

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Privacy-Preserving Algorithms for Machine Learning

Permalink

<https://escholarship.org/uc/item/43j071fj>

Author

Song, Shuang

Publication Date

2018

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

Privacy-Preserving Algorithms for Machine Learning

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Computer Science

by

Shuang Song

Committee in charge:

Professor Kamalika Chaudhuri, Chair
Professor Sanjoy Dasgupta
Professor Tara Javidi
Professor Lawrence Saul
Professor Staal Vinterbo

2018

Copyright
Shuang Song, 2018
All rights reserved.

The Dissertation of Shuang Song is approved and is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2018

DEDICATION

To my parents.

TABLE OF CONTENTS

Signature Page	iii
Dedication	iv
Table of Contents	v
List of Figures	viii
List of Tables	x
Acknowledgements	xi
Vita	xiii
Abstract of the Dissertation	xiv
Chapter 1 Introduction	1
Chapter 2 Differential Privacy and Its Variants	9
2.1 Differential Privacy	9
2.1.1 The Definition	9
2.1.2 Basic Tools	11
2.1.3 Properties of Differential Privacy	12
2.2 Rényi Differential Privacy	13
2.2.1 The Definition	13
2.2.2 Properties of Rényi Differential Privacy	14
2.3 Local Differential Privacy	15
2.4 Pufferfish Privacy	15
2.4.1 The Definition	16
2.4.2 Properties of Pufferfish Privacy	17
Chapter 3 Differentially Private Stochastic Gradient Descent	18
3.1 Overview	18
3.1.1 Related Work	19
3.2 Preliminaries	19
3.3 SGD with Differential Privacy	20
3.4 Experiments	22
3.4.1 Datasets	22
3.4.2 Procedure	22
3.4.3 Mini-batching reduces variance	24
3.4.4 Choosing appropriate parameters	24
3.5 Conclusions	25
Chapter 4 Local Differentially Private SGD with Heterogeneous Privacy Requirements	28

4.1	Overview	28
4.1.1	Related Work	31
4.2	The Model	31
4.2.1	The Heterogeneous Noise Model	32
4.3	Data order depends on learning rate	35
4.4	Adapting the learning rate to the noise level	37
4.4.1	Algorithm description	38
4.4.2	Regret Bounds	40
4.5	Experiments	41
4.6	Conclusion	44
Chapter 5	Scalable Private Learning with PATE	46
5.1	Overview	46
5.1.1	Related Work	48
5.2	Background and Overview	49
5.2.1	The PATE Framework	49
5.2.2	Rényi Differential Privacy	51
5.2.3	PATE Aggregation Mechanisms	52
5.2.4	Data-dependent Privacy in PATE	52
5.3	Improved Aggregation Mechanisms for PATE	53
5.3.1	The GNMax Aggregator and Its Privacy Guarantee	54
5.3.2	The Confident-GNMax Aggregator	56
5.3.3	The Interactive-GNMax Aggregator	57
5.4	Experimental Evaluation	58
5.4.1	Experimental Setup	59
5.4.2	Comparing the LNMax and GNMax Mechanisms	60
5.4.3	Student Training with the GNMax Aggregation Mechanisms	63
5.4.4	Noisy Threshold Checks and Privacy Costs	65
5.5	Conclusions	66
Chapter 6	Differentially Private Continual Release of Graph Statistics	68
6.1	Overview	68
6.1.1	Related Work	70
6.2	Preliminaries	72
6.2.1	Graphs and Graph Sequences	72
6.2.2	Node Differential Privacy	72
6.2.3	Bounded Degree Graphs	73
6.2.4	Graph Functions	74
6.2.5	Other Notations	74
6.3	Main Algorithm	75
6.4	Functions of Degree Distributions	77
6.4.1	Undirected Graphs	77
6.4.2	Directed Graphs	79
6.5	Functions of Subgraph Counts	79

6.5.1	Undirected Graphs	80
6.5.2	Directed Graphs	81
6.6	Experiments	81
6.6.1	Methodology	82
6.6.2	Datasets	83
6.6.3	Results	85
6.7	Conclusion	88
Chapter 7	Pufferfish Privacy Mechanisms for Correlated Data	90
7.1	Overview	90
7.1.1	Related Work	92
7.2	The Setting	94
7.2.1	The Privacy Framework	94
7.2.2	Examples	95
7.2.3	Guarantee Against Close Adveraries	97
7.2.4	Additional Notation	98
7.3	A General Mechanism	99
7.3.1	The Wasserstein Mechanism	100
7.3.2	Performance Guarantees	101
7.4	A Mechanism for Bayesian Networks	102
7.4.1	The Setting	102
7.4.2	The Markov Quilt Mechanism	104
7.4.3	Case Study: Markov Chains	106
7.5	Composition Properties	114
7.5.1	Parallel Composition	114
7.5.2	Sequential Composition	116
7.6	Experiments	118
7.6.1	Methodology	118
7.6.2	Simulations	119
7.6.3	Real Data	121
7.6.4	Discussion	125
7.7	Conclusion	125
	Bibliography	127

LIST OF FIGURES

Figure 3.1.	Objective function value vs. number of iterations for private and non-private algorithms on the MNIST and KDDCup99 data sets. Horizontal axis is scaled so that the number of samples is the same.	23
Figure 3.2.	Objective function value vs. batch size b for private and non-private algorithms on MNIST and KDDCup99.	23
Figure 3.3.	Objective function value vs. number of data points for private and non-private algorithms on synthetic data for batch size $b = 5$	25
Figure 4.1.	Column 1 plots $ f(w_{T+1}) - f(v_{T+1}) $ vs. constant c for $\lambda = 0.001$. Column 2 plots final objective function value vs. ϵ_N for $\epsilon_C = 10$. Column 3 plots final objective function value vs. c_2 for $\epsilon_N = 2$ (top) and $\epsilon_N = 1$ (bottom). Top row shows figures for MNIST and bottom row for Covertypes.	42
Figure 5.1.	Our proposed technique, Confident-GNMax, improves on the original PATE (LNMax) on all measures. <i>Left:</i> Accuracy is higher. <i>Middle:</i> Privacy cost is quartered. <i>Right:</i> Intuitive privacy is improved. These are results for a character-recognition task.	47
Figure 5.2.	Overview of the approach: (1) an ensemble of teachers is trained on disjoint subsets of the sensitive data, (2) a student model is trained on public data labeled using the ensemble.	50
Figure 5.3.	Some example inputs from the Glyph dataset along with the class they are labeled as. Note the ambiguity (between the comma and apostrophe) and the mislabeled input.	62
Figure 5.4.	Tradeoff between utility and privacy for the LNMax and GNMax aggregators on Glyph: effect of the noise distribution (left) and size of the teacher ensemble (right).	62
Figure 5.5.	Effects of the noisy threshold checking: # queries answered by LNMax, Confident-GNMax moderate ($T=3500, \sigma_1=1500$), and Confident-GNMax aggressive ($T=5000, \sigma_1=1500$). The black dots and the right axis show the expected cost of answering a single query in each bin (via GNMax, $\sigma_2=100$).	65
Figure 5.6.	Effects of the noisy threshold checking: Privacy cost of answering all (LNMax) vs only inexpensive queries (GNMax) for a given number of answered queries. The very dark area under the curve is the cost of selecting queries; the rest is the cost of answering them.	66

Figure 6.1.	L_1 error vs. privacy parameter ϵ . Publish every 1 year. Averaged over 100 runs.....	86
Figure 6.2.	L_1 error vs. number of publications T . $\epsilon = 5$. Averaged over 100 runs. ...	87
Figure 7.1.	An example of ∞ -Wasserstein distance. The frequency distributions represent two measures μ and ν , and the arrows represent the joint distribution γ that minimizes the right hand side of (7.1). Here, $W_\infty(\mu, \nu) = 1$	100
Figure 7.2.	A Bayesian Network on four variables.....	103
Figure 7.3.	Illustration of Markov Blanket and Markov Quilt	105
Figure 7.4.	L_1 error of frequency of state 1 vs. α for $\epsilon = 0.2, 1, 5$ for synthetic data. $\Theta = [\alpha, 1 - \alpha]$. GK16 does not apply left of the black dashed vertical line. GroupDP has error around 5, 1, 0.2 for $\epsilon = 0.5, 1, 5$ respectively. Reported values are averaged over 500 random trials.	119
Figure 7.5.	Exact and private aggregated physical activity relative frequency histograms for three groups of participants. Reported values are for 20 random trials and $\epsilon = 1$. Recall GK16 does not apply for this problem.	120

LIST OF TABLES

Table 5.1.	Utility and privacy of the students. For MNIST, Adult, and SVHN, we use the labels of ensembles of 250 teachers published by [111] and set $\delta = 10^{-5}$ (to the exception of SVHN where $\delta = 10^{-6}$). All Glyph results use an ensemble of 5000 teachers and δ is set to 10^{-8}	62
Table 6.1.	Summary of degree distribution results.	77
Table 6.2.	Subgraphs in undirected graphs.	80
Table 6.3.	Subgraphs in directed graphs.	80
Table 6.4.	Summary of common properties of the graph datasets. Max degree refers to the undirected version of the graph while max in-degree and max out-degree refer to the directed version.	83
Table 7.1.	L_1 error of the relative frequency histograms for individual and aggregate tasks for physical activity of three participant groups. $\epsilon = 1$. Reported values are averaged over 20 random trials.	121
Table 7.2.	Running time (in seconds) of an optimized algorithm that calculates the scale parameter of the Laplace noise (averaged over 5 runs). $\epsilon = 1$	121
Table 7.3.	L_1 error of relative frequency histogram for electricity consumption data. Reported values are averaged over 20 random trials.	123

ACKNOWLEDGEMENTS

First and foremost, I would like to convey my sincere gratitude to my PhD advisor, Professor Kamalika Chaudhuri, who has been tremendously supportive since my first day as her student. The discussions that we had not only inspired me with new research ideas, but also guided me in overcoming life and career challenges. I feel extremely fortunate to have her as my advisor.

My sincere thanks also go to the rest of my thesis committee: Professor Sanjoy Dasgupta, Professor Lawrence Saul, Professor Tara Javidi and Professor Staal Vinterbo, for their constructive feedback and comments that helped complete my thesis and dissertation.

I deeply appreciate everything, both technical and non-technical, that I have learned from my collaborators (ordered alphabetically): Úlfar Erlingsson, Joseph Geumlek, Daniel Hsu, Susan Little, Sanjay Mehta, Ilya Mironov, Nicolas Papernot, Ananth Raghunathan, Anand Sarwate, Kunal Talwar, Staal Vinterbo and Yizhen Wang.

I also would like to thank Oracle Lab, especially my host Alan Wood and my co-worker Aleksey Urmanov, and the Privacy & Security Team in Google Brain, especially my host Ananth Raghunathan for offering me great summer internship opportunities in 2015 and 2018, respectively. These internships and great mentors brought me valuable research experiences and enjoyable summers.

My PhD life could not have been so wonderful without all my lab-mates, office-mates and many other friends in UC San Diego. I would always remember the insightful discussions we have had about research and life.

Last but not least, I would like to thank my parents, Shuying Wang and Yuejin Song, and my husband, Yuliang Li, for their unconditional love and support in my life. I could not have done it without them.

Chapter 3 is based on “Stochastic gradient descent with differentially private updates” by Shuang Song, Kamalika Chaudhuri and Anand Sarwate, which appears in the Global Conference on Signal and Information Processing (GlobalSIP), 2013 [130]. The dissertation author was the

primary author of the paper.

Chapter 4 is based on “Learning from Data with Heterogenous Noise using SGD” by Shuang Song, Kamalika Chaudhuri and Anand Sarwate, which appears in the International Conference on Artificial Intelligence and Statistics (AISTATS) 2015 [131]. The dissertation author was the primary author of the paper.

Chapter 5 is based on “Scalable Private Learning with PATE” by Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar and Úlfar Erlingsson, which appears in the International Conference on Learning Representations (ICLR), 2018 [112]. The dissertation author was a primary author of the paper.

Chapter 6 is based on “Differentially Private Continual Release of Graph Statistics” by Shuang Song, Sanjay Mehta, Staal A. Vinterbo, Susan Little and Kamalika Chaudhuri which is under submission. The dissertation author was the primary author of the paper.

Chapter 7 is based on “Pufferfish Privacy Mechanisms for Correlated Data” by Shuang Song, Yizhen Wang and Kamalika Chaudhuri, which appears in the International Conference on Management of Data (SIGMOD), 2017 [132]. The dissertation author was a primary author of the paper.

Section 7.5 of Chapter 7 is based on “Composition Properties of Inferential Privacy for Time-Series Data” by Shuang Song and Kamalika Chaudhuri, which appears in the Allerton Conference on Communication, Control and Computing, 2017 [129]. The dissertation author was the primary author of the paper.

Finally, my thanks go to many anonymous reviewers for their helpful comments.

VITA

- 2012 Bachelor of Science, The Hong Kong University of Science and Technology, Hong Kong
- 2015 Master of Science, University of California, San Diego
- 2018 Doctor of Philosophy, University of California, San Diego

ABSTRACT OF THE DISSERTATION

Privacy-Preserving Algorithms for Machine Learning

by

Shuang Song

Doctor of Philosophy in Computer Science

University of California San Diego, 2018

Professor Kamalika Chaudhuri, Chair

Modern machine learning increasingly involves personal data, such as healthcare, financial and user behavioral information. However, models trained on such data can reveal detailed information about the data and cause a serious privacy breach. Consequently, it is important to design algorithms that can analyze the sensitive data while still preserving privacy.

This thesis advances the state-of-the-art of privacy-preserving machine learning in the following two major aspects.

First, this thesis addresses the challenges in differentially private large-scale machine learning. On the one hand, with a large amount of sensitive user data, privacy-preserving learning algorithms are expected to achieve improved utility. On the other hand, big data imposes

additional challenges, including performance (a.k.a Volume), data noise (a.k.a Veracity), a large number of classes and distributed sources (a.k.a Variety). This thesis presents (1) private versions of the widely used stochastic gradient descent (SGD) algorithm with generalization to data from multiple sources of different privacy requirements, and (2) an improved version of the Private Aggregation of Teacher Ensembles (PATE) framework which can scale to learning tasks with a large number of output classes and uncurated, imbalanced training data.

Second, this thesis considers privacy-preserving data analysis beyond tabular data. Differential privacy is best suited for tabular data, where each record corresponds to all the information about an individual and records are independent of each other. However, many real-world applications involve non-tabular sensitive data, such as epidemic transmission graphs and measurement of the physical activity of a single subject across time. To analyze disease transmission graphs, or more general, graphs with sensitive information stored in each node, this thesis considers privacy-preserving continual release of graph statistics, such as the percentages of highly-active patients over time. The proposed algorithm outperforms the baselines in utility over a range of parameters. For physical activity measurement, or more generally, data with correlation, this thesis looks at a recent generalization of differential privacy, called Pufferfish privacy, that addresses privacy concerns in correlated data. Two mechanisms that work under different scenarios are proposed, and one of them is evaluated on real and synthetic time-series data.

Chapter 1

Introduction

Modern machine learning increasingly involves sensitive personal data such as healthcare, financial and user behavioral information. With models trained on such data, researchers can gain invaluable insights into real-world problems across various domains. However, these trained models may unexpectedly reveal details of the sensitive data like an individual user's health condition and daily activities, which causes serious moral and legal concerns. To address the concerns while keeping the positive social impact of machine learning, it is therefore important to design algorithms that analyze sensitive personal data while protecting individuals' privacy.

There has been a long line of work on conducting data analysis while preventing identification of individual participants from the results of the analysis [58, 37, 118, 121, 59, 114] – this is formally called statistical disclosure control [34] (see [3] for a survey of the field prior to 1989). Before the era of differential privacy, one of the most commonly used privacy-protection techniques had been anonymization, where the main idea is to remove personally identifiable information, such as name, gender, age and geographical information, from a dataset before releasing it. However, it was proven by previous studies that severe privacy breach can still happen with anonymized data. In 2006, The New York Times identified and interviewed a user from an anonymized search log dataset published by AOL [9]. The attack succeeded because the search logs, though anonymized, contain user-specific queries that reveal one's age, gender, address and occupation – enough for the inference of the one's identity. Another example

proving the failure of anonymization is the Netflix Prize data leak. Using the Internet Movie Database (IMDb) as a source of background knowledge, researchers were able to identify a significant number of users from the anonymized movie rating dataset published by Netflix [103]. There have been extensions of anonymization such as k -anonymity [135], ℓ -diversity [94] and t -closeness [89]. However, most of them are known to be vulnerable to attacks with auxiliary information [63].

Previous studies have also shown that privacy breach can happen even if we hide the dataset and release only aggregate statistics. [40] proved theoretically that an adversary can reconstruct a dataset almost exactly with aggregate statistics. Such privacy breaches happened on real applications as well. [140] presented an attack on GWAS (genome-wide association study), a study that aims at finding the associations between single-nucleotide polymorphisms (SNPs) and diseases. It demonstrated that when given a large number of SNPs statistics and the genomic data of an individual, an adversary is able to determine if this individual is in the case group and therefore has the disease. Another example is the model inversion attack presented in [61]. By making queries to a neural network trained for a face-recognition task, an adversary is able to recover images of users whose data is included in the training process.

The failures of anonymization and releasing aggregate information motivate the necessity for a stronger and mathematically rigorous definition of privacy.

Differential privacy [45], a cryptographically-motivated definition of privacy, satisfies such requirements. It has gained significant attention ever since its proposal and has emerged as the gold standard in data privacy. Informally, an algorithm satisfies differential privacy if the probability of any outcome does not change significantly when a single individual's private data changes. An adversary is therefore unable to make much inference about an individual given the algorithm's output even with auxiliary information about the rest of the dataset or unlimited computational power. Individuals thus will not incur a significant privacy loss when allowing their data to be used by a differentially private algorithm. In the formal definition, a parameter ϵ quantifies the privacy loss of an algorithm, making differential privacy a mathematically rigorous

definition.

The basic techniques for achieving differential privacy in simple queries include the Laplace mechanism [45], the Gaussian mechanism [44] and the exponential mechanism [100]. All techniques guarantee differential privacy by injecting randomness in the algorithm. The amount of randomness is determined by the privacy parameter as well as a property of the query function called the global sensitivity, which measures the maximum amount of change in the function value when one individual's record changes. Based on these basic techniques, differentially private algorithms have been designed for numerous machine learning tasks such as empirical risk minimization [26, 10, 12, 62], Bayesian inference [38, 142, 152, 60, 65], recommender system [99], deep learning [128, 1], and unsupervised learning tasks such as dimensionality reduction [18, 28, 52] and clustering [18, 109, 134]. The key challenge in differentially private algorithm design is the privacy-utility tradeoff. In short, guaranteeing privacy involves randomizing the algorithms' output which impacts the performance or utility of the resulting procedures. For example, in parameter estimation, guaranteeing privacy may increase the mean-squared error of the estimator.

Differential privacy has also spawned a rich body of research on privacy definitions, including several variants of differential privacy that capture the privacy concerns in scenarios not considered by differential privacy. For instance, local differential privacy [77, 42, 43] considers the case where there is no trusted data holder who collects and analyzes the sensitive data as in differential privacy. Instead, individuals' data is perturbed locally before being transferred to and analyzed by an untrusted party. Local differential privacy is thus a strictly stronger notion compared to differential privacy and enables data collection under more restricted conditions. Another variant, Pufferfish privacy [84], is a generalization of differential privacy customized for data with correlation. While differential privacy does not adequately address privacy issues in correlated data, Pufferfish protects individuals' privacy against adversaries who possess prior knowledge about the correlation across multiple individuals.

This thesis advances privacy-preserving machine learning in two major aspects.

Differentially private large-scale machine learning

First, the presented thesis addresses the challenges in differentially private large-scale machine learning. Due to the increasing rate of data collection and generation, just like other machine learning tasks, privacy-preserving learning is required to be capable of handling a large amount of data. On the positive side, with a large amount of user data, the privacy-utility tradeoff is expected to be less intensive as it is easier to hide individuals' information in the large crowd. On the other side, however, non-private machine learning algorithms can still cause privacy breach, and big data imposes additional challenges, including performance (a.k.a Volume), data noise (a.k.a Veracity), a large number of classes and distributed sources (a.k.a Variety). It remains unclear how to perform large-scale machine learning with differential privacy. This thesis presents two methods that tackle these challenges.

- Chapter 3 considers the stochastic gradient descent (SGD) algorithm, which is one of the most commonly used algorithms in large-scale machine learning for its simplicity, scalability and convergence guarantee. We derive differentially private versions of the SGD algorithm and evaluate them on real and synthetic datasets. The experimental results show that for classification using logistic regression, differentially private single-point SGD has high variance, but a moderate increase in the batch size can improve the performance significantly. For low-dimensional problems, the private algorithm's performance is close to that of non-private SGD.

Additionally, with a small modification, the differentially private SGD algorithms can be adapted to the local differential privacy setting, where each individual provides a perturbed gradient. Specifically, we consider the scenario where private data is collected from multiple sources with heterogeneous privacy preferences. For example, a drug test might be performed at different medical institutes which operate under different privacy regulations and a researcher wants to combine the results to perform a more comprehensive analysis. In a simplified yet general case where the training data is from two sources,

one more privacy-demanding and one less, we show both theoretically and empirically that the order in which we should process the two datasets to get good performance depends on the learning rate of the private SGD algorithm. We also provide a heuristic for choosing different learning rates for the two datasets and demonstrate empirically that our algorithm outperforms the naive approaches using a single learning rate or using the less privacy-demanding data only. These results are presented in Chapter 4.

- In Chapter 5, we revisit a machine learning framework called the Private Aggregation of Teacher Ensembles, or PATE [111], and develop techniques that improve its scalability and practical applicability. PATE is a recently proposed framework that offers scalable, high-utility machine learning with strong privacy-protection guarantees. In the PATE framework, multiple “teacher” models are trained on disjoint sensitive data (e.g., different users’ data), and PATE uses the teachers’ aggregated consensus answers in a black-box fashion to supervise the training of a “student” model. PATE guarantees differential privacy by adding random noise to the aggregate answers used to train the student and releasing only the student model.

It is worth emphasizing that the privacy guarantee of PATE is agnostic to the underlying models used by the teachers and the student. Besides, PATE is, by nature, a distributed machine learning framework; the sensitive data can be stored in multiple servers where each server trains a teacher model locally and simultaneously. Thus, communication cost is only paid for transporting the trained teacher models to an aggregator which uses them to answer queries from the student. The original PATE, however, was applied to only simple tasks like MNIST, without any realistic and larger-scale evaluation.

We propose new techniques that improve PATE in terms of its privacy, utility, and scalability. When evaluated on a character recognition task with 150 output classes, the improved PATE demonstrates its applicability on uncurated, large-scale data. To this end, we claim that the improved PATE is a significant step towards large-scale learning with privacy.

Privacy-preserving data analysis on non-tabular data

Differential privacy guarantees that the participation of any individual does not make a significant difference to the outcome of the algorithm. It is therefore best suited for *tabular* data where all the information of an individual is stored as one single record and records are independent of each other. However, in many real-world applications, the sensitive data that needs to be protected is *non-tabular*, where each protected entity affects multiple records through the intrinsic structure or correlation in the data.

Consider, for example, a disease transmission graph that encodes how a disease transmits among individuals. Analyzing the general statistics on such a graph provides valuable information for epidemiology research. At the same time, due to the sensitive nature of certain diseases, we need to make sure that the infection status of any individual is protected. The structure of the graph can cause additional difficulty, as a highly infectious individual can change the structure of the graph drastically and it is therefore non-trivial to hide the evidence of the existence of such an individual while maintaining a reasonable level of utility.

Another type of non-tabular data is data with correlation. Consider, for example, a simple time-series application – measurement of physical activity of a single subject across time. The goal here is to release aggregate statistics on the subject’s activities over a long period (say a week) and the entity to be protected is the activity at any specific instant (say, 10:30am on Jan 4). If the measurements are made at small intervals, then the records form a highly correlated time-series as human activities change slowly over time. The activities performed before or after provide strong evidence about what the subject was doing at a given time, introducing additional challenges in privacy-protection for correlated data. To address privacy challenges in this kind of data, we need a generalization of differential privacy that can capture the data correlation and protect the entities under such correlation.

- To tackle the difficulty in the disease transmission graph example, in Chapter 6, we consider the problem of privacy-preserving continual release of graph statistics on sensitive graphs

where nodes and their associated edges appear over time in an online manner. We consider HIV transmission data collected from patients in a particular region over multiple years. By measuring similarities between HIV sequences of different patients, we can build an HIV transmission network where each node represents one patient and each edge an occurrence of transmission. Epidemiologists can understand how HIV propagates by studying properties of the network over time. Since there is a considerable social stigma associated with HIV, we need to ensure that privacy of the included patients is not violated when we release statistics of the network. Additionally, analyses may need to happen intermittently so that properties of the network as it evolves may be studied.

To apply differential privacy in such a scenario, it is essential to determine what a single individual's data contribute to the graph. Prior work has looked at two forms of differential privacy in graphs – edge differential privacy, where an edge corresponds to an individual's private value, and node differential privacy, where a single node corresponds to an individual. In the HIV transmission graph example, a patient corresponds to a node, and hence node differential privacy is our privacy notion of choice. The main challenge in the continual privacy-preserving release of graph statistics is maintaining a good privacy-utility tradeoff, as a node with a huge number of adjacent edges can change the statistics drastically and hiding the effect of a node can require a significant amount of noise.

In Chapter 6, we show that if there is a publicly known upper bound on the maximum degree of any node in the entire graph sequence, then we can release private sequences of simple statistics such as the number of high-degree nodes and subgraph counts with relatively high accuracy. Experiments on both real and synthetic datasets show that our algorithm outperforms the baselines in utility over a range of privacy parameters.

- To address the privacy concerns in correlated data such as that in the activity tracking example, in Chapter 7, we consider Pufferfish privacy [84], a generalization of differential privacy that captures data correlation. The standard differential privacy does not offer

enough protection for data with correlation; whereas group differential privacy, a direct adoption of differential privacy, adds noise proportional to the number of correlated records and causes unnecessary utility loss. On the contrary, Pufferfish privacy captures data correlation with a parameter Θ representing a class of distributions that plausibly generate the data. It protects the sensitive information in the data against all adversaries whose beliefs lie in Θ . With this specification, Pufferfish offers both privacy and the hope for utility when a large number of individuals or entries are correlated, yet the “average amount” of correlation is low.

The main challenge in using Pufferfish privacy is a lack of suitable mechanism. We therefore provide the first mechanism, called the Wasserstein Mechanism, which applies to any general Pufferfish instantiation. Moreover, we consider the case when the data correlation can be captured by a Bayesian network and propose a computationally efficient mechanism, called the Markov Quilt Mechanism. As a case study, we derive a simplified version of the mechanism for time-series data. We provide privacy and utility guarantees, establish composition properties, and demonstrate the practical applicability of the mechanism through experimental evaluations on synthetic as well as real data.

The rest of the thesis is structured as follows. Chapter 2 reviews the definition and properties of differential privacy, as well as some variants of differential privacy relevant to this thesis. Chapter 3 presents the differentially private SGD algorithms, with the extension to local differentially private SGD with data coming from sources of different privacy requirements presented in Chapter 4. Chapter 5 discusses the improved version of the PATE framework. Next, Chapter 6 presents the algorithm for continual release of graph statistics under node differential privacy. Algorithms achieving Pufferfish privacy for correlated data are introduced in Chapter 7.

Chapter 2

Differential Privacy and Its Variants

This chapter reviews the formal definition of differential privacy, some of its properties, and the commonly used basic algorithms that achieve it. Moreover, this chapter reviews several variants of differential privacy relevant to this thesis, including Rényi differential privacy, local differential privacy, and Pufferfish privacy.

2.1 Differential Privacy

Differential privacy considers the setting where a trusted data curator has access to a sensitive dataset D ; an untrusted data analyst send statistical queries about D to the curator, who then answers the queries with a randomized algorithm \mathcal{M} . We can assume that the sensitive dataset D contains n records from a data universe \mathcal{X} and each record contains the sensitive information from a single individual that needs to be protected. Differential privacy guarantees that each record has little influence on the output of the randomized algorithm \mathcal{M} , and an adversary thus cannot infer much about the presence or absence of any individual in the dataset from the query results.

2.1.1 The Definition

Before defining differential privacy, we present the definition of the distance between two datasets.

Definition 2.1.1 (Distance between Datasets). *Given datasets $D, D' \in \mathcal{X}^n$, the distance between them, denoted by $d(D, D')$, is the minimum number of records in D that need to be changed to convert D to D' . If $d(D, D') = 1$, we call D and D' neighboring datasets.*

Differential privacy ensures that the participation of a single individual in a dataset does not change the probability of any outcome by much, which is captured by requiring an algorithm to have similar output distributions under any pair of neighboring datasets.

Definition 2.1.2 (ϵ -Differential Privacy). *A randomized algorithm \mathcal{M} is said to guarantee ϵ -differential privacy if for any neighboring datasets D and D' , and for any $w \in \text{Range}(\mathcal{M})$, we have*

$$P(\mathcal{M}(D) = w) \leq e^\epsilon \cdot P(\mathcal{M}(D') = w),$$

where the probability is with respect to the randomness in \mathcal{M} .

In this definition, ϵ is a parameter often called the *privacy budget* or the *privacy risk*. A larger value of ϵ implies higher privacy risk. It can be shown that any non-trivial algorithm that guarantees differential privacy needs to be a randomized algorithm.

A relaxed definition – (ϵ, δ) -differential privacy, is as follows.

Definition 2.1.3 ((ϵ, δ) -Differential Privacy). *A randomized algorithm \mathcal{M} is said to guarantee (ϵ, δ) -differential privacy if for any neighboring datasets D and D' , and for any $S \subseteq \text{Range}(\mathcal{M})$, we have*

$$P(\mathcal{M}(D) \in S) \leq e^\epsilon \cdot P(\mathcal{M}(D') \in S) + \delta,$$

where the probability is with respect to the randomness in \mathcal{M} .

Intuitively, ϵ and δ together quantify the privacy risk; δ is the probability with which e^ϵ fails to bound to ratio between the two probabilities. When $\delta = 0$, (ϵ, δ) -differential privacy re-

duces to ϵ -differential privacy. Usually, ϵ -differential privacy is referred to as the *pure* differential privacy.

2.1.2 Basic Tools

Two basic algorithms that are commonly used for differentially private release of numerical queries are the Laplace Mechanism [45] and the Gaussian Mechanism [44]. Both algorithms add random noise to the exact query value, where the noise has standard deviation proportional to a quantity called the global sensitivity of the query function. Intuitively, the global sensitivity of a query function f measures the maximum amount of change in the value of f when one record in the dataset changes.

Definition 2.1.4 (Global Sensitivity). *Let $f : \mathcal{X}^n \rightarrow \mathbb{R}^d$ be a query function and $\|\cdot\|$ be a distance metric. The global sensitivity of f with respect to $\|\cdot\|$ is defined as*

$$GS(f, \|\cdot\|) = \max_{d(D, D')=1} \|f(D) - f(D')\|.$$

The L_1 and L_2 norm are the most commonly used distance metric. When f is a scalar function or the distance metric is clear from the context, we may omit $\|\cdot\|$ in the notation.

Let $\text{Lap}(\beta)$ be a random variable drawn from the Laplace distribution with mean 0 and scale parameter β , i.e., a distribution with probability density function $\rho(z) = \frac{1}{2\beta} \exp\left(-\frac{|z|}{\beta}\right)$. Given a dataset D , a numerical query function $f : \mathcal{X}^n \rightarrow \mathbb{R}^d$, and a privacy parameter ϵ , the Laplace Mechanism returns

$$f(D) + (Z_1, \dots, Z_d), \text{ with } Z_i \text{ drawn i.i.d. from } \text{Lap}\left(\frac{GS(f, \|\cdot\|_1)}{\epsilon}\right).$$

It can be shown that the Laplace Mechanism guarantees ϵ -differential privacy [45].

To achieve (ϵ, δ) -differential privacy, one can use the Gaussian Mechanism, which adds Gaussian noise to the actual query value $f(D)$. Given a dataset D , a numerical query function

$f : \mathcal{X}^n \rightarrow \mathbb{R}^d$, and privacy parameters (ϵ, δ) , the Gaussian Mechanism returns

$$f(D) + Z, \text{ with } Z \text{ drawn from } \mathcal{N}(0, \sigma^2 I_d) \text{ where } \sigma \geq \frac{\sqrt{2 \log(1.25/\delta)} \text{GS}(f, \|\cdot\|_2)}{\epsilon},$$

and I_d is the identity matrix of size d . This mechanism guarantees (ϵ, δ) -differential privacy for any $\epsilon, \delta \in (0, 1)$.

2.1.3 Properties of Differential Privacy

We review two important properties of differential privacy that allow us to use the basic tools as building blocks to construct algorithms for more sophisticated tasks.

First, differential privacy is immune to post-processing [48], which means any operation on the output of a differentially private algorithm does not increase the privacy risk.

Theorem 2.1.5 (Post-processing). *Let $\mathcal{M} : \mathcal{X}^n \rightarrow \mathcal{Y}$ be an (ϵ, δ) -differentially private algorithm and $f : \mathcal{Y} \rightarrow \mathcal{Z}$ be an arbitrary function. Then $f \circ \mathcal{M} : \mathcal{X}^n \rightarrow \mathcal{Z}$ is (ϵ, δ) -differentially private.*

To build a differentially private algorithm for a sophisticated task, we may consider composing several subroutines each guaranteeing a certain level of privacy. How do we quantify the overall privacy loss of such a composition? The composition theorems of differential privacy offer the answer.

There are two types of composition – parallel and sequential. The former captures the case where multiple differentially private subroutines run on *disjoint* subsets of the entire dataset, whereas the latter captures the case where they run on the same dataset, potentially adaptively. Now we review the composition properties of differential privacy under these two scenarios.

Theorem 2.1.6 (Parallel Composition). *For $i \in [k]$, let \mathcal{M}_i be an ϵ_i -differentially private algorithm. Given a dataset D , let $\{D^{(1)}, \dots, D^{(k)}\}$ be k disjoint subsets of D . Let \mathcal{M} be defined as $\mathcal{M}(D) = (\mathcal{M}_1(D_1), \mathcal{M}_2(D_2), \dots, \mathcal{M}_k(D_k))$. Then \mathcal{M} is $\max_{i \in [k]} \epsilon_i$ -differentially private.*

For sequential composition, we say a sequence of mechanisms $(\mathcal{M}_1, \dots, \mathcal{M}_k)$ are chosen adaptively if \mathcal{M}_i can be chosen based on the outputs of the previous mechanisms $\mathcal{M}_1(D), \dots, \mathcal{M}_{i-1}(D)$.

Theorem 2.1.7 (Basic Sequential Composition). *If a mechanism \mathcal{M} consists of a sequence of adaptive mechanisms $\mathcal{M}_1, \dots, \mathcal{M}_k$ such that for any $i \in [k]$, \mathcal{M}_i guarantees (ϵ_i, δ_i) -differential privacy, then \mathcal{M} guarantees $(\sum_{i=1}^k \epsilon_i, \sum_{i=1}^k \delta_i)$ -differential privacy.*

The advanced composition theorem [51] provides a tighter bound for sequential composition of (ϵ, δ) -differentially private algorithms.

