perspective for multiple-source transfer learning when privacy is not a concern.

The rest of the paper is organized as follows: background and related work are introduced in Section 4.2. We then present the methods and algorithms in Section 4.3. It is followed by empirical evaluation of the method on two real-world data sets in Section 4.4. Lastly, we conclude the work in Section 4.5.

4.2 Background and Related Work

The privacy notion we use is differential privacy, which is a mathematically rigorous definition of data privacy [43]. Differential privacy essentially implies that the existence of any particular data point $\boldsymbol{s}$ in a private data set $\boldsymbol{S}$ cannot be determined by analyzing the output of a differentially private algorithm $\boldsymbol{M}$ applied on $\boldsymbol{S}$. The parameter ϵ is typically known as the privacy budget which quantifies the privacy loss whenever an algorithm is applied on the private data. A differentially private algorithm with smaller ϵ

means that it is more private. The other privacy parameter δ represents the probability of the algorithm failing to satisfy differential privacy and is usually set to zero.

Differential privacy can be applied to address the problem of private machine learning. The final outputs or the intermediate computation results of machine learning algorithms can potentially reveal sensitive information of individuals who contribute data to the training set. A popular approach to achieving differential privacy is output perturbation, derived from the sensitivity method in [43]. It works by adding carefully calibrated noises to the parameters of the learned hypothesis before releasing it. In particular, Chaudhuri et al. [31] adapted the sensitivity method into a differentially private regularized empirical risk minimization (ERM) algorithm of which logistic regression is a special case. They proved that adding Laplace noise with scale inversely proportional to the privacy parameter ϵ and the regularization parameter λ to the learned hypothesis provides $(\epsilon, 0)$-differential privacy. Besides, there has been a large body of systematic theoretical work in the differentially private ERM literature deriving error bounds and efficiency [7] and investigating high-dimensional sparse regression problems [78].

More recently, research efforts have been focused on distributed privacy-preserving machine learning where private data sets are collected by multiple parties. One line of research involves exchanging differentially private information (e.g. gradients) among multiple parties during the iterative hypothesis training process [63, 143, 147, 1, 154]. An alternative line of work focuses on privacy-preserving model aggregation techniques. Pathak et al. [124] addressed the hypothesis ensemble problem by simple parameter averaging and publishing the aggregated hypothesis in a differentially private manner. Papernot et al. [122] proposed a different perspective – Private Aggregation of Teacher Ensembles (PATE), where private data set is split into disjoint subsets and different "teacher" hypotheses are trained on all subsets. The private teacher ensemble is used to produce labels on auxiliary unlabeled data by noisy voting and a "student" hypothesis is trained on the auxiliary data and released. Hamm et al. [62] explored a similar strategy of transferring local hypotheses and focused on convex loss functions. Their work involves a "trusted entity" who collects unperturbed local hypotheses trained on multiple private data sets

and uses them to generate "soft" labels on an auxiliary unlabeled data set. A global hypothesis is then trained on the soft-labeled auxiliary data and output perturbation is used to ensure its differential privacy.

In the limited literature of differentially private transfer learning, the most recent and relevant work to ours are [60, 163] which focus on multi-task learning [29], one of the variants of transfer learning. Gupta et al. [60] proposed a differentially private mutli-task learning algorithm using noisy task relationship matrix and developed a novel attribute-wise noise addition scheme. Xie et al. [163] introduced a privacy-preserving proximal gradient algorithm to asynchronously update the hypothesis parameters of the learning tasks. Both work are iterative solutions and require multiple rounds of information exchange between tasks. However, one of the key drawbacks of iterative differentially private methods is that privacy risks accumulate with each iteration. The composition theorem of differential privacy [42] provides a framework to quantify the privacy budget consumption. Given a certain level of total privacy budget, there is a limit on how many iterations can be performed on a specific private data set, which severely affects the utility-privacy trade-off of iterative solutions.

4.3 The Proposed Methods

Let us assume that there are K sources in the transfer learning system (as shown in Fig. 4.1), indexed as $k = 1, ..., K$. For the k-th source, we denote the labeled samples as $\boldsymbol{S}_l^k = \{(\boldsymbol{x}_i^k, y_i^k) : 1 \leq i \leq n_l^k\}$ and the unlabeled samples as $\boldsymbol{S}_{ul}^k = \{\boldsymbol{x}_j^k : 1 \leq j \leq n_{ul}^k\}$, where $\boldsymbol{x}_i^k, \boldsymbol{x}_j^k \in^d$ are the feature vectors, y_i^k is the corresponding label, and n_l^k, n_{ul}^k are the sizes of $\boldsymbol{S}_l^k$ and $\boldsymbol{S}_{ul}^k$ respectively. The samples in the k-th source are drawn i.i.d. according to a probability distribution $\mathcal{D}^k$. There is also a target data set $\boldsymbol{T}$ with abundant unlabeled samples and very few labeled samples drawn i.i.d. from a different yet related distribution $\mathcal{D}^T$. Following the setting of [100], $\mathcal{D}^T$ is assumed to be a mixture of the source distributions $\mathcal{D}^k$'s. A public data set $\boldsymbol{P}$ of size n^P (not necessarily labeled) is accessible to both the sources and the target serving as an information intermediary.

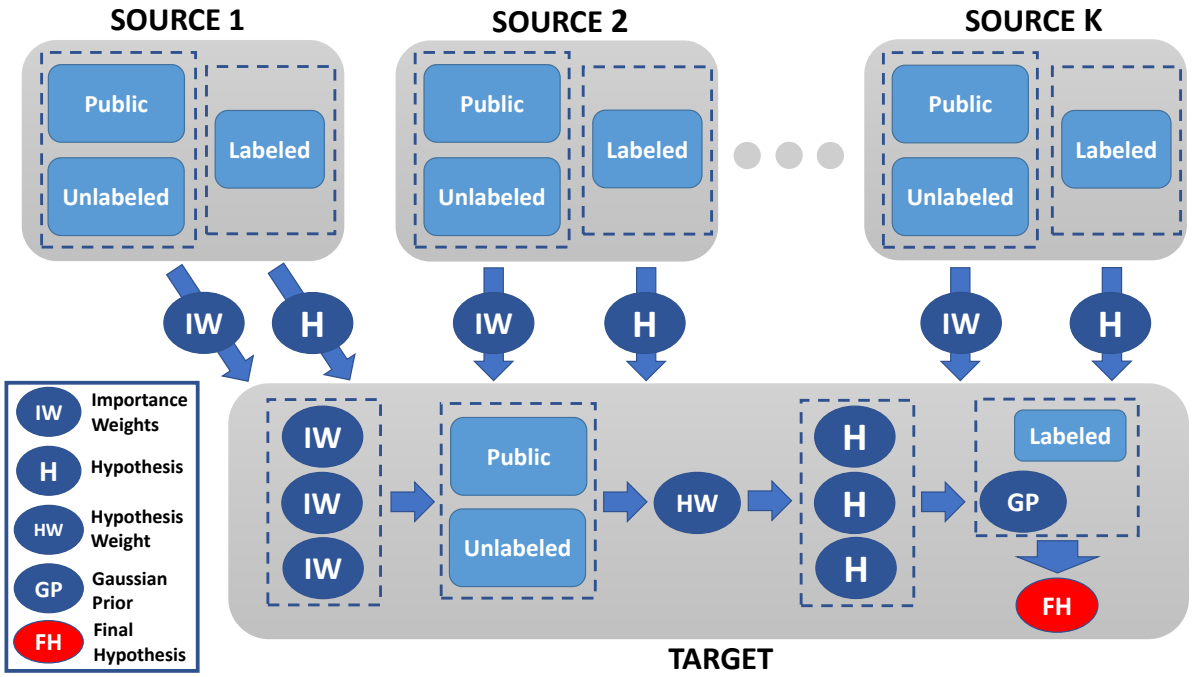


FIGURE 4.1: Diagram of the multiple-source transfer learning system

We present the full details of our method (Algorithm 4) here. Two pieces of knowledge need to be transferred to the target from each source in a differentially private manner. The first piece is the hypothesis locally trained on the labeled source samples $\boldsymbol{S}_l^k$. In order to protect the data privacy, we apply the output perturbation method in [31] to perturb the source hypotheses into $\boldsymbol{\theta}_{priv}^k$ $(k = 1, ..., K)$ before transferring them to the target (Step 1 in Algorithm 4).

The second piece of knowledge is the differentially private "importance weights" vector with respect to samples in the public data set $\boldsymbol{P}$, which is used to measure the relationship between source distribution and target distribution. It is a very difficult task to estimate high-dimensional probability densities and releasing the unperturbed distribution or the histogram can be privacy-breaching in itself [166]. No existing work addresses the differentially private source-target relationship problem explicitly. We leverage a public data set $\boldsymbol{P}$ and propose a novel method based on the importance weighting mechanism in [71] (see Algorithm 5) which was originally developed for the purpose of differentially private data publishing. We take advantage of the "importance weights" vector and successfully quantify the source-target relationship without violating the data privacy of sources.

To be more specific, every source k will compute the differentially private "importance weight" for each sample in $\boldsymbol{P}$ using its unlabeled samples $\boldsymbol{S}_{ul}^k$ (Step 2 in Algorithm 4). The "importance weight" of a data point in $\boldsymbol{P}$ is large if it is similar to the samples in $\boldsymbol{S}_{ul}^k$, and small otherwise. The "importance weights" vector $\boldsymbol{w}^k$ is therefore an n^P-dimensional vector with non-negative entries that add up to n^P. By making a multiplicative adjustment to the public data set $\boldsymbol{P}$ using $\boldsymbol{w}^k$, $\boldsymbol{P}$ will look similar to $\boldsymbol{S}_{ul}^k$ in proportion. In other words, $\boldsymbol{w}^k$ is the "recipe" of transforming $\boldsymbol{P}$ into a data set drawn from $\mathcal{D}^k$.

After receiving the "importance weights" vectors $\boldsymbol{w}^k$'s from sources, the target will also compute its own non-private "importance weight" vector $\boldsymbol{w}^T$ (Step 3 in Algorithm 4). These "recipes" bear information about $\mathcal{D}^k$'s and $\mathcal{D}^T$ and are crucial for determining how related each source is to the target. As an intuitive example, the recipe of making regular grape juice out of grapes and water is similar to the recipe of making light grape juice, but very different from that of making wine. By simply comparing the recipes, regular juice can conclude that it is alike light juice but not wine. Given $\boldsymbol{w}^k$'s and $\boldsymbol{w}^T$, the target can further compute the "hypothesis weight" to assign to each source hypothesis by solving an optimization problem that minimizes the divergence between target "importance weights" vector and a linear combination of source "importance weights" vectors (Step 4 in Algorithm 4). The "hypothesis weights" vector $\boldsymbol{w}_H$ is a probability vector of dimension K. The implied assumption here is that hypotheses trained on sources similar to the target should be assigned higher weights in the model aggregation process, whereas the impact of hypotheses trained on source domains unrelated to the target should be reduced. Finally, the target will construct an informative Bayesian prior for its logistic regression model using $\boldsymbol{\theta}_{priv}^k$'s and $\boldsymbol{w}_H$ based on the method in [101] (Step 5-6 in Algorithm 4).

To ensure differential privacy, both pieces of source knowledge transferred to the target need to be perturbed carefully. Note that the private hypothesis is trained on labeled source samples and the private "importance weights" vector is trained on the disjoint unlabeled source samples. Therefore no composition or splitting of privacy budget is involved. The privacy analysis is presented in Theorem 4.3.1 below.

Theorem 4.3.1. *Algorithm 4 is* $(\epsilon, 0)$*-differentially private with respect to both the labeled and the unlabeled samples in each source.*

Proof. We sketch the proof here for brevity. First, the perturbed source hypotheses $\boldsymbol{\theta}^k_{priv}$'s are proved to be differentially private with respect to the labeled source samples by [31]. Additionally, the "importance weights" vector $\boldsymbol{w}^k$ is $(\epsilon, 0)$-differentially private with respect to the unlabeled source samples by [71]. Therefore, the final target hypothesis $\boldsymbol{\theta}^T$ built upon $\boldsymbol{\theta}^k_{priv}$'s and $\boldsymbol{w}^k$'s is also $(\epsilon, 0)$-differentially private with respect to the sources by the post-processing guarantee of differential privacy [42]. □

Algorithm 4 Differentially Private Hypothesis Transfer Learning (DPHTL)

Require: K private labeled source data sets $\boldsymbol{S}^k_l$ and unlabeled source data sets $\boldsymbol{S}^k_{ul}$, a public data set $\boldsymbol{P}$, a target data set $\boldsymbol{T}$ with limited labels, privacy parameter ϵ, regularization parameter for importance weighting mechanism λ_{IW}, regularization parameter for logistic regression model λ_{LR},

Ensure: A final target hypothesis $\boldsymbol{\theta}^T \in^d$

1: Each source uses its labeled samples $\boldsymbol{S}^k_l$ to train a differentially private logistic regression model $\boldsymbol{\theta}^k_{priv}$ under parameters λ_{LR} and ϵ. All the hypotheses $\boldsymbol{\theta}^k_{priv}$ are sent to the target.

2: Each source fetches the public data set $\boldsymbol{P}$, compute the differentially private "importance weights" vector $\boldsymbol{w}^k \leftarrow \mathbf{DPIW}(\boldsymbol{S}^k_{ul}, \boldsymbol{P}, \epsilon, \lambda_{IW})$ and sends $\boldsymbol{w}^k$ to the target.

3: The target fetches the public data set P and compute the non-private "importance weights" vector $\boldsymbol{w}^T \leftarrow \mathbf{DPIW}(\boldsymbol{T}, \boldsymbol{P}, \infty, \lambda_{IW})$.

4: The target calculates the "hypothesis weights" vector $\boldsymbol{w}_H \in^K$ such that the Kullback-Leibler (KL) divergence between $\boldsymbol{w}^T$ and the linear combination of $\boldsymbol{w}^k$ weighted by $\boldsymbol{w}_H$ is minimized:

$$\boldsymbol{w}_H = \underset{\boldsymbol{w} \in^K_{\geq 0}, \sum \boldsymbol{w}(k)=1}{\arg\min} \boldsymbol{KL}(\boldsymbol{w}^T, \sum_{k=1}^{K} \boldsymbol{w}(k)\boldsymbol{w}^k) \tag{4.1}$$

5: In order to construct an informative Gaussian prior using $\boldsymbol{w}_H$ and $\boldsymbol{\theta}^k_{priv}$ from the sources, the target first calculates the mean $\boldsymbol{\mu}^T(j)$ and standard deviation $\boldsymbol{\sigma}^T(j)$ of each parameter $\boldsymbol{\theta}^T(j)$ $(j = 1, 2, 3..., d)$ in the target logistic regression hypothesis:

$$\boldsymbol{\mu}^T(j) = \sum_{k=1}^{K} \boldsymbol{w}_H(k)\boldsymbol{\theta}^k_{priv}(j), \quad \sigma^T(j) = \sqrt{\frac{K}{K-1} \sum_{k=1}^{K} \boldsymbol{w}_H(k)(\boldsymbol{\theta}^k_{priv}(j) - \boldsymbol{\mu}^T(j))^2} \tag{4.2}$$

6: The target trains the Bayesian logistic regression model with the limited labeled target data set and the informative Gaussian prior following the method in [101] and return the posterior parameters $\boldsymbol{\theta}^T$.

Algorithm 5 Differentially Private Importance Weights (DPIW) [71]

Require: Private data set $\boldsymbol{X}$ of size N_X, public data set $\boldsymbol{P}$ of size N_P, privacy parameter ϵ, regularization parameter λ

Ensure: Differentially private importance weights vector $\boldsymbol{w} \in^{N_P}$.

1: Label each data point in $\boldsymbol{X}$ as 1 and each data point in $\boldsymbol{P}$ as 0.