Theorem 2.1.8 (Advanced Sequential Composition). *If a mechanism \mathcal{M} consists of a sequence of adaptive mechanisms $\mathcal{M}_1, \dots, \mathcal{M}_k$ such that for any $i \in [k]$, \mathcal{M}_i guarantees (ϵ, δ) -differential privacy, then \mathcal{M} guarantees $(\sqrt{2k \log(1/\delta')} \epsilon + k\epsilon(e^\epsilon - 1), \delta' + k\delta)$ -differential privacy for any $\delta > 0$.*

2.2 Rényi Differential Privacy

Rényi Differential Privacy, or RDP [101], is a relaxation of the pure differential privacy defined with the Rényi Divergence. RDP shares important properties like post-processing and composition with differential privacy. It is strictly stronger than (ϵ, δ) -differential privacy and can be converted to a curve of $(\epsilon(\delta), \delta)$ -differential privacy guarantees. RDP allows tighter analysis for tracking the privacy loss of the composition of differentially private algorithms. In this section, we review the definition and some properties of RDP.

2.2.1 The Definition

We first define the Rényi Divergence and then RDP.

Definition 2.2.1 (Rényi Divergence). *The Rényi divergence of order λ between two probability*

distributions P and Q is defined as:

$$D_\lambda(P\|Q) \triangleq \frac{1}{\lambda-1} \log \mathbb{E}_{x \sim Q} \left[(P(x)/Q(x))^\lambda \right] = \frac{1}{\lambda-1} \log \mathbb{E}_{x \sim P} \left[(P(x)/Q(x))^{\lambda-1} \right].$$

Definition 2.2.2 (Rényi Differential Privacy (RDP)). *A randomized mechanism \mathcal{M} is said to guarantee (λ, ε) -RDP with $\lambda \geq 1$ if for any neighboring datasets D and D' ,*

$$D_\lambda(\mathcal{M}(D)\|\mathcal{M}(D')) = \frac{1}{\lambda-1} \log \mathbb{E}_{x \sim \mathcal{M}(D)} \left[\left(\frac{P(\mathcal{M}(D) = x)}{P(\mathcal{M}(D') = x)} \right)^{\lambda-1} \right] \leq \varepsilon.$$

RDP generalizes pure differential privacy in the sense that ε -differential privacy is equivalent to (∞, ε) -RDP.

2.2.2 Properties of Rényi Differential Privacy

[101] provides the following composition property of RDP and the conversion from RDP to (ε, δ) -differential privacy.

Theorem 2.2.3 (Composition of RDP). *If a mechanism \mathcal{M} consists of a sequence of adaptive mechanisms $\mathcal{M}_1, \dots, \mathcal{M}_k$ such that for any $i \in [k]$, \mathcal{M}_i guarantees (λ, ε_i) -RDP, then \mathcal{M} guarantees $(\lambda, \sum_{i=1}^k \varepsilon_i)$ -RDP.*

Theorem 2.2.4 (From RDP to DP). *If a mechanism \mathcal{M} guarantees (λ, ε) -RDP, then \mathcal{M} guarantees $(\varepsilon + \frac{\log 1/\delta}{\lambda-1}, \delta)$ -differential privacy for any $\delta \in (0, 1)$.*

Furthermore, RDP captures the privacy guarantee of Gaussian noise in a much cleaner and more accurate manner [101] compared to (ε, δ) -differential privacy.

Theorem 2.2.5 (RDP Guarantee of the Gaussian Mechanism). *Given a dataset $D \in \mathcal{X}^n$ and a query function $f : \mathcal{X}^n \rightarrow \mathbb{R}$ with $GS(f) = 1$, then the Gaussian Mechanism which outputs $f(D) + \mathcal{N}(0, \sigma)$ guarantees $(\lambda, \frac{\lambda}{2\sigma^2})$ -RDP for all $\lambda \geq 1$.*

2.3 Local Differential Privacy

Local differential privacy [144, 42, 77] is a stronger notion of privacy motivated by differential privacy [45]. Unlike differential privacy which assumes the existence of a trusted curator with direct access to the sensitive dataset, local differential privacy considers the situation in which individuals do not trust any curator. In this model, an untrusted algorithm is allowed to access a perturbed version of a sensitive dataset through a sanitization interface, and must use this perturbed data to perform some estimation. The amount of perturbation is controlled by ϵ , the privacy risk parameter. The strong guarantee offered by local differential privacy makes it an attractive choice in many industrial applications [56, 138, 39].

Definition 2.3.1 (Local Differential Privacy). *Let $D = (X_1, \dots, X_n)$ be a sensitive dataset where each $X_i \in \mathcal{D}$ corresponds to data about individual i . A randomized sanitization mechanism \mathcal{M} which outputs a disguised version (U_1, \dots, U_n) of D is said to provide ϵ -local differential privacy to individual i , if for all $x, x' \in \mathcal{D}$ and for all $S \subseteq \mathcal{S}$,*

$$P(U_i \in S | X_i = x) \leq e^\epsilon \cdot P(U_i \in S | X_i = x'), \quad (2.1)$$

where the probability is taken over the randomization in the sanitization mechanism.

As is in differential privacy, the parameter ϵ measures the privacy risk and smaller ϵ implies less privacy risk. Local differential privacy allows different ϵ value for different individual records in the dataset. We can use the Laplace mechanism to guarantee local differential privacy, where the global sensitivity measures how much U_i can change when the value of X_i changes.

2.4 Pufferfish Privacy

Pufferfish privacy [84] is a generalization of differential privacy [45] which captures the data correlation.

2.4.1 The Definition

A Pufferfish framework is instantiated by three parameters – a set \mathcal{S} of secrets, a set $\mathcal{Q} \subseteq \mathcal{S} \times \mathcal{S}$ of secret pairs, and a class of data distributions Θ . \mathcal{S} is the set of possible facts about the dataset that we might wish to hide, and could refer to a single individual’s private data or part thereof. \mathcal{Q} is the set of secret pairs that we wish to be indistinguishable. Finally, Θ is a set of distributions that can plausibly generate the data, and controls the amount and nature of correlation. Each $\theta \in \Theta$ represents a belief an adversary may hold about the data, and the goal of the privacy framework is to ensure indistinguishability in the face of these beliefs.

Definition 2.4.1 (Pufferfish Privacy). *A privacy mechanism \mathcal{M} is said to be ϵ -Pufferfish private in a framework $(\mathcal{S}, \mathcal{Q}, \Theta)$ if for all $\theta \in \Theta$ with X drawn from distribution θ , for all secret pairs $(s_i, s_j) \in \mathcal{Q}$, and for all $w \in \text{Range}(M)$, we have*

$$e^{-\epsilon} \leq \frac{P_{\mathcal{M},\theta}(\mathcal{M}(X) = w | s_i, \theta)}{P_{\mathcal{M},\theta}(\mathcal{M}(X) = w | s_j, \theta)} \leq e^{\epsilon} \quad (2.2)$$

when s_i and s_j are such that $P(s_i|\theta) \neq 0$, $P(s_j|\theta) \neq 0$.

Unlike differential privacy, the probability in (2.2) is with respect to the randomized mechanism *and* the actual data X , which is drawn from a $\theta \in \Theta$; to emphasize this, we use the notation X instead of D .

[84] shows that ϵ -differential privacy is a special case of Pufferfish, where \mathcal{S} is the set of all facts of the form $s_x^i = \text{Individual } i \text{ has value } x$ for $i \in \{1, \dots, n\}$ and x in a domain $[k]$, \mathcal{Q} is set of all pairs (s_x^i, s_z^i) for x and z in $[k]$ with $x \neq z$, and Θ is the set of all distributions where each individual is distributed independently. Moreover, we cannot have both privacy and utility when Θ is the set of all distributions [84]. Consequently, it is essential to select Θ wisely; if Θ is too restrictive, then we may not have privacy against legitimate adversaries, and if Θ is too large, then the resulting privacy mechanisms may have little utility.

2.4.2 Properties of Pufferfish Privacy

An alternative interpretation of (2.2) is that for $X \sim \theta$, $\theta \in \Theta$, for all $(s_i, s_j) \in \mathcal{Q}$, and for all $w \in \text{Range}(M)$, we have:

$$e^{-\epsilon} \cdot \frac{P(s_i|\theta)}{P(s_j|\theta)} \leq \frac{P(s_i|M(X) = w, \theta)}{P(s_j|M(X) = w, \theta)} \leq e^{\epsilon} \cdot \frac{P(s_i|\theta)}{P(s_j|\theta)}. \quad (2.3)$$

In other words, the knowledge of $M(X)$ does not affect the posterior ratio of the likelihood of s_i and s_j , compared to the initial belief.

Pufferfish privacy is immune to post-processing. However, it does not always compose [84] as in differential privacy. In Chapter 7, we will study the composition properties of Pufferfish under restricted conditions.

Chapter 3

Differentially Private Stochastic Gradient Descent

3.1 Overview

In the data-rich setting, at first blush it appears that learning algorithms can enjoy both low privacy risk and high utility. However, optimization methods for large data sets must also be scalable. Stochastic gradient descent (SGD) algorithms have received significant attention recently because they are simple and satisfy the same asymptotic guarantees as more computationally intensive learning methods [115, 127]. However, because these guarantees are asymptotic, to obtain reasonable performance on finite data sets practitioners must take care in setting parameters such as the learning rate (step size) for the updates. To alleviate some of this sensitivity and improve the performance of SGD in the finite sample setting, several works [31, 36, 137] have suggested grouping updates into “mini-batches.” This can improve the robustness of the updating at a moderate expense in terms of computation, but also introduces the batch size as a free parameter.

In this chapter, we derive differentially private versions of single-point SGD and mini-batch SGD and evaluate them on real and synthetic data sets. These algorithms work for gradient descent for general convex objectives – we illustrate the approach using logistic regression for classification. We demonstrate that differentially private single-point SGD has high variance, but a moderate increase in the batch size can improve the performance significantly. For low-

dimensional problems, the private algorithm’s performance is close to non-private SGD. However, we show that there is a limit to how much the batch size can help, and that the performance is dependent on the learning rate.

3.1.1 Related Work

The work most connected to the current chapter are those on differentially private classification [26, 123]. Duchi et al. proposed an SGD method for *local privacy* [42]. Stochastic gradient methods are an example of online learning methods. Another approach to differentially private online learning was proposed by Jain et al. [75]; however, their algorithm is more computationally intensive than ours. The PINQ [98] package uses a noisy sum operation to compute full noisy gradient steps for logistic regression [146]. There the goal was exchanging iterations for accuracy. The noisy perceptron method [17] also uses iterative noisy updates to learn a classifier. We focus here on the effect of step size and batch size for SGD methods, so we do not compare the performance for these specific classification methods.

3.2 Preliminaries

While the methods we propose work for general optimization methods, we will describe the problem in terms of a classification problem. There, the data are n labelled examples $(x_1, y_1), \dots, (x_n, y_n)$, where $x_i \in \mathbb{R}^d$, and $y_i \in \{-1, 1\}$. We assume that for all i , the norm $\|x_i\| \leq 1$. In linear classification, our goal is to find a hyperplane through the origin that largely separates the examples labeled 1 from those labeled -1 . The most popular method of training such a linear classifier based on labelled data is by solving a regularized convex optimization problem:

$$w^* = \operatorname{argmin}_{w \in \mathbb{R}^d} \frac{\lambda}{2} \|w\|^2 + \frac{1}{n} \sum_{i=1}^n \ell(w, x_i, y_i) \quad (3.1)$$

Here w is the normal vector to the hyperplane separator, and ℓ is a convex loss function. Popular choices for ℓ in the machine learning literature are the logistic loss $\ell(w, x, y) = \log(1 + e^{-yw^\top x})$,

which leads to Logistic Regression, and the hinge loss $\ell(w, x, y) = \max(0, 1 - yw^\top x)$, which leads to Support Vector Machines (SVMs).

SGD is an iterative algorithm for solving the regularized convex optimization problem in 3.1. SGD begins with an initial point w_0 , and at step t , updates the iterate as:

$$w_{t+1} = w_t - \eta_t(\lambda w_t + \nabla \ell(w_t, x_t, y_t)) \quad (3.2)$$

Here η_t is a learning rate, and the (sub)gradient $\nabla \ell(w_t, x_t, y_t)$ is computed based on a single example (x_t, y_t) .

In SGD with mini-batch updates, instead of a single example, the update at each step t is based on a small subset B_t of examples of size b . Specifically,

$$w_{t+1} = w_t - \eta_t \left(\lambda w_t + \frac{1}{b} \sum_{(x_i, y_i) \in B_t} \nabla \ell(w_t, x_i, y_i) \right) \quad (3.3)$$

Both of these methods are approximations of a full gradient update – if the point(s) at each time t are sampled uniformly from $\{1, 2, \dots, n\}$ then the expected gradient step at each iteration is equal to a gradient step on the full objective function in (3.1). More generally, we can consider general empirical risk minimization with convex loss functions. We study the L_2 -regularized objective because strong convexity allows favorable theoretical guarantees.

3.3 SGD with Differential Privacy

A differentially-private version of the SGD update can be written as:

$$w_{t+1} = w_t - \eta_t (\lambda w_t + \nabla \ell(w_t, x_t, y_t) + Z_t), \quad (3.4)$$

where each Z_t is a random noise vector in \mathbb{R}^d drawn independently from the density:

$$\rho(z) \propto e^{-(\epsilon/2)\|z\|} \quad (3.5)$$

A differentially-private version of the mini-batch update using a batch B_t of examples of size b can be written as:

$$w_{t+1} = w_t - \eta_t \left(\lambda w_t + \frac{1}{b} \sum_{(x_i, y_i) \in B_t} \nabla \ell(w_t, x_i, y_i) + \frac{1}{b} Z_t \right). \quad (3.6)$$

where Z_t is again drawn from the density in (3.5).

Theorem 3.3.1 shows that provided the initialization point w_0 is determined independent of the sensitive data, the batches B_t are disjoint, and the $\|\nabla \ell(w, x, y)\|$ is bounded for all w , these updates are ϵ -differentially private.

Theorem 3.3.1 (Privacy of SGD and Mini-Batch Updates). *Suppose we run SGD with mini-batch updates in (3.6) for T batches B_1, \dots, B_T . If the initialization point w_0 is chosen independent of the sensitive data, the batches B_t are disjoint, and if $\|\nabla \ell(w, x, y)\| \leq 1$ for all w , and all (x_i, y_i) , then SGD with mini-batch updates is ϵ -differentially private.*

The key idea of the proof is the observation that provided the conditions of the theorem hold, the global sensitivity of each update is $\frac{2\eta_t}{b}$. The proof now follows by combining this observation with results of Dwork et al. [45], and using the fact that the privacy guarantee does not degrade across batches as the samples used in the batches are disjoint.

Because we add noise at each iteration, the SGD procedure guarantees differential privacy in a “local” sense – each individual i may choose an ϵ_i and this method can guarantee differential privacy at different levels ϵ_i for different individuals by adjusting the distribution of Z_t . A slightly different notion of local privacy was also studied by Duchi et al. [42] in the statistical setting: there the algorithm can sample individuals from a distribution with unknown parameter and the goal is to estimate the parameter. At each time their algorithm can sample a new individual

and receives a noisy subgradient estimate. They use mirror descent to guarantee privacy under a variant of differential privacy based on a mutual information criterion.

3.4 Experiments

3.4.1 Datasets

We consider three classification tasks – one on a synthetic dataset and two on real data. Our synthetic dataset consists of $n = 10,000$ samples drawn uniformly from a 5-dimensional sphere, and is linearly separable with margin 0.001. For our first classification task on real data, we use the KDDCup99 dataset [119], an intrusion detection dataset on network connections. We address the normal vs. malicious classification task, and use a subsample of size 50,000. For our second task on real data, we address the “1 vs. all” classification task on the MNIST dataset [88], which consists of 60,000 training examples and 10,000 test examples of images of handwritten digits 0 to 9. In both cases, we preprocess the data by normalizing each feature, projecting each row to the unit ball, and then reducing the data dimension by random projections, which preserve differential privacy. We use a reduced dimension of $d = 9$ for KDDCup, and $d = 15$ for MNIST.

3.4.2 Procedure

We use SGD to train a logistic regression model. For each update, we use the mini-batch update from (3.6) for batch sizes $b \in \{1, 2, 5, 10, 20, 50\}$, with regularization parameter $\lambda = 0.0001$ and $\epsilon = 1$. In each case, we make a *single pass* over the entire training data. To maintain numerical stability, after each update, we project the iterate w_t onto a ball of radius $1/\lambda$. For each experiment we investigated a few different schemes for setting the learning rates. We averaged the objective function values obtained over 20 random permutations of the training data as well as fresh random samples of the noise Z_t for the private algorithm. The error bars are at a single standard deviation. Since we are interested in the optimization performance, we plot the objective function value – in future work we will also investigate the classification accuracy.

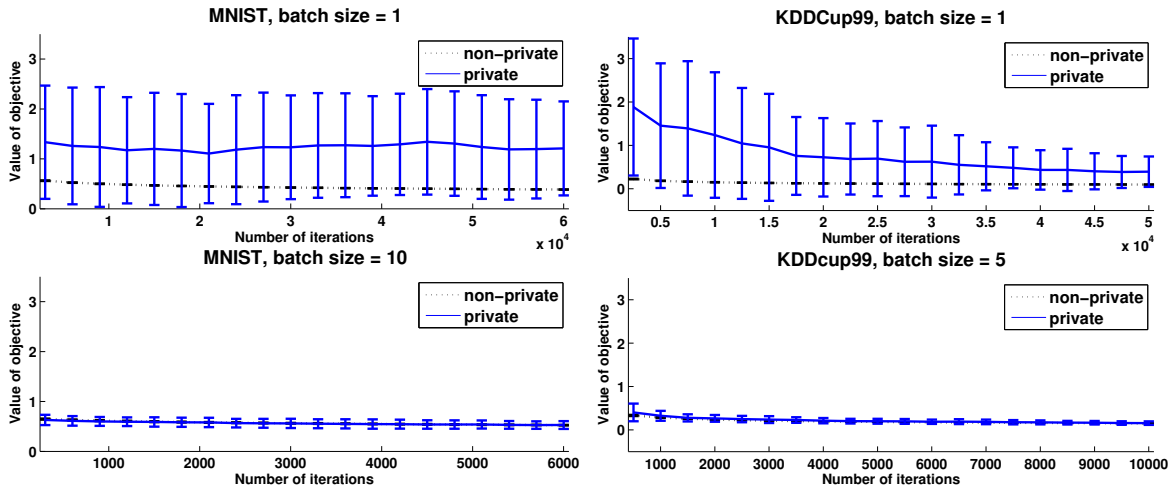


Figure 3.1. Objective function value vs. number of iterations for private and non-private algorithms on the MNIST and KDDCup99 data sets. Horizontal axis is scaled so that the number of samples is the same.

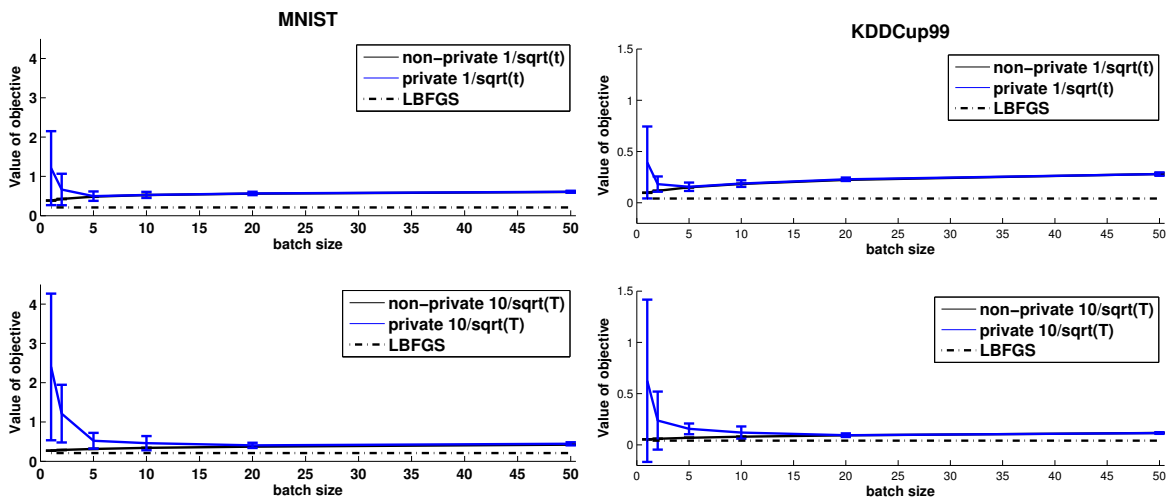


Figure 3.2. Objective function value vs. batch size b for private and non-private algorithms on MNIST and KDDCup99.

3.4.3 Mini-batching reduces variance

The first question we ask is how SGD would fare with batch size 1, since this is the case which has been most studied in the literature. The top half of Figure 3.1 shows the objective value for the MNIST data set versus the number of samples in the algorithm for learning rate $\eta_t = 1/\sqrt{t}$. For batch size $b = 1$, differentially private SGD is far from the non-private objective and furthermore has high variance. That is, the noise added in each iteration prevents the algorithm from converging. However, a modest batch size $b = 10$, as shown in the lower half of the figure, reduces the variance of differentially private SGD to the point of matching non-private SGD, even for a moderate number of data points.

The other plots in Figure 3.1 show that this behavior also holds for the KDDCup99 data set. Although the variance of the differentially private algorithm decreases slowly, choosing $b = 5$ makes the mini-batch SGD performance nearly identical to that of the non-private mini-batch SGD. What these two experiments indicate is that in terms of objective value, guaranteeing differential privacy can come for “free” using SGD with moderate batch size. We emphasize here that all of these examples are low-dimensional problems, and the privacy parameter $\epsilon = 1$. It is well-known that differentially private learning algorithms often have a sample complexity that scales linearly with the data dimension d and inversely with the privacy risk ϵ . Thus a moderate reduction in ϵ or increase in d may require more data. It would be interesting to see if increasing the batch size can still make private SGD match non-private SGD in these settings.

3.4.4 Choosing appropriate parameters

Our next experiment is to find the impact of batch size on the performance of these algorithms. Figure 3.2 shows the objective value as a function of batch size for private SGD, non-private SGD, and a centralized learning procedure which solves the optimization using all of the data points. In all cases, increasing the batch size improved the performance of private SGD, but there is a limit – for step size $1/\sqrt{t}$, much larger batch sizes actually degrade performance.

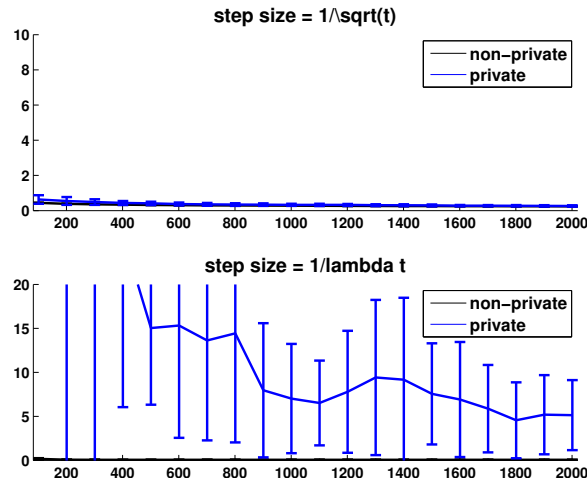


Figure 3.3. Objective function value vs. number of data points for private and non-private algorithms on synthetic data for batch size $b = 5$.

Because we choose to make a single pass over the data to limit the noise per iteration, increasing the batch size decreases the number of iterations, and therefore there is an optimal choice of b for each problem. With a larger learning rate $10/\sqrt{T}$, the performance for larger batches does not degrade as much, and the end value of the objective is closer to that of the centralized learning algorithm.

Many analyses of SGD in the strongly convex case suggest that a learning rate $\eta_t = 1/\lambda t$ guarantees fast convergence rates [115]. In our case λ is quite small, meaning the objective is not very strongly convex. To see the impact of the noise added for differential privacy, we simulated two learning rates, $1/\sqrt{t}$ and $1/\lambda t$, on the synthetic data with $b = 5$. The results in Figure 3.3 show that choosing a rapidly decreasing step size dramatically increases the variance of private SGD. In practice, choosing the step size in stochastic approximation schemes is often a matter of art, and differentially private noise complicates this choice.

3.5 Conclusions

We investigated how differential privacy affects mini-batched stochastic gradient descent (SGD). When data is plentiful, privacy is “affordable,” and SGD strategies are more computation-

ally efficient. We showed that in many cases the performance of differentially private SGD was close to that of non-private SGD, especially with larger batch sizes. In stochastic optimization, both the variability of the algorithm and the impact of privacy-preserving noise can be ameliorated by processing groups of points together. Our experiments show that privacy affects both the optimal batch size b and learning rate η_t . Some interesting future directions suggested by this chapter include: quantifying the impact of the dimension d and privacy parameter ϵ , allowing different ϵ_i 's for each point, and using multiple passes through the data to trade off iterations, total privacy loss (via composition results for differential privacy), and error. These modifications could make differentially private learning more effective in practical settings.

We note that the algorithm presented in this chapter is one of the early works on differentially private SGD. Some related work that appears later on includes [128], [1] and [147]. [128] proposed a distributed system that trains neural networks with differentially private SGD. The proposed algorithm selectively updates the parameters of the neural network and uses the sparse vector technique to improve the privacy-utility tradeoff. [1] proposed a differentially private SGD algorithm with additive Gaussian noise and evaluated it on neural networks, achieving comparable utility as non-private SGD. The privacy analysis is improved by keeping track of the privacy loss with moments accountant, a notion closely related to Rényi differential privacy. Both algorithms use gradient truncating to control the global sensitivity of the parameters. Additionally, [147] proposed an algorithm that injects random noise only at the end of the training process. The algorithm guarantees differential privacy for convex objective functions.

Acknowledgments

KC and SS would like to thank NIH U54-HL108460, the Hellman Foundation, and NSF IIS 1253942 for support.

This chapter is based on “Stochastic gradient descent with differentially private updates” by Shuang Song, Kamalika Chaudhuri and Anand Sarwate, which appears in the Global Confer-

ence on Signal and Information Processing (GlobalSIP), 2013 [130]. The dissertation author was the primary author of the paper.

Chapter 4

Local Differentially Private SGD with Heterogeneous Privacy Requirements

4.1 Overview

Modern large-scale machine learning systems often integrate sensitive data from several sources. In many cases, these sources provide data of a similar type (i.e. with the same features) but collected under different privacy requirements. For example, patient records from different studies of a particular drug may be combined to perform a more comprehensive analysis. These different studies, each conducted under different privacy regulations, require different levels of random noise for privacy-protection purposes, resulting in data of varying *quality*.

In this chapter, we adopt a model in which data is observed through heterogeneous noise, where the noise level reflects the quality of the data source. We study how to use stochastic gradient algorithms to learn from data of heterogeneous quality. In full generality, learning from heterogeneous data is essentially the problem of domain adaptation – a challenge for which good and complete solutions are difficult to obtain. Instead, we focus on the special case of heterogeneous noise and show how to use information about the data quality to improve the performance of learning algorithms which ignore this information.

The motivating example about combining patient records for a drug study can be formulated as locally differentially private learning from multiple sites. Under local differential privacy, the learner accesses the data via noisy estimates, where the noise guarantees privacy [42, 43]. In

many applications, we are required to learn from sensitive data collected from individuals with heterogeneous privacy preferences, or from multiple sites with different privacy requirements; this results in the heterogeneity of noise added to ensure privacy.

Moreover, our model of heterogeneous data does not merely suit locally differentially private machine learning; it captures a general scenario where data of different qualities are combined in a learning task. For example, a collection of images with annotations from experts as well as non-experts may be combined to learn a predictor – a situation which we call heterogeneous random classification noise (RCN). Under random classification noise [79], labels are randomly flipped before being presented to the algorithm. The heterogeneity in the noise addition comes from combining labels of variable quality – such as labels assigned by domain experts with those assigned by a crowd.

To our knowledge, [33] was the first to provide a theoretical study of how to learn classifiers from data of variable quality. In their formulation, like ours, data is observed through heterogeneous noise. Given data with known noise levels, their study focuses on finding an optimal ordering of the data and a stopping rule without any constraint on the computational complexity. We instead shift our attention to studying *computationally efficient strategies* for learning classifiers from data of variable quality.

We propose a model for variable data quality which is natural in the context of large-scale learning using stochastic gradient descent (SGD) and its variants [21, 13]. We assume that the training data are accessed through an oracle which provides an unbiased but noisy estimate of the gradient of the objective. The noise comes from two sources: the random sampling of a data point, and additional noise due to the data quality. Our two motivating applications – learning with local differential privacy and learning from data of variable quality – can both be modeled as solving a regularized convex optimization problem using SGD. Learning from data with heterogeneous noise in this framework thus reduces to running SGD with noisy gradient estimates, where the magnitude of the added noise varies across iterations.

Main results. We study noisy stochastic gradient methods when learning from multiple

data sets with different noise levels. For simplicity we consider the case where there are two data sets, which we call **Clean** and **Noisy**. We process these data sets sequentially using SGD with learning rate $O(1/t)$. We address some basic questions in this setup:

In what order should we process the data? Suppose we use standard SGD on the union of **Clean** and **Noisy**. We show theoretically and empirically that the order in which we should process the datasets to get good performance depends on the learning rate of the algorithm: in some cases we should use the order (Clean, Noisy) and in others (Noisy, Clean).

Can we use knowledge of the noise rates? We show that using separate learning rates that depend on the noise levels for the clean and noisy datasets improves the performance of SGD. We provide a heuristic for choosing these rates by optimizing an upper bound on the error for SGD that depends on the ratio of the noise levels. We analytically quantify the performance of our algorithm in two regimes of interest. For moderate noise levels, we demonstrate empirically that our algorithm outperforms using a single learning rate and using clean data only.

Does using noisy data always help? The work [33] suggests that if the noise level of noisy data is above some threshold, then noisy data will not help. Moreover, when the noise levels are very high, our heuristic does not always empirically outperform simply using the clean data. On the other hand, our theoretical results suggest that changing the learning rate can make noisy data useful. How do we resolve this apparent contradiction?

We perform an empirical study to address this question. Our experiments demonstrate that very often, there exists a learning rate at which noisy data helps; however, because the actual noise level may be far from the upper bound used in our algorithm, our optimization may not choose the best learning rate for every data set. We demonstrate that by adjusting the learning rate we can still take advantage of noisy data.

For simplicity we, like previous work [33], assume that the algorithms know the noise levels exactly. However, our algorithms can still be applied in the presence of approximate knowledge of the noise levels, and our result on the optimal data order only needs to know which dataset has more noise.

4.1.1 Related Work

There has been significant work on the convergence of SGD assuming analytic properties of the objective function, such as strong convexity and smoothness. When the objective function is λ -strongly convex, the learning rate used for SGD is $O(1/\lambda t)$ [106, 4, 115, 7], which leads to a regret of $O(1/\lambda^2 t)$ for smooth objectives. For non-smooth objectives, SGD with learning rate $O(1/\lambda t)$ followed by some form of averaging of the iterates achieves $O(1/\lambda t)$ [107, 105, 126, 148, 41].

There is also a body of literature on differentially private classification by regularized convex optimization in the batch [26, 122, 83] as well as the online [75] setting. In this chapter, we consider classification with *local differential privacy* [144, 42], a stronger form of privacy than ordinary differential privacy. [42] proposes learning a classifier with local differential privacy using SGD. Recent work [10] provides an improved privacy analysis for non-local privacy. This chapter is an extension of these papers to heterogeneous privacy requirements.

[33] studied classification when the labels in each data set are corrupted by RCN of different rates. Assuming the classifier minimizing the empirical 0/1 classification error can always be found, they propose a general theoretical procedure that processes the datasets in increasing order of noise, and determines when to stop using more data. In contrast, our noise model is more general and we provide a polynomial time algorithm for learning. Our results imply that in some cases the algorithm should process the noisy data first, and finally, our algorithm uses all the data.

4.2 The Model

Similar as in Chapter 3, we consider linear classification in the presence of noise. We are given T labelled examples $(x_1, y_1), \dots, (x_T, y_T)$, where $x_i \in \mathbb{R}^d$, and $y_i \in \{-1, 1\}$ and our goal is to find a hyperplane w that largely separates the examples labeled 1 from those labeled -1 . A

standard solution is via the following regularized convex optimization problem:

$$w^* = \operatorname{argmin}_{w \in \mathcal{W}} f(w) := \frac{\lambda}{2} \|w\|^2 + \frac{1}{T} \sum_{i=1}^T \ell(w, x_i, y_i). \quad (4.1)$$

Here ℓ is a convex loss function, and $\frac{\lambda}{2} \|w\|^2$ is a regularization term. Popular choices for ℓ include the logistic loss $\ell(w, x, y) = \log(1 + e^{-yw^\top x})$ and the hinge loss $\ell(w, x, y) = \max(0, 1 - yw^\top x)$.

Stochastic Gradient Descent (SGD) is a popular approach to solving (4.1): starting with an initial w_1 , at step t , SGD updates w_{t+1} using the point (x_t, y_t) as follows:

$$w_{t+1} = \Pi_{\mathcal{W}} (w_t - \eta_t (\lambda w_t + \nabla \ell(w_t, x_t, y_t))). \quad (4.2)$$

Here Π is a projection operator onto the convex feasible set \mathcal{W} , typically set to $\{w : \|w\|_2 \leq 1/\lambda\}$ and η_t is a learning rate (or step size) which specifies how fast w_t changes. A common choice for the learning rate for the case when $\lambda > 0$ is c/t , where $c = \Theta(1/\lambda)$.

4.2.1 The Heterogeneous Noise Model

We propose an abstract model for heterogeneous noise that can be specialized to two important scenarios: differentially private learning, and random classification noise. By heterogeneous noise we mean that the distribution of the noise can depend on the data points themselves. More formally, we assume that the learning algorithm may only access the labeled data through an oracle \mathcal{G} which, given a $w \in \mathbb{R}^d$, draws a fresh independent sample (x, y) from the underlying data distribution, and returns an unbiased noisy gradient of the objective function $\nabla f(w)$, based on the example (x, y) :

$$\mathbb{E} [\mathcal{G}(w)] = \lambda w + \nabla \ell(w, x, y), \quad \mathbb{E} [\|\mathcal{G}(w)\|^2] \leq \Gamma^2. \quad (4.3)$$

The precise manner in which $\mathcal{G}(w)$ is generated depends on the application. Define the *noise level* for the oracle \mathcal{G} as the constant Γ in (4.3); larger Γ means more noisy data. Finally, to

model finite training datasets, we assume that an oracle \mathcal{G} may be called only a limited number of times.

Observe that in this noise model, we can easily use the noisy gradient returned by \mathcal{G} to perform SGD. The update rule becomes:

$$w_{t+1} = \Pi_{\mathcal{W}}(w_t - \eta_t \mathcal{G}(w_t)). \quad (4.4)$$

The SGD estimate is w_{t+1} .

In practice, we can implement an oracle such as \mathcal{G} based on a finite labelled training set D as follows. We apply a random permutation on the samples in D , and at each invocation, compute a noisy gradient based on the next sample in the permutation. The number of calls to the oracle is limited to $|D|$. If the samples in D are drawn iid from the underlying data distribution, and if any extraneous noise added to the gradient at each iteration is unbiased and drawn independently, then this process will implement the oracle correctly.

To model heterogeneous noise, we assume that we have access to two oracles \mathcal{G}_1 and \mathcal{G}_2 implemented based on datasets D_1 and D_2 , which can be called at most $|D_1|$ and $|D_2|$ times respectively. For $j = 1, 2$, the noise level of oracle \mathcal{G}_j is Γ_j , and the values of Γ_1 and Γ_2 are known to the algorithm. In some practical situations, Γ_1 and Γ_2 will not be known exactly; however, our algorithm in Section 4.4 also applies when approximate noise levels are known, and our algorithm in Section 4.3 applies even when only the relative noise levels are known.

4.2.1.1 Local Differential Privacy

Consider learning a linear classifier from a sensitive labeled dataset while ensuring local privacy (see Chapter 2 for more details) of the participants. This problem can be expressed in our noise model by setting the sanitization mechanism as the oracle. Given a privacy risk ϵ , for $w \in \mathbb{R}^d$, the oracle \mathcal{G}^{DP} draws a random labeled sample (x, y) from the underlying data distribution, and returns the noisy gradient of the objective function at w computed based on

(x, y) as

$$\mathcal{G}^{\text{DP}}(w) = \lambda w + \nabla \ell(w, x, y) + Z, \quad (4.5)$$

where Z is independent random noise drawn from the density: $\rho(z) \propto e^{-(\epsilon/2)\|z\|}$.

[42] showed that this mechanism provides ϵ -local privacy assuming analytic conditions on the loss function, bounded data, and that the oracle generates a fresh random sample at each invocation. The following result shows how to set the parameters to fit in our heterogeneous noise model. The proof is provided in the supplement.

Theorem 4.2.1. *If $\|\nabla \ell(w, x, y)\| \leq 1$ for all w and (x, y) , then $\mathcal{G}^{\text{DP}}(w)$ is ϵ -local differentially private. Moreover, for any w such that $\|w\| \leq \frac{1}{\lambda}$, $\mathbb{E}[\mathcal{G}^{\text{DP}}(w)] = \lambda w + \nabla \mathbb{E}_{(x,y)}[\ell(w, x, y)]$, and*

$$\mathbb{E}[\|\mathcal{G}^{\text{DP}}(w)\|^2] \leq 4 + \frac{4(d^2 + d)}{\epsilon^2}.$$

In practice, we may wish to learn classifiers from multiple sensitive datasets with different privacy parameters. For example, suppose we wish to learn a classifier from sensitive patient records in two different hospitals holding data sets D_1 and D_2 , respectively. The hospitals have different privacy policies, and thus different privacy parameters ϵ_1 and ϵ_2 . This corresponds to a heterogeneous noise model in which we have two sanitizing oracles – $\mathcal{G}_1^{\text{DP}}$ and $\mathcal{G}_2^{\text{DP}}$. For $j = 1, 2$, $\mathcal{G}_j^{\text{DP}}$ implements a differentially private oracle with privacy parameter ϵ_j based on dataset D_j and may be called at most $|D_j|$ times.

4.2.1.2 Random Classification Noise

In the random classification noise model of [79], the learning algorithm is presented with labelled examples $(x_1, \tilde{y}_1), \dots, (x_T, \tilde{y}_T)$, where each $\tilde{y}_i \in \{-1, 1\}$ has been obtained by

independently flipping the *true label* y_i with some probability σ . [104] showed that solving

$$\operatorname{argmin}_w \frac{\lambda}{2} \|w\|^2 + \frac{1}{T} \sum_{i=1}^T \tilde{\ell}(w, x_i, \tilde{y}_i, \sigma) \quad (4.6)$$

yields a linear classifier from data with random classification noise, where $\tilde{\ell}$ is a surrogate loss function corresponding to a convex loss ℓ :

$$\tilde{\ell}(w, x, y, \sigma) = \frac{(1 - \sigma)\ell(w, x, y) - \sigma\ell(w, x, -y)}{1 - 2\sigma},$$

and σ is the probability that each label is flipped. This problem can be expressed in our noise model using an oracle \mathcal{G}^{RCN} which on input w draws a fresh labelled example (x, \tilde{y}) and returns

$$\mathcal{G}^{\text{RCN}}(w) = \lambda w + \nabla \tilde{\ell}(w, x, \tilde{y}, \sigma).$$

The SGD updates in (4.4) with respect to \mathcal{G}^{RCN} minimize (4.6). If $\|x\| \leq 1$ and $\|\nabla \ell(w, x, y)\| \leq 1$, we have $\mathbb{E}_{\mathcal{G}^{\text{RCN}}(w)} [=] \lambda w + \nabla \ell(w, x, y)$ and $\mathbb{E}_{\|\mathcal{G}^{\text{RCN}}(w)\|_2^2} [\leq] 3 + 1/(1 - 2\sigma)^2$, under the random classification noise assumption, so the oracle \mathcal{G}^{RCN} satisfies the conditions in (4.3) with $\Gamma^2 = 3 + 1/(1 - 2\sigma)^2$.

In practice, we may wish to learn classifiers from multiple datasets with different amounts of classification noise [33]; for example, we may have a small dataset D_1 labeled by domain experts, and a larger noisier dataset D_2 , labeled via crowdsourcing, with flip probabilities σ_1 and σ_2 . We model this scenario using two oracles $\mathcal{G}_1^{\text{RCN}}$ and $\mathcal{G}_2^{\text{RCN}}$. For $j = 1, 2$, oracle $\mathcal{G}_j^{\text{RCN}}$ is implemented based on D_j and flip probability σ_j , and may be called at most $|D_j|$ times.

4.3 Data order depends on learning rate

Suppose we have two oracles \mathcal{G}_C (for “clean”) and \mathcal{G}_N (for “noisy”) implemented based on datasets D_C, D_N with noise levels Γ_C, Γ_N (where $\Gamma_C < \Gamma_N$) respectively. In which order

should we query the oracle when using SGD? Perhaps surprisingly, it turns out that the answer depends on the learning rate. Below, we show a specific example of a convex optimization problem such that with $\eta_t = c/t$, the optimal ordering is to use \mathcal{G}_C first when $c \in (0, 1/\lambda)$, and the optimal ordering is to use \mathcal{G}_N first when $c > 1/\lambda$.

Let $|D_C| + |D_N| = T$ and consider the convex optimization problem:

$$\min_{w \in \mathcal{W}} \frac{\lambda}{2} \|w\|^2 - \frac{1}{T} \sum_{i=1}^T y_i w^\top x_i, \quad (4.7)$$

where the points $\{(x_i, y_i)\}$ are drawn from the underlying distribution by \mathcal{G}_C or \mathcal{G}_N . Suppose $\mathcal{G}(w) = \lambda w - yx + Z$ where Z is an independent noise vector such that $\mathbb{E}[Z] = 0$, $\mathbb{E}[\|Z\|^2] = V_C^2$ if \mathcal{G} is \mathcal{G}_C , and $\mathbb{E}[\|Z\|^2] = V_N^2$ if \mathcal{G} is \mathcal{G}_N with $V_N^2 \geq V_C^2$.

For our example, we consider the following three variants of SGD: CF and NF for “clean first” and “noisy first” and AO for an “arbitrary ordering”:

1. CF: For $t \leq |D_C|$, query \mathcal{G}_C in the SGD update (4.4). For $t > |D_C|$, query \mathcal{G}_N .
2. NF: For $t \leq |D_N|$, query \mathcal{G}_N in the SGD update (4.4). For $t > |D_N|$, query \mathcal{G}_C .
3. AO: Let S be an arbitrary sequence of length T consisting of $|D_C|$ C’s and $|D_N|$ N’s. In the SGD update (4.4) in round t , if the t -th element S_t of S is C, then query \mathcal{G}_C ; else, query \mathcal{G}_N .

In order to isolate the effect of the noise, we consider two additional oracles \mathcal{G}'_C and \mathcal{G}'_N ; the oracle \mathcal{G}'_C (resp. \mathcal{G}'_N) is implemented based on the dataset D_C (resp. D_N), and iterates over D_C (resp. D_N) in exactly the same order as \mathcal{G}_C (resp. \mathcal{G}_N); the only difference is that for \mathcal{G}'_C (resp. \mathcal{G}'_N), no extra noise is added to the gradient (that is, $Z = 0$). The main result of this section is stated in Theorem 4.3.1.

Theorem 4.3.1. *Let $\{w_t^{\text{CF}}\}$, $\{w_t^{\text{NF}}\}$ and $\{w_t^{\text{AO}}\}$ be the sequences of updates obtained by running SGD for objective function (4.7) under CF, NF and AO respectively, and let $\{v_t^{\text{CF}}\}$, $\{v_t^{\text{NF}}\}$ and*

$\{v_t^{\text{AO}}\}$ be the sequences of updates under CF, NF and AO with calls to \mathcal{G}_C and \mathcal{G}_N replaced by calls to \mathcal{G}'_C and \mathcal{G}'_N . Let $T = |D_C| + |D_N|$.

1. If the learning rate $\eta_t = c/t$ where $c \in (0, \frac{1}{\lambda})$, then $\mathbb{E}_{\|v_{T+1}^{\text{CF}} - w_{T+1}^{\text{CF}}\|^2} [\leq] \mathbb{E}_{\|v_{T+1}^{\text{AO}} - w_{T+1}^{\text{AO}}\|^2} [\cdot]$
2. If the learning rate $\eta_t = c/t$ where $c > 1/\lambda$, then $\mathbb{E}_{\|v_{T+1}^{\text{NF}} - w_{T+1}^{\text{NF}}\|^2} [\leq] \mathbb{E}_{\|v_{T+1}^{\text{AO}} - w_{T+1}^{\text{AO}}\|^2} [\cdot]$

This means arbitrary ordering is worse than sequentially processing one dataset after the other except when $c = 1/\lambda$. If the learning rate is small, then SGD should use the clean data first to aggressively proceed towards the optimum. If the learning rate is larger, then SGD should reserve the clean data for refining the initial estimates given by processing the noisy data.

4.4 Adapting the learning rate to the noise level

We now investigate whether the performance of SGD can be improved by using different learning rates for oracles with different noise levels. Suppose we have oracles \mathcal{G}_1 and \mathcal{G}_2 with noise levels Γ_1 and Γ_2 that are implemented based on two datasets D_1 and D_2 . Unlike the previous section, we do not assume any relation between Γ_1 and Γ_2 – we analyze the error for using oracle \mathcal{G}_1 followed by \mathcal{G}_2 in terms of Γ_1 and Γ_2 to choose a data order. Let $T = |D_1| + |D_2|$. Let $\beta_1 = \frac{|D_1|}{T}$ and $\beta_2 = 1 - \beta_1 = \frac{|D_2|}{T}$ be the fraction of the data coming from \mathcal{G}_1 and \mathcal{G}_2 , respectively. We adapt the gradient updates in (4.4) to heterogeneous noise by choosing the learning rate η_t as a function of the noise level. Algorithm 1 shows a modified SGD for heterogeneous learning rates.

Algorithm 1. SGD with varying learning rate

- 1: **Inputs:** Oracles $\mathcal{G}_1, \mathcal{G}_2$ implemented by data sets D_1, D_2 . Learning rates c_1 and c_2 .
 - 2: Set $w_1 = 0$.
 - 3: **for** $t = 1, 2, \dots, |D_1|$ **do**
 - 4: $w_{t+1} = \Pi_{\mathcal{W}}(w_t - \frac{c_1}{t} \mathcal{G}_1(w_t))$
 - 5: **end for**
 - 6: **for** $t = |D_1| + 1, |D_1| + 2, \dots, |D_1| + |D_2|$ **do**
 - 7: $w_{t+1} = \Pi_{\mathcal{W}}(w_t - \frac{c_2}{t} \mathcal{G}_2(w_t))$
 - 8: **end for return** $w_{|D_1| + |D_2| + 1}$.
-

Consider SGD with learning rate $\eta_t = c_1/t$ while querying \mathcal{G}_1 and with $\eta_t = c_2/t$ while querying \mathcal{G}_2 in the update (4.4). We must choose an order in which to query \mathcal{G}_1 and \mathcal{G}_2 as well as the constants c_1 and c_2 to get the best performance. We do this by minimizing an upper bound on the distance between the final iterate w_{T+1} and the optimal solution w^* to $\mathbb{E}[f(w)]$ where f is defined in (4.1), and the expectation is with respect to the data distribution and the gradient noise; the upper bound we choose is based on [115]. Note that for smooth functions f , a bound on the distance $\|w_{T+1} - w^*\|$ automatically translates to a bound on the regret $f(w_{T+1}) - f(w^*)$.

Theorem 4.4.1 generalizes the results of [115] to our heterogeneous noise setting; the proof is in the supplement.

Theorem 4.4.1. *If $2\lambda c_1 > 1$ and if $2\lambda c_2 \neq 1$, and if we query \mathcal{G}_1 before \mathcal{G}_2 with learning rates c_1/t and c_2/t respectively, then the SGD algorithm satisfies*

$$\mathbb{E} [\|w_{T+1} - w^*\|^2] \leq \frac{4\Gamma_1^2}{T} \cdot \frac{\beta_1^{2\lambda c_2 - 1} c_1^2}{2\lambda c_1 - 1} + \frac{4\Gamma_2^2}{T} \cdot \frac{(1 - \beta_1^{2\lambda c_2 - 1}) c_2^2}{2\lambda c_2 - 1} + O\left(\frac{1}{T^{\min(2, 2\lambda c_1)}}\right). \quad (4.8)$$

Two remarks are in order. First, the first two terms in the right hand side dominate the other term. Second, our proof techniques for Theorem 4.4.1, adapted from [115], require that $2\lambda c_1 > 1$ in order to get a $O(1/T)$ rate of convergence; without this condition, the dependence on T is $\Omega(1/T)$.

4.4.1 Algorithm description

Our algorithm for selecting c_1 and c_2 is motivated by Theorem 4.4.1. We propose an algorithm that selects c_1 and c_2 by minimizing the quantity $B(c_1, c_2)$ which represents the highest order terms in Theorem 4.4.1:

$$B(c_1, c_2) = \frac{4\Gamma_1^2 \beta_1^{2\lambda c_2 - 1} c_1^2}{T(2\lambda c_1 - 1)} + \frac{4\Gamma_2^2 (1 - \beta_1^{2\lambda c_2 - 1}) c_2^2}{T(2\lambda c_2 - 1)}. \quad (4.9)$$

Given $\lambda, \Gamma_1, \Gamma_2$ and β_1 , we use c_1^* and c_2^* to denote the values of c_1 and c_2 that minimize $B(c_1, c_2)$. We can optimize for fixed c_2 with respect to c_1 by minimizing $\frac{c_1^2}{2\lambda c_1 - 1}$; this gives $c_1^* = 1/\lambda$, and $\frac{c_1^{*2}}{2\lambda c_1^* - 1} = 1/\lambda^2$, which is independent of β_1 or the noise levels Γ_1 and Γ_2 . Minimizing $B(c_1^*, c_2)$ with respect to c_2 can be now performed numerically to yield $c_2^* = \operatorname{argmin}_{c_2} B(c_1^*, c_2)$. This yields optimal values of c_1 and c_2 .

Now suppose we have two oracles $\mathcal{G}_C, \mathcal{G}_N$ with noise levels Γ_C and Γ_N that are implemented based on datasets D_C and D_N respectively. Let $\Gamma_C < \Gamma_N$, and let $\beta_C = \frac{|D_C|}{|D_C|+|D_N|}$ and $\beta_N = \frac{|D_N|}{|D_C|+|D_N|}$ be the fraction of the total data in each data set. Define the following functions:

$$H_{CN}(c) = \frac{4\Gamma_C^2 \beta_C^{2\lambda c - 1}}{\lambda^2} + \frac{4\Gamma_N^2 (1 - \beta_C^{2\lambda c - 1}) c^2}{2\lambda c - 1},$$

$$H_{NC}(c) = \frac{4\Gamma_N^2 \beta_N^{2\lambda c - 1}}{\lambda^2} + \frac{4\Gamma_C^2 (1 - \beta_N^{2\lambda c - 1}) c^2}{2\lambda c - 1}.$$

These represent the constant of the leading term in the upper bound in Theorem 4.4.1 for $(\mathcal{G}_1, \mathcal{G}_2) = (\mathcal{G}_C, \mathcal{G}_N)$ and $(\mathcal{G}_1, \mathcal{G}_2) = (\mathcal{G}_N, \mathcal{G}_C)$, respectively.

Algorithm 2 repeats the process of choosing optimal c_1, c_2 with two orderings of the data – \mathcal{G}_C first and \mathcal{G}_N first – and selects the solution which provides the best bounds (according to the higher order terms of Theorem 4.4.1).

Algorithm 2. Selecting the Learning Rates

- 1: **Inputs:** Data sets D_C and D_N accessed through oracles \mathcal{G}_C and \mathcal{G}_N with noise levels Γ_C and Γ_N .
 - 2: Let $\beta_C = \frac{|D_C|}{|D_C|+|D_N|}$ and $\beta_N = \frac{|D_N|}{|D_C|+|D_N|}$.
 - 3: Calculate $c_{CN} = \operatorname{argmin}_c H_{CN}(c)$ and $c_{NC} = \operatorname{argmin}_c H_{NC}(c)$.
 - 4: **if** $H_{CN}(c_{CN}) \leq H_{NC}(c_{NC})$ **then**
 - 5: Run Algorithm 1 using oracles $(\mathcal{G}_C, \mathcal{G}_N)$, learning rates $c_1 = \frac{1}{\lambda}$ and $c_2 = c_{CN}$.
 - 6: **else**
 - 7: Run Algorithm 1 using oracles $(\mathcal{G}_N, \mathcal{G}_C)$, learning rates $c_1 = \frac{1}{\lambda}$ and $c_2 = c_{NC}$.
 - 8: **end if**
-

4.4.2 Regret Bounds

To provide a regret bound on the performance of SGD with two learning rates, we need to plug the optimal values of c_1 and c_2 into the right hand side of (4.9). Observe that as $c_1 = c_2$ and $c_2 = 0$ are feasible inputs to (4.9), our algorithm by construction has a superior regret bound than using a single learning rate only, or using clean data only.

Unfortunately, the value of c_2 that minimizes (4.9) does not have a closed form solution, and as such it is difficult to provide a general simplified regret bound that holds for all Γ_1, Γ_2 and β_1 . In this section, we consider two cases of interest, and derive simplified versions of the regret bound for SGD with two learning rates for these cases.

We consider the two data orders $(\Gamma_1, \Gamma_2) = (\Gamma_N, \Gamma_C)$ and $(\Gamma_1, \Gamma_2) = (\Gamma_C, \Gamma_N)$ in a scenario where $\Gamma_N/\Gamma_C \gg 1$ and both β_N and β_C are bounded away from 0 and 1. That is, the noisy data is much noisier. The following two lemmas provide upper and lower bounds on $B(c_1^*, c_2^*)$ in this setting.

Lemma 4.4.2. *Suppose $(\Gamma_1, \Gamma_2) = (\Gamma_N, \Gamma_C)$ and $0 < \beta_N < 1$. Then for sufficiently large Γ_N/Γ_C , the optimal solution c_2^* to (4.9) satisfies*

$$2c_2^*\lambda \in \left[1 + \frac{2\log(\Gamma_N/\Gamma_C) + \log\log(1/\beta_N)}{\log(1/\beta_N)}, 1 + \frac{2\log(4\Gamma_N/\Gamma_C) + \log\log(1/\beta_N)}{\log(1/\beta_N)} \right].$$

Moreover, $B(c_1^*, c_2^*)$ satisfies:

$$B(c_1^*, c_2^*) \geq \frac{4\Gamma_C^2(\log(\frac{\Gamma_N}{\Gamma_C}) + \frac{1}{2}\log\log\frac{1}{\beta_N})}{\lambda^2 T \log(\frac{1}{\beta_N})}$$

$$B(c_1^*, c_2^*) \leq \frac{4\Gamma_C^2}{\lambda^2 T} \left(4 + \frac{4 + 2\log(\frac{\Gamma_N}{\Gamma_C}) + \log\log(\frac{1}{\beta_N})}{\log(\frac{1}{\beta_N})} \right).$$

Observe that the regret bound grows logarithmically with Γ_N/Γ_C . Moreover, if we only used the cleaner data, then the regret bound would be $\frac{4\Gamma_C^2}{\lambda^2\beta_C T}$, which is better, especially for large Γ_N/Γ_C . This means that using two learning rates with the noisy data first gives poor results at

high noise levels.

Our second bound takes the opposite data order, processing the clean data first.

Lemma 4.4.3. *Suppose $(\Gamma_1, \Gamma_2) = (\Gamma_C, \Gamma_N)$ and $0 < \beta_C < 1$. Let $\sigma = (\Gamma_N/\Gamma_C)^{-2}$. Then for sufficiently large Γ_N/Γ_C , the optimal solution c_2^* to (4.9) satisfies: $2c_2^*\lambda \in \left[\sigma, \frac{8}{\beta_C}\sigma\right]$. Moreover, $B(c_1^*, c_2^*)$ satisfies:*

$$B(c_1^*, c_2^*) \geq \frac{4\Gamma_C^2}{\lambda^2\beta_C T} \beta_C^{8\sigma/\beta_C}$$

$$B(c_1^*, c_2^*) \leq \frac{4\Gamma_C^2}{\lambda^2\beta_C T} \beta_C^\sigma \left(1 + \sigma \frac{\log(1/\beta_C)}{4}\right).$$

If we only used the clean dataset, then the regret bound would be $\frac{4\Gamma_C^2}{\lambda^2\beta_C T}$, so Lemma 4.4.3 yields an improvement by a factor of $\beta_C^{(\Gamma_N/\Gamma_C)^{-2}} \left(1 + \left(\frac{\Gamma_N}{\Gamma_C}\right)^{-2} \frac{\log(1/\beta_C)}{4}\right)$. As $\beta_C < 1$, observe that this factor is always less than 1, and tends to 1 as Γ_N/Γ_C tends to infinity; therefore the difference between the regret bounds narrows as the noisy data grows noisier. We conclude that using two learning rates with clean data first gives a better regret bound than using only clean data or using two learning rates with noisy data first.

4.5 Experiments

We next illustrate our theoretical results through experiments on real data. We consider the task of training a regularized logistic regression classifier for binary classification under local differential privacy. For our experiments, we consider two real datasets – MNIST (with the task 1 vs. Rest) and `Coverttype` (Type 2 vs. Rest). The former consists of 60,000 samples in 784 dimensions, while the latter consists of 500,000 samples in 54-dimensions. We reduce the dimension of the MNIST dataset to 25 via random projections.

To investigate the effect of heterogeneous noise, we divide the training data into subsets (D_C, D_N) to be accessed through oracles $(\mathcal{G}_C, \mathcal{G}_N)$ with privacy parameters (ϵ_C, ϵ_N) respectively. We pick $\epsilon_C > \epsilon_N$, so \mathcal{G}_N is noisier than \mathcal{G}_C . To simulate typical practical situations where cleaner

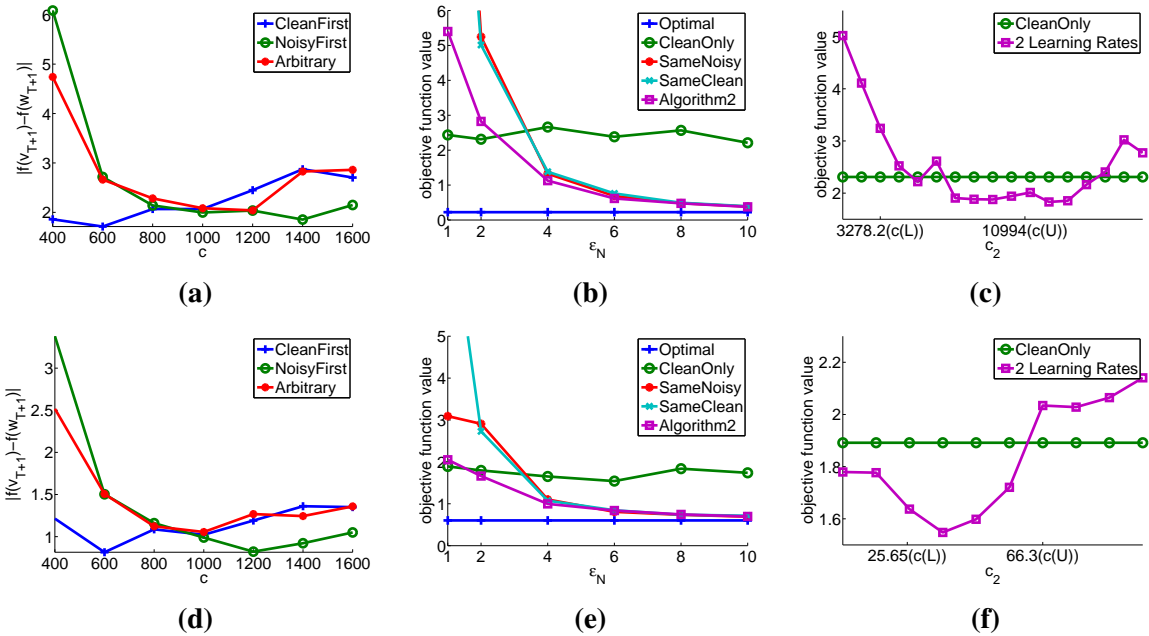


Figure 4.1. Column 1 plots $|f(w_{T+1}) - f(v_{T+1})|$ vs. constant c for $\lambda = 0.001$. Column 2 plots final objective function value vs. ϵ_N for $\epsilon_C = 10$. Column 3 plots final objective function value vs. c_2 for $\epsilon_N = 2$ (top) and $\epsilon_N = 1$ (bottom). Top row shows figures for MNIST and bottom row for Coverttype.

data is rare, we set the size of D_C to be $\beta_C = 10\%$ of the total data size. We set the regularization parameter $\lambda = 10^{-3}$, Γ_C and Γ_N according to Theorem 4.2.1 and use SGD with mini-batching (batch size 50).

Does Data Order Change Performance? Our first task is to investigate the effect of data order on performance. For this purpose, we compare three methods – CleanFirst, where all of D_C is used before D_N , NoisyFirst, where all of D_N is used before D_C , and Arbitrary, where data from $D_N \cup D_C$ is presented to the algorithm in a random order.

The results are in Figures 4.1a and 4.1d. We use $\epsilon_C = 10$, $\epsilon_N = 3$. For each algorithm, we plot $|f(w_{T+1}) - f(v_{T+1})|$ as a function of the constant c in the learning rate. Here $f(w_{T+1})$ is the function value obtained after T rounds of SGD, and $f(v_{T+1})$ is the function value obtained after T rounds of SGD if we iterate over the data in the same order, but add no extra noise to the gradient. (See Theorem 4.3.1 for more details.) As predicted by Theorem 4.3.1, the results show that for $c < \frac{1}{\lambda}$, CleanFirst has the best performance, while for $c > \frac{1}{\lambda}$, NoisyFirst performs best.

Arbitrary performs close to NoisyFirst for a range of values of c , which we expect as only 10% of the data belongs to D_C .

Are Two Learning Rates Better than One? We next investigate whether using two learning rates in SGD can improve performance. We compare five approaches. Optimal is the gold standard where we access the raw data without any intervening noisy oracle. CleanOnly uses only D_C with learning rate with the optimal value of c obtained from Section 4.4. SameClean and SameNoisy use a single value of the constant c in the learning rate for $D_N \cup D_C$, where c is obtained by optimizing (4.9)¹ under the constraint that $c_1 = c_2$. SameClean uses all of D_C before using D_N , while SameNoisy uses all of D_N before using D_C . In Algorithm2, we use Algorithm 2 to set the two learning rates and the data order (D_C first or D_N first). In each case, we set $\epsilon_C = 10$, vary ϵ_N from 1 to 10, and plot the function value obtained at the end of the optimization.

The results are plotted in Figures 4.1b and 4.1e. Each plotted point is an average of 100 runs. It is clear that Algorithm2, which uses two learning rates, performs better than both SameNoisy and SameClean. As expected, the performance difference diminishes as ϵ_N increases (that is, the noisy data gets cleaner). For moderate and high ϵ_N , Algorithm2 performs best, while for low ϵ_N (very noisy D_N), CleanOnly has slightly better performance. We therefore conclude that using two learning rates is better than using a single learning rate with both datasets, and that Algorithm2 performs best for moderate to low noise levels.

Does Noisy Data Always Help? A natural question to ask is whether using noisy data always helps performance, or if there is some threshold noise level beyond which we should not use noisy data. Lemma 4.4.3 shows that in theory, we obtain a better upper bound on performance when we use noisy data; in contrast, Figures 4.1b and 4.1e show that for low ϵ_N (high noise), Algorithm2 performs worse than CleanOnly. How do we explain this apparent contradiction?

To understand this effect, in Figures 4.1c and 4.1f we plot the performance of SGD using two learning rates (with $c_1 = \frac{1}{\lambda}$) against CleanOnly as a function of the second learning rate c_2 . The figures show that the best performance is attained at a value of c_2 which is different from

¹Note that we plug in separate noise rates for \mathcal{G}_C and \mathcal{G}_N in the learning rate calculations.

the value predicted by Algorithm2, and *this best performance is better than CleanOnly*. Thus, noisy data always improves performance; however, the improvement may not be achieved at the learning rate predicted by our algorithm.

Why does our algorithm perform suboptimally? We believe this happens because the values of Γ_N and Γ_C used by our algorithm are fairly loose upper bounds. For local differential privacy, an easy lower bound on Γ is $\sqrt{\frac{4(d^2+d)}{\epsilon^2 b}}$, where b is the mini-batch size; let $c_2(L)$ (resp. $c_2(U)$) be the value of c_2 obtained by plugging in these lower bounds (resp. upper bounds from Theorem 4.2.1) to Algorithm 1. Our experiments show that the optimal value of c_2 always lies between $c_2(L)$ and $c_2(U)$, which indicates that the suboptimal performance may be due to the looseness in the bounds.

We thus find that even in these high noise cases, theoretical analysis often allows us to identify *an interval* containing the optimal value of c_2 . In practice, we recommend running Algorithm 2 twice – once with upper, and once with lower bounds to obtain an interval containing c_2 , and then performing a line search to find the optimal c_2 .

4.6 Conclusion

We propose a model for learning from heterogeneous noise that is appropriate for studying stochastic gradient approaches to learning. In our model, data from different sites are accessed through different oracles which provide noisy versions of the gradient. Learning under local differential privacy and random classification noise are both instances of our model. We show that for two sites with different noise levels, processing data from one site followed by the other is better than randomly sampling the data, and the optimal data order depends on the learning rate. We then provide a method for choosing learning rates that depends on the noise levels and showed that these choices achieve lower regret than using a common learning rate. We validate these findings through experiments on two standard data sets and show that our method for choosing learning rates often yields improvements when the noise levels are moderate. In the

case where one data set is much noisier than the other, we provide a different heuristic to choose a learning rate that improves the regret.

There are several different directions towards generalizing the algorithms here. Firstly, extending the results to multiple sites and multiple noise levels will give more insights as to how to leverage large numbers of data sources. This leads naturally to cost and budgeting questions: how much should we pay for additional noisy data? Our results for data order do not depend on the actual noise levels, but rather their relative level. However, we use the noise levels to tune the learning rates for different sites. If bounds on the noise levels are available, we can still apply our heuristic. Adaptive approaches for estimating the noise levels while learning are also an interesting approach for future study.

Acknowledgments

The work of K. Chaudhuri and S. Song was sponsored by NIH under U54 HL108460 and the NSF under IIS 1253942.

Chapter 4 is based on “Learning from Data with Heterogenous Noise using SGD” by Shuang Song, Kamalika Chaudhuri and Anand Sarwate, which appears in the International Conference on Artificial Intelligence and Statistics (AISTATS) 2015 [131]. The dissertation author was the primary author of the paper.

Chapter 5

Scalable Private Learning with PATE

5.1 Overview

Recently, two promising, new model-training approaches have offered the hope that practical, high-utility machine learning may be compatible with strong privacy-protection guarantees for sensitive training data [2]. This chapter revisits one of these approaches, *Private Aggregation of Teacher Ensembles*, or PATE [111], and develops techniques that improve its scalability and practical applicability. PATE has the advantage of being able to learn from the aggregated consensus of separate “teacher” models trained on disjoint data, in a manner that both provides intuitive privacy guarantees and is agnostic to the underlying machine-learning techniques (cf. the approach of differentially-private stochastic gradient descent [1]). In the PATE approach multiple teachers are trained on disjoint sensitive data (e.g., different users’ data), and uses the teachers’ aggregate consensus answers in a black-box fashion to supervise the training of a “student” model. By publishing only the student model (keeping the teachers private) and by adding carefully-calibrated Laplace noise to the aggregate answers used to train the student, the original PATE work showed how to establish rigorous (ϵ, δ) differential-privacy guarantees [111]. However, to date, PATE has been applied to only simple tasks, like MNIST, without any realistic, larger-scale evaluation.

The techniques presented in this chapter allow PATE to be applied on a larger scale to build more accurate models, in a manner that improves both on PATE’s intuitive privacy-

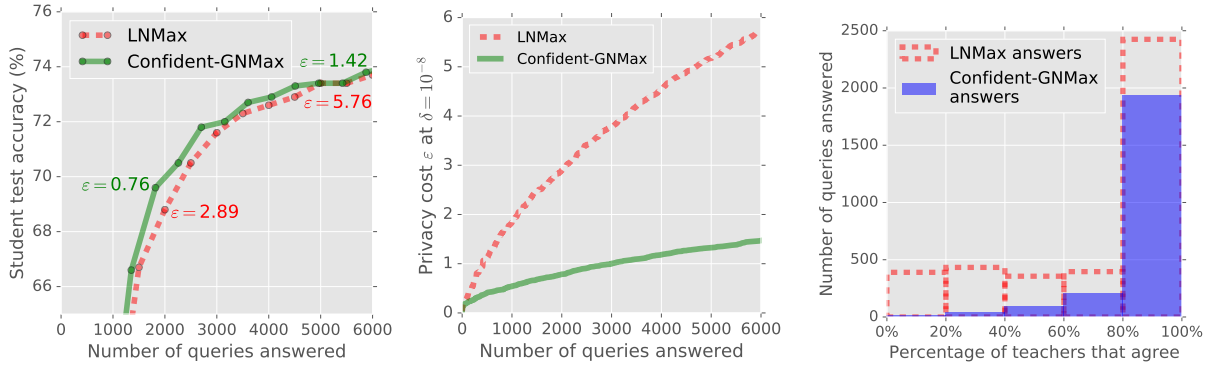


Figure 5.1. Our proposed technique, Confident-GNMax, improves on the original PATE (LNMax) on all measures. *Left:* Accuracy is higher. *Middle:* Privacy cost is quartered. *Right:* Intuitive privacy is improved. These are results for a character-recognition task.

protection due to the teachers’ independent consensus as well as its differential-privacy guarantees. As shown in our experiments, the result is a gain in privacy, utility, and practicality – an uncommon joint improvement.

The primary technical contributions of this chapter are new mechanisms for aggregating teachers’ answers that are more selective and add less noise. On all measures, our techniques improve on the original PATE mechanism when evaluated on the same tasks using the same datasets, as described in Section 5.4. Furthermore, we evaluate both variants of PATE on a new, large-scale character recognition task with 150 output classes, inspired by MNIST. The results show that PATE can be successfully utilized even to uncurated datasets – with significant class imbalance as well as erroneous class labels – and that our new aggregation mechanisms improve both privacy and model accuracy.

To be more selective, our new mechanisms leverage some pleasant synergies between privacy and utility in PATE aggregation. For example, when teachers disagree, and there is no real consensus, the privacy cost is much higher; however, since such disagreement also suggest that the teachers may not give a correct answer, the answer may simply be omitted. Similarly, teachers may avoid giving an answer where the student already is confidently predicting the right answer. Additionally, we ensure that these selection steps are themselves done in a private manner.

To add less noise, our new PATE aggregation mechanisms sample Gaussian noise, since the tails of that distribution diminish far more rapidly than those of the Laplace noise used in the original PATE work. This reduction greatly increases the chance that the noisy aggregation of teachers’ votes results in the correct consensus answer, which is especially important when PATE is scaled to learning tasks with large numbers of output classes. However, changing the sampled noise requires redoing the entire PATE privacy analysis from scratch (see the full paper [112] for more details).