2: Fit the regularized logistic regression model on the combination of $\boldsymbol{X}$ and $\boldsymbol{P}$ with the new binary labels:

$$\begin{aligned}\boldsymbol{\beta}^* = \arg\min_{\beta} & - \sum_{\boldsymbol{x} \in \boldsymbol{P}} \frac{\log(p(\boldsymbol{x} \in \boldsymbol{P} | \boldsymbol{x} \in \boldsymbol{X} \cup \boldsymbol{P}))}{N_P} \\ & - \sum_{\boldsymbol{x} \in \boldsymbol{X}} \frac{\log(p(\boldsymbol{x} \in \boldsymbol{X} | \boldsymbol{x} \in \boldsymbol{X} \cup \boldsymbol{P}))}{N_X} + \frac{\lambda}{2} \left\| \boldsymbol{\beta} \right\|^2 ,\end{aligned} \tag{4.3}$$

where $p(x \in \boldsymbol{X} | \boldsymbol{x} \in \boldsymbol{X} \cup \boldsymbol{P}) = 1 - p(\boldsymbol{x} \in \boldsymbol{P} | \boldsymbol{x} \in \boldsymbol{X} \cup \boldsymbol{P}) = 1/(1 + \exp(-\boldsymbol{\beta}^\intercal \boldsymbol{x}))$.

3: Add Laplace noise to $\boldsymbol{\beta}^*$ to get the differentially private $\boldsymbol{\beta}_{priv}$: $\boldsymbol{\beta}_{priv} = \boldsymbol{\beta}^* + \boldsymbol{\delta}$, where $Pr(\boldsymbol{\delta}) \sim \exp(-\epsilon ||\boldsymbol{\delta}||_2 N_X \lambda / \sqrt{d})$.

4: Output differentially private importance weight $\boldsymbol{w}(\boldsymbol{x}) = \exp(\boldsymbol{\beta}_{priv}^\intercal \boldsymbol{x}) N_P / Z$ for each $\boldsymbol{x}$ in $\boldsymbol{P}$, where $Z = \sum_{x \in \boldsymbol{P}} \exp(\boldsymbol{\beta}_{priv}^\intercal \boldsymbol{x})$.

4.4 Experiments

In this section we empirically evaluate the effectiveness of our differentially private hypothesis transfer learning method (**DPHTL**) using publicly available real-world data sets. The experiment results show that the hypothesis obtained by our method provides a performance boost over the hypothesis trained on the limited labeled target data while guaranteeing differential privacy of sources and is also better than the hypotheses obtained by several baselines under various level of privacy requirements.

The first data set is the text classification data set 20NewsGroup (**20NG**)[1] [83], a popular benchmark data set for transfer learning and domain adaptation [128, 40, 121, 98]. It is a collection of approximately 20,000 newsgroup documents with stop words removed and partitioned evenly across 20 different topics. Some topics can be further grouped into a broader subject, for example, computer-related, recreation-related. To construct a binary classification problem, we randomly paired each of the 5 computer-related topics with 2 non-computer-related topics in each source domain. Our purpose is to build a logistic regression model to classify each document as either computer-related or

[1] http://scikit-learn.org/stable/datasets/twenty_newsgroups.html

non-computer-related. Specifically, there are 5 source domains with documents sampled from different topic groups as listed in Table 4.1. The target data set and the public data set are assumed to be mixtures of the source domains.

TABLE 4.1: Topics of documents in each source domain for **20NG**

Domain	**Topic group**
Source 1	comp.graphics rec.autos sci.space
Source 2	comp.os.ms-windows.misc talk.politics.guns sci.med
Source 3	comp.sys.ibm.pc.hardware soc.religion.christian talk.politics.mideast
Source 4	comp.sys.mac.hardware misc.forsale rec.sport.baseball
Source 5	comp.windows.x talk.religion.misc sci.crypt

The second data set is the Amazon review sentiment data (**AMAZON**)[2], a famous benchmark multi-domain sentiment data set collected by Blitzer et al [16]. It contains product reviews from 4 different types – *Book*, *DVD*, *Kitchen* and *Electronics*. Reviews with star rating > 3 are labeled positive, and those with star rating < 3 are labeled negative. Each source domain contains reviews from one type and the target domain and public data set are mixtures of the source domains. After initial preprocessing, each product review is represented by its count of unigrams and bigrams in the document and the amounts of positive reviews and negative reviews are balanced. Note that among the four product types, *DVD* and *Book* are similar domains sharing many common sentiment words, whereas *Kitchen* and *Electronics* are more correlated with each other because most of kitchen appliances belong to electronics [171].

[2] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

4.4.1 Data Preprocessing

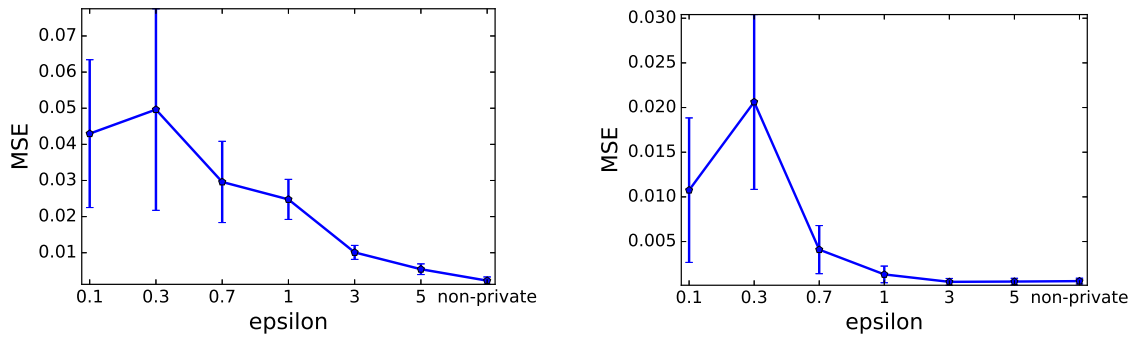
For the **20NG** data set, we removed the "headers", "footers" and "quotes". When building the vocabulary for each source domain, we ignored terms that have a document frequency strictly higher than 0.5 or strictly lower than 0.005. The total number of terms in the combination of source vocabularies is 5,588. To simulate the transfer learning setup, we sampled 1,000 documents from each source domain as our unlabeled source data set and another 800 documents with labels from each source domain as the labeled source data set. The public data set is an even mixture of the 5 sources with 1,500 documents in total. The target data set is also a mixture of all the source domains with insufficient labels. The final logistic regression model will be tested on a hold-out data set of size 320 sampled from the target domain. For the **AMAZON** data set, we selected the 1,500 most-frequently used unigrams/bigrams from each source domain and combined them to construct the feature set used in the logistic regression model. The total number of features is 2,969. We sampled 1,000 unlabeled reviews from each source domain as the unlabeled source data sets and another 1,000 labeled reviews as the labeled source data sets. The public data set is an even mixture of the 4 product types with 2,000 reviews in total. The counts of features were also transformed to a normalized tf-idf representation. The size of the target data set is 1,500 and the performance will be evaluated on a hold-out test set of size 600 sampled from the target domain. For both data sets, validation sets were reserved for the grid search of regularization parameters. We explored the performance of our method (**DPHTL**) across varying mixtures of sources, varying percentage of target labels and varying privacy requirements.

4.4.2 Differentially Private Importance Weighting

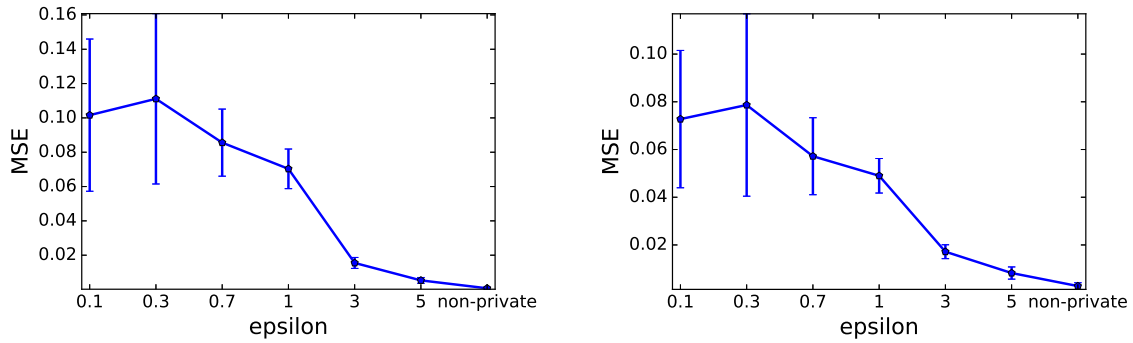
For the first set of experiments, we study the utility of the differentially private importance weighting mechanism (Step 2 - 4 in Algorithm 4) on both data sets. As shown by the results, our method can indeed reveal the proportion of each source domain in the target domain even under stringent privacy requirements. We set the regularization parameter

λ_{IW} to be 0.001 after grid search among several candidates and plot the mean squared error (MSE) between the original proportion vector and the "hypothesis weights" vector determined by our method across varying privacy requirements ϵ. The experiments were repeated 20 times with different samples and the the error bars in the figures represent the standard errors.

The results (see Fig. 4.2 and Fig. 4.3) clearly show that the "hypothesis weights" vector is a good privacy-preserving estimation of the proportion of source domains in the target domain for both **20NG** and **AMAZON**. As the privacy requirement is relaxed (ϵ increases), the MSE will approach zero, reflecting the fact that the weight assigned to each source hypothesis is directly affected by the relationship between the source and the target. It is crucial in avoiding negative transfer as brute-force transfer may actually hurt performance if the sources are too dissimilar to the target [134].

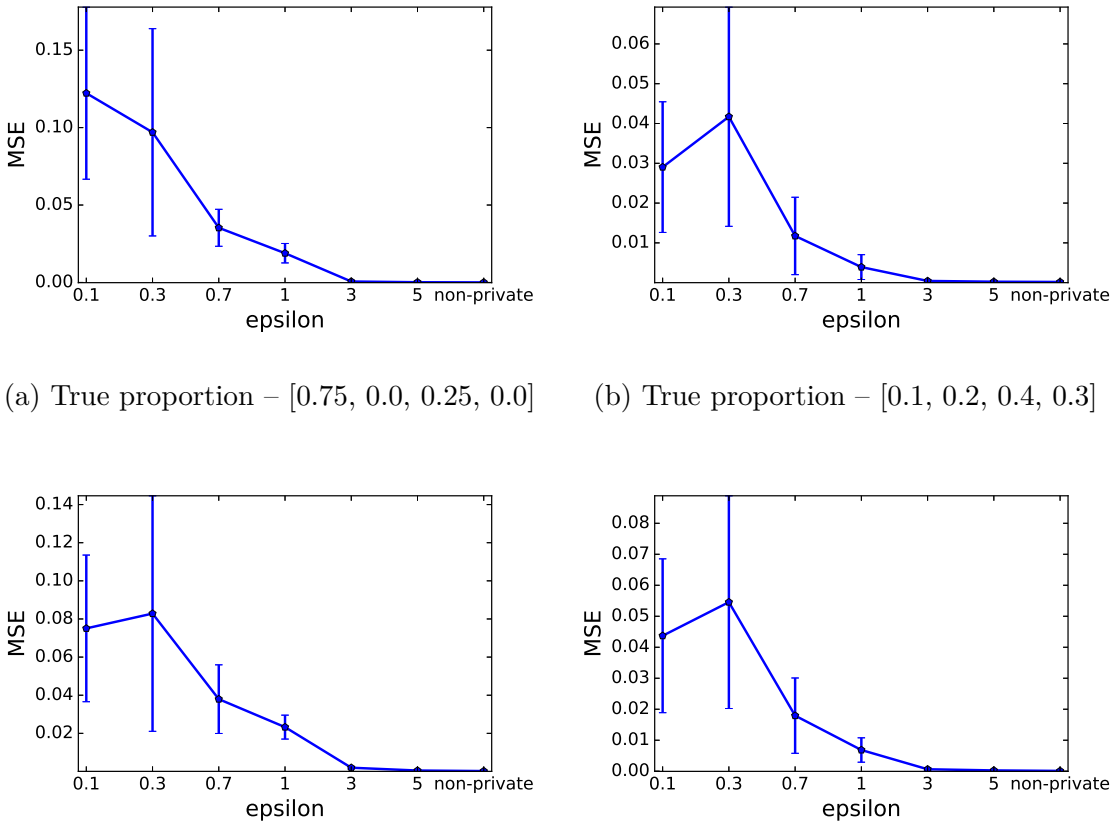


(a) True proportion – [0.2, 0.0, 0.4, 0.0, 0.4] (b) True proportion – [0.2, 0.2, 0.2, 0.2, 0.2]



(c) True proportion – [0.2, 0.0, 0.0, 0.8, 0.0] (d) True proportion – [0.0, 0.0, 0.6, 0.0, 0.4]

FIGURE 4.2: MSE between differentially private hypothesis weight vector and true proportion (**20NG**)



(a) True proportion – [0.75, 0.0, 0.25, 0.0]

(b) True proportion – [0.1, 0.2, 0.4, 0.3]

(c) True proportion – [0.0, 0.5, 0.5, 0.0]

(d) True proportion – [0.0, 0.2, 0.4, 0.4]

FIGURE 4.3: MSE between differentially private hypothesis weight vector and true proportion (**AMAZON**)

4.4.3 Differentially Private Hypothesis Transfer Learning

For the second set of experiments, we investigate the privacy-utility trade-off of our **DPHTL** method and compare its performance with those of several baselines. The first baseline **SOURCE** is the best performer on the target test set among all the differentially private source hypotheses. The second baseline **AVERAGE** is the posterior Bayesian logistic regression model which assigns equal weights to all the source hypotheses in constructing the Gaussian prior. The third baseline is referred to as **SOFT**, which represents building the logistic regression model using soft labels. It is an adaptation of the work proposed by Hamm et al. in [62] under our transfer learning setting. More specifically, all the unlabeled samples in the target will be soft-labeled by the fraction $\alpha(\boldsymbol{x})$ of positive votes from all the differentially private source hypotheses (see Algorithm 2 in [62]). For

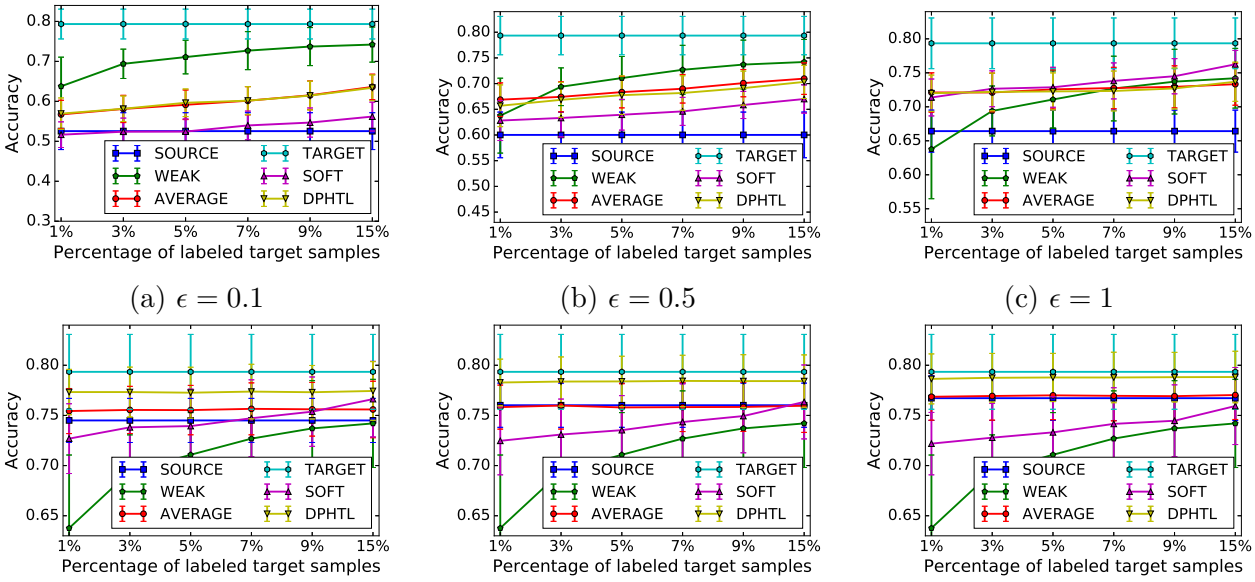
a certain target sample $\boldsymbol{x}$,

$$\alpha(\boldsymbol{x}) = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\theta}_{priv}^{k}(\boldsymbol{x}) \tag{4.4}$$

The minimizer of the loss function of the regularized logistic regression with labeled target samples and soft-labeled target samples will be output as the final hypothesis for the **SOFT** baseline. Similar to our method, Hamm et al. also explored the idea of leveraging auxiliary unlabeled data and proposed a hypothesis ensemble approach. However, there are two key discrepancies. Firstly, no trusted third party is required in our setting so the local hypotheses are perturbed before being transferred to the target. Secondly, the algorithms in [62] were not designed for transfer learning and by assumption the samples are drawn from a distribution common to all parties. Therefore, no mechanism is in place to prevent negative transfer.