Finally, of independent interest are the details of our evaluation extending that of the original PATE work. In particular, we find that the virtual adversarial training (VAT) technique of [102] is a good basis for semi-supervised learning on tasks with many classes, outperforming the improved GANs by [124] used in the original PATE work. Furthermore, we explain how to tune the PATE approach to achieve very strong privacy ($\epsilon \approx 1.0$) along with high utility, for our real-world character recognition learning task.

This chapter is structured as follows: Section 5.1.1 is the related work section; Section 5.2 gives a background on PATE; Section 5.3 describes our improved aggregation mechanisms; Section 5.4 details our experimental evaluation; Section 5.5 offers conclusions; and proofs can be found in the full paper [112].

5.1.1 Related Work

The first learning algorithms adapted to provide differential privacy with respect to their training data were often linear and convex [113, 26, 130, 12, 68]. More recently, successful developments in deep learning called for differentially private stochastic gradient descent algorithms [1], some of which have been tailored to learn in federated [96] settings.

Differentially private selection mechanisms like GNMax (Section 5.3.1) are commonly used in hypothesis testing, frequent itemset mining, and as building blocks of more complicated private mechanisms. The most commonly used differentially private selection mechanisms are exponential mechanism [100] and LNMax [14]. Recent works offer lower bounds on sample

complexity of such problem [133, 8].

The Confident and Interactive Aggregator proposed in this chapter (Section 5.3.2 and Section 5.3.3 resp.) use the intuition that selecting samples under certain constraints could result in better training than using samples uniformly at random. In machine learning theory, active learning [30] has been shown to allow learning from fewer labeled examples than the passive case (see e.g. [69]). Similarly, in model stealing [139], a goal is to learn a model from limited access to a teacher network. There is previous work in differential privacy literature [71, 120] where the mechanism first *decides* whether or not to answer a query, and then privately answers the queries it chooses to answer using a traditional noise-addition mechanism. In these cases, the sparse vector technique [49, Chapter 3.6] helps bound the privacy cost in terms of the number of answered queries. This is in contrast to our work where a constant *fraction* of queries get answered and the sparse vector technique does not seem to help reduce the privacy cost. Closer to our work, [23] considers a setting where the answer to a query of interest is often either very large or very small. They show that a sparse vector-like analysis applies in this case, where one pays only for queries that are in the middle.

5.2 Background and Overview

We introduce essential components of our approach towards a generic and flexible framework for machine learning with provable privacy guarantees for training data.

5.2.1 The PATE Framework

Here, we provide an overview of the PATE framework. To protect the privacy of training data during learning, PATE transfers knowledge from an ensemble of teacher models trained on partitions of the data to a student model. Privacy guarantees may be understood intuitively and expressed rigorously in terms of differential privacy.

Illustrated in Figure 5.2, the PATE framework consists of three key parts: (1) an ensemble of n teacher models, (2) an aggregation mechanism and (3) a student model.

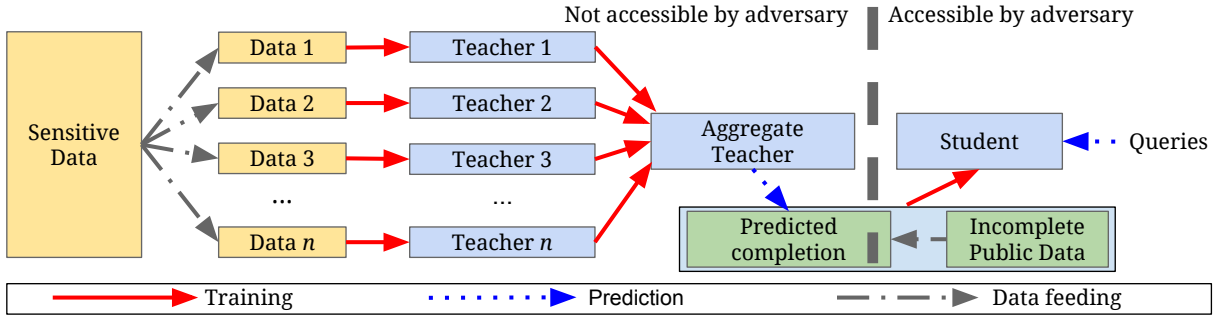


Figure 5.2. Overview of the approach: (1) an ensemble of teachers is trained on disjoint subsets of the sensitive data, (2) a student model is trained on public data labeled using the ensemble.

Teacher models: Each teacher is a model trained independently on a subset of the data whose privacy one wishes to protect. The data is partitioned to ensure no pair of teachers will have trained on overlapping data. Any learning technique suitable for the data can be used for any teacher. Training each teacher on a *partition* of the sensitive data produces n different models solving the same task. At inference, teachers independently predict labels.

Aggregation mechanism: When there is a strong consensus among teachers, the label they almost all agree on does not depend on the model learned by any given teacher. Hence, this collective decision is intuitively private with respect to any given training point – because such a point could have been included only in one of the teachers’ training set. To provide rigorous guarantees of differential privacy, the aggregation mechanism of the original PATE framework counts votes assigned to each class, adds carefully calibrated Laplace noise to the resulting vote histogram, and outputs the class with the most noisy votes as the ensemble’s prediction. This mechanism is referred to as the max-of-Laplace mechanism, or LNMax, going forward.

For samples x and classes $1, \dots, m$, let $f_j(x) \in [m]$ denote the j -th teacher model’s prediction and n_i denote the vote count for the i -th class (i.e., $n_i \triangleq |f_j(x) = i|$). The output of the mechanism is $\mathcal{A}(x) \triangleq \operatorname{argmax}_i (n_i(x) + \operatorname{Lap}(1/\gamma))$. Through a rigorous analysis of this mechanism, the PATE framework provides a differentially private API: the privacy cost of each aggregated prediction made by the teacher ensemble is known.

Student model: PATE’s final step involves the training of a student model by knowledge transfer

from the teacher ensemble using access to public – but *unlabeled* – data. To limit the privacy cost of labeling them, queries are only made to the aggregation mechanism for a subset of public data to train the student in a semi-supervised way using a fixed number of queries. The authors note that every additional ensemble prediction increases the privacy cost spent and thus cannot work with unbounded queries. Fixed queries fixes privacy costs as well as diminishes the value of attacks analyzing model parameters to recover training data [151]. The student only sees public data and privacy-preserving labels.

5.2.2 Rényi Differential Privacy

[111] note that the natural approach to bounding PATE’s privacy loss – by bounding the privacy cost of each label queried and using strong composition [51] to derive the total cost – yields loose privacy guarantees. Instead, their approach uses *data-dependent* privacy analysis. This takes advantage of the fact that when the consensus among the teachers is very strong, the plurality outcome has overwhelming likelihood leading to a very small privacy cost whenever the consensus occurs. To capture this effect quantitatively, [111] rely on the *moments accountant*, introduced by [1] and building on previous work [22, 50].

Rényi Differential Privacy or RDP [101] generalizes pure differential privacy ($\delta = 0$) and is closely related to the moments accountant. We choose to use RDP as a more natural analysis framework when dealing with our mechanisms that use Gaussian noise. The definition of RDP and its composition properties, relation to (ϵ, δ) -differential privacy are introduced in Chapter 2.

While both (ϵ, δ) -differential privacy and RDP are relaxations of pure ϵ -differential privacy, the two main advantages of RDP are as follows. First, it composes nicely; second, it captures the privacy guarantee of Gaussian noise in a much cleaner manner compared to (ϵ, δ) -differential privacy. This lets us do a careful privacy analysis of the GNMax mechanism as stated in Theorem 5.3.1. While the analysis of [111] leverages the first aspect of such frameworks with the Laplace noise (LNMax mechanism), our analysis of the GNMax mechanism relies on both.

5.2.3 PATE Aggregation Mechanisms

The aggregation step is a crucial component of PATE. It enables knowledge transfer from the teachers to the student while enforcing privacy. We improve the LNMax mechanism used by [111] which adds Laplace noise to teacher votes and outputs the class with the highest votes.

First, we add Gaussian noise with an accompanying privacy analysis in the RDP framework. This modification effectively reduces the noise needed to achieve the same privacy cost per student query.

Second, the aggregation mechanism is now *selective*: teacher votes are analyzed to decide which student queries are *worth* answering. This takes into account both the privacy cost of each query and its payout in improving the student’s utility. Surprisingly, our analysis shows that these two metrics are not at odds and in fact align with each other: the privacy cost is the smallest when teachers agree, and when teachers agree, the label is more likely to be correct thus being more useful to the student.

Third, we propose and study an *interactive* mechanism that takes into account not only teacher votes on a queried example but possible student predictions on that query. Now, queries worth answering are those where the teachers agree on a class but the student is not confident in its prediction on that class. This third modification aligns the two metrics discussed above even further: queries where the student already agrees with the consensus of teachers are not worth expending our privacy budget on, but queries where the student is less confident are useful and answered at a small privacy cost.

5.2.4 Data-dependent Privacy in PATE

A direct privacy analysis of the aggregation mechanism, for reasonable values of the noise parameter, allows answering only few queries before the privacy cost becomes prohibitive. The original PATE proposal used a data-dependent analysis, exploiting the fact that when the teachers have large agreement, the privacy cost is usually much smaller than the data-independent

bound would suggest.

In this chapter, we perform a data-dependent privacy analysis of the aggregation mechanism with Gaussian noise. This change of noise distribution turns out to be technically much more challenging than the Laplace noise case. The full analysis can be found in the full version [112]. This increased complexity of the analysis however does not make the algorithm any more complicated and thus allows us to improve the privacy-utility tradeoff.

Sanitizing the privacy cost via smooth sensitivity analysis. An additional challenge with data-dependent privacy analyses arises from the fact that the privacy cost itself is now a function of the private data. Further, the data-dependent bound on the privacy cost has large global sensitivity (a metric used in differential privacy to calibrate the noise injected) and is therefore difficult to sanitize. To remedy this, we use the smooth sensitivity framework proposed in [110].

The full paper [112] describes how we add noise to the computed privacy cost using this framework to publish a sanitized version of the privacy cost. The final analysis shows that the incremental cost of sanitizing our privacy estimates is modest – less than 50% of the raw estimates – thus enabling us to use precise data-dependent privacy analysis while taking into account its privacy implications.

5.3 Improved Aggregation Mechanisms for PATE

The privacy guarantees provided by PATE stem from the design and analysis of the aggregation step. Here, we detail our improvements to the mechanism used by [111]. As outlined in Section 5.2.3, we first replace the Laplace noise added to teacher votes with Gaussian noise, adapting the data-dependent privacy analysis. Next, we describe the Confident and Interactive Aggregators that select queries worth answering in a privacy-preserving way: the privacy budget is shared between the query selection and answer computation. The aggregators use different heuristics to select queries: the former does not take into account student predictions, while the

latter does.

5.3.1 The GNMax Aggregator and Its Privacy Guarantee

This section uses the following notation. For a sample x and classes 1 to m , let $f_j(x) \in [m]$ denote the j -th teacher model’s prediction on x and $n_i(x)$ denote the vote count for the i -th class (i.e., $n_i(x) = |\{j: f_j(x) = i\}|$). We define a Gaussian NoisyMax (GNMax) aggregation mechanism as:

$$\mathcal{M}_\sigma(x) \triangleq \underset{i}{\operatorname{argmax}} \{n_i(x) + \mathcal{N}(0, \sigma^2)\},$$

where $\mathcal{N}(0, \sigma^2)$ is the Gaussian distribution with mean 0 and variance σ^2 . The aggregator outputs the class with noisy plurality after adding Gaussian noise to each vote count. In what follow, *plurality* more generally refers to the highest number of teacher votes assigned among the classes.

The Gaussian distribution is more concentrated than the Laplace distribution used by [111]. This concentration directly improves the aggregation’s utility when the number of classes m is large. The GNMax mechanism satisfies $(\lambda, \lambda/\sigma^2)$ -RDP, which holds for all inputs and all $\lambda \geq 1$ (precise statements and proofs of claims in this section can be found in the full paper [112]). A straightforward application of composition theorems leads to loose privacy bounds. As an example, the standard advanced composition theorem applied to experiments in the last two rows of Table 5.1 would give us $\varepsilon = 8.42$ and $\varepsilon = 10.14$ resp. at $\delta = 10^{-8}$ for the Glyph dataset.

To refine these, we work out a careful *data-dependent* analysis that yields values of ε smaller than 1 for the same δ . The following theorem translates data-independent RDP guarantees for higher orders into a data-dependent RDP guarantee for a smaller order λ . We use it in conjunction with Theorem 5.3.2 to bound the privacy cost of each query to the GNMax algorithm as a function of \tilde{q} , the probability that the most common answer will not be output by the mechanism.

Theorem 5.3.1 (informal). *Let \mathcal{M} be a randomized algorithm with (μ_1, ε_1) -RDP and (μ_2, ε_2) -*

RDP guarantees and suppose that given a dataset D , there exists a likely outcome i^* such that $P(\mathcal{M}(D) \neq i^*) \leq \tilde{q}$. Then the data-dependent Rényi differential privacy for \mathcal{M} of order $\lambda \leq \mu_1, \mu_2$ at D is bounded by a function of $\tilde{q}, \mu_1, \varepsilon_1, \mu_2, \varepsilon_2$, which approaches 0 as $\tilde{q} \rightarrow 0$.

The new bound improves on the data-independent privacy for λ as long as the distribution of the algorithm’s output *on that input* has a strong peak (i.e., $\tilde{q} \ll 1$). Values of \tilde{q} close to 1 could result in a looser bound. Therefore, in practice we take the minimum between this bound and λ/σ^2 (the data-independent one). The theorem generalizes Theorem 3 from [111], where it was shown for a mechanism satisfying ε -differential privacy (i.e., $\mu_1 = \mu_2 = \infty$ and $\varepsilon_1 = \varepsilon_2$).

The final step in our analysis uses the following lemma to bound the probability \tilde{q} when i^* corresponds to the class with the true plurality of teacher votes.

Proposition 5.3.2. *For any $i^* \in [m]$, we have $P(\mathcal{M}_\sigma(D) \neq i^*) \leq \frac{1}{2} \sum_{i \neq i^*} \operatorname{erfc}\left(\frac{n_{i^*} - n_i}{2\sigma}\right)$, where erfc is the complementary error function.*

In the full paper [112], we detail how these results translate to privacy bounds. In short, for each query to the GNMax aggregator, given teacher votes n_i and the class i^* with maximal support, Theorem 5.3.2 gives us the value of \tilde{q} to use in Theorem 5.3.1. We optimize over μ_1 and μ_2 to get a data-dependent RDP guarantee for any order λ . Finally, we use composition properties of RDP to analyze a sequence of queries, and translate the RDP bound back to an (ε, δ) -DP bound.

Expensive queries. This data-dependent privacy analysis leads us to the concept of an *expensive* query in terms of its privacy cost. When teacher votes largely disagree, some $n_{i^*} - n_i$ values may be small leading to a large value for \tilde{q} : i.e., the lack of consensus amongst teachers indicates that the aggregator is likely to output a wrong label. Thus expensive queries from a privacy perspective are often bad for training too. Conversely, queries with strong consensus enable tight privacy bounds. This synergy motivates the aggregation mechanisms discussed in the following sections: they evaluate the strength of the consensus before answering a query.

5.3.2 The Confident-GNMax Aggregator

In this section, we propose a refinement of the GNMax aggregator that enables us to filter out queries for which teachers do not have a sufficiently strong consensus. This filtering enables the teachers to avoid answering expensive queries. We also take note to do this selection step itself in a private manner.

The proposed *Confident Aggregator* is described in Algorithm 3. To select queries with overwhelming consensus, the algorithm checks if the plurality vote crosses a threshold T . To enforce privacy in this step, the comparison is done after adding Gaussian noise with variance σ_1^2 . Then, for queries that pass this noisy threshold check, the aggregator proceeds with the usual GNMax mechanism with a smaller variance σ_2^2 . For queries that do not pass the noisy threshold check, the aggregator simply returns \perp and the student discards this example in its training.

In practice, we often choose significantly higher values for σ_1 compared to σ_2 . This is because we pay the cost of the noisy threshold check *always*, and without the benefit of knowing that the consensus is strong. We pick T so that queries where the plurality gets less than half the votes (often very expensive) are unlikely to pass the threshold after adding noise, but we still have a high enough yield amongst the queries with a strong consensus. This tradeoff leads us to look for T 's between $0.6\times$ to $0.8\times$ the number of teachers.

The privacy cost of this aggregator is intuitive: we pay for the threshold check for every query, and for the GNMax step only for queries that pass the check. In the work of [111], the mechanism paid a privacy cost for every query, expensive or otherwise. In comparison, the Confident Aggregator expends a much smaller privacy cost to check against the threshold, and by answering a significantly smaller fraction of expensive queries, it expends a lower privacy cost overall.

Algorithm 3. – Confident-GNMax Aggregator: given a query, consensus among teachers is first estimated in a privacy-preserving way to then only reveal confident teacher predictions.

Require: input x , threshold T , noise parameters σ_1 and σ_2

- 1: **if** $\max_i \{n_j(x)\} + \mathcal{N}(0, \sigma_1^2) \geq T$ **then** ▷ Privately check for consensus
 - 2: **return** $\operatorname{argmax}_j \{n_j(x) + \mathcal{N}(0, \sigma_2^2)\}$ ▷ Run the usual max-of-Gaussian
 - 3: **else**
 - 4: **return** \perp
 - 5: **end if**
-

5.3.3 The Interactive-GNMax Aggregator

While the Confident Aggregator excludes expensive queries, it ignores the possibility that the student might receive labels that contribute little to learning, and in turn to its utility. By incorporating the student’s current predictions for its public training data, we design an *Interactive Aggregator* that discards queries where the student already confidently predicts the same label as the teachers.

Given a set of queries, the Interactive Aggregator (Algorithm 4) selects those answered by comparing student predictions to teacher votes for each class. Similar to Step 1 in the Confident Aggregator, queries where the plurality of these noised differences crosses a threshold are answered with GNMax. This noisy threshold suffices to enforce privacy of the first step because student predictions can be considered public information (the student is trained in a differentially private manner).

For queries that fail this check, the mechanism reinforces the predicted student label if the student is confident enough and does this without looking at teacher votes again. This limited form of supervision comes at a small privacy cost. Moreover, the order of the checks ensures that a student falsely confident in its predictions on a query is not accidentally reinforced if it disagrees with the teacher consensus. The privacy accounting is identical to the Confident Aggregator except in considering the difference between teachers and the student instead of only the teachers votes.

In practice, the Confident Aggregator can be used to start training a student when it can

Algorithm 4. – Interactive-GNMax Aggregator: the protocol first compares student predictions to the teacher votes in a privacy-preserving way to then either (a) reinforce the student prediction for the given query or (b) provide the student with a new label predicted by the teachers.

Require: input x , confidence γ , threshold T , noise parameters σ_1 and σ_2 , total number of teachers M

- 1: Ask the student to provide prediction scores $\mathbf{p}(x)$
 - 2: **if** $\max_j \{n_j(x) - Mp_j(x)\} + \mathcal{N}(0, \sigma_1^2) \geq T$ **then** ▷ Student does not agree with teachers
 - 3: **return** $\operatorname{argmax}_j \{n_j(x) + \mathcal{N}(0, \sigma_2^2)\}$ ▷ Teachers provide new label
 - 4: **else if** $\max\{p_i(x)\} > \gamma$ **then** ▷ Student agrees with teachers and is confident
 - 5: **return** $\operatorname{argmax}_j p_j(x)$ ▷ Reinforce student’s prediction
 - 6: **else**
 - 7: **return** \perp ▷ No output given for this label
 - 8: **end if**
-

make no meaningful predictions and training can be finished off with the Interactive Aggregator after the student gains some proficiency.

5.4 Experimental Evaluation

Our goal is first to show that the improved aggregators introduced in Section 5.3 enable the application of PATE to uncurated data, thus departing from previous results on tasks with balanced and well-separated classes. We experiment with the Glyph dataset described below to address two aspects left open by [111]: (a) the performance of PATE on a task with a larger number of classes (the framework was only evaluated on datasets with at most 10 classes) and (b) the privacy-utility tradeoffs offered by PATE on data that is class imbalanced and partly mislabeled. In Section 5.4.2, we evaluate the improvements given by the GNMax aggregator over its Laplace counterpart (LNMax) and demonstrate the necessity of the Gaussian mechanism for uncurated tasks.

In Section 5.4.3, we then evaluate the performance of PATE with both the Confident and Interactive Aggregators on all datasets used to benchmark the original PATE framework, in addition to Glyph. With the right teacher and student training, the two mechanisms from Section 5.3 achieve high accuracy with very tight privacy bounds. Not answering queries for which

teacher consensus is too low (Confident-GNMax) or the student’s predictions already agree with teacher votes (Interactive-GNMax) better aligns utility and privacy: queries are answered at a significantly reduced cost.

5.4.1 Experimental Setup

MNIST, SVHN, and the UCI Adult databases. We evaluate with two computer vision tasks (MNIST and Street View House Numbers [108]) and census data from the UCI Adult dataset [85]. This enables a comparative analysis of the utility-privacy tradeoff achieved with our Confident-GNMax aggregator and the LNMax originally used in PATE. We replicate the experimental setup and results found in [111] with code and teacher votes made available online. The source code for the privacy analysis as well as supporting data required to run this analysis is available on Github.¹

A detailed description of the experimental setup can be found in [111]; we provide here only a brief overview. For MNIST and SVHN, teachers are convolutional networks trained on partitions of the training set. For UCI Adult, each teacher is a random forest. The test set is split in two halves: the first is used as unlabeled inputs to simulate the student’s public data and the second is used as a hold out to evaluate test performance. The MNIST and SVHN students are convolutional networks trained using semi-supervised learning with GANs à la [124]. The student for the Adult dataset are fully supervised random forests.

Glyph. This optical character recognition task has *an order of magnitude more classes* than all previous applications of PATE. The Glyph dataset also possesses many characteristics shared by real-world tasks: e.g., it is imbalanced and some inputs are mislabeled. Each input is a 28×28 grayscale image containing a single glyph generated synthetically from a collection of over 500K computer fonts.² Samples representative of the difficulties raised by the data are depicted in Figure 5.3. The task is to classify inputs as one of the 150 Unicode symbols used to

¹https://github.com/tensorflow/models/tree/master/research/differential_privacy

²Glyph data is not public but similar data is available publicly as part of the notMNIST dataset.

generate them.

This set of 150 classes results from pre-processing efforts. We discarded additional classes that had few samples; some classes had at least 50 times fewer inputs than the most popular classes, and these were almost exclusively incorrectly labeled inputs. We also merged classes that were too ambiguous for even a human to differentiate them. Nevertheless, a manual inspection of samples grouped by classes—favorably to the human observer – led to the conservative estimate that some classes remain 5 times more frequent, and mislabeled inputs represent at least 10% of the data.

To simulate the availability of private and public data (see Section 5.2.1), we split data originally marked as the training set (about 65M points) into partitions given to the teachers. Each teacher is a ResNet [73] made of 32 leaky ReLU layers. We train on batches of 100 inputs for 40K steps using SGD with momentum. The learning rate, initially set to 0.1, is decayed after 10K steps to 0.01 and again after 20K steps to 0.001. These parameters were found with a grid search.

We split holdout data in two subsets of 100K and 400K samples: the first acts as public data to train the student and the second as its testing data. The student architecture is a convolutional network learnt in a semi-supervised fashion with virtual adversarial training (VAT) from [102]. Using unlabeled data, we show how VAT can regularize the student by making predictions constant in *adversarial*³ directions. Indeed, we found that GANs did not yield as much utility for Glyph as for MNIST or SVHN. We train with Adam for 400 epochs and a learning rate of $6 \cdot 10^{-5}$.

5.4.2 Comparing the LNMax and GNMax Mechanisms

Section 5.3.1 introduces the GNMax mechanism and the accompanying privacy analysis. With a Gaussian distribution, whose tail diminishes more rapidly than the Laplace distribution,

³In this context, the adversarial component refers to the phenomenon commonly referred to as adversarial examples [15, 136] and not to the adversarial training approach taken in GANs.

we expect better utility when using the new mechanism (albeit with a more involved privacy analysis).

To study the tradeoff between privacy and accuracy with the two mechanisms, we run experiments training several ensembles of M teachers for $M \in \{100, 500, 1000, 5000\}$ on the Glyph data. Recall that 65 million training inputs are partitioned and distributed among the M teachers with each teacher receiving between 650K and 13K inputs for the values of M above. The test data is used to query the teacher ensemble and the resulting labels (after the LNMax and GNMax mechanisms) are compared with the ground truth labels provided in the dataset. This predictive performance of the teachers is essential to good student training with accurate labels and is a useful proxy for utility.

For each mechanism, we compute (ϵ, δ) -differential privacy guarantees. As is common in literature, for a dataset on the order of 10^8 samples, we choose $\delta = 10^{-8}$ and denote the corresponding ϵ as the privacy cost. The total ϵ is calculated on a subset of 4,000 queries, which is representative of the number of labels needed by a student for accurate training (see Section 5.4.3). We visualize in Figure 5.4 the effect of the noise distribution (left) and the number of teachers (right) on the tradeoff between privacy costs and label accuracy.

Observations. On the left of Figure 5.1, we compare our GNMax aggregator to the LNMax aggregator used by the original PATE proposal, on an ensemble of 1000 teachers and for varying noise scales σ . At fixed test accuracy, the GNMax algorithm consistently outperforms the LNMax mechanism in terms of privacy cost. To explain this improved performance, recall notation from Section 5.3.1. For both mechanisms, the data dependent privacy cost scales linearly with \tilde{q} – the likelihood of an answer other than the true plurality. The value of \tilde{q} falls off as e^{-x^2} for GNMax and e^{-x} for LNMax, where x is the ratio $(n_{i^*} - n_i)/\sigma$. Thus, when $n_{i^*} - n_i$ is (say) 4σ , LNMax would have $\tilde{q} \approx e^{-4} = 0.018\dots$, whereas GNMax would have $\tilde{q} \approx e^{-16} \approx 10^{-7}$, thereby leading to a much higher likelihood of returning the true plurality. Moreover, this reduced \tilde{q} translates to a smaller privacy cost for a given σ leading to a better utility-privacy tradeoff.

As long as each teacher has sufficient data to learn a good-enough model, increasing



Figure 5.3. Some example inputs from the Glyph dataset along with the class they are labeled as. Note the ambiguity (between the comma and apostrophe) and the mislabeled input.

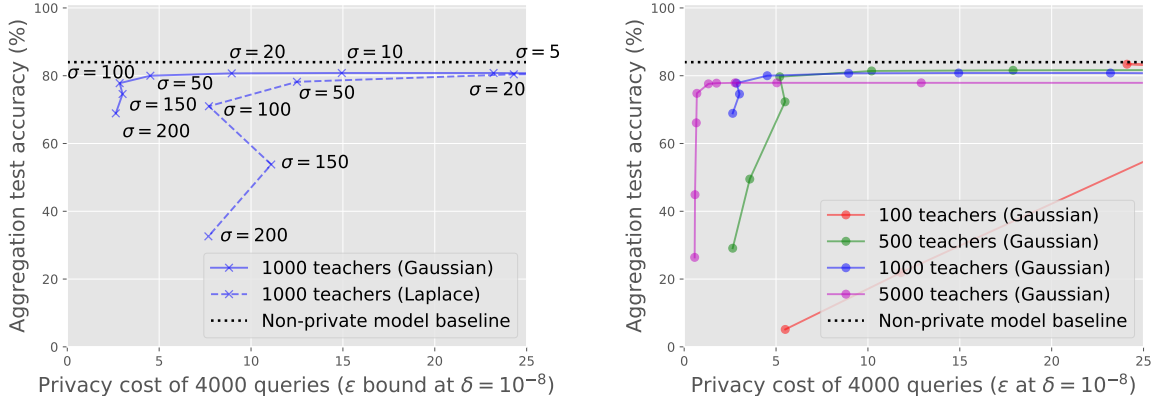


Figure 5.4. Tradeoff between utility and privacy for the LNMax and GNMax aggregators on Glyph: effect of the noise distribution (left) and size of the teacher ensemble (right).

Table 5.1. Utility and privacy of the students. For MNIST, Adult, and SVHN, we use the labels of ensembles of 250 teachers published by [111] and set $\delta = 10^{-5}$ (to the exception of SVHN where $\delta = 10^{-6}$). All Glyph results use an ensemble of 5000 teachers and δ is set to 10^{-8} .

Dataset	Aggregator	Queries answered	Privacy bound ϵ	Accuracy	
				Student	Baseline
MNIST	LNMax [111]	100	2.04	98.0%	99.2%
	LNMax [111]	1,000	8.03	98.1%	
	Confident-GNMax ($T=200, \sigma_1=150, \sigma_2=40$)	286	1.97	98.5%	
SVHN	LNMax [111]	500	5.04	82.7%	92.8%
	LNMax [111]	1,000	8.19	90.7%	
	Confident-GNMax ($T=300, \sigma_1=200, \sigma_2=40$)	3,098	4.96	91.6%	
Adult	LNMax [111]	500	2.66	83.0%	85.0%
	Confident-GNMax ($T=300, \sigma_1=200, \sigma_2=40$)	524	1.90	83.7%	
Glyph	LNMax	4,000	4.3	72.4%	82.2%
	Confident-GNMax ($T=1000, \sigma_1=500, \sigma_2=100$)	10,762	2.03	75.5%	
	Interactive-GNMax, two rounds	4,341	0.837	73.2%	

the number M of teachers improves the tradeoff – as illustrated on the right of Figure 5.4 with GNMax. The larger ensembles lower the privacy cost of answering queries by tolerating larger σ 's. Combining the two observations made in this Figure, for a fixed label accuracy, we lower privacy costs by switching to the GNMax aggregator and training a larger number M of teachers.

5.4.3 Student Training with the GNMax Aggregation Mechanisms

As outlined in Section 5.2, we train a student on public data labeled by the aggregation mechanisms. We take advantage of PATE's flexibility and apply the technique that performs best on each dataset: semi-supervised learning with Generative Adversarial Networks [124] for MNIST and SVHN, Virtual Adversarial Training [102] for Glyph, and fully-supervised random forests for UCI Adult. In addition to evaluating the total privacy cost associated with training the student model, we compare its utility to a non-private baseline obtained by training on the sensitive data (used to train teachers in PATE): we use the baselines of 99.2%, 92.8%, and 85.0% reported by [111] respectively for MNIST, SVHN, and UCI Adult, and we measure a baseline of 82.2% for Glyph. We compute (ϵ, δ) -privacy bounds and denote the privacy cost as the ϵ value at a value of δ set accordingly to number of training samples.

Confident-GNMax Aggregator. Given a pool of 500 to 12,000 samples to learn from (depending on the dataset), the student submits queries to the teacher ensemble running the Confident-GNMax aggregator from Section 5.3.2. A grid search over a range of plausible values for parameters T , σ_1 and σ_2 yielded the values reported in Table 5.1, illustrating the tradeoff between utility and privacy achieved. We additionally measure the number of queries selected by the teachers to be answered and compare student utility to a non-private baseline.

The Confident-GNMax aggregator outperforms LNMax for the four datasets considered in the original PATE proposal: it reduces the privacy cost ϵ , increases student accuracy, or both simultaneously. On the uncurated Glyph data, despite the imbalance of classes and mislabeled data (as evidenced by the 82.2% baseline), the Confident Aggregator achieves 73.5% accuracy with a privacy cost of just $\epsilon = 1.02$. Roughly 1,300 out of 12,000 queries made are not answered,

indicating that several expensive queries were successfully avoided. This selectivity is analyzed in more details in Section 5.4.4.

Interactive-GNMax Aggregator. On Glyph, we evaluate the utility and privacy of an interactive training routine that proceeds in *two rounds*. Round one runs student training with a Confident Aggregator. A grid search targeting the best privacy for roughly 3,400 answered queries (out of 6,000) – sufficient to bootstrap a student – led us to setting $(T=3500, \sigma_1=1500, \sigma_2=100)$ and a privacy cost of $\varepsilon \approx 0.59$.

In round two, this student was then trained with 10,000 more queries made with the Interactive-GNMax Aggregator $(T=3500, \sigma_1=2000, \sigma_2=200)$. We computed the resulting (total) privacy cost and utility at an *exemplar* data point through another grid search of plausible parameter values. The result appears in the last row of Table 5.1. With just over 10,422 answered queries in total at a privacy cost of $\varepsilon = 0.84$, the trained student was able to achieve 73.2% accuracy. Note that this students required fewer answered queries compared to the Confident Aggregator. The best overall cost of student training occurred when the privacy costs for the first and second rounds of training were roughly the same. (The total ε is less than $0.59 \times 2 = 1.18$ due to better composition – via Theorems 2.2.3 and 2.2.4.)

Comparison with Baseline. Note that the Glyph student’s accuracy remains seven percentage points below the non-private model’s accuracy achieved by training on the 65M sensitive inputs. We hypothesize that this is due to the uncurated nature of the data considered. Indeed, the class imbalance naturally requires more queries to return labels from the less represented classes. For instance, a model trained on 200K queries is only 77% accurate on test data. In addition, the large fraction of mislabeled inputs are likely to have a large privacy cost: these inputs are sensitive because they are outliers of the distribution, which is reflected by the weak consensus among teachers on these inputs.

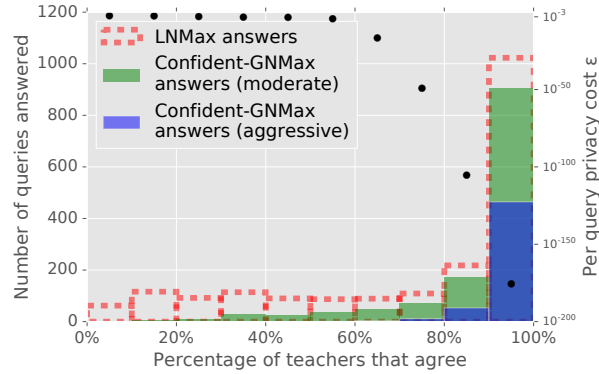


Figure 5.5. Effects of the noisy threshold checking: # queries answered by LNMax, Confident-GNMax moderate ($T=3500, \sigma_1=1500$), and Confident-GNMax aggressive ($T=5000, \sigma_1=1500$). The black dots and the right axis show the expected cost of answering a single query in each bin (via GNMax, $\sigma_2=100$).

5.4.4 Noisy Threshold Checks and Privacy Costs

Sections 5.3.1 and 5.3.2 motivated the need for a noisy threshold checking step before having the teachers answer queries: it prevents most of the privacy budget being consumed by few queries that are expensive and also likely to be incorrectly answered. In Figure 5.5 and Figure 5.6, we compare the privacy cost ϵ of answering all queries to only answering confident queries for a fixed number of queries.

We run additional experiments to support the evaluation from Section 5.4.3. With the votes of 5,000 teachers on the Glyph dataset, we plot in Figure 5.5 the histogram of the plurality vote counts (n_{i^*} in the notation of Section 5.3.1) across 25,000 student queries. We compare these values to the vote counts of queries that passed the noisy threshold check for two sets of parameters T and σ_1 in Algorithm 3. Smaller values imply weaker teacher agreements and consequently more expensive queries.