In addition, we compared with the non-private hypotheses **WEAK** and **TARGET** trained on the target. The hypothesis **WEAK** is trained using the limited labeled samples in the target data set. The hypothesis **TARGET** is trained using the true labels of all the samples (both labeled and unlabeled) in the target data set and can be considered as the best possible performer since it has access to all the true labels and ignores privacy. The test accuracy of the obtained hypotheses is plotted as a function of the percentage of labeled samples in the target set. The experiments were repeated 20 times and the error bars are standard errors. In order to save space, we illustrate the utility-privacy trade-off by figures for one mixture only.

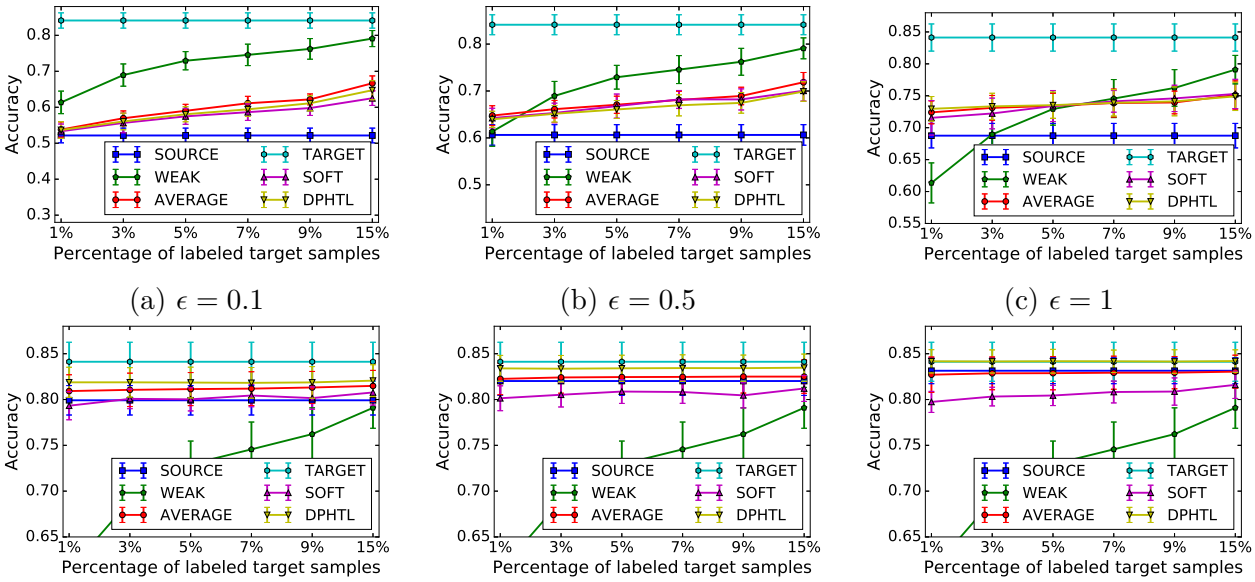
We set the logistic regression regularization parameter at $\lambda_{LR} = 0.003$ for **20NG** and $\lambda_{LR} = 0.005$ for **AMAZON** after preliminary grid search on the validation set. Fig. 4.4 and 4.5 show the test accuracy for all the methods across increasing privacy parameter ϵ. **DPHTL** and **AVERAGE** start to emerge as better performers at an intermediate privacy level for both **20NG** and **AMAZON**. When the privacy requirement is further relaxed, **DPHTL** becomes the best performer by a large margin at all range of labeled target sample percentage. Moreover, **DPHTL** is the only method performing as well as **TARGET** when privacy is not a concern. In comparison, the utilities of **AVERAGE**



(a) $\epsilon = 0.1$ (b) $\epsilon = 0.5$ (c) $\epsilon = 1$

(d) $\epsilon = 3$ (e) $\epsilon = 5$ (f) non-private

FIGURE 4.4: The test accuracy comparison of **DPHTL** and baselines as a function of percentage of labeled target samples for **20NG**. In the target set, 60% are sampled from source domain 1 and 40% are sampled from source domain 4 .



(a) $\epsilon = 0.1$ (b) $\epsilon = 0.5$ (c) $\epsilon = 1$

(d) $\epsilon = 3$ (e) $\epsilon = 5$ (f) non-private

FIGURE 4.5: The test accuracy comparison of **DPHTL** and baselines as a function of percentage of labeled target samples for **AMAZON**. In the target set, 25% are sampled from source domain 2 and 75% are sampled from source domain 4.

and **SOFT** are hindered by unrelated source domains. More test results are presented in Table 4.3 when the percentage of labeled target samples is fixed at 5%. In general, **DPHTL** gains a performance improvement over other baselines by taking advantage of the "hypotheses weights" while preserving the privacy of sources. Besides, it presents a promising solution to the notorious situation where very few labeled data is available at the target domain.

ϵ	WEAK	SOURCE	AVERAGE	SOFT	DPHTL	TARGET
Target mixture 1: $[0.25, 0.0, 0.75, 0.0]$						
0.1	**0.650(0.03)**	0.523(0.03)	0.563(0.03)	0.554(0.02)	0.555(0.03)	0.780(0.02)
0.5	**0.650(0.03)**	0.574(0.03)	0.623(0.02)	0.625(0.03)	0.614(0.03)	0.780(0.02)
1	0.650(0.03)	0.629(0.03)	**0.681(0.02)**	0.676(0.02)	0.670(0.02)	0.780(0.02)
3	0.650(0.03)	0.723(0.02)	0.751(0.02)	0.744(0.02)	**0.755(0.02)**	0.780(0.02)
5	0.650(0.03)	0.753(0.03)	0.767(0.02)	0.754(0.02)	**0.773(0.02)**	0.780(0.02)
∞	0.650(0.03)	0.770(0.02)	0.776(0.02)	0.764(0.02)	**0.788(0.02)**	0.780(0.02)
Target mixture 2: $[0.0, 0.5, 0.0, 0.5]$						
0.1	**0.718(0.03)**	0.521(0.03)	0.586(0.03)	0.549(0.02)	0.575(0.03)	0.834(0.02)
0.5	**0.718(0.03)**	0.598(0.02)	0.662(0.02)	0.646(0.03)	0.657(0.03)	0.834(0.02)
1	0.718(0.03)	0.672(0.02)	0.729(0.02)	0.726(0.02)	**0.734(0.01)**	0.834(0.02)
3	0.718(0.03)	0.786(0.02)	0.801(0.02)	0.786(0.02)	**0.814(0.02)**	0.834(0.02)
5	0.718(0.03)	0.808(0.02)	0.815(0.01)	0.792(0.02)	**0.829(0.02)**	0.834(0.02)
∞	0.718(0.03)	0.823(0.02)	0.818(0.01)	0.794(0.02)	**0.838(0.02)**	0.834(0.02)

TABLE 4.2: The test accuracy on **AMAZON** for different target mixtures when the percentage of labeled target samples is set at 5%. The best performer outside **TARGET** at each privacy level is highlighted.

ϵ	WEAK	SOURCE	AVERAGE	SOFT	DPHTL	TARGET
Target mixture 1: [0.0, 0.6, 0.2, 0.0, 0.2]						
0.1	**0.735(0.05)**	0.537(0.05)	0.639(0.03)	0.533(0.03)	0.636(0.03)	0.848(0.02)
0.5	0.735(0.05)	0.651(0.04)	**0.740(0.03)**	0.659(0.03)	0.732(0.04)	0.848(0.02)
1	0.735(0.05)	0.740(0.03)	0.777(0.03)	0.768(0.03)	**0.779(0.03)**	0.848(0.02)
3	0.735(0.05)	0.821(0.03)	0.813(0.02)	0.784(0.03)	**0.832(0.02)**	0.848(0.02)
5	0.735(0.05)	0.832(0.02)	0.819(0.02)	0.777(0.02)	**0.840(0.02)**	0.848(0.02)
∞	0.735(0.05)	0.840(0.02)	0.824(0.02)	0.771(0.03)	**0.853(0.02)**	0.848(0.02)
Target mixture 2: [0.5, 0.0, 0.0, 0.5, 0.0]						
0.1	**0.704(0.04)**	0.540(0.04)	0.588(0.04)	0.533(0.02)	0.585(0.04)	0.792(0.02)
0.5	**0.704(0.04)**	0.602(0.04)	0.671(0.03)	0.628(0.03)	0.664(0.04)	0.792(0.02)
1	0.704(0.04)	0.654(0.04)	0.714(0.02)	**0.725(0.03)**	0.716(0.03)	0.792(0.02)
3	0.704(0.04)	0.745(0.03)	0.747(0.02)	0.734(0.02)	**0.765(0.02)**	0.792(0.02)
5	0.704(0.04)	0.771(0.02)	0.751(0.02)	0.730(0.02)	**0.778(0.01)**	0.792(0.02)
∞	0.704(0.04)	0.781(0.02)	0.764(0.02)	0.728(0.02)	**0.788(0.02)**	0.792(0.02)
Target mixture 3: [0.2, 0.0, 0.3, 0.3, 0.2]						
0.1	**0.703(0.03)**	0.544(0.05)	0.609(0.04)	0.547(0.02)	0.608(0.05)	0.790(0.02)
0.5	**0.703(0.03)**	0.625(0.04)	0.703(0.02)	0.657(0.03)	0.694(0.03)	0.790(0.02)
1	0.703(0.03)	0.688(0.03)	0.741(0.02)	**0.752(0.01)**	0.742(0.02)	0.790(0.02)
3	0.703(0.03)	0.753(0.03)	0.773(0.02)	0.748(0.02)	**0.779(0.02)**	0.790(0.02)
5	0.703(0.03)	0.765(0.03)	0.775(0.02)	0.742(0.02)	**0.784(0.02)**	0.790(0.02)
∞	0.703(0.03)	0.779(0.02)	0.782(0.02)	0.738(0.02)	**0.791(0.02)**	0.790(0.02)

TABLE 4.3: The test accuracy on **20NG** for different target mixtures when the percentage of labeled target samples is set at 5%. The best performer outside **TARGET** at each privacy level is highlighted.

4.5 Chapter Conclusion

This paper proposes a multiple-source hypothesis transfer learning system that protects the differential privacy of sources. By leveraging the relatively abundant supply of unlabeled samples and an auxiliary public data set, we derive the relationship between sources and target in a privacy-preserving manner. Our hypothesis ensemble approach incorporates this relationship information to avoid negative transfer when constructing the Gaussian prior for the target logistic regression model. Moreover, our approach provides a promising and effective solution when the labeled target samples are scarce. Experimental results on benchmark data sets confirm our performance improvement over several baselines from recent work.

Chapter 5

PrivLogit: Efficient Multi-Party Privacy Preserving Logistic Regression via Tailored Newton's Method

Privacy preserving machine learning based on cryptography techniques is increasingly popular for safeguarding personal privacy in collaborative data analytics across multiple organizations. However, in practice existing secure solutions suffer from excessive computational overhead, partially due to the naive adoption of mainstream model fitting algorithms that are not tailored for secure computing. In this chapter, we propose an improved numerical optimizer based on Newton's method for secure logistic regression. The theoretical analysis and extensive empirical evaluations have further shown the competitive performance of our proposal.

5.1 Introduction

Logistic regression is a fundamental statistical model with a wide adoption in a variety of domains, such as healthcare study, medical informatics and quantitative psychology. To reach powerful and reliable statistical conclusions, it is becoming more prevalent to train a logistic regression model collaboratively across a federation of organizations [103, 55, 102,

161]. Such a trend, however, is often hampered by serious privacy concerns as human data subject underlying these studies are typically considered sensitive and strictly protected by various privacy laws and regulations. Meanwhile, many organizations are reluctant to reveal their data to external entities (due to concerns around privacy and business secrets), even though they would jointly benefit from the pooled data. In this chapter, we address this problem by proposing an efficient optimization algorithm to train a logistic regression model collaboratively in a privacy-preserving manner. Cryptography [76], or secure multi-party computation (SMC) in particular, proves promising for tackling this challenge (albeit posing increased computational overhead). In various domains such as machine learning, data mining and database management, numerous attempts have been made to support statistical modeling without disclosing raw and/or intermediate data. These proposals have been further enhanced in performance by leveraging distributed computing and decomposition of data and computations among various participating organizations [3, 161, 19, 165, 90], a generic scenario known as privacy-preserving data mining (PPDM) on horizontally or vertically partitioned data. With respect to logistic regression, there have been various secure solutions as well, such as Wolfson et al. [159], Gaye et al. [55], El Emam et al. [44], Wu et al. [162], Li et al. [90], Slavkovic, Nardi, and Tibbits [146], and Kim et al. [79].

Despite encouraging progress, very few proposals have seen wide adoption in real world applications for privacy-preserving logistic regression. A major concern seems to be the excessive computational overhead of cryptographic protocols, in addition to insufficient privacy protection from many existing proposals [44]. While it is generally expected for secure computation to be slower than its non-secure counterparts, we also make an observation that much of the computational overhead indeed traces back to the sub-optimal technical decisions made by humans experts (e.g., cryptographers) and could have been avoided. For instance, nearly all existing secure protocols [162, 90] naïvely apply mainstream (distributed) model estimation algorithms (e.g., Newton's method for logistic regression [157]), failing to account for secure computing-specific characteristics and thus missing valuable opportunities for performance improvement.

In this chapter, we present a contrasting perspective on privacy-preserving logistic regression, and propose an improved Newton's method tailored for secure computing which significantly accelerates the computation while guaranteeing privacy and model quality. In our proposal (which we call PrivLogit), we derive a constant lower bound to approximate the second-order curvature information (the Hessian matrix) in the Newton's method for logistic regression. At first glance, this adapted optimizer seems counter-intuitive and unfavorable due to its elongated convergence and increased network interactions during the iterative model estimation process, but eventually turns out to be highly competitive for secure distributed logistic regression in terms of model accuracy and computational performance.

Following our new PrivLogit optimizer, we implement two highly-efficient cryptographic protocols for privacy-preserving logistic regression, which we call PrivLogit-Hessian and PrivLogit-Local, respectively. To make our work generally applicable, we also support the widely-used ridge or ℓ_2-regularization for logistic regression. We validate the performance of our proposals with extensive empirical evaluations, on both simulated and real-world data sets. In summary, we make the following major contributions in this work:

- We propose an approximate Newton's method (PrivLogit) tailored for secure multi-party logistic regression and provide detailed theoretical analysis on our proposals.
- We implement two efficient secure protocols based on PrivLogit (PrivLogit-Hessian and PrivLogit-Local) for privacy-preserving logistic regression.
- We extensively evaluate our implementations on a variety of simulated and real-world studies of very large scale, many of which are significantly larger than previously reported in the literature.

This chapter is organized as follows: In Section 5.2 and 5.3, we describe our improved optimizer PrivLogit for training logistic regression, and two new secure implementation. This is then followed by detailed experimental results in Section 5.4. We discuss and conclude this work in Section 5.5.