When ($T=3500, \sigma_1=1500$) we capture a significant fraction of queries where teachers have a strong consensus (roughly > 4000 votes) while managing to filter out many queries with poor consensus. This *moderate check* ensures that although many queries with plurality votes between 2,500 and 3,500 are answered (i.e., only 50–70% of teachers agree on a label) the

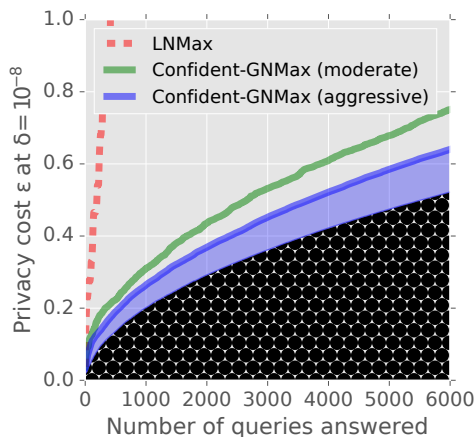


Figure 5.6. Effects of the noisy threshold checking: Privacy cost of answering all (LNMax) vs only inexpensive queries (GNMax) for a given number of answered queries. The very dark area under the curve is the cost of selecting queries; the rest is the cost of answering them.

expensive ones are most likely discarded. For $(T=5000, \sigma_1=1500)$, queries with poor consensus are completely culled out. This selectivity comes at the expense of a noticeable drop for queries that might have had a strong consensus and little-to-no privacy cost. Thus, this *aggressive check* answer fewer queries with very strong privacy guarantees. We reiterate that this threshold checking step itself is done in a private manner. Empirically, in our Interactive Aggregator experiments, we expend about a third to a half of our privacy budget on this step, which still yields a very small cost *per query* across 6,000 queries.

5.5 Conclusions

The key insight motivating the addition of a noisy thresholding step to the two aggregation mechanisms proposed in this chapter is that there is a form of synergy between the privacy and accuracy of labels output by the aggregation: *labels that come at a small privacy cost also happen to be more likely to be correct*. As a consequence, we are able to provide more quality supervision to the student by choosing not to output labels when the consensus among teachers is too low to provide an aggregated prediction at a small cost in privacy. This observation was further confirmed in some of our experiments where we observed that if we trained the student

on either private or non-private labels, the former almost always gave better performance than the latter – for a fixed number of labels.

Complementary with these aggregation mechanisms is the use of a Gaussian (rather than Laplace) distribution to perturb teacher votes. In our experiments with Glyph data, these changes proved essential to preserve the accuracy of the aggregated labels – because of the large number of classes. The analysis presented in Section 5.3 details the delicate but necessary adaptation of analogous results for the Laplace NoisyMax.

As was the case for the original PATE proposal, semi-supervised learning was instrumental to ensure the student achieves strong utility given a limited set of labels from the aggregation mechanism. However, we found that virtual adversarial training outperforms the approach from [124] in our experiments with Glyph data. These results establish lower bounds on the performance that a student can achieve when supervised with our aggregation mechanisms; future work may continue to investigate virtual adversarial training, semi-supervised generative adversarial networks and other techniques for learning the student in these particular settings with restricted supervision.

Acknowledgments

We are grateful to Martín Abadi, Vincent Vanhoucke, and Daniel Levy for their useful inputs and discussions.

This chapter is based on “Scalable Private Learning with PATE” by Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar and Úlfar Erlingsson, which appears in the International Conference on Learning Representations (ICLR), 2018 [112]. The dissertation author was a primary author of the paper.

Chapter 6

Differentially Private Continual Release of Graph Statistics

6.1 Overview

Dynamic social networks are ubiquitous models of social and economic phenomena, and analyzing them over a period of time can allow researchers to understand various aspects of human behavior. Many social networks, however, include sensitive and personal information about the people involved. Consequently, we need to design privacy-preserving algorithms that can summarize properties of dynamic social networks over time while still preserving the privacy of the participants.

As a concrete motivating example, consider data on HIV transmission collected from patients in a particular region over multiple years [90, 141, 145]. Advances in sequencing technology allow scientists to infer putative transmission links by measuring similarities between HIV sequences obtained from different patients. These links can then be resolved into transmission networks, reflecting the patterns of transmission in that population. Epidemiologists would like to study properties of these networks as they grow over time to understand how HIV propagates. Since there is considerable social stigma associated with HIV, these networks are highly sensitive information, and public release of their properties needs to ensure that privacy of the included individuals is not violated. Additionally, analyses of these networks need to happen intermittently – for example, once a year – so that properties of the network as it evolves may be

studied.

In this chapter, we consider continual privacy-preserving release of graph statistics, such as degree distributions and subgraph counts, from sensitive networks where nodes and their associated edges appear over time in an online manner. For our privacy notion, we use differential privacy [45] – the gold standard in private data analysis. Differential privacy guarantees privacy by ensuring that the participation of a single person in the dataset does not change the probability of any outcome by much; this is enforced by adding enough noise to either the input data or to the output of a function computed on the data so as to obscure the private value of a single individual. Since in our applications, a node corresponds to a single person, we use node differential privacy [72], where the goal is to hide the participation of any single node.

There are two main challenges in continually releasing graph statistics with node differential privacy. The first is that node differential privacy itself is often very difficult to attain, and can only be attained in either bounded degree graphs or graphs that can be projected to be degree-bounded. The second challenge pertains to the online nature of the problem. Prior work has looked at continual release of statistics based on streaming tabular data [19, 24, 25, 80, 29, 47]; however, these works rely on the fact that in tabular data, at any time t , we only get information about the t -th individual, and not about individuals who already exist in the data. This property no longer holds in online graphs, as an incoming node may bring in new information about existing nodes in the form of connecting edges, and therefore these solutions do not directly apply.

In this work, we show that if there is a publicly known upper bound on the maximum degree of any node in the entire graph sequence, then, a *difference sequence* – namely, the sequence of differences in the statistics computed on subsequent graphs – has low sensitivity. The assumption of bounded maximum degree holds for many real networks, as many real-world graphs, such as social interaction networks, collaboration networks, computer networks and disease transmission networks, that are scale-free with power-law degree distributions have low maximum degree. Given this assumption holds, we show in particular that the sensitivity of

the entire difference sequence only depends on the publicly known upper bound, and not on the length of the sequence. This implies that we can release a private version of the difference sequence with relatively high accuracy, which can be used to continually release the target statistic with high privacy-accuracy tradeoff.

It is commonly believed that many real-world networks, such as social interaction networks, collaboration networks, computer networks and disease transmission networks are scale-free with degree distributions following a power law; such graphs have low maximum degrees.

We derive the sensitivity of the difference sequence for a number of common graph statistics, such as degree distribution, number of high degree nodes, as well as counts of fixed subgraphs. We then implement our algorithms and evaluate them on three real and two synthetic datasets against two natural baselines. Our experimental results show that the algorithm outperforms these baselines in terms of utility for these datasets over a range of privacy parameters.

6.1.1 Related Work

To apply differential privacy to graph data, it is important to determine what a single person’s data contributes to the graph. Prior work has looked at two forms of differential privacy in graphs – *edge differential privacy*, where an edge corresponds to a person’s private value, and *node differential privacy*, where a single node corresponds to a person. In our motivating application, a patient corresponds to a node, and hence node differential privacy is our privacy notion of choice.

Prior work on edge differential privacy [109, 72] has looked at how to compute a number of statistics for *static graphs* while preserving privacy. For example, [76] computes subgraph counts, and [72] degree distributions with edge differential privacy. It is also known how to successfully calculate more complex graph parameters under this notion; for example, [92] fits exponential random graph models and [143, 5] computes spectral graph statistics such as pagerank.

In contrast, achieving node differential privacy is considerably more challenging. Changing a single node and its associated edges can alter even simple statistics of a static graph significantly; this means that any differentially private solution needs to add a considerable amount of noise to hide the effect of a single node, resulting in low utility. Prior work has addressed this challenge in two separate ways. The first is to assume that there is a publicly known upper bound on the maximum degree of any node in the graph [20, 64].

The second is to use a carefully-designed projection from the input graph to a bounded degree graph, where adding or removing a single node has less effect, and then release statistics of the projected graph with privacy. To ensure that the entire process is privacy-preserving, the projection itself is required to be *smooth* – in the sense that changing a single node should not change the statistics of the projected graph by much. This approach has been taken by [78], who releases degree distributions and subgraph counts for static graphs and [16], who releases subgraph counts and local profile queries. [116] releases degree distributions by using a flow-based projection algorithm. Finally, [35] proposes an improved projection method for releasing degree distributions, and is the state-of-the-art in this area. In this chapter, we show that when the graph arrives *online*, existing projection-based approaches can yield poor utility, and therefore, we consider bounded degree graphs, where domain knowledge suggests an upper bound on the maximum degree.

Finally, while we are not aware of any work on differentially private statistics on streaming graph data, prior work has looked at releasing private statistics on streaming *tabular data* in an online manner [19, 24, 25, 80, 29, 47]. In these settings, however, complete information about a single person (or a group of people) arrives at each time step, which makes the problem of private release considerably easier than online graph data, where newly arriving nodes may include information in the form of edges to already existing nodes. Thus, these approaches do not directly translate to online graphs.

6.2 Preliminaries

6.2.1 Graphs and Graph Sequences

Formally, we consider a graph $G = (V, E)$, where a node $v \in V$ represents a person and an edge $(u, v) \in E$ a relationship. G may be directed or undirected, depending on the application. We assume that each node $v \in V$ is associated with a time stamp, denoted by $v.\text{time}$, that records when v enters the graph.

In our setting, a graph arrives online as more and more of its vertices and some of their adjacent edges become visible. More specifically, at time t , a set of vertices ∂V_t arrives, along with a set of edges ∂E_t ; each edge in ∂E_t has at least one end-point in ∂V_t , and the other end-point may be a vertex that arrived earlier. These vertices and edges, along with vertices and edges that arrived earlier comprise a graph G_t . Given a function f that operates on graphs, our goal is to output (a private approximation to) $f(G_t)$ at each time step t .

More formally, the arrival process comprises a *graph sequence* $\mathcal{G} = (G_1, G_2, \dots)$, which is defined as a sequence of graphs with $G_t = (V_t, E_t)$ such that $V_0 = \emptyset$, $\partial V_t = \{v : v.\text{time} = t\}$ is the set of all nodes with time stamp t and $V_t = V_{t-1} \cup \partial V_t$ for $t \geq 1$ is the set of all nodes with time stamps $\leq t$. Additionally, we let $E_0 = \emptyset$, $\partial E_t = \{(u, v) | u \in \partial V_t, v \in V_t \text{ or } u \in V_t, v \in \partial V_t\}$, and $E_t = E_{t-1} \cup \partial E_t$. Given a function f that operates on a graph, we define f applied to the graph sequence $f(\mathcal{G})$ as the sequence $(f(G_1), f(G_2), \dots)$.

If the graph sequence is $\mathcal{G} = (G_1, G_2, \dots, G_T)$, then the error of $\mathcal{A}(G)$ is defined as: $\sum_{t=1}^T |\mathcal{A}(G_t) - f(G_t)|$. Our goal is to design an algorithm \mathcal{A} that has as low error as possible.

6.2.2 Node Differential Privacy

Recall that in Chapter 2, we have defined differential privacy and the Laplace mechanism, which adds Laplace noise with scale parameter $\text{GS}(f)/\epsilon$, with $\text{GS}(f)$ as the global sensitivity of a query function f and ϵ as the privacy parameter, to the true value of the query result.

To apply differential privacy to graphs, we need to determine what constitutes a single

person’s data in a graph. For the kind of graphs that we will study, a node v corresponds to a single person. This is known as *node differential privacy* [72], which ensures that the addition or removal of a single node along with its adjacent edges does not change the probability of any outcome by much.

6.2.3 Bounded Degree Graphs

A major challenge with ensuring node differential privacy is that the global sensitivity $GS(f)$ may be very large even for simple graph functions f , which in turn requires the addition of a large amount of noise to ensure privacy. For example, if f is the number of nodes with degree ≥ 1 , and we have an empty graph G on n nodes, then adding a single node connected to every other node can increase f by as much as n .

Prior work has addressed this challenge in two separate ways. The first is by considering *Bounded Degree Graphs* [78, 16, 20], where an a-priori bound on the degree of any node is known to the user and the algorithm designer. This is the solution that we will consider in this chapter.

A second line of prior work [16, 78, 116, 35] presents *Graph Projections* algorithms that may be used to project graphs into lower degree graphs such that the resulting projections have low global sensitivity for some graph functions. In Section 6.4.1.1, we show that natural extensions of some of these projections may be quite unstable when a graph appears online.

We first define bounded degree graphs. Let $\deg_G(v)$ denote the degree of node v in an undirected graph G , $\text{out-deg}_G(v)$ and $\text{in-deg}_G(v)$ denote the out-degree and in-degree of v in a directed graph G .

Definition 6.2.1. *An undirected graph $G = (V, E)$ is D -bounded if $\deg_G(v) \leq D$ for any $v \in V$. A graph sequence $\mathcal{G} = (G_1, G_2, \dots)$ is D -bounded if for all t , G_t is D -bounded. In other words, the degree of all nodes remain bounded by D in the entire graph sequence.*

A directed graph $G = (V, E)$ is D_{out} -out-bounded if $\text{out-deg}_G(v) \leq D_{out}$ for any $v \in V$. A graph sequence $\mathcal{G} = (G_1, G_2, \dots)$ is D_{out} -out-bounded if for all t , G_t is D_{out} -out-bounded.

Similarly, G is D_{in} -in-bounded if $\text{in-deg}_G(v) \leq D_{in}$ for any $v \in V$. \mathcal{G} is D_{in} -in-bounded if for all t , G_t is D_{in} -in-bounded.

We say a directed graph or a graph sequence is (D_{in}, D_{out}) -bounded if it is both D_{in} -in-bounded and D_{out} -out-bounded.

In this work, we assume that the domain consists only of degree bounded graphs. This ensures that the global sensitivity of certain common graph functions, such as degree distribution and subgraph counts, is low, and allows us to obtain privacy with relatively low noise. Additionally, many common sensitive graphs, such as the HIV transmission graph and co-authorship networks, typically have relatively low maximum degree, thus ensuring that the assumption holds for low or moderate values of D .

6.2.4 Graph Functions

This work will consider two types of functions on graph sequences. The first consists of functions of the degree distribution. The specific functions we will look at for undirected graphs are $\text{highDeg}_\tau(G)$, which counts the number of nodes in G with degree $\geq \tau$ and the degree histogram $\text{hist}(G)$, which counts the number of nodes with degree d for any $d \in \mathbb{N}_+$. Similarly, for directed graph, we consider $\text{highOutDeg}_\tau(G)$, the number of nodes with out-degree $\geq \tau$, and the out-degree histogram $\text{histOut}(G)$, which counts the number of nodes with out-degree d for any $d \in \mathbb{N}_+$.

The second class of functions will involve subgraph counts. Given a subgraph S , we will count the number of occurrences of this subgraph $S(G)$ in the entire graph G . For example, when S is a triangle, $S(G)$ will count the number of triangles in the graph. When G is directed, so will be the corresponding subgraphs.

6.2.5 Other Notations

In a directed graph, an edge is denoted by an ordered tuple, i.e., (u, v) represents a directed edge pointing from node u to v .

We use $(a_t)_{t=1}^T$ as an abbreviation for vector (a_1, a_2, \dots, a_T) .

For any integer i , we use $[i]$ to denote the set $\{1, 2, \dots, i\}$.

A degree histogram h is a mapping from degrees to counts, i.e., given $d \in \mathbb{N}_+$, $h(d)$ is the number of nodes with degree equal to d . We define the distance between two histograms h and h' as $\|h(d) - h'(d)\|_1 = \sum_{d \in \mathbb{N}_+} |h(d) - h'(d)|$. Given two sequences of histograms $(h_t)_{t=1}^T$ and $(h'_t)_{t=1}^T$, we define the generalized L_1 distance between them as $\sum_{t=1}^T \|h_t(d) - h'_t(d)\|_1$.

6.3 Main Algorithm

Recall that we are given as input a D -bounded (or $(D_{\text{in}}, D_{\text{out}})$ -bounded) graph sequence $\mathcal{G} = (G_1, G_2, \dots, G_T)$ that arrives online, a privacy budget ϵ and a function f . Our goal is to publish an ϵ -differentially private approximation to the sequence $f(\mathcal{G})$ in an online manner. Specifically, at time t , an incoming vertex set ∂V_t and edges ∂E_t adjacent to it and the existing vertices arrive, and our goal is to release a private approximation to $f(G_t)$ with low additive L_1 -error.

Baseline Approaches. A naive approach is to calculate $f(G_t)$ at each t and add noise proportional to its global sensitivity over ϵ . Since ∂E_t may contain information on individuals in G_{t-1} in the form of adjacent edges, this procedure will not provide ϵ -differential privacy.

The correct way to do privacy accounting for this method is by sequential composition [45]. Suppose the graph sequence has total length T and we allocate privacy budget ϵ/T to each time step; then at time t , we calculate $f(G_t)$ and add noise proportional to its global sensitivity divided by ϵ/T . If the global sensitivity of $f(G_t)$ is $O(1)$, then, we add $O(T/\epsilon)$ noise to $f(G_t)$, which results in a $\Theta(T^2/\epsilon)$ expected L_1 -error between $f(\mathcal{G})$ and the output of the algorithm.

A second approach is to calculate $f(G_t)$ and add noise proportional to the global sensitivity of the (entire) sequence $f(\mathcal{G})$ divided by ϵ . This preserves ϵ -differential privacy. However, the global sensitivity of the sequence $f(\mathcal{G})$ typically grows linearly with T , the length of the

entire graph sequence, even if the graph sequence itself is degree-bounded. For example, if $f(G)$ is the number of nodes in G with degree $\geq \tau$, then, a single extra node with degree $\tau + 1$, added at time $t = 1$, can increase $f(G_t)$ by 1 for *every* t , resulting in a global sensitivity of $\Omega(T)$. Consequently, the expected L_1 -error between the true value of $f(\mathcal{G})$ and the output of this approach is as again large as $\Theta(T^2/\epsilon)$.

Our Approach. The main observation in this work is that for a number of popular functions, the *difference sequence* $\Delta = (f(G_1), f(G_2) - f(G_1), f(G_3) - f(G_2), \dots)$ has considerably better properties. Observe that unlike certain functions on tabular data [19, 24, 25, 80, 29, 47], releasing $f(G_t) - f(G_{t-1})$ after adding noise proportional to its sensitivity over ϵ will still not be ϵ -differentially private – this is because ∂E_t can still include edges adjacent to people in G_{t-1} .

However, the difference sequence Δ does have considerably less global sensitivity than $f(\mathcal{G})$. In particular, we show that if the graph sequence \mathcal{G} is D -bounded, then, the global sensitivity $\text{GS}(\Delta)$ of the entire difference sequence for a number of popular functions f depends only on D and not on the sequence length T . For example, in Section 6.4.1.1, we show that when G is an undirected graph and f is the number of nodes with degree $\geq \tau$, the global sensitivity of the entire difference sequence is at most $2D + 1$.

This immediately suggests the following algorithm. At time t , calculate the difference $\Delta_t = f(G_t) - f(G_{t-1})$, and add Laplace noise proportional to its global sensitivity over ϵ to get a private perturbed version $\tilde{\Delta}_t$. Release the partial sum $\sum_{s=1}^t \tilde{\Delta}_s$, which is an approximation to $f(G_t)$. Since the expected value of $\tilde{\Delta}_t - \Delta_t$ is independent of T , the maximum standard deviation of any partial sum is at most $O(\sqrt{T}/\epsilon)$, which results in an expected L_1 -error of $O(T^{3/2}/\epsilon)$ – better than the $O(T^2/\epsilon)$ -error achieved by the two baseline approaches.

The full algorithm, applied to a generic function f , is described in Algorithm 5. We call it **SENSDIFF** as it uses the global sensitivity of the difference sequence Δ . The rest of the chapter is devoted to analyzing the global sensitivity of the difference sequence for a number of popular graph functions f . Our analysis exploits specific combinatorial properties of the graph functions in question, and is carried out for two popular classes of graph functions – functions of

Algorithm 5. SENSDIFF(Graph sequence \mathcal{G} , query f , privacy parameter ϵ)

for $t = 1, \dots, T$ **do**
 Receive ∂V_t and ∂E_t , and construct G_t .
 Calculate $\Delta_t = f(G_t) - f(G_{t-1})$.
 Let $\text{GS}_D(\Delta)$ be the global sensitivity of the difference sequence;
 Calculate $\tilde{\Delta}_t = \Delta_t + \text{Lap}\left(\frac{\text{GS}_D(\Delta)}{\epsilon}\right)$, and the partial sum $\sum_{s=1}^t \tilde{\Delta}_s$.
end for
return $\left(\sum_{s=1}^t \tilde{\Delta}_s\right)_{t=1}^T$

the degree distribution and subgraph counts.

6.4 Functions of Degree Distributions

We begin with functions of the degree distributions of the graph sequence, and consider both directed and undirected graphs. A summary of the results in this section is provided in Table 6.1.

Table 6.1. Summary of degree distribution results.

	Undirected graph	Directed graph
(out-)degree histogram	$4D^2 + 2D + 1$ (for D -bounded)	$4D_{\text{out}}D_{\text{in}} + 2D_{\text{out}} + 1$ (for $(D_{\text{in}}, D_{\text{out}})$ -bounded)
high-(out-)degree nodes	$2D + 1$ (for D -bounded)	$2D_{\text{in}} + 1$ (for D_{in} -in-bounded)

6.4.1 Undirected Graphs

For undirected graphs, we will consider two functions applied to graph sequences – first, the number of nodes with degree greater than or equal to a threshold τ , and second, the degree histogram.

6.4.1.1 Number of High Degree Nodes

Recall that $\text{highDeg}_\tau(G)$ is the number of nodes in G with degree $\geq \tau$. We show below, that for D -bounded graphs, the difference sequence corresponding to $\text{highDeg}_\tau(G)$ has global sensitivity at most $2D + 1$.

Lemma 6.4.1. *Let $f(G) = \text{highDeg}_\tau(G)$. For D -bounded graphs, the difference sequence corresponding to f has global sensitivity at most $2D + 1$. In fact, the global sensitivity is $2D + 1$ for any $\tau < D$.*

Notice that $\tau \leq D$ is needed for the statistic to be meaningful; if $\tau > D$, there is no high-degree node.

Projection Yields High Sensitivity in Graph Sequence A common idea in static graph analysis with node differential privacy is to project the original graph into a bounded-degree graph. The sensitivity of some common statistics on this projected graph scales with the degree bound instead of the total number of nodes. The current state-of-the-art projection algorithm is proposed in [35]. Given a projection threshold \tilde{D} , a graph $G = (V, E)$ and an ordering of the nodes in V , the algorithm constructs a bounded-degree graph $G^{\tilde{D}}$ as follows. First, it adds all nodes in V to $G^{\tilde{D}}$; then it orders all edges in E according to the ordering of V , and for each edge (u, v) , adds it to $G^{\tilde{D}}$ if and only if the addition does not make the degree of either u or v exceed \tilde{D} .

This algorithm can be easily adapted to the online graph setting. However, it can be shown that the global sensitivity of the difference sequence is proportional to the total number of publications.

Lemma 6.4.2. *Let $f(G) = \text{highDeg}_\tau(G^{\tilde{D}})$. For D -bounded graphs, the corresponding difference sequence that ends at time T has global sensitivity at least $\tilde{D}T$ for any $\tilde{D} > \tau > 0$.*

Notice that $\tilde{D} > \tau$ is needed for $\text{highDeg}_\tau(G^{\tilde{D}})$ to be meaningful; otherwise, we would have $\text{highDeg}_\tau(G^{\tilde{D}}) = |V|$ for any G .

6.4.1.2 Degree Histogram

Degree histogram is another informative statistic of a graph. However, we can show that even for bounded graph, the sensitivity can scale quadratically with the degree bound. We use the generalized L_1 distance defined in Section 6.2.5 as the distance metric for the global sensitivity.

Lemma 6.4.3. *Let $f(G) = \text{hist}(G)$. For D -bounded graphs, the difference sequence corresponding to f has global sensitivity $4D^2 + 2D + 1$.*

6.4.2 Directed Graphs

For Directed graphs, we show similar results for the number of nodes with out-degree greater than or equal to a threshold τ and the out-degree histogram. Similar results can be obtained for in-degree.

6.4.2.1 Number of High Out-Degree Nodes

Recall that $\text{highOutDeg}_\tau(G)$ denotes the number of nodes in G with degree $\geq \tau$. We show below that for D_{in} -in-bounded graphs, the difference sequence corresponding to $\text{highOutDeg}_\tau(G)$ has global sensitivity $2D_{\text{in}} + 1$.

Lemma 6.4.4. *Let $f(G) = \text{highOutDeg}_\tau(G)$. For D_{in} -in-bounded graphs, the difference sequence corresponding to f has global sensitivity $2D_{\text{in}} + 1$.*

6.4.2.2 Out-Degree Histogram

We show that for bounded directed graphs, the sensitivity of the histogram scales quadratically with the degree bounds as well.

Lemma 6.4.5. *Let $f(G) = \text{histOut}(G)$. For $(D_{\text{in}}, D_{\text{out}})$ -bounded graphs, the difference sequence corresponding to f has global sensitivity $4D_{\text{out}}D_{\text{in}} + 2D_{\text{out}} + 1$.*

6.5 Functions of Subgraph Counts

In this section, we consider the count of some common directed and undirected subgraphs.

Popular subgraphs In undirected graphs, we consider three subgraphs. 1) an edge, including two nodes and the edge between them, 2) a triangle, including three nodes with edges between any two of them and 3) a k -star, including one center node c , k boundary nodes $\{b_1, \dots, b_k\}$ and edges $\{(c, b_i) : i \in [k]\}$. Table 6.2 summarizes these subgraphs.

In directed graphs, we consider five subgraphs. 1) an edge, including two nodes and an directed edge between them, 2) triangle I, including nodes $\{v_1, v_2, v_3\}$ and edges $\{(v_1, v_2), (v_2, v_3), (v_3, v_1)\}$, 3) triangle II, including nodes $\{v_1, v_2, v_3\}$ and edges $\{(v_1, v_2), (v_1, v_3), (v_2, v_3)\}$, 4) an out- k -star, including one center node c , k boundary nodes $\{b_1, \dots, b_k\}$ and edges $\{(c, b_i) : i \in [k]\}$, 5) an in- k -star, including one center node c , k boundary nodes $\{b_1, \dots, b_k\}$ and edges $\{(b_i, c) : i \in [k]\}$. Table 6.3 summarizes these subgraphs.

Table 6.2. Subgraphs in undirected graphs.









Edge	Triangle	k -star
S_E^u	S_{Δ}^u	S_{\star}^u
		

Table 6.3. Subgraphs in directed graphs.

Edge	Triangle I	Triangle II	Out- k -star	In- k -star
S_E^d	$S_{\Delta^1}^d$	$S_{\Delta^2}^d$	$S_{\star^o}^d$	$S_{\star^i}^d$
				

6.5.1 Undirected Graphs

First, we present a general result that applies to any undirected subgraph S . Recall that $S(G)$ denotes the total number of copies of S in graph G .

Lemma 6.5.1. *Given any undirected subgraph S , if $S(G)$ changes by at most S_+ with an additional node with degree D (and the corresponding edges), then the difference sequence corresponding to $S(\cdot)$ has global sensitivity S_+ for any D -bounded graph G .*

Now we show the values of S_+ for the subgraphs listed in Table 6.2.

Lemma 6.5.2. *Given the degree bound D , the value of S_+ for some common subgraphs are:*

1. for S_E^u , $S_+ = D$;

2. for S_{Δ}^u , $S_+ = \binom{D}{2}$;
3. for S_{\star}^u , $S_+ = D \binom{D-1}{k-1} + \binom{D}{k}$.

6.5.2 Directed Graphs

Again, we first present a lemma that applies to any subgraph S , and then show the values of S_+ , the maximum change in the subgraph count caused by an additional node, for the subgraphs in Table 6.3.

Lemma 6.5.3. *Given any directed subgraph S , if $S(G)$ changes by at most S_+ with an additional node with D_{in} in-degree and D_{out} out-degree (and the corresponding edges), then the difference sequence corresponding to $S(\cdot)$ has global sensitivity S_+ for any (D_{in}, D_{out}) -bounded graph G .*

Lemma 6.5.4. *Given degree bounds D_{in} and D_{out} , the value of S_+ for some common subgraphs are:*

1. for S_E^d , $S_+ = D_{in} + D_{out}$;
2. for $S_{\Delta 1}^d$, $S_+ = D_{in}D_{out}$;
3. for $S_{\Delta 2}^d$, $S_+ = \binom{D_{in}+D_{out}}{2}$;
4. for $S_{\star o}^d$, $S_+ = D_{in} \binom{D_{out}-1}{k-1} + \binom{D_{out}}{k}$;
5. for $S_{\star i}^d$, $S_+ = D_{out} \binom{D_{in}-1}{k-1} + \binom{D_{in}}{k}$.

6.6 Experiments

We next demonstrate the practical applicability of the proposed algorithm by comparing it with some natural baselines. In particular, we investigate the following questions:

1. What is the utility offered by SENSDIFF as a function of the privacy parameter ϵ and the number of releases?

2. How does its utility compare with existing baselines, such as composition across time steps, and composition coupled with graph projection?

These questions are considered in the context of five datasets – two synthetic and three real online graphs. We consider two versions of each dataset – directed and undirected, and two graph statistics – the number of high-degree nodes and the number of edges.

6.6.1 Methodology

6.6.1.1 Baseline Algorithms

We consider two natural baselines based on sequential differential privacy composition – COMPOSE- D -BOUNDED and COMPOSE-PROJECTION. COMPOSE- D -BOUNDED considers each graph G_t in the sequence separately, and adds noise to $f(G_t)$ that is proportional to its D -bounded global sensitivity divided by ϵ/T . Here ϵ is the privacy parameter and T is the number of releases. COMPOSE-PROJECTION uses a state-of-the-art projection algorithm – the one proposed in [35] – to project each G_t into \tilde{G}_t , and releases $f(\tilde{G}_t)$ after adding noise proportional to its global sensitivity divided by ϵ/T .

There are two other natural approaches. The first is to compute $f(\mathcal{G})$ and add noise proportional to its D -bounded global sensitivity divided by ϵ ; the second is to use the projection algorithm to obtain a sequence of projected graphs $\tilde{\mathcal{G}} = (\tilde{G}_1, \dots, \tilde{G}_T)$, compute $f(\tilde{\mathcal{G}})$ and add noise proportional to its global sensitivity divided by ϵ . However, we can show that the utility of either of these approaches is guaranteed to be at most that of COMPOSE- D -BOUNDED and COMPOSE-PROJECTION; details are omitted due to space constraints.

6.6.1.2 Choice of Parameters

SENSDIFF and COMPOSE- D -BOUNDED both require an a-priori bound on the graph degree. We set this bound to be the actual maximum degree rounded up to the nearest 5-th integer.

COMPOSE-PROJECTION requires a projection threshold \tilde{D} (or, \tilde{D}_{in} and \tilde{D}_{out} for directed

graphs). This parameter is chosen by parameter tuning – we pick the parameter value out of a predetermined list that leads to the lowest error. Note that for the sake of fairness, we do not allocate any extra privacy budget to parameter tuning, which would be the case in reality; thus our estimate of the performance of COMPOSE-PROJECTION is *optimistic*.

The threshold τ in the high-degree or high-out-degree nodes experiments is set to be the 90-th percentile of the (non-private) degree distribution.

6.6.2 Datasets

We use five datasets – three real and two synthetic. In addition to the standard directed version of each dataset, we also consider an undirected version that is obtained by ignoring the edge directions. A brief summary of these datasets is presented in Table 6.4.

Table 6.4. Summary of common properties of the graph datasets. Max degree refers to the undirected version of the graph while max in-degree and max out-degree refer to the directed version.

	# nodes	# edges	timespan (year)	max degree	max in-degree	max out-degree
HIV transmission	1660	456	21	13	12	8
Patent citation	91614	475427	15	247	211	246
Paper citation	9038	5249	24	36	19	36
Synthetic I	1990	665	20	5	1	4
Synthetic II	1088	588	20	6	1	5

6.6.2.1 Real Data Sets

HIV transmission graph This is a graph of potential HIV transmissions where a node represents a patient and an edge connects two patients whose viral sequences have high similarity. An edge thus represents a plausible transmission; the graph also has spurious edges that may correspond to a patient transmitting the disease to multiple others within a short period of time. About two-thirds of the patients have an estimated date of infection (EDI) ranging from 1996 to 2016, which is taken as the time stamp of the corresponding node. For the remaining nodes, EDI could not be estimated as the patient was admitted long after infection; we set the corresponding

time stamps as 1995.

Patents citation graph The patents citation graph [67] contains all US patents granted between 1963 and 1999, and all citations made by patents from 1975 to 1999. A patent u 's citing patent v naturally yields a directed edge from v to u . We pick all patents under subcategory *Computer Hardware & Software* (indexed 22) to form both a directed and an undirected graph, and publish statistics from years 1985 to 1999.

Paper citation graph This is a graph of articles and their citations from ACL derived by [87]; each article has a recorded publication date from 1975 to 2013. We select the positive citations, where an edge from v to u implies that u endorsed the article v , and publish statistics from 1990 to 2013.

6.6.2.2 Synthetic Data Sets

In addition to the real data sets, we consider two synthetic graphs that are generated from two separate disease transmission models.

Synthetic disease transmission graph I This is a synthetic graph of disease transmissions based on the Barabási–Albert preferential attachment model. In the Barabási–Albert model, there are m_0 initial nodes, followed by a number of nodes that arrive sequentially. A node on arrival connects to k existing nodes, with a higher chance of connecting to nodes with higher degree.

We make three modifications to this model so that the generated graph is a more realistic disease transmission network. First, we assume there is a total of Y years with n nodes added per year; each node has a time stamp – its year of arrival – and the initial nodes have time stamp 0. All edges are directed from nodes with lower time stamps to those with higher time stamps. Second, to model the large number of isolated nodes that exist in real disease transmission networks, we ensure that each new node is isolated with probability proportional to a parameter P_{isolated} . Third, in practice, an infected individual is usually less likely to spread infection as time passes due to treatment or death. We build this property into the model by adding

an extra decaying factor to the connection probability, i.e., the probability of a new node’s connecting to an existing node v is proportional to $\deg(v) \times (\text{current_time} - v.\text{time} + 1)^{-c}$ (or $\text{out-deg}(v) \times (\text{current_time} - v.\text{time} + 1)^{-c}$ in directed graph) where c is the decay parameter. For our experiments, we generate a graph with parameters $P_{\text{isolated}} = 0.5$, $k = 1$, $m_0 = 500$, $n = 70$, $Y = 20$ and $c = 1$.

Synthetic disease transmission graph II This is a synthetic disease transmission graph drawn from the popular SIR model [81] of infection overlaid on an underlying Barabási–Albert social network. In the SIR model, a node or individual has three statuses – susceptible (S), infectious (I) and recovered (R). The infectious individuals transmit the disease to susceptible individuals through social links with a transmission probability P_t . With probability P_r , an infectious individual can recover; once recovered, an individual will not get infected again.

We generate an undirected social network $G_{\text{interact}} = (V, E)$ from the Barabási–Albert model, where a node represents a person and an edge a social interaction. We then simulate the transmission process as follows. Initially, every node is susceptible (S) except for n_0 randomly picked infectious (I) nodes. At time step t , an infectious node changes status to recovered (R) with probability P_r ; then, each node u that is still infectious infects each one of its neighbors in G_{interact} with probability $P_i / \deg_{G_{\text{interact}}}(u)$. This gives us a transmission graph where a directed edge is a disease transmission. The time stamp of each node is the time when it is infected. We note that any node in the social interaction graph that has never been infected will not appear in the transmission graph. For our experiments, we use the parameters $P_r = 0.1$, $P_i = 0.18$ and a underlying interaction graph G_{interact} of size 10000 with attachment parameter $k = 2$.

6.6.3 Results

We compare SENSDIFF with the baseline algorithms COMPOSE- D -BOUNDED and COMPOSE-PROJECTION under varying ϵ – the privacy parameter and varying T – the number of releases. We measure utility by the relative L_1 error $\sum_{t=1}^T |\mathcal{A}(G_t) - f(G_t)| / f(G_t)$, where $f(G_t)$ is the non-private statistic and $\mathcal{A}(G_t)$ is the value estimated by an algorithm.

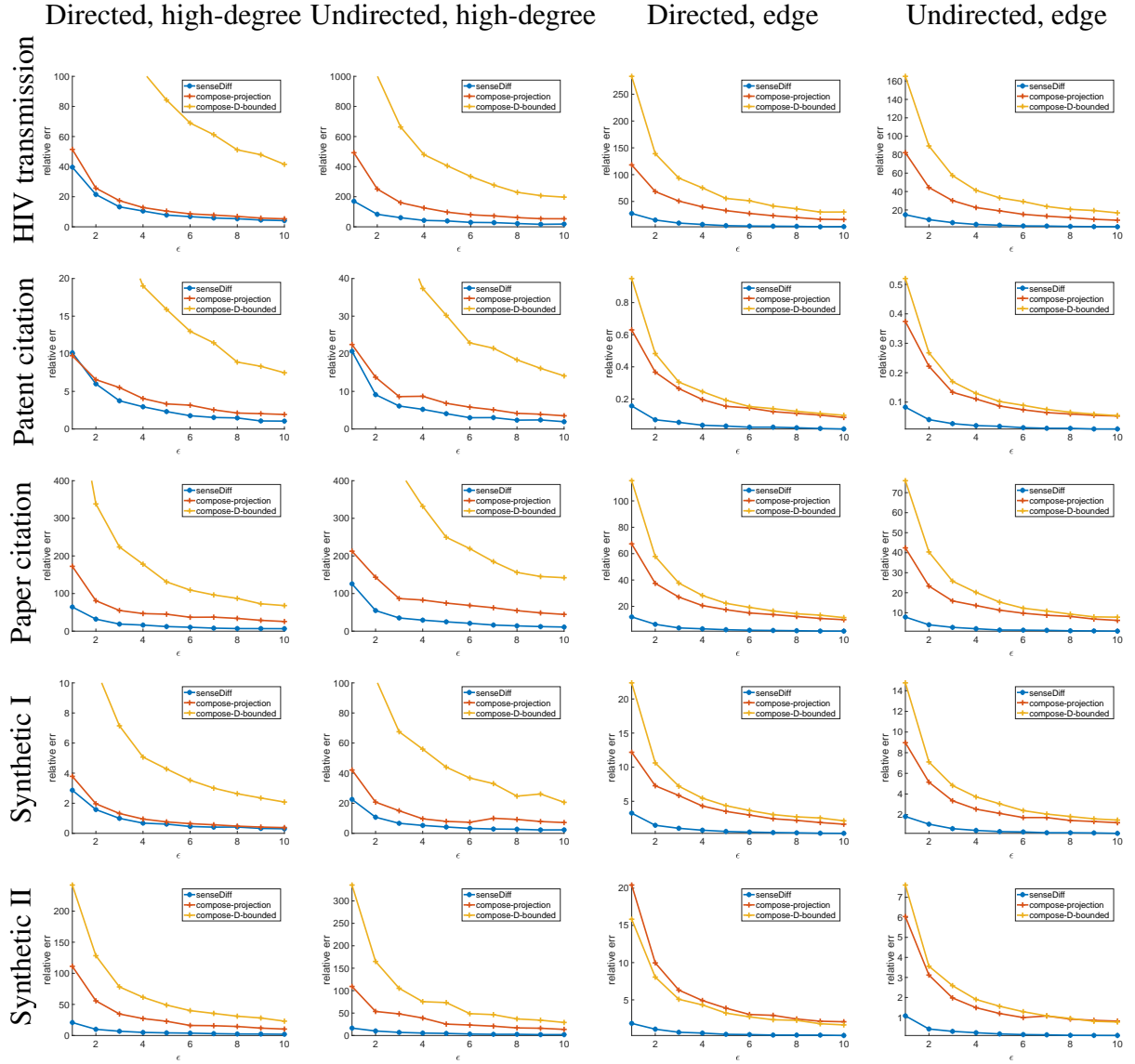


Figure 6.1. L_1 error vs. privacy parameter ϵ . Publish every 1 year. Averaged over 100 runs.

Figure 6.1 shows the privacy-utility tradeoffs of the three algorithms for the number of high-degree nodes (1st and 2nd columns) and number of edges (3rd and 4th columns) across all datasets (different rows). There is a clear trend of decreasing error as ϵ increases for all algorithms; however, SENSEDIFF yields smaller error compared to the other two baselines in all the cases. We point out that these results are overly optimistic for COMPOSE-PROJECTION – as we do not spend any privacy budget tuning the projection thresholds.

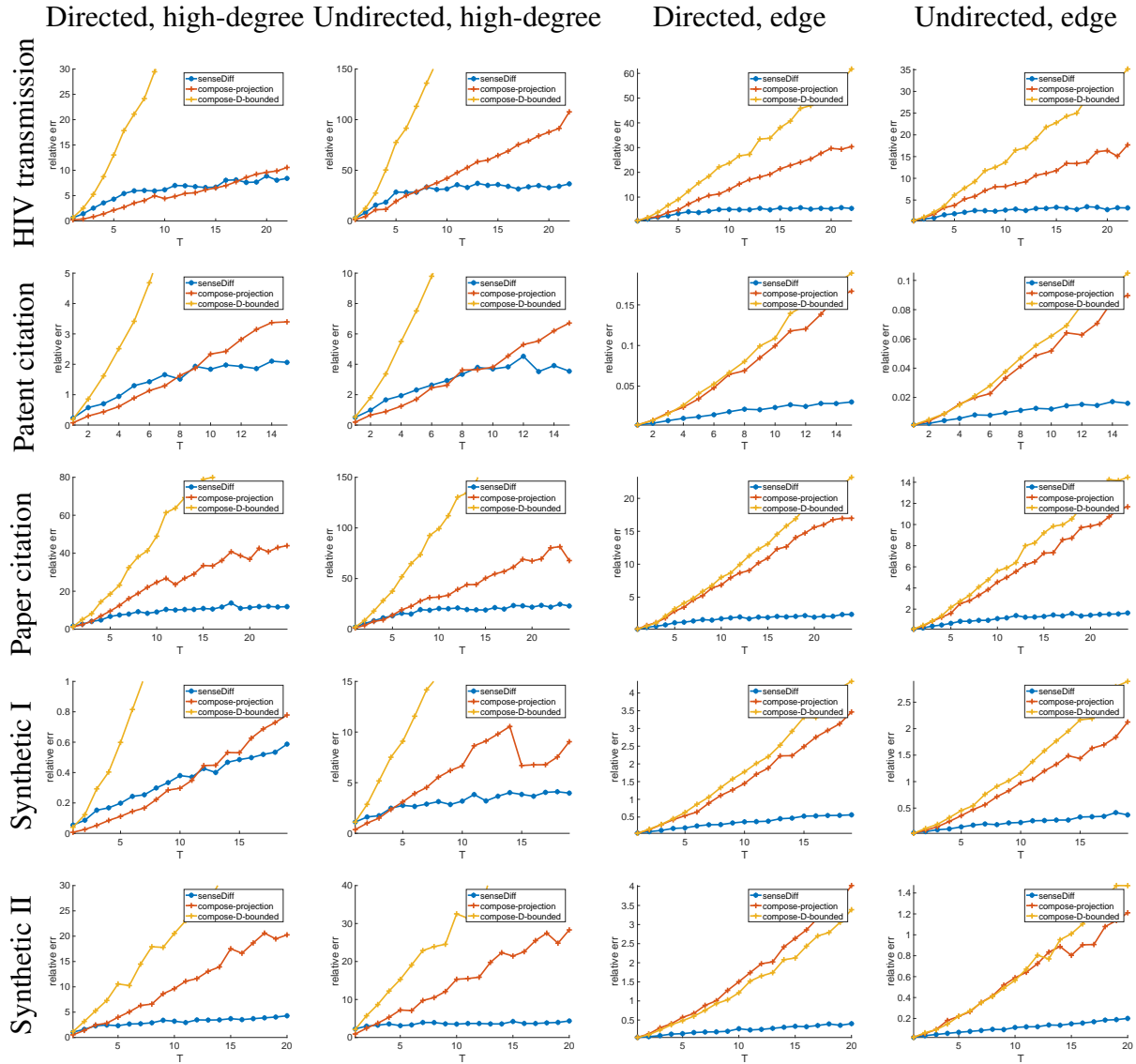


Figure 6.2. L_1 error vs. number of publications T . $\epsilon = 5$. Averaged over 100 runs.

Figure 6.2 presents the utility of all algorithms across different datasets (different rows) and statistics (different columns) as a function of the number of releases T . We fix ϵ to be 5. We see that the relative error increases as T increases for all algorithms, across all datasets and statistics. Similar to the previous experiment, SENSEDIFF achieves lower errors in most cases for the number of high-degree nodes and significantly lower errors in all cases for the number of edges. In addition, we observe that the error of SENSEDIFF increases at a smaller rate with increasing number of releases compared to the two baselines. This implies that SENSEDIFF

achieves better utility for large T , and confirms the theoretical arguments in Section 6.3.

Reconsidering the two questions proposed in the beginning of the section, we conclude that `SENSDIFF` offers better utility under a wide range of ϵ and T compared to both baselines. `COMPOSE- D -BOUNDED` yields the worst utility across all datasets for both statistics, which is to be expected as it does not take advantage of either projections or additional properties of the graph sequence. `SENSDIFF` outperforms `COMPOSE-PROJECTION` in most cases, and has a more significant advantage for large T .

6.7 Conclusion

In summary, we present a general algorithm for continually releasing statistics of a graph sequence. Our proposed algorithm exploits the difference sequence of the statistics, which has lower sensitivity compared to the original sequence, to achieve improved utility. We derive the global sensitivity of the difference sequences for common statistics including degree statistics and subgraph counts for bounded-degree graphs. Evaluations on real and synthetic graphs demonstrate the practical applicability of the proposed algorithm by showing that it outperforms two natural baselines over a wide range of parameters. In particular, the proposed algorithm is much less sensitive with respect to the number of releases compared with the baselines.

Our work is thus a step towards privacy-preserving analysis of online graphs, and has the potential to lead to insights in epidemiology of stigma-inducing diseases such as HIV, while still preserving the patient privacy.

Acknowledgments

The work was sponsored by NIH under MH100974, AI106039, ONR under N00014-16-1-261, and NSF under IIS 1253942.

This chapter is based on “Differentially Private Continual Release of Graph Statistics” by Shuang Song, Sanjay Mehta, Staal A. Vinterbo, Susan Little and Kamalika Chaudhuri which

is under submission. The dissertation author was the primary author of the paper.

Chapter 7

Pufferfish Privacy Mechanisms for Correlated Data

7.1 Overview

Many modern machine learning applications, such as those involving healthcare, power usage and building management, also increasingly involve a different setting – correlated data – privacy issues in which are not as well-understood. Consider, for example, a simple time-series application – measurement of physical activity of a *single subject* across time. The goal here is to release aggregate statistics on the subject’s activities over a long period (say a week) while hiding the evidence of an activity at any specific instant (say, 10:30am on Jan 4). If the measurements are made at small intervals, then the records form a *highly correlated time-series* as human activities change slowly over time.

What is a good notion of privacy for this example? Since the data belongs to a *single* subject, differential privacy is not directly applicable; however, a modified version called *entry-privacy* [70] applies. Entry-privacy ensures that the inclusion of a single time-series entry does not affect the probability of any outcome by much. It will add noise with standard deviation ≈ 1 to each bin of the histogram of activities. While this is enough for independent activities, it is insufficient to hide the evidence of correlated activities that may continue for several time points. A related notion is group differential privacy [48], which extends the definition to participation of *entire groups* of correlated individuals or entries. Here, all entries are correlated, and hence

group differential privacy will add $\approx O(T)$ noise to a histogram over T measurements, thus destroying all utility. Thus to address privacy challenges in this kind of data, we need a different privacy notion.

A generalized version of differential privacy called Pufferfish was proposed by [84]. In Pufferfish, privacy requirements are specified through three components – \mathcal{S} , a set of secrets that represents what may need to be hidden, \mathcal{Q} , a set of secret pairs that represents pairs of secrets that need to be indistinguishable to the adversary and Θ , a class of distributions that can plausibly generate the data. Privacy is provided by ensuring that the secret pairs in \mathcal{Q} are indistinguishable when data is generated from any $\theta \in \Theta$. In the time-series example, \mathcal{S} is the set of activities at each time t , and secret pairs are all pairs of the form (Activity a at time t , Activity b at time t). Assuming that activities transition in a Markovian fashion, Θ is a set of Markov Chains over activities. Pufferfish captures correlation in such applications effectively in two ways – first, unlike differential privacy, it can hide private values against correlation across multiple entries/individuals; second, unlike group privacy, it also allows utility in cases where a large number of individuals or entries are correlated, yet the “average amount” of correlation is low.

Because of these properties, we adopt Pufferfish as our privacy definition. To bolster the case for Pufferfish, we provide a general result showing that even if the adversary’s belief $\tilde{\theta}$ lies outside the class Θ , the resulting loss in privacy is low if $\tilde{\theta}$ is close to Θ .

The main challenge in using Pufferfish is a lack of suitable mechanisms. While mechanisms are known for specific instantiations [84, 74, 150], there is no mechanism for general Pufferfish. In this chapter, we provide the first mechanism, called the Wasserstein Mechanism, that can be adopted to any general Pufferfish instantiation. Since this mechanism may be computationally inefficient, we consider the case when correlation between variables can be described by a Bayesian network, and the goal is to hide the private value of each variable. We provide a second mechanism, called the Markov Quilt Mechanism, that can exploit properties of the Bayesian network to reduce the computational complexity. As a case study, we derive a

simplified version of the mechanism for the physical activity measurement example. We provide privacy and utility guarantees, establish composition properties, and finally demonstrate the practical applicability of the mechanism through experimental evaluation on synthetic as well as real data. Specifically, our contributions are as follows:

- We establish that when we guarantee Pufferfish privacy with respect to a distribution class Θ , but the adversary’s belief $\tilde{\theta}$ lies outside Θ , the resulting loss in privacy is small when $\tilde{\theta}$ is close to Θ .
- We provide the first mechanism that applies to any Pufferfish instantiation and is a generalization of the Laplace mechanism for differential privacy. We call this the Wasserstein Mechanism.
- Since the above mechanism may be computationally inefficient, we provide a more efficient mechanism called the Markov Quilt Mechanism when correlation between entries is described by a Bayesian Network.
- We show that under certain conditions, applying the Markov Quilt Mechanism multiple times over the same dataset leads to a gracefully decaying privacy parameter. This makes the mechanism particularly attractive as Pufferfish privacy does not always compose [84].
- We derive a simplified and computationally efficient version of this mechanism for time series applications such as physical activity monitoring.
- Finally, we provide an experimental comparison between Markov Quilt Mechanism and standard baselines as well as some concurrent work [66]; experiments are performed on simulated data, a moderately-sized real dataset ($\approx 10,000$ observations per person) on physical activity, and a large dataset (over 1 million observations) on power consumption.

7.1.1 Related Work

There is a body of work on differentially private mechanisms [109, 97, 26, 27, 46] – see surveys [125, 48]. As we explain earlier, differential privacy is not the right formalism for the

kind of applications we consider. A related framework is coupled-worlds privacy [11]; while it can take data distributions into account through a distribution class Θ , it requires that the participation of a single individual or entry does not make a difference, and is not suitable for our applications. We remark that while mechanisms for *specific* coupled-worlds privacy frameworks exist, there is also no *generic* coupled-worlds privacy mechanism.

This chapter instead uses Pufferfish, a recent generalization of differential privacy [84] (see Chapter 2 for more details). [84, 74] provide some specific instances of Pufferfish frameworks along with associated privacy mechanisms; they do not provide a mechanism that applies to any Pufferfish instance, and their examples do not apply to Bayesian networks. [150] uses a modification of Pufferfish, where instead of a distribution class Θ , they consider a single generating distribution θ . They consider the specific case where correlations can be modeled by Gaussian Markov Random Fields, and thus their work also does not apply to our physical activity monitoring example. [91] designs Pufferfish privacy mechanisms for distribution classes that include Markov Chains. Their mechanism adds noise proportional to a parameter ρ that measures correlation between entries. However, they do not specify how to calculate ρ as a function of the distribution class Θ , and as a result, their mechanism cannot be implemented when only Θ is known. [149] releases time-varying location trajectories under differential privacy while accounting for temporal correlations using a Hidden Markov Model with publicly known parameters. Finally, [82] apply Pufferfish to smart-meter data; instead of directly modeling correlation through Markov models, they add noise to the wavelet coefficients of the time series corresponding to different frequencies.

In concurrent work, [66] provide an alternative algorithm for Pufferfish privacy when data can be written as $X = (X_1, \dots, X_n)$ and the goal is to hide the value of each X_i . Their mechanism is less general than the Wasserstein Mechanism, but applies to a broader class of models than the Markov Quilt Mechanism. In Section 7.6, we experimentally compare their method with ours for Markov Chains and show that our method works for a broader range of distribution classes than theirs.

Finally, there has also been some previous work on differentially private time-series release [93, 57, 117]; however, they relate to aggregates over trajectories from a large number of people, unlike trajectories of single subjects as in our work.

7.2 The Setting

To motivate our privacy framework, we use two applications – physical activity monitoring of a single subject and flu statistics in a social network. We begin by describing them at a high level, and provide more details in Section 7.2.1.

Example 1: Physical Activity Monitoring. The dataset consists of a time-series $X = \{X_1, X_2, \dots, X_T\}$ where record X_t denotes a discrete physical activity (e.g, running, sitting, etc) of a subject at time t . Our goal is to release (an approximate) histogram of activities over a period (say, a week), while preventing an adversary from inferring the subject’s activity at a specific time (say, 10:30am on Jan 4).

Example 2: Flu Status. The dataset consists of a set of records $X = \{X_1, \dots, X_n\}$, where record X_i , which is 0 or 1, represents person i ’s flu status. The goal is to release (an approximation to) $\sum_i X_i$, the number of infected people, while ensuring privacy against an adversary who wishes to detect whether a particular person Alice in the dataset has flu. The dataset includes people who interact socially, and hence the flu statuses are highly correlated. Additionally, the decision to participate in the dataset is made at a group-level (for example, workplace-level or school-level), so individuals do not control their participation.

7.2.1 The Privacy Framework

The privacy framework of our choice is Pufferfish [84], an elegant generalization of differential privacy [45]. A detailed review of Pufferfish privacy framework can be found in Chapter 2. Recall that a Pufferfish framework consists of three parameters: a set of secrets \mathcal{S} , a set of secret pairs \mathcal{Q} , and a set of data distributions \mathcal{Q} . An algorithm \mathcal{M} guarantees ϵ -Pufferfish privacy if the ratio between $P_{\mathcal{M},\theta}(\mathcal{M}(X) = w|_{s_i}, \theta)$ and $P_{\mathcal{M},\theta}(\mathcal{M}(X) = w|_{s_j}, \theta)$ is upper

bounded by e^ϵ for any secret pair $(s_i, s_j) \in \mathcal{Q}$, any distribution $\theta \in \Theta$ and any $w \in \text{Range}(\mathcal{M})$, where X denotes the dataset drawn from θ .

A related framework is Group Differential Privacy [48], which applies to datasets with groups of correlated records.

Definition 7.2.1 (Group Differential Privacy). *Let D be a dataset with n records, and let $\mathcal{G} = \{G_1, \dots, G_k\}$ be a collection of subsets $G_i \subseteq \{1, \dots, n\}$. A privacy mechanism \mathcal{M} is said to be ϵ -group differentially private with respect to \mathcal{G} if for all G_i in \mathcal{G} , for all pairs of datasets D and D' which differ in the private values of the individuals in G_i , and for all $S \subseteq \text{Range}(\mathcal{M})$, we have:*

$$P(\mathcal{M}(D) \in S) \leq e^\epsilon P(\mathcal{M}(D') \in S).$$

In other words, Definition 7.2.1 implies that the participation of each group G_i in the dataset does not make a substantial difference to the outcome of mechanism \mathcal{M} .

7.2.2 Examples

We next illustrate how instantiating these examples in Pufferfish would provide effective privacy and utility.

Example 1: Physical Activity Measurement. Let A be the set of all activities – for example, $\{\text{walking}, \text{running}, \text{sitting}\}$ – and let s_a^t denote the event that activity a occurs at time t – namely, $X_t = a$. In the Pufferfish instantiation, we set \mathcal{S} as $\{s_a^t : t = 1, \dots, T, a \in A\}$ – so the activity at each time t is a secret. \mathcal{Q} is the set of all pairs (s_a^t, s_b^t) for a, b in A and for all t ; in other words, the adversary cannot distinguish whether the subject is engaging in activity a or b at any time t for all pairs a and b . Finally, Θ is a set of time series models that capture how people switch between activities. A plausible modeling decision is to restrict Θ to be a set of Markov Chains $X_1 \rightarrow X_2 \rightarrow \dots \rightarrow X_T$ where each state X_t is an activity in A . Each such Markov Chain can be described by an initial distribution q and a transition matrix P . For example, when

there are three activities, Θ can be the set:

$$\left\{ \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.1 & 0.5 & 0.4 \\ 0.4 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.4 \end{bmatrix} \right), \left(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.9 & 0.1 & 0.0 \\ 0.0 & 0.9 & 0.1 \\ 0.1 & 0.0 & 0.9 \end{bmatrix} \right) \right\}.$$

In this example,

- Differential privacy does not directly apply since we have a single person's data.
- Entry differential privacy [70, 84] and coupled worlds privacy [11] add noise with standard deviation $\approx 1/\epsilon$ to each bin of the activity histogram. However, this does not erase the evidence of an activity at time t . Moreover, an activity record does not have its agency, and hence its not enough to hide its participation.
- As all entries are correlated, group differential privacy adds noise with standard deviation $\approx T/\epsilon$, where T is the length of the chain; this destroys all utility.

In contrast, in Section 7.4 we show an ϵ -Pufferfish mechanism which will add noise approximately equal to the mixing time over ϵ , and thus offer both privacy and utility for rapidly mixing chains.

Example 2: Flu Status over Social Network. Let s_0^i (resp. s_1^i) denote the event that person i 's flu status is 0 (resp. 1). In the Pufferfish instantiation, we let \mathcal{S} be the set $\{s_0^i, s_1^i : i = 1, \dots, n\}$ – so the flu status of each individual is a secret. \mathcal{Q} is $\{(s_0^i, s_1^i) : i = 1, \dots, n\}$ – so the adversary cannot tell if each person i has flu or not. Θ is a set of models that describe the spread of flu; a plausible $\theta \in \Theta$ is a tuple (G_θ, p_θ) where $G_\theta = (X, E)$ is a graph of personal interactions, and p_θ is a probability distributions over the connected components of G_θ . As a concrete example, G_θ could be an union of cliques C_1, \dots, C_k , and p_θ could be the following distribution on the number N of infected people in each clique C_i : $P(N = j) = e^{2j} / \sum_{i=0}^{|C_i|} e^{2i}$, $j = 0, \dots, |C_i|$.

Similarly, for the flu status example, we have:

- Both differential privacy and coupled worlds privacy add noise with standard deviation

$\approx 1/\epsilon$ to the number of infected people. This will hide whether an Alice participates in the data, but if flu is contagious, then it is not enough to hide evidence of Alice’s flu status. Note that unlike differential privacy, as the decision to participate is made at group level, Alice has no agency over whether she participates, and hence we cannot argue it is enough to hide her participation.

- Group differential privacy will add noise proportional to the size of the largest connected component; this may result in loss of utility if this component is large, even if the “average spread” of flu is low.

Again, we can achieve Pufferfish privacy by adding noise proportional to the “average spread” of flu which may be less noise than group differential privacy. For a concrete numerical example, see Section 7.3.

7.2.3 Guarantee Against Close Adveraries

A natural question is what happens when we offer Pufferfish privacy with respect to a distribution class Θ , but the adversary’s belief $\tilde{\theta}$ does not lie in Θ . Our first result is to show that the loss in privacy is not too large if $\tilde{\theta}$ is close to Θ conditioned on each secret $s_i \in S$, provided closeness is quantified according to a measure called max-divergence.

Definition 7.2.2 (Max-divergence). *Let p and q be two distributions with the same support. The max-divergence $D_\infty(p||q)$ between them is defined as:*

$$D_\infty(p||q) = \sup_{x \in \text{support}(p)} \log \frac{p(x)}{q(x)}.$$

For example, suppose p and q are distributions over $\{1, 2, 3\}$ such that p assigns probabilities $\{1/3, 1/2, 1/6\}$ and q assigns probabilities $\{1/2, 1/4, 1/4\}$ to 1, 2 and 3 respectively. Then, $D_\infty(p||q) = \log 2$. Max-divergence belongs to the family of Renyi-divergences [32], which also includes the popular Kullback-Leibler divergence.

Notation. Given a belief distribution θ (which may or may not be in the distribution class Θ), we use the notation $\theta_{|s_i}$ to denote the conditional distribution of θ given secret s_i .

Theorem 7.2.3. *Let \mathcal{M} be a mechanism that is ϵ -Pufferfish private with respect to parameters $(\mathcal{S}, \mathcal{Q}, \Theta)$ and suppose that Θ is a closed set. Suppose an adversary has beliefs represented by a distribution $\tilde{\theta} \notin \Theta$. Then,*

$$e^{-\epsilon-2\Delta} \leq \frac{P_{\mathcal{M},\theta}(\mathcal{M}(X) = w|s_i, \theta)}{P_{\mathcal{M},\theta}(\mathcal{M}(X) = w|s_j, \theta)} \leq e^{\epsilon+2\Delta}$$

where

$$\Delta = \inf_{\theta \in \Theta} \max_{s_i \in \mathcal{S}} \max \left(D_\infty(\tilde{\theta}_{|s_i} || \theta_{|s_i}), D_\infty(\theta_{|s_i} || \tilde{\theta}_{|s_i}) \right).$$

The conditional dependence on s_i cannot be removed unless additional conditions are met. To see this, suppose θ places probability mass $\{0.9, 0.05, 0.05\}$ and $\tilde{\theta}$ places probability mass $\{0.01, 0.95, 0.04\}$ on datasets D_1 , D_2 and D_3 respectively. In this case, we have $\max\{D_\infty(\theta||\tilde{\theta}), D_\infty(\tilde{\theta}||\theta)\} = \log 90$. Suppose now that conditioning on s_i tells us that the probability of occurrence of D_3 is 0, but leaves the relative probabilities of D_1 and D_2 unchanged. Then $\theta_{|s_i}$ places probability mass $\{0.9474, 0.0526\}$ while $\tilde{\theta}_{|s_i}$ places probability mass $\{0.0104, 0.9896\}$ on D_1 and D_2 respectively. In this case, $\max\{D_\infty(\theta_{|s_i}||\tilde{\theta}_{|s_i}), D_\infty(\tilde{\theta}_{|s_i}||\theta_{|s_i})\} = \log 91.0962$ which is larger than $\max\{D_\infty(\theta||\tilde{\theta}), D_\infty(\tilde{\theta}||\theta)\}$.

7.2.4 Additional Notation

We conclude this section with some additional definitions and notation that we will use throughout the chapter.

Definition 7.2.4 (Lipschitz). *A query $F : [k]^n \rightarrow \mathbb{R}^k$ is said to be L -Lipschitz in L_1 norm if for all i and $X_i, X'_i \in [k]$,*

$$\|F(X_1, \dots, X_i, \dots, X_n) - F(X_1, \dots, X'_i, \dots, X_n)\|_1 \leq L.$$

This means that changing one record out of n in a dataset changes the L_1 norm of the query output by at most L .

For example, if $F(X_1, \dots, X_n)$ is a (vector-valued) histogram over records X_1, \dots, X_n , then F is 2-Lipschitz in the L_1 norm, as changing a single X_i can affect the count of at most two bins.

We use the notation $\text{Lap}(\sigma)$ to denote a Laplace distribution with mean 0 and scale parameter σ . Recall that this distribution has the density function: $h(x) = \frac{1}{2\sigma} e^{-|x|/\sigma}$. Additionally, we use the notation $\text{card}(S)$ to denote the cardinality, or, the number of items in a set S .

7.3 A General Mechanism

While a number of mechanisms for specific Pufferfish instantiations are known [84, 74], there is no mechanism that applies to any general Pufferfish instantiation. We next provide the first such mechanism. Given a dataset represented by random variable X , a Pufferfish instantiation $(\mathcal{S}, \mathcal{Q}, \Theta)$, and a query F that maps X into a scalar, we design mechanism \mathcal{M} that satisfies ϵ -Pufferfish privacy in this instantiation and approximates $F(X)$.

Our proposed mechanism is inspired by the Laplace mechanism in differential privacy; the latter adds noise to the result of the query F proportional to the sensitivity, which is the worst case distance between $F(D)$ and $F(D')$ where D and D' are two datasets that differ in the value of a single individual. In Pufferfish, the quantities analogous to D and D' are the distributions $P(F(X)|s_i, \theta)$ and $P(F(X)|s_j, \theta)$ for a secret pair (s_i, s_j) , and therefore, the added noise should be proportional to the worst case distance between these two distributions according to some metric.

Wasserstein Distances. It turns out that the relevant metric is the ∞ -Wasserstein distance – a modification of the Earthmover’s distance used in information retrieval and computer vision.

Definition 7.3.1 (∞ -Wasserstein Distance). *Let μ, ν be two probability distributions on \mathbb{R} , and let $\Gamma(\mu, \nu)$ be the set of all joint distributions with marginals μ and ν . The ∞ -Wasserstein*

distance between μ and ν is defined as:

$$W_\infty(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu, \nu)} \max_{(x, y) \in \text{support}(\gamma)} |x - y|. \quad (7.1)$$

Intuitively, each $\gamma \in \Gamma(\mu, \nu)$ is a way to *shift* probability mass between μ and ν ; the cost of a shift γ is: $\max_{(x, y) \in \text{support}(\gamma)} |x - y|$, and the cost of the min-cost shift is the ∞ -Wasserstein distance. It can be interpreted as the maximum “distance” that any probability mass moves while transforming μ to ν in the most optimal way possible. A pictorial example is shown in Figure 7.1.

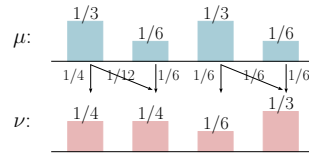


Figure 7.1. An example of ∞ -Wasserstein distance. The frequency distributions represent two measures μ and ν , and the arrows represent the joint distribution γ that minimizes the right hand side of (7.1). Here, $W_\infty(\mu, \nu) = 1$.

7.3.1 The Wasserstein Mechanism

The main intuition behind our mechanism is based on this interpretation. Suppose for some (s_i, s_j) and θ , we would like to transform $P(F(X)|s_i, \theta)$ to $P(F(X)|s_j, \theta)$. Then, the maximum “distance” that any probability mass moves is

$$W_{i,j,\theta} = W_\infty(P(F(X)|s_i, \theta), P(F(X)|s_j, \theta)),$$

and adding Laplace noise with scale $W_{i,j,\theta}/\epsilon$ to F will guarantee that the likelihood ratio of the outputs under s_i and s_j lies in $[e^{-\epsilon}, e^\epsilon]$. Iterating over all pairs $(s_i, s_j) \in \mathcal{Q}$ and all $\theta \in \Theta$ and taking the maximum over $W_{i,j,\theta}$ leads to a mechanism for the entire instantiation $(\mathcal{S}, \mathcal{Q}, \Theta)$.

The full mechanism is described in Algorithm 6, and its privacy properties in Theorem 7.3.2. Observe that when Pufferfish reduces to differential privacy, then the corresponding

Wasserstein Mechanism reduces to the Laplace mechanism; it is thus a generalization of the Laplace mechanism.

Algorithm 6. Wasserstein Mechanism (Dataset D , query F , Pufferfish framework $(\mathcal{S}, \mathcal{Q}, \Theta)$, privacy parameter ϵ)

for all $(s_i, s_j) \in \mathcal{Q}$ and all $\theta \in \Theta$ such that $P(s_i|\theta) \neq 0$ and $P(s_j|\theta) \neq 0$ **do**
 Set $\mu_{i,\theta} = P(F(X) = \cdot | s_i, \theta)$, $\mu_{j,\theta} = P(F(X) = \cdot | s_j, \theta)$. Calculate $W_\infty(\mu_{i,\theta}, \mu_{j,\theta})$
end for
Set $W = \sup_{(s_i, s_j) \in \mathcal{Q}, \theta \in \Theta} W_\infty(\mu_{i,\theta}, \mu_{j,\theta})$.
return $F(D) + Z$, where $Z \sim \text{Lap}(\frac{W}{\epsilon})$

Example. Consider a Pufferfish instantiation of the flu status application. Suppose that the dataset has size 4, and $\Theta = \{(G_\theta, p_\theta)\}$ where G_θ is a clique on 4 nodes, and p_θ is the following symmetric joint distribution on the number N of infected individuals:

i	0	1	2	3	4
$P(N = j)$	0.1	0.15	0.5	0.15	0.1

From symmetry, this induces the following conditional distributions:

j	0	1	2	3	4
$P(N = j X_i = 0)$	0.2	0.225	0.5	0.075	0
$P(N = j X_i = 1)$	0	0.075	0.5	0.225	0.2

In this case, the parameter W in Algorithm 6 is 2, and the Wasserstein Mechanism will add $\text{Lap}(2/\epsilon)$ noise to the number of infected individuals. As all the X_i 's are correlated, the sensitivity mechanism with Group Differential Privacy would add $\text{Lap}(4/\epsilon)$ noise, which gives worse utility.

7.3.2 Performance Guarantees

Theorem 7.3.2 (Wasserstein Privacy). *The Wasserstein Mechanism provides ϵ -Pufferfish privacy in the framework $(\mathcal{S}, \mathcal{Q}, \Theta)$.*

Utility. Because of the extreme generality of the Pufferfish framework, it is difficult to make general statements about the utility of the Wasserstein mechanism. However, we show that the phenomenon illustrated by the flu status example is quite general. When X is written as $X = (X_1, \dots, X_n)$ where X_i is the i -th individual's private value, and the goal is to keep each individual's value private, the Wasserstein Mechanism for Pufferfish never performs worse than the Laplace mechanism for the corresponding group differential privacy framework. The proof is in the full paper [132].

Theorem 7.3.3 (Wasserstein Utility). *Let $(\mathcal{S}, \mathcal{Q}, \Theta)$ be a Pufferfish framework, and let \mathcal{G} be the corresponding group differential privacy framework (so that \mathcal{G} includes a group G for each set of correlated individuals in Θ). Then, for a L -Lipschitz query F , the parameter W in the Wasserstein Mechanism is less than or equal to the global sensitivity of F in the \mathcal{G} -group differential privacy framework.*

7.4 A Mechanism for Bayesian Networks

The Wasserstein Mechanism, while general, may be computationally expensive. We next consider a more restricted setting where the dataset X can be written as a collection of variables $X = (X_1, \dots, X_n)$ whose dependence is described by a Bayesian network, and the goal is to keep the value of each X_i private. This setting is still of practical interest, as it can be applied to physical activity monitoring and power consumption data.