5.2 PrivLogit for Logistic Regression

In this section, we introduce our adapted Newton's method tailored for secure logistic regression – PrivLogit. Most existing solutions for secure logistic regression directly apply the (distributed) Newton's method with appropriate cryptographic protections in place [159, 162, 90, 146, 79]. We argue that mainstream statistical model estimation methods designed for general privacy-free settings may not necessarily be competitive when directly translated into their privacy-preserving counterparts. Here we first describe distributed Newton's method used for logistic regression and state the limitations of traditional Newton's method; and then we proceed to introduce our new optimizer for privacy-preserving logistic regression.

5.2.1 Limitations of Distributed Newton's Method.

As described in Section 2.2.1 of Chapter 2, the log likelihood function in logistic regression is usually optimized with Newton's method. Under a distributed setting where S parties train a common model collaboratively, both the gradients and the Hessian are decomposable.

$$\boldsymbol{g}(\boldsymbol{\beta}) = \sum_{i=1}^{S} \boldsymbol{g}_i(\boldsymbol{\beta}) - \lambda\boldsymbol{\beta}, \tag{5.1}$$

$$\boldsymbol{H}(\boldsymbol{\beta}) = \sum_{i=1}^{S} \boldsymbol{H}_i(\boldsymbol{\beta}) - \lambda\boldsymbol{I}\ , \tag{5.2}$$

where $\boldsymbol{g}_i(\boldsymbol{\beta})$ and $\boldsymbol{H}_i(\boldsymbol{\beta})$ are evaluated on the local data sets from the i-th party. However, for an iterative method such as Newton's method, the evaluation and inversion of the Hessian matrix have to be repeatedly performed for every iteration until model convergence. It can be prohibitively expensive especially when the data set is huge and high dimensional [24]. This in fact serves as the main motivation for numerous improvement over Newton's method, which are usually referred to as Quasi-Newton or Hessian-free method in machine learning and optimization [34, 26, 48, 96].

In the data security and privacy domain, the issue of expensive Newton's method is exacerbated as secure matrix inversion requires complex operations (such as repeated

secure division and square root) which often have to resort to highly computation- and communication-intensive primitives and approximations from secure multi-party computation (SMC) [165, 44]. As a result, almost all existing secure logistic regression proposals based on (distributed) Newton's method have to trade strong privacy protection for better performance in some way (e.g., to selectively reveal intermediate data/computations [161, 90]), which otherwise are computationally impractical for even modest-size studies. In addition, the lack of model convergence guarantee in Newton's method is also a known issue, especially when poor initialization of regression parameters is provided [17]. Given the limitations of Newton's method, we are motivated to design a more efficient and robust optimizer to speed up the secure computation and obtain reliable model estimates.

5.2.2 PrivLogit for Privacy-Preserving Logistic Regression.

Per prior analysis, to overcome the computational bottleneck of the (distributed) Newton's method, we must avoid the evaluation or inversion of the Hessian matrix. A straightforward solution we come up with is to replace the Hessian matrix with some constant values across all iterations. However, this seems counter-intuitive and nullifies the primary advantage of Newton's method in incorporating curvature (i.e., second-order) information for fewer iterations and faster convergence. Also, the convergence guarantee and model quality of the new proposal may become obscure.

In this section, on the contrary, we prove that with careful design, our proposed Hessian approximation is indeed plausible and turns out to be highly efficient when coupled with secure computation. The main intuition underlying our new PrivLogit proposal is to substitute the varying Hessian $\boldsymbol{H}(\boldsymbol{\beta})$ at each iteration with one carefully chosen constant approximation $\tilde{\boldsymbol{H}}$, thus avoiding repeated evaluation and inversion of the original Hessian. Specifically, for ℓ_2-regularized logistic regression, we propose and prove that the global lower bound $\tilde{\boldsymbol{H}}$ of $\boldsymbol{H}(\boldsymbol{\beta})$ could serve as an appropriate Hessian approximation:

$$\tilde{\boldsymbol{H}} = -\frac{1}{4}\boldsymbol{X}^\intercal\boldsymbol{X} - \lambda\boldsymbol{I}. \tag{5.3}$$

Here $\tilde{\boldsymbol{H}}$ is a tight lower bound of $\boldsymbol{H}(\boldsymbol{\beta})$. Moreover it is negative-definite, which guarantees the existence of the maximum for this quadratic approximation to the logistic regression log-likelihood $l_2(\boldsymbol{\beta})$. Note that our new approximation $\tilde{\boldsymbol{H}}$ is decoupled from the diagonal matrix $\boldsymbol{A}$ (and thus independent of the regression parameters $\boldsymbol{\beta}_k$). The update in PrivLogit becomes

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k - \tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}(\boldsymbol{\beta}_k) \ , \tag{5.4}$$

We note that the calculation of approximate Hessian $\tilde{\boldsymbol{H}}$ can also be decomposed and computed in a distributed manner:

$$\tilde{\boldsymbol{H}} = -\frac{1}{4}\sum_{i=1}^{S} \boldsymbol{X}_i^{\intercal}\boldsymbol{X}_i - \lambda \boldsymbol{I} = \sum_{i=1}^{S} \tilde{\boldsymbol{H}}_i - \lambda \boldsymbol{I}, \tag{5.5}$$

where $\boldsymbol{X}_i$ is the (privacy-sensitive) raw data stored locally at the i-th party. Substituting this approximate Hessian into Newton's method, along with the distributed evaluation of gradient, the iterative updating formula for PrivLogit now becomes:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k - \left(\sum_{i=1}^{S} \tilde{\boldsymbol{H}}_i - \lambda \boldsymbol{I}\right)^{-1} \left(\sum_{i=1}^{S} \boldsymbol{g}_i(\boldsymbol{\beta}_k) - \lambda \boldsymbol{\beta}_k\right). \tag{5.6}$$

5.2.3 Theoretical Properties of PrivLogit

Here we present theoretical analysis of PrivLogit regarding its convergence property, which is based on the quadratic function approximation work of Böhning and Lindsay [18]. We show that our PrivLogit optimizer is guaranteed to converge to the optimum at a linear convergence rate. Specifically, we prove the following proposition:

Proposition 1. *Assume the optimal solution $\boldsymbol{\beta}^*$ to the objective function $l_2(\boldsymbol{\beta})$ (Equation 2.10) of the ℓ_2-regularized logistic regression exists and is unique. Let $\{\boldsymbol{\beta}_k\}$ be a sequence generated by PrivLogit with the update formula in Equation 5.4. The sequence has the following properties:*

(a) $l_2(\boldsymbol{\beta}_{k+1}) > l_2(\boldsymbol{\beta}_k)$ and $\boldsymbol{\beta}_k$ will converge to the optimal solution $\boldsymbol{\beta}^*$.

(b) The rate of convergence of PrivLogit is linear.

Proof. (a) By the negative definiteness of $\tilde{\boldsymbol{H}}$ and the second-order Taylor expansion of $l_2(\boldsymbol{\beta})$, we have,

$$\begin{aligned} l_2(\boldsymbol{\beta}_{k+1}) - l_2(\boldsymbol{\beta}_k) &= -\boldsymbol{g}(\boldsymbol{\beta}_k)^\intercal \tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}(\boldsymbol{\beta}_k) + \frac{1}{2}\boldsymbol{g}(\boldsymbol{\beta}_k)^\intercal \tilde{\boldsymbol{H}}^{-1}\boldsymbol{H}(\hat{\boldsymbol{\beta}})\tilde{\boldsymbol{H}}^{-1} g(\boldsymbol{\beta}_k) \\ &> -\boldsymbol{g}(\boldsymbol{\beta}_k)^\intercal \tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}(\boldsymbol{\beta}_k) + \frac{1}{2}\boldsymbol{g}(\boldsymbol{\beta}_k)^\intercal \tilde{\boldsymbol{H}}^{-1}\tilde{\boldsymbol{H}}\tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}(\boldsymbol{\beta}_k) \\ &= -\frac{1}{2}\boldsymbol{g}(\boldsymbol{\beta}_k)^\intercal \tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}(\boldsymbol{\beta}_k) \\ &> 0 \end{aligned} \tag{5.7}$$

where $\hat{\boldsymbol{\beta}}$ is between $\boldsymbol{\beta}_k$ and $\boldsymbol{\beta}_{k+1}$.

The objective function $l_2(\boldsymbol{\beta})$ is strictly concave with a negative definite Hessian matrix and therefore is maximized at the optimal solution $\boldsymbol{\beta}^*$. From the previous derivation, we obtain the lower bound of the increment of the objective function at each iteration. If $\boldsymbol{g}(\boldsymbol{\beta}_k)$ is bounded away from 0 for all the iteration, in other words, $||\boldsymbol{g}(\boldsymbol{\beta}_k)|| > \epsilon$ for some positive constant ϵ, then the increment of each iteration is also bounded above 0, which contradicts the upper boundedness of the objective function. Therefore, $\boldsymbol{g}(\boldsymbol{\beta})_k \to 0$ as $k \to \infty$, which means the sequence $\{\boldsymbol{\beta}_k\}$ converges to the optimal solution $\boldsymbol{\beta}^*$.

(b) Since $\boldsymbol{X}^\intercal\boldsymbol{X}$ is positive semi-definite, its eigenvalues are all non-negative. Let the biggest eigenvalue of $\boldsymbol{X}^\intercal\boldsymbol{X}$ be denoted as λ_{max}. Furthermore, we also assume $\boldsymbol{X}^\intercal\boldsymbol{A}\boldsymbol{X}$ is positive definite at every iteration, with the smallest eigenvalue $\lambda_{min} > 0$. Then we have

$$-\frac{1}{\frac{1}{4}\lambda_{max} + \lambda}\boldsymbol{I} \succeq \tilde{\boldsymbol{H}}^{-1} \tag{5.8}$$

and

$$-(\lambda_{min} + \lambda)\mathbf{I} \succeq \boldsymbol{H}(\boldsymbol{\beta}) \succeq -(\frac{1}{4}\lambda_{max} + \lambda)\boldsymbol{I} \tag{5.9}$$

Let $M = \frac{1}{4}\lambda_{max} + \lambda$ and $m = \lambda_{min} + \lambda$. By the strong concavity assumption and the second-order Taylor expansion of l_2, we have for any $\boldsymbol{\upsilon}$ and $\boldsymbol{\omega}$ in the regression parameter

space,

$$\begin{aligned} l_2(\boldsymbol{\omega}) &< l_2(\boldsymbol{\upsilon}) + \boldsymbol{g}(\boldsymbol{\upsilon})^\intercal(\boldsymbol{\omega} - \boldsymbol{\upsilon}) - \frac{1}{2}m||\boldsymbol{\omega} - \boldsymbol{\upsilon}||_2^2 \\ &< l_2(\boldsymbol{\upsilon}) + \frac{||\boldsymbol{g}(\boldsymbol{\upsilon})||_2^2}{2m} \end{aligned} \tag{5.10}$$

Since the inequality holds everywhere in the regression parameter space, we have $||\boldsymbol{g}(\boldsymbol{\upsilon})||_2^2 > 2m(l_2(\boldsymbol{\beta}^*) - l_2(\boldsymbol{\upsilon}))$ for any $\boldsymbol{\upsilon}$. Next we need to investigate the relation between $l_2(\boldsymbol{\beta}^*) - l_2(\boldsymbol{\beta}_{k+1})$ and $l_2(\boldsymbol{\beta}^*) - l_2(\boldsymbol{\beta}_k)$ for all k. From part (a), we have

$$\begin{aligned} l_2(\boldsymbol{\beta}_{k+1}) &> l_2(\boldsymbol{\beta}_k) - \frac{1}{2}\boldsymbol{g}(\boldsymbol{\beta}_k)^\intercal \tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}(\boldsymbol{\beta}_k) \\ &> l_2(\boldsymbol{\beta}_k) + \frac{1}{2M}||\boldsymbol{g}(\boldsymbol{\beta}_k)||_2^2 \end{aligned} \tag{5.11}$$

Subtracting both sides from $l_2(\boldsymbol{\beta}^*)$, we get

$$\begin{aligned} l_2(\boldsymbol{\beta}^*) - l_2(\boldsymbol{\beta}_{k+1}) &< l_2(\boldsymbol{\beta}^*) - l_2(\boldsymbol{\beta}_k) - \frac{1}{2M}||\boldsymbol{g}(\boldsymbol{\beta}_k)||_2^2 \\ &< (1 - \frac{m}{M})(l_2(\boldsymbol{\beta}^*) - l_2(\boldsymbol{\beta}_k)) \\ &< (1 - \frac{m}{M})^k(l_2(\boldsymbol{\beta}^*) - l_2(\boldsymbol{\beta}_1)), \end{aligned} \tag{5.12}$$

where the factor $1 - \frac{m}{M} < 1$. It shows that $l_2(\boldsymbol{\beta}_k)$ converges in a linear rate to $l_2(\boldsymbol{\beta}^*)$ as $k \to \infty$. □

5.2.4 Advantages of PrivLogit

PrivLogit has a few attractive properties, which seem highly promising for efficient secure logistic regression. First of all, the constant Hessian approximation adaption comes at the cost of elongated convergence compared to Newton's method. However, in secure implementations, the local computation at each participating party in the collaborative study is essentially "free" because each party has full control of its private data and standard non-secure computation can be directly performed without any privacy concerns; but the secure computation at the aggregation center is usually orders of magnitudes slower than its non-secure counterparts (due to expensive cryptographic operations). This

implies that reducing the computational cost of secure computation in the aggregation center (the current bottleneck in Newton's method) can potentially lead to significant speedup.

Secondly, the proposed Hessian approximation $\tilde{\boldsymbol{H}}$ (or $\tilde{\boldsymbol{H}}_i$) stays constant and independent of the varying regression parameters $\boldsymbol{\beta}_k$. This indicates that it only needs to be evaluated and inverted once in the beginning and can be reused across all iterations, leading to a dramatic reduction in computation time compared with repeated expensive evaluation and inversion of the varying Hessian $\boldsymbol{H}(\boldsymbol{\beta}_k)$ in Newton's method.

Thirdly, the new adaption allows for easy decomposition of the computation among participating organizations, which can be leveraged to enable secure outsourcing to distributed computing nodes for improved efficiency. The computation decomposition of gradient and Hessian aforementioned (Equation 5.6) implies that, once the inverted Hessian is obtained (i.e., $\tilde{\mathbf{H}}^{-1}$) and properly protected (e.g., via strong encryption functions, which we symbolically denote as $Enc(.)$), the computation of the PrivLogit update direction can be outsourced to the local participating party to avoid expensive secure matrix multiplication at the center. Following this strategy, each local party derives its respective local summary-level information (denoted ss_i) in encrypted form as:

$$Enc(ss_i) = Enc(\tilde{\boldsymbol{H}}^{-1}\ g_i(\boldsymbol{\beta}_k)) = Enc(\tilde{\boldsymbol{H}}^{-1}) \otimes Enc(g_i(\boldsymbol{\beta}_k))\ , \tag{5.13}$$

where $\otimes$ denotes the secure multiplication operation which yields the encrypted product using only encryption of multipliers. Later, the aggregation center only needs to securely aggregate these (encrypted) local summaries, which is trivial to perform even in secure implementation. This further accelerates the whole system by avoiding the repeated expensive secure matrix multiplication between the inverted Hessian and aggregated gradient.

Lastly, PrivLogit will generate a sequence of parameter estimates which monotonically increase the value of the objective function, leading to guaranteed convergence to the optimal solution no matter what initialization values the algorithm starts with. In contrast,

this desirable property is lacking in Newton's method, which could fail to converge given poor initialization values (as evidenced by our empirical evaluation and [17]). Moreover, unlike the gradient-based method, there is no need for the complicated and expensive line searching for best step size.

5.3 Cryptographic Implementations of PrivLogit

In this section, we propose two secure implementations of PrivLogit for logistic regression. The first, which we call PrivLogit-Hessian, is a straightforward cryptographic implementation of PrivLogit. The second, which we call PrivLogit-Local, leverages outsourced computation to take advantage of the "free" local computation for significant speedup.