7.4.1 The Setting

Bayesian Networks. Bayesian networks are a class of probabilistic models that are commonly used to model dependencies among random variables or vectors, and include popular models such as Markov Chains and trees. A Bayesian network is described by a set of variables $X = \{X_1, \dots, X_n\}$ (where each X_i is a scalar or a vector) and a *directed acyclic graph* $G = (X, E)$ whose vertices are variables in X . Since G is directed acyclic, its edges E induce a

parent-child relationship parent among the nodes X . The probabilistic dependence on X induced by the network can be written as:

$$P(X_1, \dots, X_n) = \prod_i P(X_i | \text{parent}(X_i)).$$

For example, Figure 7.2 shows a Bayesian Network on 4 variables, whose joint distribution is described by:

$$P(X_1, X_2, X_3, X_4) = P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2, X_3).$$

A node X_i may have more than one parent, and as such these networks can describe complex probabilistic dependencies.

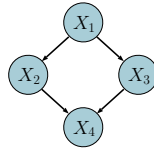


Figure 7.2. A Bayesian Network on four variables.

The Framework. Specifically, we assume that the dataset X can be written as $X = \{X_1, \dots, X_n\}$, where each X_i lies in a bounded domain $[k]$. Let s_a^i denote the event that X_i takes value a . The set of secrets is $\mathcal{S} = \{s_a^i : a \in [k], i \in [n]\}$, and the set of secret pairs is $\mathcal{Q} = \{(s_a^i, s_b^i) : a, b \in [k], a \neq b, i \in [n]\}$. We also assume that there is an underlying known Bayesian network $G = (X, E)$ connecting the variables. Each $\theta \in \Theta$ that describes the distribution of the variables is based on this Bayesian network G , but may have different parameters.

For example, the Pufferfish instantiation in Example 1 will fall into this framework, with $n = T$ and G a Markov Chain $X_1 \rightarrow X_2 \rightarrow \dots \rightarrow X_T$.

Notation. We use X with a lowercase subscript, for example, X_i , to denote a single node in G , and X with an uppercase subscript, for example, X_A , to denote a set of nodes in G . For a

set of nodes X_A we use the notation $\text{card}(X_A)$ to denote the number of nodes in X_A .

7.4.2 The Markov Quilt Mechanism

The main insight behind our mechanism is that if nodes X_i and X_j are “far apart” in G , then, X_j is largely independent of X_i . Thus, to obscure the effect of X_i on the result of a query, it is sufficient to add noise proportional to the number of nodes that are “local” to X_i plus a correction term to account for the effect of the distant nodes. The rest of the section will explain how to calculate this correction term, and how to determine how many nodes are local.

First, we quantify how much changing the value of a variable $X_i \in X$ can affect a set of variables $X_A \subset X$, where $X_i \notin X_A$ and the dependence is described by a distribution θ in a class Θ . To this end, we define the *max-influence* of a variable X_i on a set of variables X_A under a distribution class Θ as follows.

Definition 7.4.1 (Max-influence). *We define the max-influence of a variable X_i on a set of variables X_A under Θ as:*

$$e_{\Theta}(X_A|X_i) = \sup_{\theta \in \Theta} \max_{a, b \in [k]} \max_{x_A \in [k]^{\text{card}(X_A)}} \log \frac{P(X_A = x_A | X_i = a, \theta)}{P(X_A = x_A | X_i = b, \theta)}.$$

Here $[k]$ is the range of any X_j . The max-influence is thus the maximum max-divergence between the distributions $X_A|X_i = a, \theta$ and $X_A|X_i = b, \theta$ where the maximum is taken over any pair $(a, b) \in [k] \times [k]$ and any $\theta \in \Theta$. If X_A and X_i are independent, then the max-influence of X_i on X_A is 0, and a large max-influence means that changing X_i can have a large impact on the distribution of X_A . In a Bayesian network, the max-influence of any X_i and X_A can be calculated given the probabilistic dependence.

Our mechanism will attempt to find large sets X_A such that X_i has low max-influence on X_A under Θ . The naive way to do so is through brute force search, which takes time exponential in the size of G . We next show how structural properties of the Bayesian network G can be exploited to perform this search more efficiently.

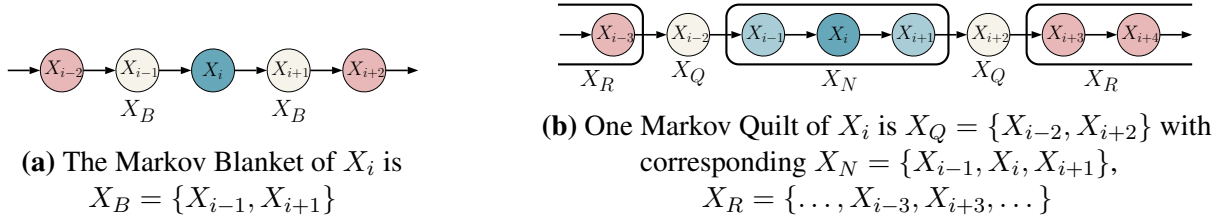


Figure 7.3. Illustration of Markov Blanket and Markov Quilt

Markov Blankets and Quilts. For this purpose, we provide a second definition that generalizes the Markov Blanket, a standard notion in probabilistic graphical models [86]. The Markov Blanket of a node X_u in a Bayesian network consists of its parents, its children and the other parents of its children, and the rest of the nodes in the network are independent of X_u conditioned on its Markov Blanket. We define its generalization, the Markov Quilt, as follows.

Definition 7.4.2 (Markov Quilt). *A set of nodes X_Q , $Q \subset [n]$ in a Bayesian network $G = (X, E)$ is a Markov Quilt for a node X_i if the following conditions hold:*

1. *Deleting X_Q partitions G into parts X_N and X_R such that $X = X_N \cup X_Q \cup X_R$ and $X_i \in X_N$.*
2. *For all $x_R \in [k]^{\text{card}(X_R)}$, all $x_Q \in [k]^{\text{card}(X_Q)}$ and for all $a \in [k]$, $P(X_R = x_R | X_Q = x_Q, X_i = a) = P(X_R = x_R | X_Q = x_Q)$.*

Thus, X_R is independent of X_i conditioned on X_Q .

Intuitively, X_R is a set of “remote” nodes that are far from X_i , and X_N is the set of “nearby” nodes; X_N and X_R are separated by the Markov Quilt X_Q . Observe that unlike Markov Blankets, a node can have many Markov Quilts. Figure 7.3 shows an example. We also allow the “trivial Markov Quilt” with $X_Q = \emptyset$, $X_N = X$ and $X_R = \emptyset$.

Suppose we would like to release the result of a L -Lipschitz scalar query F while protecting a node X_i . If we can find a Markov Quilt (X_N, X_Q, X_R) such that the max-influence of X_i on X_Q under Θ is at most δ , then, it is sufficient to add Laplace noise to F with scale parameter $L \cdot \text{card}(X_N) / (\epsilon - \delta)$. This motivates the following mechanism, which we call the Markov Quilt Mechanism. To protect X_i , we search over a set $S_{Q,i}$ of Markov Quilts for X_i and

pick the one which requires adding the least amount of noise. We then iterate over all X_i and add to F the maximum amount of noise needed to protect any X_i ; this ensures that the private values of *all nodes* are protected. Details are presented in Algorithm 7, and Theorem 7.4.3 establishes its privacy properties.

Algorithm 7. Markov Quilt Mechanism (Dataset D , L -Lipschitz query F , Pufferfish instantiation $(\mathcal{S}, \mathcal{Q}, \Theta)$, privacy parameter ϵ , Markov Quilts set $S_{Q,i}$ for each X_i)

```

for all  $X_i$  do
  for all Markov Quilts  $X_Q$  (with  $X_N, X_R$ ) in  $S_{Q,i}$  do
    if max-influence  $e_{\Theta}(X_Q|X_i) < \epsilon$  then
       $\sigma(X_Q) = \frac{\text{card}(X_N)}{\epsilon - e_{\Theta}(X_Q|X_i)}$  /*score of  $X_Q$ */
    else
       $\sigma(X_Q) = \infty$ 
    end if
  end for
   $\sigma_i = \min_{X_Q \in S_{Q,i}} \sigma(X_Q)$ 
end for
 $\sigma_{\max} = \max_i \sigma_i$ 
return  $F(D) + (L \cdot \sigma_{\max}) \cdot Z$ , where  $Z \sim \text{Lap}(1)$ 

```

Vector-Valued Functions. The mechanism can be easily generalized to vector-valued functions. If F is L -Lipschitz with respect to L_1 norm, then from Proposition 1 of [45], adding noise drawn from $L \cdot \sigma_{\max} \cdot \text{Lap}(1)$ to each coordinate of F guarantees ϵ -Pufferfish privacy.

Theorem 7.4.3 (Markov Quilt Mechanism Privacy). *If F is L -Lipschitz, and if each $S_{Q,i}$ contains the trivial quilt $X_Q = \emptyset$ (with $X_N = X$, $X_R = \emptyset$), then the Markov Quilt Mechanism preserves ϵ -Pufferfish privacy in the instantiation $(\mathcal{S}, \mathcal{Q}, \Theta)$ described in Section 7.4.1.*

7.4.3 Case Study: Markov Chains

Algorithm 7 can still be computationally expensive, especially if the underlying Bayesian network is complex and the set Θ is large and unstructured. In this section, we show that when the underlying graph is a Markov Chain, the mechanism can be made more efficient.

The Setting. We use the same setting as Example 1 that was described in detail in Section 7.2.2. We assume that there are k activities so that the state space $[k] = [k]$. Thus each

$\theta \in \Theta$ corresponds to a tuple (q_θ, P_θ) where q_θ is a $k \times 1$ vector that describes the distribution of the first state and P_θ is a $k \times k$ transition matrix.

A Running Example. As a concrete running example in this section, consider a dataset of $T = 100$ records represented by a Markov Chain $X_1 \rightarrow \dots \rightarrow X_{100}$ where each state X_i can take $k = 2$ values, 0 and 1. Also let:

$$\Theta = \left\{ \theta_1 = \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.9 & 0.1 \\ 0.4 & 0.6 \end{bmatrix} \right), \theta_2 = \left(\begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}, \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix} \right) \right\}$$

be the set of distributions. Suppose we set $\epsilon = 1$.

Sources of Inefficiency. There are three potential sources of computational inefficiency in the Markov Quilt Mechanism. First, searching over all Markov Quilts in a set is inefficient if there are a large number of quilts. Second, calculating the max-influence of a fixed quilt is expensive if the probabilistic dependence between the variables is complex, and finally, searching over all $\theta \in \Theta$ is inefficient if Θ is large and unstructured.

We show below how to mitigate the computational complexity of all three sources for Markov Chains. First we show how to exploit structural information to improve the computational efficiency of the first two. Next we propose an approximation using tools from Markov Chain theory that will considerably improve the computational efficiency of searching over all $\theta \in \Theta$. The improvement preserves ϵ -Pufferfish privacy, but might come at the expense of some additional loss in accuracy.

7.4.3.1 Exploiting Structural Information

First, we show how to exploit structural information to improve the running time.

Bounding the Number of Markov Quilts. Technically, the number of Markov Quilts for a node X_i in a Markov Chain of length T is exponential in T . Fortunately however, Lemma 7.4.4 shows that for the purpose of finding the quilt with the lowest score for any X_i , it is sufficient to search over only $O(T^2)$ quilts. The proof is in the full paper [132].

Lemma 7.4.4. *In the setting of Section 7.4.3, let the set of Markov Quilts $S_{Q,i}$ be as follows:*

$$S_{Q,i} = \{\{X_{i-a}, X_{i+b}\}, \{X_{i-a}\}, \{X_{i+b}\}, \emptyset \mid 1 \leq a \leq i-1, 1 \leq b \leq T-i\} \quad (7.2)$$

Consider any Markov Quilt (X_N, X_Q, X_R) for X_i that may or may not lie in $S_{Q,i}$. Then, the score of this quilt is greater than or equal to the score of some $(X_{N'}, X_{Q'}, X_{R'})$ in $S_{Q,i}$.

Calculating the max-influence. Consider the max-influence of a set of variables X_Q on a variable X_i under a fixed $\theta \in \Theta$. Calculating it may be expensive if the probabilistic dependency among the variables in X_Q and X_i is complex under θ . We show below that for Markov Chains, this max-influence may be computed relatively efficiently.

Suppose $X_Q = \{X_{i-a}, X_{i+b}\}$, and recall that a state can take values in $[k] = \{1, \dots, k\}$.

Then, we can write:

$$\begin{aligned} e_{\{\theta\}}(X_Q|X_i) &= \max_{x, x', x_{i-a}, x_{i+b} \in [k]} \log \frac{P(X_{i-a} = x_{i-a}, X_{i+b} = x_{i+b} | X_i = x, \theta)}{P(X_{i-a} = x_{i-a}, X_{i+b} = x_{i+b} | X_i = x', \theta)} \\ &= \max_{x, x' \in [k]} \left(\log \frac{P(X_i = x', \theta)}{P(X_i = x, \theta)} + \max_{x_{i+b} \in [k]} \log \frac{P(X_{i+b} = x_{i+b} | X_i = x, \theta)}{P(X_{i+b} = x_{i+b} | X_i = x', \theta)} \right. \\ &\quad \left. + \max_{x_{i-a} \in [k]} \log \frac{P(X_i = x | X_{i-a} = x_{i-a}, \theta)}{P(X_i = x' | X_{i-a} = x_{i-a}, \theta)} \right). \end{aligned} \quad (7.3)$$

(7.3) allows for efficient computation of the max-influence. Given an initial distribution and a transition matrix, for each x and x' pair, the first term in (7.3) can be calculated in $O(ik^2)$ time, and the second and third in $O((a+b)k^3)$ time each. Taking the maximum over all x and x' involves another $O(k^2)$ operations, leading to a total running time of $O(ik^2 + (a+b)k^3)$. Similar formulas can be obtained for quilts of the form $X_Q = \{X_{i-a}\}$ and $X_Q = \{X_{i+b}\}$. Note for the special case of the trivial Markov Quilt, the max-influence is 0, which takes $O(1)$ time to compute.

Algorithm MQMExact. Thus, using structural properties of Markov Chains gives a more efficient instantiation of the Markov Quilt Mechanism that we describe in Algorithm 8.

Algorithm 8. MQMExact(Dataset D , L -Lipschitz query F , Θ , privacy parameter ϵ , maximum Markov Quilt size parameter ℓ)

```

for all  $\theta \in \Theta$  do
  for all  $X_i$  do
     $S_{Q,i} = \{\{X_{i-a}, X_{i+b}\} : i-a > 0, i+b \leq T, a+b < \ell\} \cup \{\{X_{i-a}\} : i-a \geq T-\ell\}$ 
     $\cup \{\{X_{i+b}\} : i+b < \ell\} \cup \{\emptyset\}$     /*all quilts with end-points at distance  $\leq \ell$  plus the
    trivial quilt */
    for all Markov Quilts  $X_Q \in S_{Q,i}$  do
      Calculate  $e_{\{\theta\}}(X_Q|X_i)$  from (7.3)
      if  $e_{\{\theta\}}(X_Q|X_i) < \epsilon$  then
         $\sigma(X_Q) = \frac{\text{card}(X_N)}{\epsilon - e_{\{\theta\}}(X_Q|X_i)}$     /*score of  $X_Q$ */
      else
         $\sigma(X_Q) = \infty$ 
      end if
    end for
     $\sigma_i = \min_{X_Q \in S_{Q,i}} \sigma(X_Q)$ 
  end for
   $\sigma_{\max}^\theta = \max_i \sigma_i$ 
end for
 $\sigma_{\max} = \max_{\theta \in \Theta} \sigma_{\max}^\theta$ 
return  $F(D) + (L \cdot \sigma_{\max}) \cdot Z$ , where  $Z \sim \text{Lap}(1)$ 

```

We remark that to improve efficiency, instead of searching over all $O(T^2)$ Markov Quilts for a node X_i , we search only over Markov Quilts whose endpoints lie at a distance $\leq \ell$ from X_i . Since there are $O(\ell^2)$ such Markov Quilts, this reduces the running time if $\ell \ll T$.

Concretely, consider using Algorithm MQMExact on our running example with $\ell = T$. For θ_1 , a search over each X_i gives that X_8 has the highest score, which is 13.0219, achieved by Markov Quilt $\{X_3, X_{13}\}$. For θ_2 , X_6 has the highest score, 10.6402, achieved by the Markov Quilt $\{X_{10}\}$. Thus, the algorithm adds noise $Z \sim \text{Lap}(13.0219 \times L)$ to the exact query value.

Running Time Analysis. A naive implementation of Algorithm 8 would run in time $O(T\ell^2|\Theta|(\ell k^3 + Tk^2))$. However, we can use the following observations to speed up the algorithm considerably.

First, observe that for a fixed θ , we can use dynamic programming to compute and store *all* the probabilities $P(X_i|\theta)$, $P(X_i|X_{i-a})$ and $P(X_{i+b}|X_i)$ together in time $O(Tk^3)$. Second,

note that once these probabilities are stored, the rest is a matter of maximization; for fixed X_i , Markov Quilt X_Q and θ , we can calculate $e_{\{\theta\}}(X_Q|X_i)$ from (7.3) in time $O(k^3)$; iterating over all X_i , and all $O(\ell^2)$ Markov Quilts for X_i gives a running time of $O(Tk^3 + T\ell^2k^3)$. Finally, iterating over all $\theta \in \Theta$ gives a final running time of $O(T\ell^2k^3|\Theta|)$ which much improves the naive implementation.

Additional optimizations may be used to improve the efficiency even further. In the full paper [132], we show that if Θ includes all possible initial distributions for a set of transition matrices, then we can avoid iterating over all initial distributions by a direct optimization procedure. Finally, another important observation is that when the initial distribution under θ is the stationary distribution of the Markov Chain – as can happen when data consists of samples drawn from a Markov process in a stable state, such as, household electricity consumption in a steady state – then for any $i \in [a, T - b]$, the max-influence $e_{\{\theta\}}(\{X_{i-a}, X_{i+b}\}|X_i)$ depends only on a and b and is independent of i . This eliminates the need to conduct a separate search for each i , and further improves efficiency by a factor of T .

7.4.3.2 Approximating the max-influence

Unfortunately, Algorithm 8 may still be computationally inefficient when Θ is large. In this section, we show how to mitigate this effect by computing an *upper bound* on the max-influence under a set of distributions Θ in closed form using tools from Markov Chain theory. Note that now we can no longer compute an exact score; however, since we use an *upper bound* on the score, the resulting mechanism remains ϵ -Pufferfish private.

Definitions from Markov Chain Theory. We begin with reviewing some definitions and notation from Markov Chain theory that we will need. For a Markov Chain with transition matrix P_θ , we use π_θ to denote its stationary distribution [6]. We define the time reversal Markov Chain corresponding to P_θ as follows.

Definition 7.4.5 (time-reversal). *Let P_θ be the transition matrix of a Markov Chain θ . If π_θ is the stationary distribution of θ , then, the corresponding time-reversal Markov Chain is defined*

as the chain with transition matrix P_θ^* where:

$$P_\theta^*(x, y)\pi_\theta(x) = P_\theta(y, x)\pi_\theta(y).$$

Intuitively, P_θ^* is the transition matrix when we run the Markov process described by P_θ backwards from X_T to X_1 .

In our running example, for both θ_1 and θ_2 , the time-reversal chain has the same transition matrix as the original chain, i.e., $P_{\theta_1}^* = P_{\theta_1}$, $P_{\theta_2}^* = P_{\theta_2}$.

We next define two more parameters of a set Θ of Markov Chains and show that an upper bound to the max-influence under Θ can be written as a function of these two parameters. First, we define π_Θ^{\min} as the minimum probability of any state under the stationary distribution π_θ of any Markov Chain $\theta \in \Theta$. Specifically,

$$\pi_\Theta^{\min} = \min_{x \in [k], \theta \in \Theta} \pi_\theta(x). \quad (7.4)$$

In our running example, the stationary distribution of the transition matrix for θ_1 is $[0.8, 0.2]$ and thus $\pi_{\{\theta_1\}}^{\min} = 0.2$; similarly, θ_2 has stationary distribution $[0.6, 0.4]$ and thus $\pi_{\{\theta_2\}}^{\min} = 0.4$. We have $\pi_\Theta^{\min} = 0.2$.

Additionally, we define g_Θ as the minimum eigengap of $P_\theta P_\theta^*$ for any $\theta \in \Theta$. Formally,

$$g_\Theta = \min_{\theta \in \Theta} \min\{1 - |\lambda| : P_\theta P_\theta^* x = \lambda x, |\lambda| < 1\}. \quad (7.5)$$

In our running example, the eigengap for both $P_{\theta_1} P_{\theta_1}^*$ and $P_{\theta_2} P_{\theta_2}^*$ is 0.75, and thus we have $g_\Theta = 0.75$.

Algorithm MQMApprox. The following Lemma characterizes an upper bound of the max-influence of a variable X_i on a X_Q under a set Θ of Markov Chains as a function of π_Θ^{\min} and g_Θ .

Lemma 7.4.6. *Suppose the Markov Chains induced by each $P_\theta \in \Theta$ are irreducible and aperiodic with $\pi_\theta > 0$ and $g_\theta > 0$. If $a, b \geq \frac{2 \log(1/\pi_\theta^{\min})}{g_\theta}$, then the max-influence $e_\Theta(X_Q|X_i)$ of a Markov Quilt $X_Q = \{X_{i-a}, X_{i+b}\}$ on a node X_i under Θ is at most:*

$$\log \left(\frac{\pi_\Theta^{\min} + \exp(-g_\Theta b/2)}{\pi_\Theta^{\min} - \exp(-g_\Theta b/2)} \right) + 2 \log \left(\frac{\pi_\Theta^{\min} + \exp(-g_\Theta a/2)}{\pi_\Theta^{\min} - \exp(-g_\Theta a/2)} \right).$$

The proof is in the full paper [132]. Observe that the irreducibility and aperiodicity conditions may be necessary – without these conditions, the Markov Chain may not mix, and hence we may not be able to offer privacy.

Results similar to Lemma 7.4.6 can be obtained for Markov Quilts of the form $X_Q = \{X_{i-a}\}$ or $\{X_{i+b}\}$ as well. Finally, in the special case that the chains are reversible, a tighter upper bound may be obtained; these are stated in the full paper [132].

Lemma 7.4.6 indicates that when Θ is parametrized by g_Θ and π_Θ^{\min} , (an upper bound on) the score of each X_Q may thus be calculated in $O(1)$ time based on Lemma 7.4.6. This gives rise to Algorithm 9. Like Algorithm 8, we can again improve efficiency by confining our search to Markov Quilts where the local set X_N has size at most ℓ .

Running Time Analysis. A naive analysis shows that Algorithm 9 has a worst case running time of $O(T\ell^2) - T$ iterations to go over all X_i , $O(\ell^2)$ Markov Quilts per X_i , and $O(1)$ time to calculate an upper bound on the score of each quilt. However, the following Lemma shows that this running time can be improved significantly when Θ has some nice properties.

Lemma 7.4.7. *Suppose the Markov Chains induced by Θ are aperiodic and irreducible with $g_\Theta, \pi_\Theta^{\min} > 0$. Let $a^* = 2 \lceil \log \left(\frac{\exp(\epsilon/6)+1}{\exp(\epsilon/6)-1} \frac{1}{\pi_\Theta^{\min}} \right) / g_\Theta \rceil$. If the length of the chain T is $\geq 8a^*$, then, the optimal Markov Quilt for the middle node $X_{\lceil T/2 \rceil}$ of the chain is of the form $X_Q = \{X_{\lceil T/2 \rceil - a}, X_{\lceil T/2 \rceil + b}\}$ where $a + b \leq 4a^*$. Additionally, the maximum score $\sigma_{\max} = \sigma_{\lceil T/2 \rceil}$.*

Lemma 7.4.7 implies that for long enough chains, it is sufficient to search over Markov Quilts of length $\ell = 4a^*$, and only over $X_i = X_{\lceil T/2 \rceil}$; this leads to a running time of $O((a^*)^2)$, which is considerably better and independent of the length of the chain.

Algorithm 9. MQMApprox(Dataset D , L -Lipschitz query F , Θ containing Markov chains of length T , privacy parameter ϵ , maximum Markov Quilts length ℓ)

for all X_i **do**
 $S_{Q,i} = \{\{X_{i-a}, X_{i+b}\} : i - a > 0, i + b \leq T, a + b < \ell\} \cup \{\{X_{i-a}\} : i - a \geq T - \ell\}$
 $\cup \{\{X_{i+b}\} : i + b < \ell\} \cup \{\emptyset\}$ **/*all quilts with end-points at distance $\leq \ell$ plus the trivial quilt */**
for all Markov Quilts X_Q in $S_{Q,i}$ **do**
Calculate $e_\Theta(X_Q|X_i)$ from Lemma 7.4.6
if max-influence $e_\Theta(X_Q|X_i) < \epsilon$ **then**
 $\sigma(X_Q) = \frac{\text{card}(X_N)}{\epsilon - e_\Theta(X_Q|X_i)}$ **/*score of X_Q */**
else
 $\sigma(X_Q) = \infty$
end if
end for
 $\sigma_i = \min_{X_Q \in S_{Q,i}} \sigma(X_Q)$
end for
 $\sigma_{\max} = \max_i \sigma_i$
return $F(D) + (L \cdot \sigma_{\max}) \cdot Z$, where $Z \sim \text{Lap}(1)$

Utility. We conclude with an utility analysis of Algorithm 9.

Theorem 7.4.8 (Utility Guarantees). *Suppose we apply Algorithm 9 to release an approximation to a 1-Lipschitz query F of the states in the Pufferfish instantiation in Example 1. If the length T of the chain satisfies: $T \geq 8 \lceil \log \left(\frac{\exp(\epsilon/6)+1}{\exp(\epsilon/6)-1} \frac{1}{\pi_\Theta} \right) / g_\Theta \rceil + 3$, then the Laplace noise added by the Markov Quilt Mechanism has scale parameter $\leq C/\epsilon$ for some positive constant C that depends only on Θ .*

The proof is in the full paper [132]. Theorem 7.4.8 implies that the noise added does not grow with T and the *relative accuracy* improves with more and more observations. A careful examination of the proof also shows that the amount of noise added is an upper bound on the mixing time of the chain. Thus if Θ consists of rapidly mixing chains, then Algorithm 9 provides both privacy and utility.

7.5 Composition Properties

Unlike differential privacy which has good composition properties (see Chapter 2 for more details), general Pufferfish mechanisms do not compose linearly [84]. However, we can exploit the properties of the specific Pufferfish mechanism – the Markov Quilt Mechanism, and the properties of data distributions – Markov Chains to obtain parallel and sequential composition guarantees.

7.5.1 Parallel Composition

Consider the Pufferfish framework $(\mathcal{S}, \mathcal{Q}, \Theta)$ as described in Section 7.4.3. Parallel composition can be formulated as follows. Suppose there are two subchains of the Markov Chain, $X^A = X^{[T_1, T_2]}$ and $X^B = X^{[T_3, T_4]}$ where $1 \leq T_1 < T_2 < T_3 < T_4 \leq T$; and correspondingly, let $S_A = \mathcal{S}^{[T_1, T_2]}$, $Q_A = \mathcal{Q}^{[T_1, T_2]}$ and $S_B = \mathcal{S}^{[T_3, T_4]}$, $Q_B = \mathcal{Q}^{[T_3, T_4]}$. Suppose Alice has access to subchain X^A and wants to release Lipchitz query \mathcal{F}_A while guaranteeing ϵ_A -Pufferfish Privacy under framework (S_A, Q_A, Θ) ; and Bob has access to X^B and wants to release Lipchitz query \mathcal{F}_B while guaranteeing ϵ_B -Pufferfish Privacy under framework (S_B, Q_B, Θ) . Our goal is to determine how strong the Pufferfish privacy guarantee we can get for releasing $(\mathcal{M}_A(X^A), \mathcal{M}_B(X^B))$.

First, we present a general result for Markov Chains.

Theorem 7.5.1. *Suppose $\mathcal{M}_A, \mathcal{M}_B$ are two mechanisms such that $\mathcal{M}_A(X^A)$ guarantees ϵ_A -Pufferfish privacy under framework (S_A, Q_A, Θ) and $\mathcal{M}_B(X^B)$ guarantees ϵ_B -Pufferfish privacy under framework (S_B, Q_B, Θ) . Then releasing $(\mathcal{M}_A(X^A), \mathcal{M}_B(X^B))$ guarantees $\max\{\min\{\epsilon_A + \epsilon_B, \epsilon_A + e_\Theta(X_{T_2}|X_{T_3})\}, \min\{\epsilon_B + \epsilon_A, \epsilon_B + e_\Theta(X_{T_3}|X_{T_2})\}\}$ -Pufferfish Privacy under framework $(S_A \cup S_B, Q_A \cup Q_B, \Theta)$.*

Comparing with parallel composition for differential privacy, here we have the extra terms $e_\Theta(X_{T_2}|X_{T_3})$ and $e_\Theta(X_{T_3}|X_{T_2})$ which capture the correlation between X_{T_2} and X_{T_3} – the end point of the first subchain and the starting point of the second. This is to be expected, since

there is correlation among states in the Markov Chain. Intuitively, if the two subchains are close enough, releasing information on one can cause a privacy breach of the other.

Now we present a result on the Markov Quilt Mechanism on Markov Chains.

Let $\text{MQM}(D, F, \epsilon, (\mathcal{S}, \mathcal{Q}, \Theta))$ denote the output of the Markov Quilt Mechanism on dataset D , query function F , privacy parameter ϵ and Pufferfish framework $(\mathcal{S}, \mathcal{Q}, \Theta)$. Suppose Alice and Bob use MQMApprox to publish $\text{MQM}(X^A, \mathcal{F}_A, \epsilon_A, (S_A, Q_A, \Theta))$ and $\text{MQM}(X^B, \mathcal{F}_B, \epsilon_B, (S_B, Q_B, \Theta))$ respectively.

We first define active Markov Quilt $X_{Q,i}^*$ for a node X_i and then establish a parallel composition result.

Definition 7.5.2 (Active Markov Quilt). *Consider an instance of the Markov Quilt Mechanism M . We say that a Markov Quilt X_Q (with corresponding X_N, X_R) for a node X_i is active with respect to $\theta \in \Theta$ if $X_Q = \arg \min_{X_Q \in S_{Q,i}} \sigma_i^\theta(X_Q)$, and thus $\sigma_i^\theta(X_Q) = \sigma_i^\theta$.*

Theorem 7.5.3. *Suppose we run MQMApprox to release $(\text{MQM}(X^A, \mathcal{F}_A, \epsilon_A, (S_A, Q_A, \Theta)))$ and $\text{MQM}(X^B, \mathcal{F}_B, \epsilon_B, (S_B, Q_B, \Theta))$. If the following conditions hold:*

1. *for any $\theta \in \Theta$, there exists some $X_i \in X^A$ and $X_j \in X^B$ such that the active Markov Quilts of X_i and X_j with respect to θ are of the form $\{X_{i-a}, X_{i+b}\}$ and $\{X_{j-a'}, X_{j+b'}\}$ respectively for some a, b, a', b' , and*
2. *$T_3 - T_2 \geq \max\{T_2 - T_1, T_4 - T_3\}$, i.e., X^A, X^B are far from each other compared to their lengths,*

then the release guarantees $\max(\epsilon_A, \epsilon_B)$ -Pufferfish Privacy under the framework $(S_A \cup S_B, Q_A \cup Q_B, \Theta)$.

The main intuition is as follows. Note that we require the active Markov Quilt of some $X_i \in X^A$ to be of the form $\{X_{i-a}, X_{i+b}\}$. For any $X_{i'} \in X^A$, the correction factor added to account for the effect of the nodes $\{X_k \in X^A\}_{k \geq i+b}$ also automatically accounts for the effect

of $X^{\mathcal{B}} = X^{[T_3, T_4]}$, provided that $[T_3, T_4]$ does not overlap with $[i' - a, i' + b]$. This is ensured by the second condition in Theorem 7.5.3.

7.5.2 Sequential Composition

First, we show that an arbitrary Pufferfish mechanism does not compose linearly even when Θ consists of Markov Chains.

Theorem 7.5.4. *There exists a Markov Chain X , a function \mathcal{F} and mechanisms $\mathcal{M}_A, \mathcal{M}_B$ such that both $\mathcal{M}_A(X)$ and $\mathcal{M}_B(X)$ guarantee ϵ -Pufferfish privacy under framework $(\mathcal{S}, \mathcal{Q}, \Theta)$, yet releasing $(\mathcal{M}_A(X), \mathcal{M}_B(X))$ does not guarantee 2ϵ -Pufferfish privacy under framework $(\mathcal{S}, \mathcal{Q}, \Theta)$.*

Now we show that arbitrary Pufferfish mechanisms compose with a correction factor that depends on the max-divergence between the joint and product distributions of $\mathcal{M}_A(X)$ and $\mathcal{M}_B(X)$.

Theorem 7.5.5. *Suppose $\mathcal{M}_A, \mathcal{M}_B$ are two mechanisms used by Alice and Bob which guarantee ϵ_A and ϵ_B -Pufferfish privacy respectively under framework $(\mathcal{S}, \mathcal{Q}, \Theta)$. If there exists E , such that for all $s_i \in \mathcal{S}, \theta \in \Theta$,*

$$D_\infty(P(\mathcal{M}_A(X), \mathcal{M}_B(X)|s_i, \theta) \parallel P(\mathcal{M}_A(X)|s_i, \theta)P(\mathcal{M}_B(X)|s_i, \theta)) \leq E$$

$$D_\infty(P(\mathcal{M}_A(X)|s_i, \theta)P(\mathcal{M}_B(X)|s_i, \theta) \parallel P(\mathcal{M}_A(X), \mathcal{M}_B(X)|s_i, \theta)) \leq E,$$

then the releasing $(\mathcal{M}_A(X), \mathcal{M}_B(X))$ guarantees $(\epsilon_A + \epsilon_B + 2E)$ -Pufferfish Privacy under framework $(\mathcal{S}, \mathcal{Q}, \Theta)$.

The max-divergence between the joint and product distributions of $\mathcal{M}_A(X)$ and $\mathcal{M}_B(X)$ measures the amount of dependence between the two releases. The more independent they are, the smaller the max-divergence would be and the stronger privacy the algorithm guarantees.

Now we show two sequential composition results of the Markov Quilt Mechanism for general Bayesian networks and Markov Chains respectively.

Theorem 7.5.6 (Sequential Composition of MQM on General Bayesian Networks). *For the Pufferfish framework $(\mathcal{S}, \mathcal{Q}, \Theta)$ defined in Section 7.4.1, releasing $MQM(X, \mathcal{F}_k, \epsilon_k, (\mathcal{S}, \mathcal{Q}, \Theta))$ for all $k \in [K]$ under fixed Markov Quilt sets $\{S_{Q,i}\}_{i=1}^n$ for all X_i guarantees $K\epsilon$ -Pufferfish privacy under $(\mathcal{S}, \mathcal{Q}, \Theta)$.*

We note that the condition for the Markov Quilt Mechanism to compose linearly in general Bayesian networks is that all \mathcal{M}_k 's use the same active Markov Quilt. This holds automatically if the privacy parameter ϵ and Markov Quilt set $S_{Q,i}$ are the same for all \mathcal{M}_k . If different levels of privacy $\{\epsilon_k\}_{k=1}^K$ are required, we can guarantee $K \cdot \max_{k \in [K]} \epsilon_k$ -Pufferfish privacy as long as the same $S_{Q,i}$ is used across \mathcal{M}_k .

Next, we show that we can further exploit the properties of the Markov Chains to provide a tighter composition guarantee. Specifically, we show that when Θ consists of Markov Chains, even if the two runs of Markov Quilt Mechanism use different Markov Quilts, the Markov Quilt Mechanism still compose linearly. This result applies to both MQMExact and MQMApprox.

Theorem 7.5.7 (Sequential Composition of MQM on Markov Chains). *For the Pufferfish framework $(\mathcal{S}, \mathcal{Q}, \Theta)$ defined in Section 7.4.3, releasing $MQM(X, \mathcal{F}_k, \epsilon_k, (\mathcal{S}, \mathcal{Q}, \Theta))$ for all $k \in [K]$ guarantees $\sum_{k \in [K]} \epsilon_k$ -Pufferfish privacy under framework $(\mathcal{S}, \mathcal{Q}, \Theta)$.*

This result shows that the Markov Quilt Mechanism on Markov Chain achieves the same composition guarantee as pure differential privacy. Comparing to Theorem 7.5.6, it does not require the same Markov Quilts to be used in the two runs of Markov Quilt Mechanism. Moreover, the privacy guarantee is better when ϵ_k 's are different – $\sum_k \epsilon_k$ as opposite to $K \cdot \max_k \epsilon_k$.

7.6 Experiments

We next demonstrate the practical applicability of the Markov Quilt Mechanism when the underlying Bayesian network is a discrete time homogeneous Markov chain and the goal is to hide the private value of a single time entry – in short, the setting of Section 7.4.3. Our goal in this section is to address the following questions:

1. What is the privacy-utility tradeoff offered by the Markov Quilt Mechanism as a function of the privacy parameter ϵ and the distribution class Θ ?
2. How does this tradeoff compare against existing baselines, such as [66] and Group-differential privacy?
3. What is the accuracy-run time tradeoff offered by the MQMAprox algorithm as compared with MQMExact?

These questions are considered in three different contexts – (a) a small problem involving a synthetic dataset generated by a two-state Markov Chain, (b) a medium-sized problem involving real physical activity measurement data and a four-state Markov Chain, and (c) a large problem involving real data on power consumption in a single household over time and a fifty-one state Markov Chain.

7.6.1 Methodology

Experimental Setup. Our experiments involve the Pufferfish instantiation of Example 1 described in Section 7.2.2. To ensure that results across different chain lengths are comparable, we release a private *relative frequency histogram* over states of the chain which represents the (approximate) fraction of time spent in each state. This is a vector valued query, and is $\frac{2}{T}$ -Lipschitz in its L_1 -norm.

For our experiments, we consider three values of the privacy parameter ϵ that are representative of three different privacy regimes – 0.2 (high privacy), 1 (moderate privacy), 5 (low

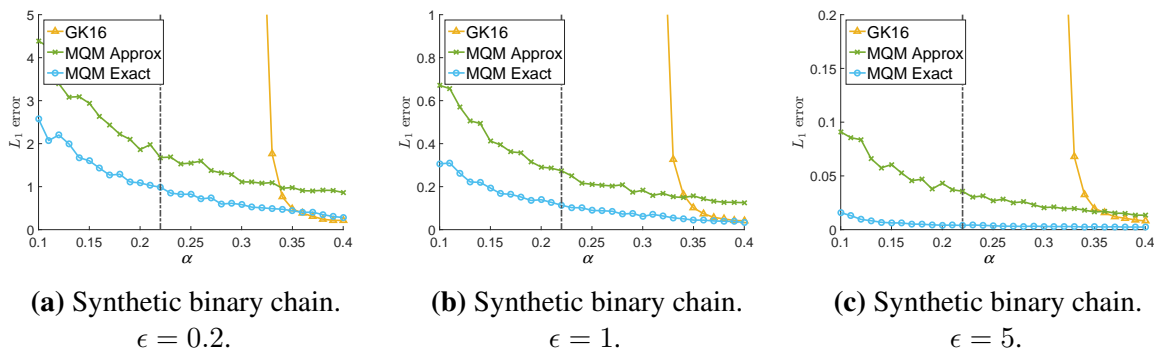


Figure 7.4. L_1 error of frequency of state 1 vs. α for $\epsilon = 0.2, 1, 5$ for synthetic data. $\Theta = [\alpha, 1 - \alpha]$. GK16 does not apply left of the black dashed vertical line. GroupDP has error around 5, 1, 0.2 for $\epsilon = 0.5, 1, 5$ respectively. Reported values are averaged over 500 random trials.

privacy). All run-times are reported for a desktop with a 3.00 GHz Intel Core i5-3330 CPU and 8GB memory.

Algorithms. Our experiments involve four mechanisms that guarantee ϵ -Pufferfish privacy – GroupDP, GK16, MQMApprox and MQMExact.

GroupDP is a simple baseline that assumes that all entries in a *connected* chain are completely correlated, and therefore adds $\text{Lap}(1/\epsilon)$ noise to each bin. **GK16** is the algorithm proposed by [66], which defines and computes an “influence matrix” for each $\theta \in \Theta$. The algorithm applies only when the spectral norm of this matrix is less than 1, and the standard deviation of noise added increases as the spectral norm approaches 1. We also use two variants of the Markov Quilt Mechanism – **MQMExact** and **MQMApprox**.

7.6.2 Simulations

We first consider synthetic data generated by a binary Markov Chain of length $T = 100$ with states $\{0, 1\}$; the setup is chosen so that all algorithms are computationally tractable when run on reasonable classes Θ . The transition matrix of such a chain is completely determined by two parameters – $p_0 = P(X_{i+1} = 0|X_i = 0)$ and $p_1 = P(X_{i+1} = 1|X_i = 1)$, and its initial distribution by a single parameter $q_0 = P(X_1 = 0)$. Thus a distribution $\theta \in \Theta$ is

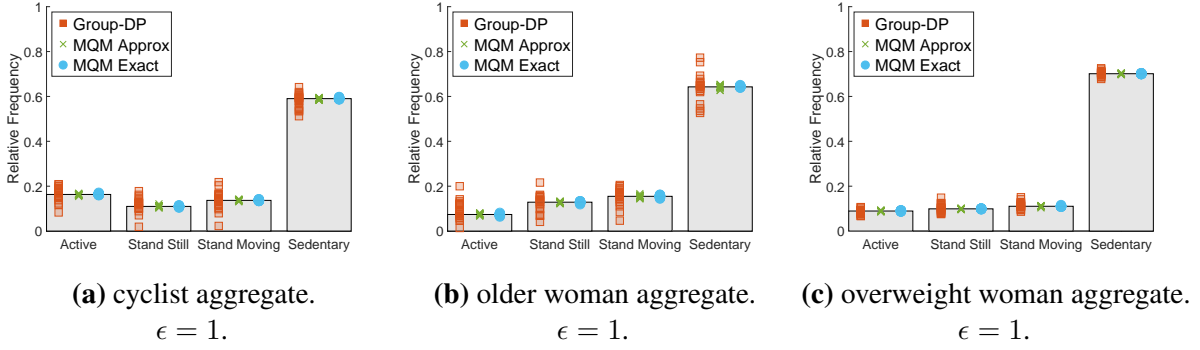


Figure 7.5. Exact and private aggregated physical activity relative frequency histograms for three groups of participants. Reported values are for 20 random trials and $\epsilon = 1$. Recall GK16 does not apply for this problem.

represented by a tuple (q_0, p_0, p_1) , and a distribution class Θ by a set of such tuples. To allow for effective visualization, we represent the distribution class Θ by an interval $[\alpha, \beta]$ which means that Θ includes all transition matrices for which $p_0, p_1 \in [\alpha, \beta]$ and all initial distribution q in the 2-dimensional probability simplex. When $0 < p_0, p_1 < 1$, the chain is guaranteed to be aperiodic, irreducible and reversible, and we can use the approximation in the full paper [132] for MQMApprox. We use the optimization procedure described in the full paper [132] to improve the efficiency of MQMExact. Finally, since the histogram has only two bins, it is sufficient to look at the query $F(X) = \frac{1}{T} \sum_{i=1}^T X_i$ which is $\frac{1}{T}$ -Lipschitz.

To generate synthetic data from a family $\Theta = [\alpha, \beta]$, we pick p_0 and p_1 uniformly from $[\alpha, \beta]$, and an initial state distribution uniformly from the probability simplex. We then generate a state sequence X_1, \dots, X_T of length T from the corresponding distribution. For ease of presentation, we restrict Θ to be intervals where $\beta = 1 - \alpha$, and vary α from in 0.1 to 0.4. We vary ϵ in $\{0.2, 1, 5\}$, repeat each experiment 500 times, and report the average error between the actual $F(X)$ and its reported value. For the run-time experiments, we report the average running time of the procedure that computes the scale parameter for the Laplace noise in each algorithm; the average is taken over all $\theta = (p_0, p_1)$ in a grid where p_0, p_1 vary in $\{0.1, 0.11, \dots, 0.9\}$.

Figure 7.4 shows the accuracy results, and Table 7.2 (Column 2) the run-time results for $\epsilon = 1$. The values for GroupDP have high variance and are not plotted in the figure; these values

are around 5, 1, 0.2 respectively. As expected, for a given ϵ , the errors of GK16, MQMAprox and MQMExact decrease as α increases, i.e, the distribution class Θ becomes narrower. When α is to the left of the black dashed line in the figure, GK16 does not apply as the spectral norm of the influence matrix becomes > 1 ; the position of this line does not change as a function of ϵ . In contrast, MQMAprox and MQMExact still provide privacy and reasonable utility. As expected, MQMExact is more accurate than MQMAprox, but requires higher running time. Thus, the Markov Quilt Mechanism applies to a wider range of distribution families than GK16; in the region where all mechanisms work, MQMAprox and MQMExact perform significantly better than GK16 for a range of parameter values, and somewhat worse for the remaining range.

7.6.3 Real Data

Table 7.1. L_1 error of the relative frequency histograms for individual and aggregate tasks for physical activity of three participant groups. $\epsilon = 1$. Reported values are averaged over 20 random trials.

Algorithm	cyclist		older woman		overweight woman	
	Agg	Indi	Agg	Indi	Agg	Indi
DP	0.2918	N/A	0.8746	N/A	0.4763	N/A
GroupDP	0.0834	2.3157	0.1138	1.7860	0.0458	1.1492
GK16	N/A	N/A	N/A	N/A	N/A	N/A
MQMAprox	0.0107	0.6319	0.0156	0.2790	0.0048	0.1967
MQMExact	0.0074	0.4077	0.0098	0.1742	0.0033	0.1316

Table 7.2. Running time (in seconds) of an optimized algorithm that calculates the scale parameter of the Laplace noise (averaged over 5 runs). $\epsilon = 1$.

Algorithm	Synthetic	cyclist	older woman	overweight woman	electricity power
GK16	6.3589×10^{-4}	N/A	N/A	N/A	N/A
MQMAprox	1.8458×10^{-4}	0.0064	0.0060	0.0028	0.0567
MQMExact	7.6794×10^{-4}	1.5186	1.2786	0.6299	282.2273

We next apply our algorithm to two real datasets on physical activity measurement and power consumption. Since these are relatively large problems with large state-spaces, it is extremely difficult to search over all Markov Chains in a class Θ , and both GK16 as well as MQMExact become highly computation intensive. To ensure a fair comparison, we pick Θ to be a singleton set $\{\theta\}$, where $\theta = (q_\theta, P_\theta)$; here P_θ is the transition matrix obtained from the data, and q_θ is its stationary distribution. For MQMApprox, we use ℓ from Lemma 7.4.7, while for MQMExact we use as ℓ the length of the optimal Markov Quilt that was returned by MQMApprox.

7.6.3.1 Physical Activity Measurement

We use an activity dataset provided by [53, 55, 54], which includes monitoring data from a study of daily habits of 40 cyclists, 16 older women, and 36 overweight women. The dataset includes four activities – active, standing still, standing moving and sedentary – for all three groups of participants,¹ and thus the underlying θ is a four-state Markov Chain. Activities are recorded about every 12 seconds for 7 days when the participants are awake, which gives us more than 9,000 observations per person on an average in each group. To address missing values, we treat gaps of more than 10 minutes as the starting point of a new independent Markov Chain. Observe that this improves the performance of GroupDP, since the noise added is $\text{Lap}(M/T\epsilon)$, where M is the length of the longest chain. For each group of participants, we calculate a single empirical transition matrix P_θ based on the entire group; this P_θ is used in the experiments. For this application, we consider two tasks – aggregate and individual. In the aggregate task, the goal is to publish a private aggregated relative frequency histogram² over participants in each group in order to analyze their comparative activity patterns. While in theory this task can be achieved with differential privacy, this gives poor utility as the group sizes are small. In the individual task, we publish the relative frequency histogram for each individual and report the average error

¹For cyclists, data also includes cycling, which we merge with active for ease of analysis and presentation.

²Recall that in a relative frequency histogram, we report the number of observations in each bin divided by the total number of observations.

across individuals in each group.

Table 7.1 summarizes the L_1 errors for the three groups and both tasks for $\epsilon = 1$. For the aggregate task, we also report the error of a differentially private release (denoted as DP). Note that the error of the individual task is the average L_1 error of each individual in the group. Figure 7.5 presents the exact and private aggregated relative frequency histograms for the three groups. Table 7.2 (Columns 2-4) presents the time taken to calculate the scale parameter of the Laplace noise for each group of participants; as this running time depends on both the size of the group as well as the activity patterns, larger groups do not always have higher running times.

The results in Table 7.1 show that the utilities of both MQMAprox and MQMExact are significantly better than that of GroupDP for all datasets and tasks, and are significantly better than DP for the aggregated task. As expected, MQMExact is better than MQMAprox, but has a higher running time. We also find that GK16 cannot be applied to any of the tasks, since the spectral norm of the influence matrix is > 1 ; this holds for all ϵ as the spectral norm condition does not depend on the value of ϵ . Figure 7.5 shows the activity patterns of the different groups: the active time spent by the cyclist group is significantly longer than the other two groups, and the sedentary time spent by overweight women group is the longest. These patterns are visible from the private histograms published by MQMAprox and MQMExact, but not necessarily from those published by GroupDP.

7.6.3.2 Electricity Consumption

Table 7.3. L_1 error of relative frequency histogram for electricity consumption data. Reported values are averaged over 20 random trials.

Algorithm	$\epsilon = 0.2$	$\epsilon = 1$	$\epsilon = 5$
GroupDP	516.1555	102.8868	19.8712
GK16	N/A	N/A	N/A
MQMAprox	0.3369	0.0614	0.0113
MQMExact	0.1298	0.0188	0.0022

We use data on electricity consumption of a single household in the greater Vancouver area provided by [95]. Power consumption (in Watt) is recorded every 1 minute for about two years. Missing values in the original recording have been filled in before the dataset is published. We discretize the power values into 51 intervals, each of length 200(W), resulting in a Markov chain with 51 states of length $T \approx 1,000,000$. Our goal again is to publish a private approximation to the relative frequency histogram of power levels.

Table 7.3 reports the L_1 errors of the four algorithms on electricity power dataset for different ϵ values, and Table 7.2 the times required to calculate the scale parameter. We again find that GK16 does not apply as the spectral norm condition is not satisfied. The error of GroupDP is very large, because the data forms a single Markov chain, and the number of states is relatively large. We find that in spite of the large number of bins, MQMApprox and MQMExact have high utility; for example, even for $\epsilon = 0.2$, the *per bin* error of MQMExact is about 0.25 percent, and for $\epsilon = 1$, the *per bin* error is 0.04 percent. Finally, while the running time of MQMExact is an order of magnitude higher than MQMApprox, it is still manageable (< 5 minutes) for this problem.

7.6.4 Discussion

We now reconsider our initial questions. First, as expected, the experiments show that the utility of GK16 and both versions of MQM decreases with decreasing ϵ and increasing size of Θ . The utility of GroupDP does not change with Θ and is not very high; again this is to be expected as GroupDP depends only on the worst-case correlation. Overall the utility of both versions of the Markov Quilt Mechanism improve for longer chains.

Comparing with GK16, we find that for synthetic data, the Markov Quilt Mechanism applies to a much wider range of distribution families; in the region where all mechanisms work, MQMApprox and MQMExact perform significantly better than GK16 for a range of parameter values, and somewhat worse for the remaining range. For the two real datasets we consider, GK16 does not apply. We suspect this is because the influence matrix in [66] is calculated based on local transitions between successive time intervals (for example, X_t as a function of X_{t-1}); instead, the Markov Quilt Mechanism implicitly takes into account transitions across periods (for example, how X_{t+k} as a function of X_t). Our experiments imply that this spectral norm condition may be quite restrictive in practical applications.

Finally, our experiments show that there is indeed a gap between the performance of MQMExact and MQMApprox, as well as their running times, although the running time of MQMExact still remains manageable for relatively large problems. Based on these results, we recommend using MQMExact for medium-sized problems where the state space is smaller (and thus computing the max-influence is easier) but less data is available, and MQMApprox for larger problems where the state space is larger but there is a lot of data to mitigate the effect of the approximation.

7.7 Conclusion

We present a detailed study of how Pufferfish may be applied to achieve privacy in correlated data problems. We establish robustness properties of Pufferfish against adversarial

beliefs, and we provide the first mechanism that applies to any Pufferfish instantiation. We provide a more computationally efficient mechanism for Bayesian networks, and establish its composition properties. We derive a version of our mechanism for Markov Chains, and evaluate it experimentally on a small, medium and a large problem on time series data. Our results demonstrate that Pufferfish offers a good solution for privacy in these problems.

We believe that our work is a first step towards a comprehensive study of privacy in correlated data. There are many interesting privacy problems – such as privacy of users connected into social networks and privacy of spatio-temporal information gathered from sensors. With the proliferation of sensors and “internet-of-things” devices, these privacy problems will become increasingly pressing. We believe that an important line of future work is to model these problems in rigorous privacy frameworks such as Pufferfish and design novel mechanisms for these models.

Acknowledgments

We thank Mani Srivastava and Supriyo Chakravarty for introducing us to the physical activity problem and early discussions and anonymous reviewers for feedback. We thank Joseph Geumlek, Sewoong Oh and Yizhen Wang for initial discussions for the composition results. This work was partially supported by NSF under IIS 1253942 and ONR under N00014-16-1-2616.

This chapter is based on “Pufferfish Privacy Mechanisms for Correlated Data” by Shuang Song, Yizhen Wang and Kamalika Chaudhuri, which appears in the International Conference on Management of Data (SIGMOD), 2017 [132]. The dissertation author was a primary author of the paper.

Section 7.5 of this chapter is based on “Composition Properties of Inferential Privacy for Time-Series Data” by Shuang Song and Kamalika Chaudhuri, which appears in the Allerton Conference on Communication, Control and Computing, 2017 [129]. The dissertation author was the primary author of the paper.

Bibliography

- [1] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.
- [2] Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, H. Brendan McMahan, Nicolas Papernot, Ilya Mironov, Kunal Talwar, and Li Zhang. On the protection of private information in machine learning systems: Two recent approaches. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 1–6, 2017.
- [3] Nabil R Adam and John C Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys (CSUR)*, 21(4):515–556, 1989.
- [4] Alekh Agarwal, Peter L. Bartlett, Pradeep Ravikumar, and Martin J. Wainwright. Information-theoretic lower bounds on the oracle convexity of convex optimization. In *Adv. NIPS 22*. MIT Press, 2009.
- [5] Faraz Ahmed, Rong Jin, and Alex X Liu. A random matrix approach to differential privacy and structure preserved social network graph publishing. *arXiv preprint arXiv:1307.0475*, 2013.
- [6] David Aldous and Jim Fill. Reversible markov chains and random walks on graphs, 2002.
- [7] F. Bach and E. Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Adv. NIPS 24*. MIT Press, 2011.
- [8] Mitali Bafna and Jonathan Ullman. The price of selection in differential privacy. In *Proceedings of the 2017 Conference on Learning Theory (COLT)*, volume 65 of *Proceedings of Machine Learning Research*, pages 151–168, July 2017.
- [9] Michael Barbaro, Tom Zeller, and Saul Hansell. A face is exposed for aol searcher no. 4417749. *New York Times*, 9(2008):8For, 2006.
- [10] R. Bassily, A. Thakurta, and A. Smith. Private empirical risk minimization, revisited. In

Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS 2014), Philadelphia, PA, USA, October 2014.

- [11] Raef Bassily, Alex Groce, Justin Katz, and Adam Smith. Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy. In *FOCS*, 2013.
- [12] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 464–473, 2014.
- [13] Ron Bekkerman, Mikhail Bilenko, and John Langford. *Scaling up Machine Learning, Parallel and Distributed Approaches*. Cambridge University Press, 2011.
- [14] Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 503–512. ACM, 2010.
- [15] Battista Biggio, Iginio Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402, 2013.
- [16] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96. ACM, 2013.
- [17] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *Proc. PODS*, pages 128–138, New York, NY, USA, 2005. ACM.
- [18] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138. ACM, 2005.
- [19] Jean Bolot, Nadia Fawaz, S Muthukrishnan, Aleksandar Nikolov, and Nina Taft. Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory*, pages 284–295. ACM, 2013.
- [20] Christian Borgs, Jennifer Chayes, and Adam Smith. Private graphon estimation for sparse graphs. In *Advances in Neural Information Processing Systems*, pages 1369–1377, 2015.
- [21] Leon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proc. 19th Int’l Conf. on Comp. Stat.*, pages 177–187, 2010.
- [22] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications,

- extensions, and lower bounds. In *Theory of Cryptography Conference (TCC)*, pages 635–658, 2016.
- [23] Mark Bun, Thomas Steinke, and Jonathan Ullman. Make up your mind: The price of online queries in differential privacy. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1306–1325. SIAM, 2017.
- [24] Jianneng Cao, Qian Xiao, Gabriel Ghinita, Ninghui Li, Elisa Bertino, and Kian-Lee Tan. Efficient and accurate strategies for differentially-private sliding window queries. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 191–202. ACM, 2013.
- [25] T-H Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):26, 2011.
- [26] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *JMLR*, 12:1069–1109, March 2011.
- [27] Kamalika Chaudhuri, Daniel Hsu, and Shuang Song. The large margin mechanism for differentially private maximization. In *NIPS*, 2014.
- [28] Kamalika Chaudhuri, Anand D Sarwate, and Kaushik Sinha. A near-optimal algorithm for differentially-private principal components. *The Journal of Machine Learning Research*, 14(1):2905–2943, 2013.
- [29] Yan Chen, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Pegasus: Data-adaptive differentially private stream processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1375–1388. ACM, 2017.
- [30] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.
- [31] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *Adv. NIPS 24*, pages 1647–1655, 2011.
- [32] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.
- [33] K Crammer, M Kearns, and J Wortman. Learning from data of variable quality. In Y. Weiss, B. Schölkopf, and J.C. Platt., editors, *Advances in Neural Information Processing Systems 18*, pages 219–226. MIT Press, 2006.
- [34] Tore Dalenius. Towards a methodology for statistical disclosure control. *statistik Tidskrift*, 15(429-444):2–1, 1977.

- [35] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*, pages 123–138. ACM, 2016.
- [36] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *J. Mach. Learn. Res.*, 13:165–202, 2012.
- [37] Dorothy E. Denning. Secure statistical databases with random sample queries. *ACM Trans. Database Syst.*, 5(3):291–315, September 1980.
- [38] Christos Dimitrakakis, Blaine Nelson, Aikaterini Mitrokotsa, and Benjamin IP Rubinstein. Robust and private bayesian inference. In *International Conference on Algorithmic Learning Theory*, pages 291–305. Springer, 2014.
- [39] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3574–3583, 2017.
- [40] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210. ACM, 2003.
- [41] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 2009.
- [42] John Duchi, Michael Jordan, and Martin Wainwright. Privacy aware learning. In *Adv. NIPS 25*, pages 1439–1447, 2012.
- [43] John Duchi, Martin J. Wainwright, and Michael Jordan. Local privacy and minimax bounds: Sharp rates for probability estimation. In *Advances in Neural Information Processing Systems 26*, pages 1529–1537, 2013.
- [44] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503, Berlin, Heidelberg, 2006. Springer-Verlag.
- [45] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, 2006.
- [46] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *STOC*, 2009.
- [47] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

- [48] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *TCS*, 9(3-4):211–407, 2013.
- [49] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.
- [50] Cynthia Dwork and Guy N Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.
- [51] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 51–60, 2010.
- [52] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 11–20. ACM, 2014.
- [53] Katherine Ellis, Suneeta Godbole, Jacqueline Chen, Simon Marshall, Gert Lanckriet, and Jacqueline Kerr. Physical activity recognition in free-living from body-worn sensors. In *Proceedings of the 4th International SenseCam & Pervasive Imaging Conference*, pages 88–89. ACM, 2013.
- [54] Katherine Ellis, Jacqueline Kerr, Suneeta Godbole, and Gert Lanckriet. Multi-sensor physical activity recognition in free-living. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 431–440. ACM, 2014.
- [55] Katherine Ellis, Jacqueline Kerr, Suneeta Godbole, John Staudenmayer, and Gert Lanckriet. Hip and wrist accelerometer algorithms for free-living behavior classification. *Medicine and science in sports and exercise*, 48(5):933, 2016.
- [56] 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*, pages 1054–1067. ACM, 2014.
- [57] L. Fan, L. Xiong, and V. Sunderam. Differentially private multi-dimensional time series release for traffic monitoring. In *DBSec*, 2013.
- [58] I. P. Fellegi. On the question of statistical confidentiality. *Journal of the American Statistical Association*, 67(337):7–18, 1972.
- [59] Stephen E Fienberg and Russell J Steele. Disclosure limitation using perturbation and related methods for categorical data. *Journal of Official Statistics*, 14(4):485, 1998.

- [60] James Foulds, Joseph Geumlek, Max Welling, and Kamalika Chaudhuri. On the theory and practice of privacy-preserving bayesian data analysis. *arXiv preprint arXiv:1603.07294*, 2016.
- [61] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.
- [62] Kazuto Fukuchi, Quang Khai Tran, and Jun Sakuma. Differentially private empirical risk minimization with input perturbation. In *International Conference on Discovery Science*, pages 82–90. Springer, 2017.
- [63] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 265–273. ACM, 2008.
- [64] Johannes Gehrke, Edward Lui, and Rafael Pass. Towards privacy for social networks: A zero-knowledge based definition of privacy. In *Theory of Cryptography Conference*, pages 432–449. Springer, 2011.
- [65] Joseph Geumlek, Shuang Song, and Kamalika Chaudhuri. Renyi differential privacy mechanisms for posterior sampling. In *Advances in Neural Information Processing Systems*, pages 5295–5304, 2017.
- [66] Arpita Ghosh and Robert Kleinberg. Inferential privacy guarantees for differentially private mechanisms. *arXiv preprint arXiv:1603.01508*, 2016.
- [67] Bronwyn H Hall, Adam B Jaffe, and Manuel Trajtenberg. The nber patent citation data file: Lessons, insights and methodological tools. Technical report, National Bureau of Economic Research, 2001.
- [68] Jihun Hamm, Yingjun Cao, and Mikhail Belkin. Learning privately from multiparty data. In *International Conference on Machine Learning (ICML)*, pages 555–563, 2016.
- [69] Steve Hanneke. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3):131–309, 2014.
- [70] M. Hardt and A. Roth. Beyond worst-case analysis in private singular vector computation. In *STOC*, 2013.
- [71] Moritz Hardt and Guy N Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *51st Annual IEEE Symposium on Foundations of Computer*

Science (FOCS), pages 61–70, 2010.

- [72] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 169–178. IEEE, 2009.
- [73] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [74] Xi He, Ashwin Machanavajjhala, and Bolin Ding. Blowfish privacy: tuning privacy-utility trade-offs using policies. In *SIGMOD '14*, pages 1447–1458, 2014.
- [75] P. Jain, P. Kothari, and A. Thakurta. Differentially private online learning. *JMLR – COLT*, 2012.
- [76] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, 2011.
- [77] S. A. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *FOCS*, 2008.
- [78] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography*, pages 457–476. Springer, 2013.
- [79] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *JACM*, 1998.
- [80] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12):1155–1166, 2014.
- [81] W.O. Kermack and A.G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 115(772):700–721, 1927.
- [82] Stephan Kessler, Erik Buchmann, and Klemens Böhm. Deploying and evaluating pufferfish privacy for smart meter data. *Karlsruhe Reports in Informatics*, 1, 2015.
- [83] D. Kifer, A. Smith, and A. Thakurta. Private convex optimization for empirical risk minimization with applications to high-dimensional regression. In *COLT*, 2012.
- [84] Daniel Kifer and Ashwin Machanavajjhala. Pufferfish: A framework for mathematical

- privacy definitions. *ACM Trans. Database Syst.*, 39(1):3, 2014.
- [85] Ron Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In *KDD*, volume 96, pages 202–207, 1996.
- [86] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [87] Srijan Kumar. Structure and dynamics of signed citation networks. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 63–64. International World Wide Web Conferences Steering Committee, 2016.
- [88] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proc. IEEE*, pages 2278–2324, 1998.
- [89] 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*, pages 106–115. IEEE, 2007.
- [90] Susan J Little, Sergei L Kosakovsky Pond, Christy M Anderson, Jason A Young, Joel O Wertheim, Sanjay R Mehta, Susanne May, and Davey M Smith. Using hiv networks to inform real time prevention interventions. *PloS one*, 9(6):e98443, 2014.
- [91] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. Dependence makes you vulnerable: Differential privacy under dependent tuples. In *NDSS 2016*, 2016.
- [92] Wentian Lu and Gerome Miklau. Exponential random graph estimation under differential privacy. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 921–930. ACM, 2014.
- [93] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, 2008.
- [94] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkatasubramanian. l-diversity: Privacy beyond k-anonymity. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 24–24. IEEE, 2006.
- [95] Stephen Makonin, Bradley Ellert, Ivan V. Bajic, and Fred Popowich. Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014. *Scientific Data*, 3(160037):1–12, 2016.
- [96] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private language models without losing accuracy. *arXiv preprint arXiv:1710.06963*,

2017.

- [97] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, 2007.
- [98] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Communications of the ACM*, 53(9):89–97, September 2010.
- [99] Frank McSherry and Ilya Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 627–636. ACM, 2009.
- [100] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, 2007.
- [101] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275, 2017.
- [102] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.
- [103] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 111–125. IEEE, 2008.
- [104] N Natarajan, I Dhillon, P Ravikumar, and A Tewari. Learning with noisy labels. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1196–1204. Curran Associates, Inc., 2013.
- [105] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alex Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. Opt.*, 19(4):1574–1609, 2009.
- [106] A.S. Nemirovsky and D.B. Yudin. *Problem complexity and method efficiency in optimization*. Wiley & Sons, 1983.
- [107] Y. Nesterov and J. Vial. Confidence level solutions for stochastic programming. *Automatica*, 2008.
- [108] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop*

on *Deep Learning and Unsupervised Feature Learning*, page 5, 2011.

- [109] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2007.
- [110] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing (STOC)*, pages 75–84, 2007.
- [111] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.
- [112] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with PATE. In *International Conference on Learning Representations*, 2018.
- [113] Manas Pathak, Shantanu Rane, and Bhiksha Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Information Processing Systems*, pages 1876–1884, 2010.
- [114] Trivellore E Raghunathan, Jerome P Reiter, and Donald B Rubin. Multiple imputation for statistical disclosure limitation. *Journal of official statistics*, 19(1):1, 2003.
- [115] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. Technical Report arXiv:1109.5647 [cs.LG], ArXiv, 2012.
- [116] Sofya Raskhodnikova and Adam Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. *arXiv preprint arXiv:1504.07912*, 2015.
- [117] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, 2010.
- [118] Steven P Reiss. Practical data-swapping: The first steps. *ACM Transactions on Database Systems (TODS)*, 9(1):20–37, 1984.
- [119] Saharon Rosset and Aron Inger. KDD-cup 99: knowledge discovery in a charitable organization’s donor database. *SIGKDD Explor. Newsl.*, 1(2):85–90, January 2000.
- [120] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In

- Proceedings of the Forty-second ACM Symposium on Theory of Computing (STOC)*, pages 765–774, 2010.
- [121] Donald B Rubin. Discussion statistical disclosure limitation. *Journal of official Statistics*, 9(2):461, 1993.
- [122] B. Rubinstein, P. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *JPC*, 2013.
- [123] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *Journal of Privacy and Confidentiality*, 4(1):65–100, 2012.
- [124] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.
- [125] A.D. Sarwate and K. Chaudhuri. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *Signal Processing Magazine, IEEE*, 30(5):86–94, Sept 2013.
- [126] S Shalev-Shwartz, O Shamir, N Srebro, and K Sridaran. Stochastic convex optimization. In *COLT*, 2009.
- [127] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming, Series B*, 127(1):3–30, 2011.
- [128] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321. ACM, 2015.
- [129] Shuang Song and Kamalika Chaudhuri. Composition properties of inferential privacy for time-series data. In *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 814–821, 2017.
- [130] Shuang Song, Kamalika Chaudhuri, and Anand Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248, Dec 2013.
- [131] Shuang Song, Kamalika Chaudhuri, and Anand Sarwate. Learning from Data with Heterogeneous Noise using SGD. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 894–902, San

Diego, California, USA, 09–12 May 2015. PMLR.

- [132] Shuang Song, Yizhen Wang, and Kamalika Chaudhuri. Pufferfish privacy mechanisms for correlated data. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD '17*, pages 1291–1306, New York, NY, USA, 2017. ACM.
- [133] Thomas Steinke and Jonathan Ullman. Tight lower bounds for differentially private selection. In *58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 552–563, 2017.
- [134] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 26–37. ACM, 2016.
- [135] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.
- [136] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.
- [137] Martin Takáč, Avleen Bijral, Peter Richtárik, and Nathan Srebro. Mini-batch primal and dual methods for SVMs. In *Proc. ICML*, 2013.
- [138] Abhradeep Guha Thakurta, Andrew H Vyrros, Umesh S Vaishampayan, Gaurav Kapoor, Julien Freudingier, Vipul Ved Prakash, Arnaud Legendre, and Steven Duplinsky. Emoji frequency detection and deep link frequency, July 18 2017. US Patent 9,712,550.
- [139] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium*, pages 601–618, 2016.
- [140] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. 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*, pages 534–544. ACM, 2009.
- [141] Xicheng Wang, Yasong Wu, Lin Mao, Wei Xia, Weiwei Zhang, Lili Dai, Sanjay R Mehta, Joel O Wertheim, Xingqi Dong, Tong Zhang, Hao Wu, and Davey Smith. Targeting hiv prevention based on molecular epidemiology among deeply sampled subnetworks of men who have sex with men. *Clinical Infectious Diseases*, 61(9):1462–1468, 2015.
- [142] Yu-Xiang Wang, Stephen Fienberg, and Alex Smola. Privacy for free: Posterior sampling and stochastic gradient monte carlo. In *International Conference on Machine Learning*,

pages 2493–2502, 2015.

- [143] Yue Wang, Xintao Wu, and Leting Wu. Differential privacy preserving spectral graph analysis. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 329–340. Springer, 2013.
- [144] L. Wasserman and S. Zhou. A statistical framework for differential privacy. *JASA*, 2010.
- [145] Joel O Wertheim, Sergei L Kosakovsky Pond, Lisa A Forgione, Sanjay R Mehta, Ben Murrell, Sharmila Shah, Davey M Smith, Konrad Scheffler, and Lucia V Torian. Social and genetic networks of hiv-1 transmission in new york city. *PLoS pathogens*, 13(1):e1006000, 2017.
- [146] Oliver Williams and Frank McSherry. Probabilistic inference and differential privacy. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Adv. NIPS 23*, pages 2451–245, 2010.
- [147] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1307–1322. ACM, 2017.
- [148] Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *JMLR*, 2010.
- [149] Yonghui Xiao and Li Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC CCS*, pages 1298–1309. ACM, 2015.
- [150] Bin Yang, Issei Sato, and Hiroshi Nakagawa. Bayesian differential privacy on correlated data. In *SIGMOD '15*, pages 747–762. ACM, 2015.
- [151] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.
- [152] Zuhe Zhang, Benjamin IP Rubinstein, and Christos Dimitrakakis. On the differential privacy of bayesian inference. In *AAAI*, pages 2365–2371, 2016.