We note that various secure schemes and primitives can be leveraged to safeguard the data and computation in PrivLogit. Options include various SMC techniques such as Yao's garbled circuit [168], additive homomorphic encryption [119], fully homomorphic encryption (FHE) [56], secret sharing [141] and some more efficient implementations and enhancements to existing primitives [39, 95]. For demonstration purpose, we base our implementations on two widely used secure primitives – Yao's garbled circuit [168] and the Paillier cryptosystem [119]. In fact, most state-of-the-art protocols for privacy-preserving machine learning also rely on (the hybrid of) these two primitives [117, 165, 90].

Both our secure implementations adopt a hybrid architecture of distributed Nodes (local parties) and an aggregation Center, as illustrated in Figure 5.1. In brief, participating parties (i.e., Nodes) are responsible for protecting their respective data and only generating (secure) summary-level data (e.g., gradients and Hessian), which would be securely consumed by the Center for model estimation. In a strongly protected system such as ours, all data and computations at the Center are encrypted and not visible even to the Center itself. The role of Center is typically played by two or more mutually independent semi-trusted authorities (denoted as different Servers in Figure 5.1), as is common for SMC applications [161, 118, 165, 90]. As long as there is no major collusion between the authorities, the security of the system is guaranteed. For practical deployment in areas

such as biomedical and social sciences, the role of Center could be assumed by the coordinating center (of a consortium, federation or association) in addition to a third-party authority (e.g., audit organizations or even a respectful member organization).

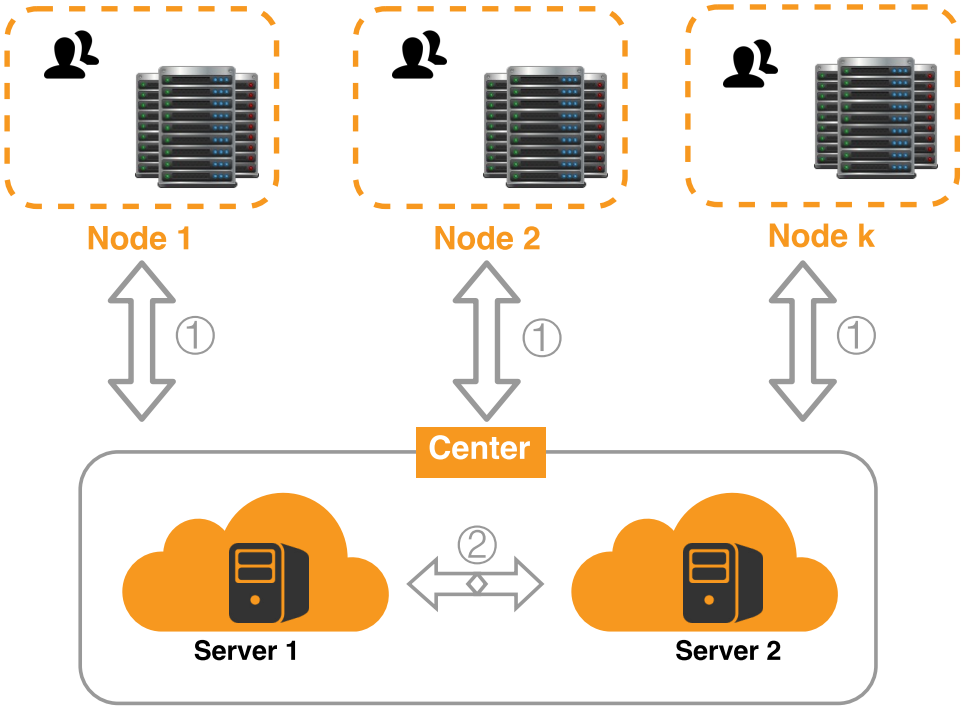


FIGURE 5.1: Architecture of PrivLogit-Hessian and PrivLogit-Local for privacy-preserving logistic regression. The distributed Nodes denote local parties possessing their respective private data in the collaborative study; The aggregation Center consists of independent authorities to securely aggregate and perform model updating, whose responsibilities are often split between several independent authorities backed by SMC. Two main types of computations are involved between: 1) local Nodes and the Center; 2) different servers/authorities at the Center.

For easier reference, we denote a few common secure mathematical arithmetic (leveraging Yao's garbled circuit or Paillier encryption) using intuitive symbols. Each of these operations take encrypted operands as inputs, and securely compute (without decryption) to output an encrypted result. Encrypted data are represented as $Enc(.)$, and we denote secure addition, subtraction, multiplication, division and square root as: $\oplus, \ominus, \otimes, \oslash, E_{sqrt}(.)$, respectively.

5.3.1 PrivLogit-Hessian

PrivLogit-Hessian is our straightforward secure and distributed implementation of the PrivLogit optimizer, as presented in Algorithm 6. This secure protocol consists of two phases of computation: a one-time setup phase of securely aggregating and inverting the approximate Hessian, and a repeated (iterative) secure parameter update phase.

Algorithm 6 PrivLogit-Hessian

Require: Random initial $\boldsymbol{\beta}_0$, regularization parameter λ
Ensure: Optimal regression parameters estimate $\boldsymbol{\beta}^*$
[Setup phase] :
1: Securely approximate and Cholesky-decompose the negated approximate Hessian: $Enc(\boldsymbol{L}) = SetupOnce()$ (where $Enc(\boldsymbol{L}\boldsymbol{L}^\intercal) = Enc(-\tilde{\boldsymbol{H}})$)
[Iterative phase] :
2: **while** regression model not converged **do**
[Local Nodes] :
3: **for** each participating parties $i = 1$ **to** S **do**
4: Compute local gradient $\boldsymbol{g}_i(\boldsymbol{\beta}_k)$ and encrypt
5: Compute local log-likelihood $l_i(\boldsymbol{\beta}_k)$ and encrypt
6: Securely transmit encryptions $Enc(\boldsymbol{g}_i(\boldsymbol{\beta}_k)), Enc(l_i)$ to Center
7: **end for**
[Center] :
8: Securely aggregate gradients from participating parties: $Enc(\boldsymbol{g}) = Enc(\boldsymbol{g}_1) \oplus ... \oplus Enc(\boldsymbol{g}_S) \ominus Enc(\lambda\boldsymbol{\beta}_k)$
9: Secure back-substitution: $Enc(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}) \leftarrow Enc(\boldsymbol{L})$, $Enc(\boldsymbol{g})$
10: Securely update coefficient estimates via PrivLogit: $\boldsymbol{\beta}_{k+1} \leftarrow \boldsymbol{\beta}_k$ (Equation 5.6)
11: Securely aggregate log-likelihood across organizations: $Enc(l) = Enc(l_1) \oplus ... \oplus Enc(l_S) \ominus Enc(\lambda\boldsymbol{\beta}_k^\intercal\boldsymbol{\beta}_k/2)$
12: Securely check model convergence
13: **end while**
14: **return** $\boldsymbol{\beta}$ from the last iteration.

The first phase (Step 1 in Algorithm 6 or the *SetupOnce()* function in Algorithm 7) focuses on securely computing and inverting the approximate Hessian $\tilde{\boldsymbol{H}}$. Specifically, each participating party compute their local Hessian approximation $\tilde{\boldsymbol{H}}_i$ based on its data set $\boldsymbol{X}_i$ and encrypt it before sharing with the Center. The Center securely aggregates these encrypted local Hessian approximation (and the regularization term as necessary), yielding an encrypted global Hessian approximation $Enc(\tilde{\mathbf{H}})$. Later, the Center performs secure Cholesky decomposition on the encrypted Hessian approximation and obtain its encrypted inversion $Enc(\boldsymbol{L})$ such that $Enc(\boldsymbol{L}\boldsymbol{L}^\intercal) = Enc(-\tilde{\boldsymbol{H}})$ [118]. Note that the

Algorithm 7 *SetupOnce()* for securely approximating and inverting Hessian.

Require: Local participating parties with their respective data
Ensure: Encrypted triangular matrix $Enc(\boldsymbol{L})$ from Cholesky decomposition (where $Enc(\boldsymbol{L}\boldsymbol{L}^\intercal) = Enc(-\tilde{\boldsymbol{H}})$)
[Local Nodes] :
1: **for** each organization $i = 1$ **to** S **do**
2: Compute local approximate Hessian $\tilde{\boldsymbol{H}}_i$
3: Encrypt and securely transmit $Enc(\tilde{\boldsymbol{H}}_i)$ to Center
4: **end for**
[Center] :
5: Securely aggregate Hessians across organizations: $Enc(\tilde{\boldsymbol{H}}) = Enc(\tilde{\boldsymbol{H}}_1) \oplus ... \oplus Enc(\tilde{\boldsymbol{H}}_S) \ominus Enc(\lambda \boldsymbol{I})$
6: Secure Cholesky decomposition to obtain $Enc(\mathbf{L})$
7: **return** $Enc(\mathbf{L})$

whole computation only happens once, which is a significant improvement over Newton's method-based protocols.

The second phase (Steps 2 to 14 in Algorithm 6) of PrivLogit-Hessian resembles that of the privacy-preserving distributed Newton's method, except for the elimination of repeated Hessian evaluation and inversion. Model convergence is checked at each iteration, as measured by the relative change of the regularized log-likelihood and compared against a predefined threshold (e.g., 10^{-6}) [84]. For each iteration, local parties only need to compute their local gradients $\boldsymbol{g}_i$ and log-likelihood l_i, and securely transmit their encrypted values to the Center. The Center securely aggregates the gradients and log-likelihood submissions, and compose the encrypted global gradient and log-likelihood. The Center then updates current regression parameters estimation following the PrivLogit updating formula (Equation 5.6). This iterative process continues until convergence.

5.3.2 PrivLogit-Local

Our second secure implementation is presented in Algorithm 8, which we refer to as PrivLogit-Local. This secure protocol takes advantage of the fact that global $\tilde{\boldsymbol{H}}^{-1}$ (or $Enc(\tilde{\boldsymbol{H}}^{-1})$) can be regarded as a (private) constant value. This means that we can further distribute the secure computation of the expensive matrix-vector multiplication $Enc(\tilde{\boldsymbol{H}}^{-1}) \otimes Enc(\boldsymbol{g}) = Enc(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{g})$ to the local parties.

Algorithm 8 PrivLogit-Local

Require: Random initial $\boldsymbol{\beta}_0$, regularization parameter λ
Ensure: Optimal regression parameters estimate $\boldsymbol{\beta}^*$

[Setup phase] :

Securely approximate and Cholesky-decompose the negated Hessian approximation: $Enc(\boldsymbol{L}) = SetupOnce()$ $(Enc(\boldsymbol{L}\boldsymbol{L}^\intercal) = Enc(-\tilde{\boldsymbol{H}}))$

[Iterative phase] :

while regression model not converged **do**

[Local Nodes] :

for each organization $i = 1$ **to** S **do**

Compute local gradient $\boldsymbol{g}_i(\boldsymbol{\beta}_k)$ and encrypt

Compute local log-likelihood $l_i(\boldsymbol{\beta}_k)$ and encrypt

Secure multiplication: $Enc(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}_i) \leftarrow Enc(\tilde{\boldsymbol{H}}^{-1}),\ g_i$;

Securely send encrypted values $Enc(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}_i),\ Enc(l_i)$ to Center

end for

[Center] :

Securely compose global numerical updating step: $Enc(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}) = Enc(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}_1) \oplus ... \oplus Enc(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}_S) \ominus Enc(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{\beta}_k)$

Securely update coefficient estimates via PrivLogit: $\boldsymbol{\beta}^{(t+1)} \leftarrow \boldsymbol{\beta}_k$ (Equation 5.6)

Securely aggregate log-likelihood across organizations: $Enc(l) = Enc(l_1) \oplus ... \oplus Enc(l_S) \ominus Enc(\lambda\boldsymbol{\beta}_k^\intercal\boldsymbol{\beta}_k/2)$

Securely check model convergence

end while

Return $\boldsymbol{\beta}$ from the last iteration

In greater details, the first step of PrivLogit-Local still involves the local Noeds and Center securely approximating and inverting the Hessian, similar to Phase 1 of PrivLogit-Hessian. Next, we directly materialize the inversion of approximate Hessian in encrypted form, i.e., $Enc(\tilde{\boldsymbol{H}}^{-1})$. After that, this encrypted inversion is disseminated to each local Nodes where local computation of gradients only involves privacy-free operations.

Later on, at each iteration, local organizations derive their local summaries, (log-likelihood and gradients). Then they compute their respective versions of (partial) Newton's method updating step, by using efficient SMC primitives. Since the local gradients $\boldsymbol{g}_i$ do not involve privacy concerns at their respective local organizations (thus can be regarded as a public constant value), the computation is greatly simplified to highly efficient secure multiplication-by-constant primitives. Afterwards, local organizations send their encrypted summaries $Enc(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{g}_i), Enc(l_i)$ back to the Center. For regularized logistic regression, the regularization term also needs to be securely composed, which can be

performed either by any local Nodes or the Center. Finally, the Center only needs to perform trivial secure aggregation to complete the Newton updating process and convergence check.

Building on top of PrivLogit-Hessian, our second protocol PrivLogit-Local further avoids expensive secure matrix multiplication at the center, which leads to significantly simplified computation and less overhead than PrivLogit-Hessian and baseline Newton's method. Due to simplicity of computation equation, this also makes PrivLogit-Local more widely amenable to a variety of cryptographic schemes.

5.4 Experiments

We implement our two PrivLogit proposals in the Julia programming language [13]. Secure and distributed computation are implemented in Java and Scala, using ObliVM-GC [95] and SecureMA [165]. We use common privacy-preserving floating-point representations [118]. We use 2048-bit security parameter for encryption and other latest security parameters in ObliVM-GC. Secure Newton's method for logistic regression has been explored by different communities and with different relaxations, but most were proposed prior to the ObliVM-GC framework and no open-source code is available as our baseline, making it difficult for direct and fair comparison. To set a directly comparable baseline, we thus also implement state-of-the-art privacy-preserving distributed Newton method using latest cryptography (same as our protocols), which may be of separate interest. We run all experiments between two commodity PCs with 2.5 GHz quad-core CPU and 16 GB memory, connected via Ethernet.

Our empirical evaluations focus on the following criteria: 1) Model estimation quality (the accuracy of estimated regression parameters); 2) Model convergence performance. We use the relative change of likelihood function value as our convergence checking criteria, and a threshold of 10^{-6} is adopted [84]. To be more specific, we stop the iterative process when the likelihood value l_k at iteration k satisfies $\frac{|l_k - l_{k-1}|}{|l_{k-1}|} < 10^{-6}$ (we also tested other thresholds such as 10^{-7} and 10^{-8}, which do not affect our main results and conclusions).

5.4.1 Data Sets

In our empirical studies, we include a series of simulated and real world data sets, which represent a wide spectrum of applications from different domains and of different scales. The real-world data sets include: 1) the *Wine* quality study (with 6,497 samples and 12 features) [35] for predicting wine quality from physicochemical test results, 2) online *Loans* data (with 122, 578 samples and 33 features) from Lending Club [86] for studying loan default status from loan application data, 3) company *Insurance* study (of dimension: $9,882 \times 38$) for predicting caravan insurance from demographic attributes and other personal financial factors, and 4) a *News* dataset (of dimension $39,082 \times 52$) [47] for predicting the popularity of news article on Mashable.com.

To make our evaluations more comprehensive, we have also simulated a series of data set with varying sample sizes and dimensionality: *SimuX10* $(50,000 \times 10)$, *SimuX12* $(1,000,000 \times 12)$, *SimuX50* $(1,000,000 \times 50)$, *SimuX100* $(3,000,000 \times 100)$, *SimuX150* $(4,000,000 \times 150)$, *SimuX200* $(5,000,000 \times 200)$, *SimuX400* $(50,000,000 \times 400)$. In brief, we adopt a similar simulation approach to [90] by randomly generating covariates $\boldsymbol{X} \in \mathcal{R}^{n \times p}$ and regression parameters $\boldsymbol{\beta} \in \mathcal{R}^{p \times 1}$, and deriving responses $\boldsymbol{y} \in \mathcal{R}^{n \times p}$ according to Bernoulli distribution.

These evaluation data sets are representative for most large-scale studies in the domains we are interested in. We also randomly partition the data sets into subsets horizontally in order to simulate the multi-party setting in collaborative studies.

5.4.2 Model Accuracy

First and foremost, we want to ensure that our proposals are scientifically sound and reliable. To do so, we examine the model quality (as measured by accuracy of regression parameter estimates) estimated by PrivLogit. The standard non-secure distributed Newton's method serves as the ground truth. Our hypothesis is that despite the significant change in our numerical optimizer and reliance on cryptographic operations, the accuracy of our model estimation is still guaranteed.

Numerical results have confirmed our hypothesis, as is illustrated in Figure 5.2. Specifically, the regression parameters estimation $\boldsymbol{\beta}$ by PrivLogit are in perfect alignment with the ground-truth estimation by Newton's method across all studies. This implies that the approximate Hessian adaption we introduce to the Newton's method does not affect model quality.

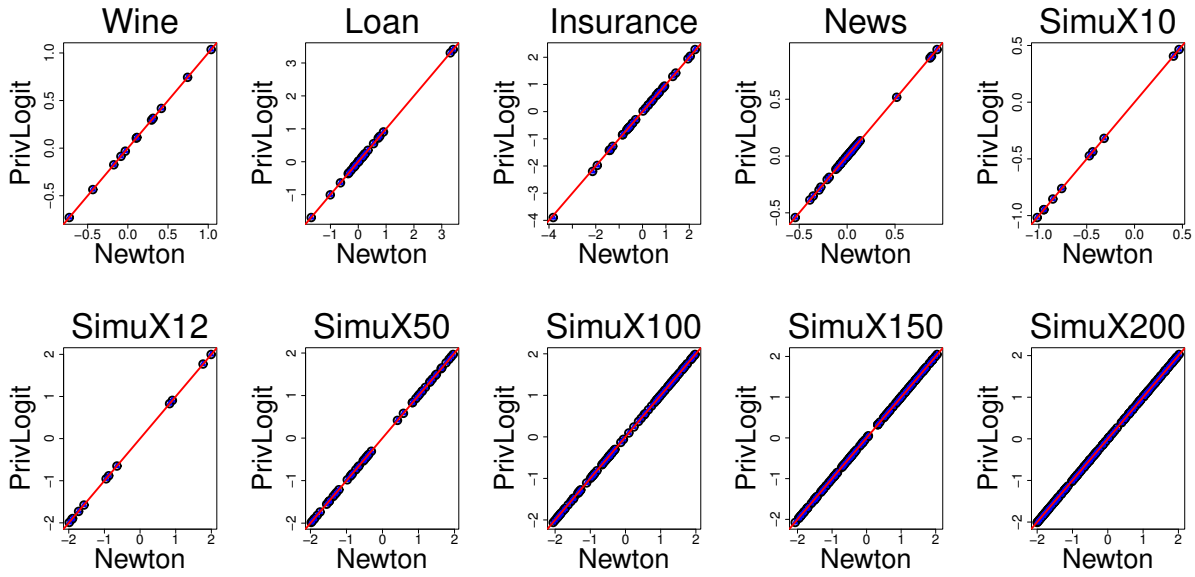


FIGURE 5.2: Comparison of regression parameters estimated by PrivLogit vs. the estimation by the baseline Newton's method across all the data sets in our experiment.

5.4.3 Computational Performance

Next, we evaluate the computational performance of PrivLogit-Hessian and PrivLogit-Local in terms of iterations count and total run-time. We partition each evaluation data set into 5 parties. As is illustrated in Figure 5.3, PrivLogit manage to converge within a reasonable number of iterations for all evaluation data sets. The convergence rate for Newton's method is quadratic, whereas the rate for PrivLogt is linear.

However, as shown in Table 5.1, detailed run-time analysis reveals that our two secure protocols, PrivLogit-Hessian and PrivLogit-Local, turn out to be quite competitive in performance. For instance, regarding the *Loans* study, while Newton method takes only 6

iterations, its actual run-time reaches as many as 492 seconds (because of expensive per-iteration computation); On the other hand, despite requiring substantially more iterations (i.e., 18), the PrivLogit-Hessian and PrivLogit-Local protocols only take around 272 and 104 seconds, respectively, leading to a big speedup . Other evaluation studies also confirm the relatively good performance of PrivLogit proposals. Furthermore, we also tested on data sets with dimension as high as 400, a scale that has seldomly been tested for privacy-preserving logistic regression before. Only PrivLogit-Local converged within reasonable time (110,597.86 seconds or roughly 1.28 days).

To better demonstrate the relative performance of PrivLogit-Hessian and PrivLogit-Local over Newton's method, we present the relative speedup of our methods over the baseline Newton's method. As illustrated in Figure 5.4, PrivLogit-Hessian is always more efficient than Newton's method across all data set. For PrivLogit-Local, the speedup is even more significant, with a speedup of up to 10x. This provides further evidence that PrivLogit proposals have better performance compared to Newton's method, and our relative competitive advantage increases along with data scale.

TABLE 5.1: Model convergence rate and run-time (seconds) for PrivLogit and Newton's method.

Data set	**Iterations (Newton)**	**Iterations (PrivLogit)**	**Time (Newton)**	**Time (PrivLogit-Hessian)**	**Time (PrivLogit-Local)**
Wine	5	13	32	24	17
Loans	6	17	492	260	104
Insurance	7	28	843	520	144
News	5	13	1,442	621	313
SimuX10	6	20	26	24	13
SimuX12	6	22	38	37	17
SimuX50	6	32	1,550	1052	383
SimuX100	7	59	13,138	7,817	1,807
SimuX150	7	83	42,951	25030	6,055
SimuX200	8	105	114,522	56,917	14105
SimuX400	8	206	N/A	N/A	110,598

5.4.4 Model Convergence Guarantee

Another advantage of PrivLogit-based proposals is that it is guaranteed to converge to the global optima, which is not true for Newton method. It is widely acknowledged that

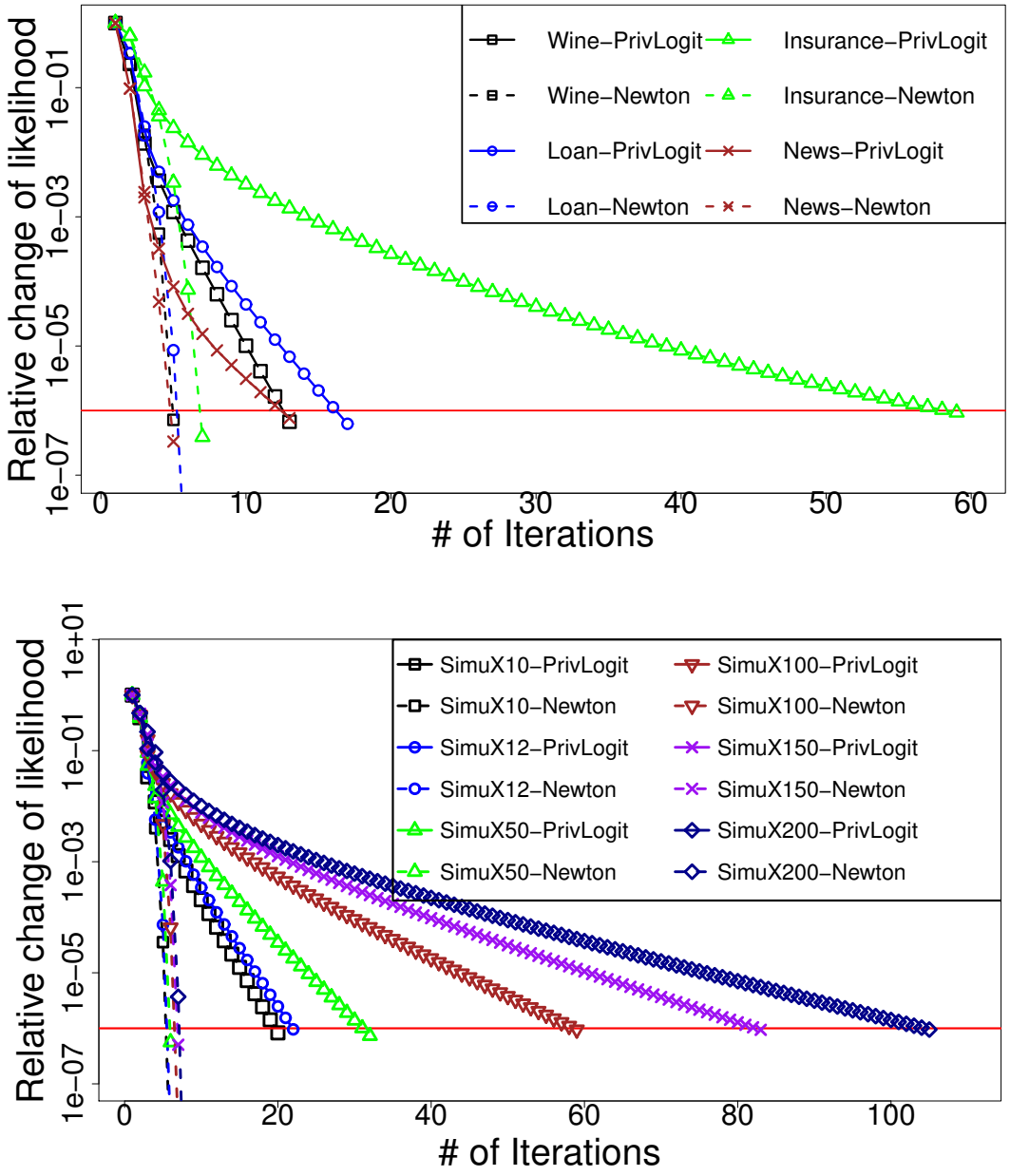


FIGURE 5.3: Convergence rate of PrivLogit and the baseline Newton's method on real-world (upper panel) and simulated (lower panel) data sets. Iterations stop when relative change of likelihood is smaller than threshold 10^{-6}. Both PrivLogit and Newton's method converge within a reasonable number of iterations; and the former converges much slower than the latter in terms of raw counts of iterations.

Newton is sensitive to initial values of $\boldsymbol{\beta}$. Certain suboptimal choice of initialization may cause divergence of the model, leading to indefinite iteration loops.

To compare the convergence of PrivLogit protocols and Newton methods, we use the *SimuX50* dataset and run a series of random initializations by initializing all the

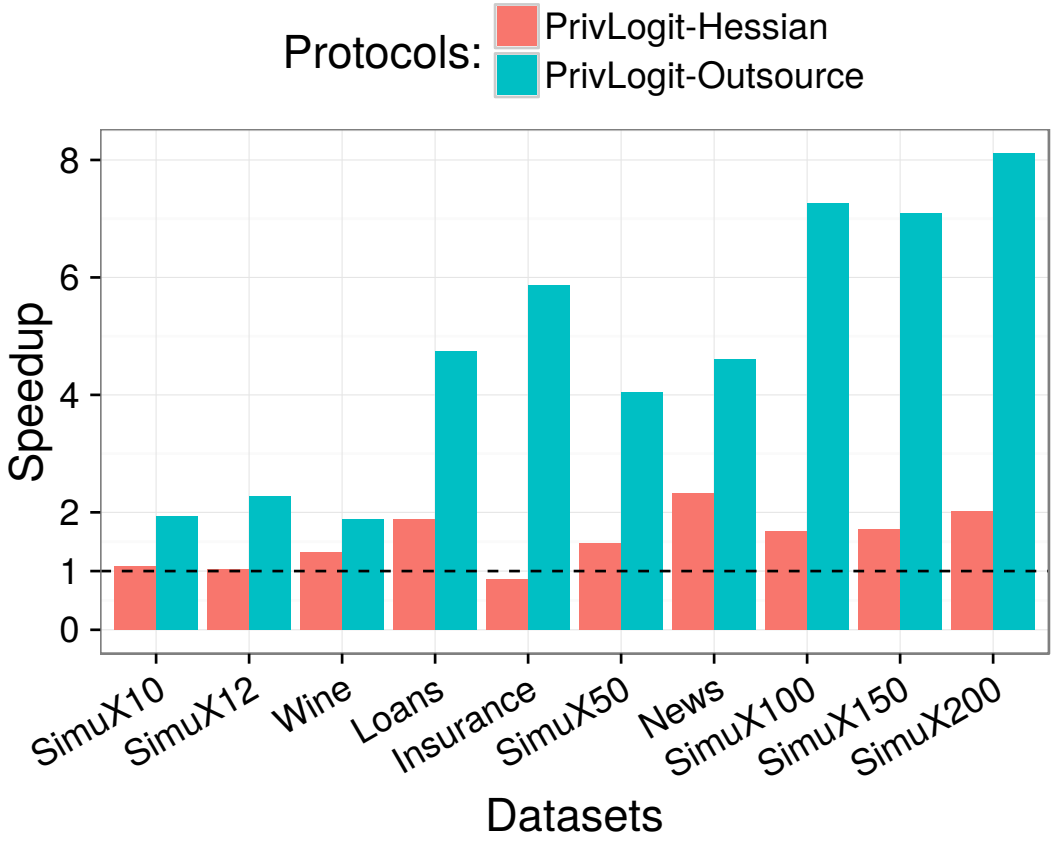


FIGURE 5.4: Relative speedup of PrivLogit-Hessian and PrivLogit-Local over the distributed Newton baseline (the $y = 1$ line), across various datasets. PrivLogit-Hessian can speed-up the computation by around 1.2x∼2.2x on the various studies we evaluated. For PrivLogit-Local, the improvement is even more encouraging, with upto 10x speedup (e.g., for SimuX150 and SimuX200).

regression parameters at the following values: $0.8, 1, 1.5, 2$, respectively. We report on the discrepancy (measured in terms of Euclidean distance) between the regression parameters estimates at each iteration and the ground-truth $\boldsymbol{\beta}$ estimated by Newton's method.

The superior convergence guarantee of PrivLogit is manifested in Figure 5.5, where PrivLogit always converges within a reasonable number of iterations. Newton method, however, diverge significantly from the first few iterations and its coefficient estimation is getting worse and worse. Our extensive evaluations on other data sets indicate that the divergence of Newton is not uncommon in practice.

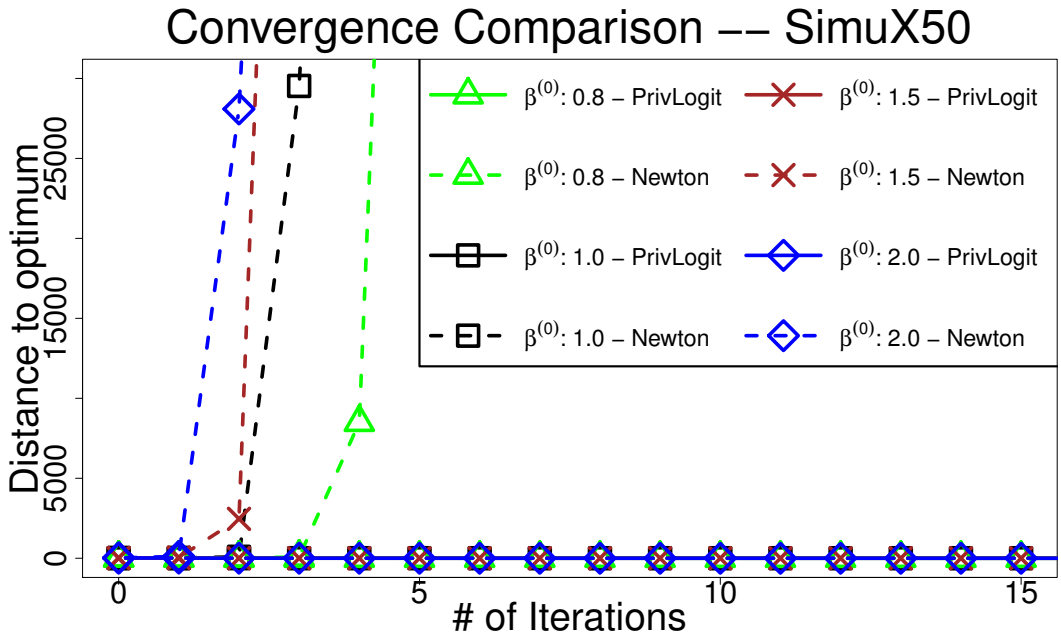


FIGURE 5.5: Comparison of model convergence for PrivLogit and Newton methods, as tested with different initializations of coefficient estimates. We measure the Euclidean distance between per-iteration coefficient estimates and the ground-truth coefficients. Distance close or equal to 0 implies perfect estimation (no discrepancy). Both PrivLogit-Hessian and PrivLogit-Local always converge within reasonable iterations, while Newton method could diverge significantly and end up with indefinite iterations.

5.5 Chapter Conclusion

In this chapter, we have made a novel observation about a generic performance bottleneck in privacy-preserving logistic regression, and proposed an improved numerical optimizer (i.e., PrivLogit) and demonstrate its obscure but surprisingly competitive performance for privacy-preserving logistic regression. This contrasts to common practice in privacy-preserving data mining (i.e., directly applying mainstream numerical optimization), which often disregards secure computing-specific characteristics and thus misses valuable opportunities for significant performance boost. Based on PrivLogit, we also propose two secure and highly-efficient protocols for privacy-preserving logistic regression. We validate our proposals extensively using both analytical and empirical evaluations. Results indicate that our proposals outperform baselines while ensuring privacy and accuracy. Our methods should be helpful for making privacy-preserving logistic regression more scalable and

practical for large collaborative studies. And our novel and generic perspective on tailoring optimizers for secure computing should also inspire other research in secure data management in general.

Chapter 6

Conclusion

In this chapter, we conclude this dissertation with a summary of our work in Chapter 3 to 5, as well as ideas for future research.

6.1 Our Contributions

In this dissertation, we explore the multi-party privacy-preserving machine learning problem from different perspectives. To be more specific:

(a) In Chapter 3, we design a multi-party differentially private algorithm to train an MLP in a more efficient way. The key idea is to focus on the large gradients and apply the newly proposed idea of Terngrad to further reduce communication burden. The algorithm is implemented and tested on a large real-world credit card fraud detection data set contributed by a renowned bank. Contrary to the common belief, the evaluation of utility-privacy trade-off suggests that privacy doesn't necessarily impose a negative impact on the model performance. This is consistent with previous findings that adding random noise to gradient actually improve the learning of deep networks.

(b) In Chapter 4, we switch focus to the transfer learning framework and study the multi-source differentially private hypothesis transfer learning problem. Specifically, we take advantage of the relatively abundant supply of unlabeled data and quantify the relationship between the source domains and the target domain by applying a

differentially private importance weighting algorithm. Given the relationship and the differentially private hypotheses transferred from sources, we construct an informative prior for the logistic regression model on the target domain. Empirical experiments on 2 famous benchmark data sets confirm the utility of our method even when labeled data are extremely scarce in the target domain.

(c) In Chapter 5, we propose to use a constant approximation of the Hessian matrix when training logistic regression with Newton's method. This seemingly counter-intuitive approach turns out to be an effective solution to the expensive computation in secure multi-party logistic regression. Despite the elongated convergence, this method avoids the repeated evaluation, inversion and encryption of the Hessian matrix at each iteration, therefore leading to significant speedup. In addition, this simple approximation is guaranteed to converge to the optimal solution at a linear rate and less dependent on the initialization of parameters than Newton's method. We further propose two secure implementations of the method and test it on a variety of real-word and simulated data sets. The experimental results show that our methods is helpful in facilitating large-scale collaborative studies in a privacy-preserving manner.

6.2 Future Work

There are many exciting future directions that we intend to pursue in the future following the work in each chapter.

(a) We are considering further evaluate the multi-party privacy preserving deep learning algorithm in Chapter 3 and conduct a set of more comprehensive experiments under the multi-party heterogeneous data set setting, for instance, the ratio of fraud transaction, the number of transactions, the merchant category for transactions, etc. varies in each participating bank. In addition, we are interested to introduce a framework where participating parties can adaptively set their privacy budget

during the model training process. The method can also be extended to other ANN models (CNN, RNN) or any machine learning models that are trained with gradient-based methods.

(b) We intend to combine our differentially private hypothesis transfer learning mechanism with other machine learning models and other model aggregation methods. Moreover, the current experiment setting is that the target domain is a mixture of the sources domain. We are interested to investigate the performance of our method when the target domain contains samples from a population unseen by the sources.

(c) For PrivLogit, there is room for further acceleration on our protocols as cryptography techniques continue to improve and we leave it as future work to explore alternative cryptographic schemes. While our work focuses on logistic regression model, our proposal of tailoring optimizers for secure computing seems widely applicable to other machine learning models as well.

Bibliography

[1] Martín Abadi et al. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pp. 308–318.

[2] Nishant Agarwal and Meghna Sharma. "Fraud Risk Prediction in Merchant-Bank Relationship using Regression Modeling". In: *Vikalpa* 39.3 (2014), pp. 67–76.

[3] Charu C Aggarwal and S Yu Philip. *A general survey of privacy-preserving data mining models and algorithms*. Springer, 2008.

[4] Mário S Alvim et al. "Differential privacy: on the trade-off between utility and information leakage". In: *Formal Aspects of Security and Trust*. Springer, 2012, pp. 39–54.

[5] Guozhong An. "The effects of adding noise during backpropagation training on a generalization performance". In: *Neural Computation* 8.3 (1996), pp. 643–674.

[6] Steven C Bagley, Halbert White, and Beatrice A Golomb. "Logistic regression in the medical literature:: Standards for use and reporting, with particular attention to one medical domain". In: *Journal of Clinical Epidemiology* 54.10 (2001), pp. 979–985.

[7] Raef Bassily, Adam Smith, and Abhradeep Thakurta. "Private empirical risk minimization: Efficient algorithms and tight error bounds". In: *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE. 2014, pp. 464–473.

[8] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. “Completeness theorems for non-cryptographic fault-tolerant distributed computation”. In: *Proceedings of the 20th annual ACM Symposium on Theory of Computing*. ACM. 1988, pp. 1–10.

[9] Kristin P Bennett and Emilio Parrado-Hernández. “The interplay of optimization and machine learning research”. In: *Journal of Machine Learning Research* 7 (2006), pp. 1265–1281.

[10] Dimitri P Bertsekas. “Incremental gradient, subgradient, and proximal methods for convex optimization: A survey”. In: *Optimization for Machine Learning* 2010 (2011), pp. 1–38.

[11] Dimitri P Bertsekas. *Nonlinear Programming*. Athena scientific, 1999.

[12] Dimitri P Bertsekas and John N Tsitsiklis. “Gradient convergence in gradient methods with errors”. In: *SIAM Journal on Optimization* 10.3 (2000), pp. 627–642.

[13] Jeff Bezanson et al. “Julia: A fast dynamic language for technical computing”. In: *arXiv preprint arXiv:1209.5145* (2012).

[14] Doron Blatt, Alfred O Hero, and Hillel Gauchman. “A convergent incremental gradient method with a constant step size”. In: *SIAM Journal on Optimization* 18.1 (2007), pp. 29–51.

[15] Pierre-Alexandre Bliman and Giancarlo Ferrari-Trecate. “Average consensus problems in networks of agents with delayed communications”. In: *Automatica* 44.8 (2008), pp. 1985–1995.

[16] John Blitzer, Mark Dredze, and Fernando Pereira. “Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification”. In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. 2007, pp. 440–447.

[17] Dankmar Böhning. “11 Construction of reliable maximum likelihood algorithms with application to logistic and cox regression”. In: *Handbook of Statistics* 9 (1993), pp. 409–422.

[18] Dankmar Böhning and Bruce G Lindsay. "Monotonicity of quadratic-approximation algorithms". In: *Annals of the Institute of Statistical Mathematics* 40.4 (1988), pp. 641–663.

[19] Steven M Boker et al. "Maintained individual data distributed likelihood estimation (MIDDLE)". In: *Multivariate Behavioral Research* 50.6 (2015), pp. 706–720.

[20] Richard J Bolton and David J Hand. "Statistical fraud detection: a review". In: *Statistical Science* (2002), pp. 235–249.

[21] Joseph-Frédéric Bonnans et al. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

[22] Léon Bottou. "Online learning and stochastic approximations". In: *On-line Learning in Neural Networks* 17.9 (1998), p. 142.

[23] Olivier Bousquet and André Elisseeff. "Stability and generalization". In: *Journal of Machine Learning Research* 2 (2002), pp. 499–526.

[24] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[25] Stephen Boyd et al. "Randomized gossip algorithms". In: *IEEE/ACM Transactions on Networking (TON)* 14.SI (2006), pp. 2508–2530.

[26] Charles George Broyden. "The convergence of a class of double-rank minimization algorithms 1. general considerations". In: *IMA Journal of Applied Mathematics* 6.1 (1970), pp. 76–90.

[27] C Sidney Burrus. "Iterative reweighted least squares". In: *OpenStax-CNX Web site. http://cnx. org/content/m45285/1.12* (2012).

[28] Ming Cao, A Stephen Morse, and Brian DO Anderson. "Reaching a consensus in a dynamically changing environment: A graphical approach". In: *SIAM Journal on Control and Optimization* 47.2 (2008), pp. 575–600.

[29] Rich Caruana. "Multitask learning". In: *Learning to learn*. Springer, 1998, pp. 95–133.

[30] Kamalika Chaudhuri and Claire Monteleoni. “Privacy-preserving logistic regression”. In: *Advances in Neural Information Processing Systems*. 2009, pp. 289–296.

[31] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. “Differentially private empirical risk minimization”. In: *The Journal of Machine Learning Research* 12 (2011), pp. 1069–1109.

[32] David Chaum, Claude Crépeau, and Ivan Damgard. “Multiparty unconditionally secure protocols”. In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*. ACM. 1988, pp. 11–19.

[33] I-An Chen. “Fast distributed first-order methods”. PhD thesis. Massachusetts Institute of Technology, 2012.

[34] Andrew R Conn, Nicholas IM Gould, and Ph L Toint. “Convergence of quasi-Newton matrices generated by the symmetric rank one update”. In: *Mathematical Programming* 50.1-3 (1991), pp. 177–195.

[35] Paulo Cortez et al. “Modeling wine preferences by data mining from physicochemical properties”. In: *Decision Support Systems* 47.4 (2009), pp. 547–553.

[36] David R Cox. “The regression analysis of binary sequences”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1958), pp. 215–242.

[37] David W Craig et al. “Assessing and managing risk when sharing aggregate genetic variant data”. In: *Nature Reviews Genetics* 12.10 (2011), p. 730.

[38] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314.

[39] Daniel Demmler, Thomas Schneider, and Michael Zohner. “ABY-A Framework for Efficient Mixed-Protocol Secure Two-Party Computation.” In: *22nd Annual Network and Distributed System Security Symposium (NDSS)*. 2015.

[40] Chuong B Do and Andrew Y Ng. “Transfer learning for text classification”. In: *Advances in Neural Information Processing Systems*. 2006, pp. 299–306.

[41] Cynthia Dwork. “Differential privacy”. In: *Automata, Languages and Programming*. Springer, 2006, pp. 1–12.

[42] Cynthia Dwork and Aaron Roth. “The algorithmic foundations of differential privacy”. In: *Foundations and Trends in Theoretical Computer Science* 9.3-4 (2014), pp. 211–407.

[43] Cynthia Dwork et al. “Calibrating noise to sensitivity in private data analysis”. In: *Theory of Cryptography*. Springer, 2006, pp. 265–284.

[44] Khaled El Emam et al. “A secure distributed logistic regression protocol for the detection of rare adverse drug events”. In: *Journal of the American Medical Informatics Association* 20.3 (2013), pp. 453–461.

[45] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. “Rappor: Randomized aggregatable privacy-preserving ordinal response”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2014, pp. 1054–1067.

[46] Li Fei-Fei. “Knowledge transfer in learning to recognize visual objects classes”. In: *Proceedings of the International Conference on Development and Learning (ICDL)*. 2006, p. 11.

[47] Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. “A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News”. In: *Progress in Artificial Intelligence*. Springer, 2015, pp. 535–546.

[48] Roger Fletcher. “A new approach to variable metric algorithms”. In: *The Computer Journal* 13.3 (1970), pp. 317–322.

[49] Vojtech Franc, Alexander Zien, and Bernhard Schölkopf. “Support vector machines as probabilistic models”. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 665–672.

[50] Kunihiko Fukushima and Sei Miyake. “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition”. In: *Competition and Cooperation in Neural Nets*. Springer, 1982, pp. 267–285.

[51] Keke Gai and Meikang Qiu. “Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers”. In: *IEEE Transactions on Industrial Informatics* (2017).

[52] Keke Gai, Meikang Qiu, and Xiaotong Sun. “A survey on FinTech”. In: *Journal of Network and Computer Applications* (2017).

[53] Keke Gai et al. “Privacy-preserving multi-channel communication in Edge-of-Things”. In: *Future Generation Computer Systems* 85 (2018), pp. 190–200.

[54] Jochen Garcke and Thomas Vanck. “Importance weighted inductive transfer learning for regression”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2014, pp. 466–481.

[55] Amadou Gaye et al. “DataSHIELD: taking the analysis to the data, not the data to the analysis”. In: *International Journal of Epidemiology* 43.6 (2014), pp. 1929–1944.

[56] Craig Gentry. “A fully homomorphic encryption scheme”. PhD thesis. Stanford University, 2009.

[57] Oded Goldreich, Silvio Micali, and Avi Wigderson. “How to play any mental game”. In: *Proceedings of the nineteenth Annual ACM symposium on Theory of Computing*. ACM. 1987, pp. 218–229.

[58] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[59] Peter J Green. “Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1984), pp. 149–192.

[60] Sunil Kumar Gupta, Santu Rana, and Svetha Venkatesh. “Differentially private multi-task learning”. In: *Pacific-Asia Workshop on Intelligence and Security Informatics*. Springer. 2016, pp. 101–113.

[61] MT Hale and M Egerstedty. “Differentially private cloud-based multi-agent optimization with constraints”. In: *American Control Conference (ACC), 2015*. IEEE. 2015, pp. 1235–1240.

[62] Jihun Hamm, Yingjun Cao, and Mikhail Belkin. “Learning privately from multiparty data”. In: *International Conference on Machine Learning*. 2016, pp. 555–563.

[63] Jihun Hamm et al. “Crowd-ml: A privacy-preserving learning framework for a crowd of smart devices”. In: *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE. 2015, pp. 11–20.

[64] Frank Harrell. *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015.

[65] Frank E Harrell. “Ordinal logistic regression”. In: *Regression Modeling Strategies*. Springer, 2001, pp. 331–343.

[66] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. “Deep models under the GAN: information leakage from collaborative deep learning”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2017, pp. 603–618.

[67] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[68] Nils Homer et al. “Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays”. In: *PLoS Genet* 4.8 (2008), e1000167.

[69] Dusan Jakovetic, Joao Xavier, and Jose MF Moura. "Fast distributed gradient methods". In: *Automatic Control, IEEE Transactions on* 59.5 (2014), pp. 1131–1146.

[70] Felicia Y Jakubiec and Alejandro Ribeiro. "D-map: Distributed maximum a posteriori probability estimation of dynamic systems". In: *Signal Processing, IEEE Transactions on* 61.2 (2013), pp. 450–466.

[71] Zhanglong Ji and Charles Elkan. "Differential privacy based on importance weighting". In: *Machine learning* 93.1 (2013), pp. 163–183.

[72] Zhanglong Ji et al. "Differentially private distributed logistic regression using private and public data". In: *BMC Medical Genomics* 7.1 (2014), S14.

[73] Jing Jiang and ChengXiang Zhai. "Instance weighting for domain adaptation in NLP". In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. 2007, pp. 264–271.

[74] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. "Secure multi-party differential privacy". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2008–2016.

[75] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. "The composition theorem for differential privacy". In: *IEEE Transactions on Information Theory* (2017).

[76] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC Press, 2014.

[77] S Sathiya Keerthi and Dennis DeCoste. "A modified finite Newton method for fast solution of large scale linear SVMs". In: *Journal of Machine Learning Research*. 2005, pp. 341–361.

[78] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. "Private convex empirical risk minimization and high-dimensional regression". In: *Journal of Machine Learning Research* 1 (2012), p. 41.

[79] Miran Kim et al. “Secure Logistic Regression Based on Homomorphic Encryption: Design and Evaluation”. In: *JMIR Medical Informatics* 6.2 (2018).

[80] Gary King and Langche Zeng. “Logistic regression in rare events data”. In: *Political Analysis* 9.2 (2001), pp. 137–163.

[81] Ilja Kuzborskij and Francesco Orabona. “Fast rates by transferring from auxiliary hypotheses”. In: *Machine Learning* 106.2 (2017), pp. 171–195.

[82] Ilja Kuzborskij and Francesco Orabona. “Stability and hypothesis transfer learning”. In: *Proceedings of The 30th International Conference on Machine Learning (ICML-13)*. 2013, pp. 942–950.

[83] Ken Lang. “Newsweeder: Learning to filter netnews”. In: *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 331–339.

[84] Su-In Lee et al. “Efficient l˜ 1 regularized logistic regression”. In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 21. 1. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 2006, p. 401.

[85] Jing Lei. “Differentially private m-estimators”. In: *Advances in Neural Information Processing Systems*. 2011, pp. 361–369.

[86] LengingClub. *Loans Data*. Last accessed: 02-02-2016. 2016.

[87] Cathryn M Lewis and Jo Knight. “Introduction to genetic association studies”. In: *Cold Spring Harbor Protocols* 2012.3 (2012), pdb–top068163.

[88] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. “t-closeness: Privacy beyond k-anonymity and l-diversity”. In: *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE. 2007, pp. 106–115.

[89] Tiancheng Li and Ninghui Li. “On the tradeoff between privacy and utility in data publishing”. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2009, pp. 517–526.

[90] Wenfa Li et al. “Supporting Regularized Logistic Regression Privately and Efficiently”. In: *arXiv preprint arXiv:1510.00095* (2015).

[91] Xuejun Liao, Ya Xue, and Lawrence Carin. “Logistic regression with an auxiliary data source”. In: *Proceedings of the 22nd International Conference on Machine Learning*. ACM. 2005, pp. 505–512.

[92] Chi-Chen Lin et al. “Detecting the financial statement fraud: The analysis of the differences between data mining techniques and experts’ judgments”. In: *Knowledge-Based Systems* 89 (2015), pp. 459–470.

[93] Yehuda Lindell and Benny Pinkas. “A proof of security of Yao’s protocol for two-party computation”. In: *Journal of Cryptology* 22.2 (2009), pp. 161–188.

[94] Qing Ling and Alejandro Ribeiro. “Decentralized dynamic optimization through the alternating direction method of multipliers”. In: *Signal Processing Advances in Wireless Communications (SPAWC), 2013 IEEE 14th Workshop on*. IEEE. 2013, pp. 170–174.

[95] Chang Liu et al. “Oblivm: A programming framework for secure computation”. In: *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE. 2015, pp. 359–376.

[96] Dong C Liu and Jorge Nocedal. “On the limited memory BFGS method for large scale optimization”. In: *Mathematical Programming* 45.1-3 (1989), pp. 503–528.

[97] Edmund G Lowrie and Nancy L Lew. “Death risk in hemodialysis patients: the predictive value of commonly measured variables and an evaluation of death rate differences between facilities”. In: *American Journal of Kidney Diseases* 15.5 (1990), pp. 458–482.

[98] Zhongqi Lu et al. “Source Free Transfer Learning for Text Classification.” In: *AAAI Conference on Artificial Intelligence*. 2014, pp. 122 –128.

[99] Ashwin Machanavajjhala et al. “l-diversity: Privacy beyond k-anonymity”. In: *Data Engineering, 2006. ICDE’06. Proceedings of the 22nd International Conference on*. IEEE. 2006, pp. 24–24.

[100] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. “Domain adaptation with multiple sources”. In: *Advances in Neural Information Processing Systems*. 2009, pp. 1041–1048.

[101] Zvika Marx et al. “Two algorithms for transfer learning”. In: *Inductive Transfer: 10 Years Later* (2008).

[102] Tara C Matise et al. “The Next PAGE in understanding complex traits: design for the analysis of Population Architecture Using Genetics and Epidemiology (PAGE) Study”. In: *American Journal of Epidemiology* 174.7 (2011), pp. 849–859.

[103] Catherine A McCarty et al. “The eMERGE Network: a consortium of biorepositories linked to electronic medical records data for conducting genomic studies”. In: *BMC Medical Genomics* 4.1 (2011), p. 13.

[104] H Brendan McMahan et al. “Ad click prediction: a view from the trenches”. In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2013, pp. 1222–1230.

[105] Frank McSherry and Kunal Talwar. “Mechanism design via differential privacy”. In: *Foundations of Computer Science, 2007. FOCS’07. 48th Annual IEEE Symposium on*. IEEE. 2007, pp. 94–103.

[106] Lilyana Mihalkova and Raymond J Mooney. “Transfer learning by mapping with minimal target data”. In: *Proceedings of the AAAI-08 Workshop on Transfer Learning for Complex Tasks*. 2008.

[107] Tomáš Mikolov. “Statistical language models based on neural networks”. In: *Presentation at Google, Mountain View, 2nd April* (2012).

[108] Sayan Mukherjee et al. “Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization”. In: *Advances in Computational Mathematics* 25.1-3 (2006), pp. 161–193.

[109] Arvind Narayanan and Vitaly Shmatikov. “Robust de-anonymization of large sparse datasets”. In: *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE. 2008, pp. 111–125.

[110] Yuval Nardi, Stephen E Fienberg, and Robert J Hall. “Achieving both valid and secure logistic regression analysis on aggregated data from different private sources”. In: *Journal of Privacy and Confidentiality* 4.1 (2012), p. 9.

[111] Angelia Nedic. “On the rate of convergence of distributed subgradient methods for multi-agent optimization”. In: *Decision and Control, 2007 46th IEEE Conference on*. IEEE. 2007, pp. 4711–4716.

[112] Angelia Nedić and Dimitri Bertsekas. “Convergence rate of incremental subgradient algorithms”. In: *Stochastic Optimization: Algorithms and Applications*. Springer, 2001, pp. 223–264.

[113] Angelia Nedic and Dimitri P Bertsekas. “Incremental subgradient methods for nondifferentiable optimization”. In: *SIAM Journal on Optimization* 12.1 (2001), pp. 109–138.

[114] Angelia Nedić and Asuman Ozdaglar. “Distributed subgradient methods for multi-agent optimization”. In: *Automatic Control, IEEE Transactions on* 54.1 (2009), pp. 48–61.

[115] Arvind Neelakantan et al. “Adding gradient noise improves learning for very deep networks”. In: *arXiv preprint arXiv:1511.06807* (2015).

[116] A Michael Nielsen. “Neural Networks and Deep Learning”. In: *Determination Press* (2015).

[117] Valeria Nikolaenko et al. “Privacy-preserving matrix factorization”. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM. 2013, pp. 801–812.

[118] Valeria Nikolaenko et al. “Privacy-preserving ridge regression on hundreds of millions of records”. In: *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE. 2013, pp. 334–348.

[119] Pascal Paillier. “Public-key cryptosystems based on composite degree residuosity classes”. In: *Advances in Cryptology—EUROCRYPT’99*. Springer. 1999, pp. 223–238.

[120] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359.

[121] Weike Pan, Erheng Zhong, and Qiang Yang. “Transfer learning for text mining”. In: *Mining Text Data*. Springer, 2012, pp. 223–257.

[122] Nicolas Papernot et al. “Scalable Private Learning with PATE”. In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rkZB1XbRZ.

[123] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “Understanding the exploding gradient problem”. In: *CoRR, abs/1211.5063* (2012).

[124] Manas Pathak, Shantanu Rane, and Bhiksha Raj. “Multiparty differential privacy via aggregation of locally trained classifiers”. In: *Advances in Neural Information Processing Systems*. 2010, pp. 1876–1884.

[125] Martin Pettai and Peeter Laud. “Combining differential privacy and secure multiparty computation”. In: *Proceedings of the 31st Annual Computer Security Applications Conference*. ACM. 2015, pp. 421–430.

[126] Michael Rabbat and Robert Nowak. “Distributed optimization in sensor networks”. In: *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*. ACM. 2004, pp. 20–27.

[127] Michael G Rabbat and Robert D Nowak. “Quantized incremental algorithms for distributed optimization”. In: *Selected Areas in Communications, IEEE Journal on* 23.4 (2005), pp. 798–808.

[128] Rajat Raina, Andrew Y Ng, and Daphne Koller. "Constructing informative priors using transfer learning". In: *Proceedings of the 23rd International Conference on Machine Learning*. ACM. 2006, pp. 713–720.

[129] Rajat Raina et al. "Self-taught learning: transfer learning from unlabeled data". In: *Proceedings of the 24th International Conference on Machine Learning*. ACM. 2007, pp. 759–766.

[130] Arun Rajkumar and Shivani Agarwal. "A differentially private stochastic gradient descent algorithm for multiparty classification". In: *International Conference on Artificial Intelligence and Statistics*. 2012, pp. 933–941.

[131] S Sundhar Ram, A Nedić, and Venugopal V Veeravalli. "Distributed stochastic subgradient projection algorithms for convex optimization". In: *Journal of Optimization Theory and Applications* 147.3 (2010), pp. 516–545.

[132] S Sundhar Ram, A Nedic, and VV Veeravalli. "Stochastic incremental gradient descent for estimation in sensor networks". In: *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on*. IEEE. 2007, pp. 582–586.

[133] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), p. 386.

[134] Michael T Rosenstein et al. "To transfer or not to transfer". In: *NIPS 2005 Workshop on Transfer Learning*. Vol. 898. 2005, pp. 1–4.

[135] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), p. 533.

[136] Gabriel Rushin et al. "Horse race analysis in credit card fraud-deep learning, logistic regression, and Gradient Boosted Tree". In: *Systems and Information Engineering Design Symposium (SIEDS), 2017*. IEEE. 2017, pp. 117–121.

[137] Pierangela Samarati and Latanya Sweeney. “Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression”. In: *Technical report, SRI International* (1998).

[138] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (2015), pp. 85–117.

[139] Devavrat Shah. *Gossip algorithms*. Now Publishers Inc, 2009.

[140] Shai Shalev-Shwartz et al. “Learnability, stability and uniform convergence”. In: *Journal of Machine Learning Research* 11.Oct (2010), pp. 2635–2670.

[141] Adi Shamir. “How to share a secret”. In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

[142] Wei Shi et al. “Extra: An exact first-order algorithm for decentralized consensus optimization”. In: *SIAM Journal on Optimization* 25.2 (2015), pp. 944–966.

[143] Reza Shokri and Vitaly Shmatikov. “Privacy-preserving deep learning”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2015, pp. 1310–1321.

[144] Andrea Simonetto, Leon Kester, and Geert Leus. “Distributed Time-Varying Stochastic Optimization and Utility-based Communication”. In: *arXiv preprint arXiv:1408.5294* (2014).

[145] Andrea Simonetto and Geert Leus. “Distributed asynchronous time-varying constrained optimization”. In: *Signals, Systems and Computers, 2014 48th Asilomar Conference on*. IEEE. 2014, pp. 2142–2146.

[146] Aleksandra B Slavkovic, Yuval Nardi, and Matthew M Tibbits. “" Secure" Logistic Regression of Horizontally and Vertically Partitioned Distributed Databases”. In: *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*. IEEE. 2007, pp. 723–728.

[147] Shuang Song, Kamalika Chaudhuri, and Anand Sarwate. “Learning from data with heterogeneous noise using sgd”. In: *Artificial Intelligence and Statistics*. 2015, pp. 894–902.

[148] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. “Stochastic gradient descent with differentially private updates”. In: *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE. 2013, pp. 245–248.

[149] S Sundhar Ram, A Nedić, and Venugopal V Veeravalli. “A new class of distributed optimization algorithms: Application to regression of distributed data”. In: *Optimization Methods and Software* 27.1 (2012), pp. 71–88.

[150] Latanya Sweeney. “k-anonymity: A model for protecting privacy”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 557–570.

[151] John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. “Distributed asynchronous deterministic and stochastic gradient optimization algorithms”. In: *IEEE Transactions on Automatic Control* 31.9 (1986), pp. 803–812.

[152] Lorenzo Valerio, Andrea Passarella, and Marco Conti. “Hypothesis transfer learning for efficient data computing in smart cities environments”. In: *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1–8.

[153] Rui Wang et al. “Learning your identity and disease from research papers: information leaks in genome wide association study”. In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. ACM. 2009, pp. 534–544.

[154] Yang Wang et al. “Privacy Preserving Distributed Deep Learning and its Application in Credit Card Fraud Detection”. In: *TrustCom/BigDataSE, 2018*. IEEE. 2018 (in press).

[155] Yu-Xiang Wang, Jing Lei, and Stephen E Fienberg. “Learning with differential privacy: Stability, learnability and the sufficiency and necessity of ERM principle”. In: *arXiv preprint arXiv:1502.06309* (2015).

[156] Stanley L Warner. “Randomized response: A survey technique for eliminating evasive answer bias”. In: *Journal of the American Statistical Association* 60.309 (1965), pp. 63–69.

[157] Larry Wasserman. *All of statistics*. Springer Science & Business Media, 2011.

[158] Wei Wen et al. “Terngrad: Ternary gradients to reduce communication in distributed deep learning”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 1508–1518.

[159] Michael Wolfson et al. “DataSHIELD: resolving a conflict in contemporary bioscience—performing a pooled analysis of individual-level data without sharing the data”. In: *International Journal of Epidemiology* 39.5 (2010), pp. 1372–1382.

[160] Genqiang Wu et al. “Inherit differential privacy in distributed setting: Multiparty randomized function computation”. In: *Trustcom/BigDataSE/I SPA, 2016 IEEE*. IEEE. 2016, pp. 921–928.

[161] Yuan Wu, Xiaoqian Jiang, and Lucila Ohno-Machado. “Preserving institutional privacy in distributed binary logistic regression”. In: *AMIA Annual Symposium Proceedings*. Vol. 2012. American Medical Informatics Association. 2012, p. 1450.

[162] Yuan Wu et al. “Grid Binary LOgistic REgression (GLORE): building shared models without sharing data”. In: *Journal of the American Medical Informatics Association* 19.5 (2012), pp. 758–764.

[163] Liyang Xie et al. “Privacy-preserving distributed multi-task learning with asynchronous updates”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017, pp. 1195–1204.

[164] Wei Xie et al. “Privlogit: Efficient privacy-preserving logistic regression by tailoring numerical optimizers”. In: *arXiv preprint arXiv:1611.01170* (2016).

[165] Wei Xie et al. “SecureMA: protecting participant privacy in genetic association meta-analysis”. In: *Bioinformatics* 30.23 (2014), pp. 3334–3341.

[166] Jia Xu et al. “Differentially private histogram publication”. In: *The VLDB Journal* 22.6 (2013), pp. 797–822.

[167] Ching-Nung Yang et al. “Enhancing security for two-party comparison over encrypted data”. In: *Systems and Informatics (ICSAI), 2016 3rd International Conference on*. IEEE. 2016, pp. 439–443.

[168] Andrew C Yao. “Protocols for secure computations”. In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE. 1982, pp. 160–164.

[169] M. F. Zeager et al. “Adversarial learning in credit card fraud detection”. In: *2017 Systems and Information Engineering Design Symposium (SIEDS)*. 2017, pp. 112–116. DOI: 10.1109/SIEDS.2017.7937699.

[170] Tong Zhang. “Solving large scale linear prediction problems using stochastic gradient descent algorithms”. In: *Proceedings of the 21st International Conference on Machine Learning*. ACM. 2004, p. 116.

[171] Yu Zhang and Dit-Yan Yeung. “A convex formulation for learning task relationships in multi-task learning”. In: *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2010, pp. 733–742.

[172] Yong Zhuang et al. “Distributed Newton Methods for Regularized Logistic Regression”. In: *Advances in Knowledge Discovery and Data Mining*. Springer, 2015, pp. 690–703.