UNIVERSITY OF CALIFORNIA, SAN DIEGO

Learning Information from Data while Preserving Differential Privacy

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

in

Computer Science

by

Zhanglong Ji

Committee in charge:

Professor Charles Elkan, Chair
Professor Lucila Ohno-Machado, Co-Chair
Professor Kamalika Chaudhuri
Professor Sanjoy Dasgupta
Professor Kevin Patrick

2017

The dissertation of Zhanglong Ji is approved and is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California, San Diego

2017

DEDICATION

I dedicate this dissertation to Yiren Hu, for her support and help in my daily life and career. She provides me with encouragement when I am down; gives me useful suggestions when I make decisions; and helps me accomplish my goals. Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

When I interned in Microsoft Research Asia as an undergraduate in 2010, I felt the power and beauty of data mining, and thus applied to the Computer Science PhD program at the University of California, San Diego. Here I learned how medical informatics could help understand mechanisms of diseases, effects of drugs, treatments of patients, mistake prevention, etc. However, data mining on medical data, unlike other data, has privacy issues. Therefore, specially designed algorithms to account for these privacy needs are needed. Since then, I have been working to combine data mining and privacy together. This dissertation includes some of my work during this period.

This dissertation is for researchers who are interested in privacy-preserving algorithms, and assumes readers have some knowledge of data mining.

ACKNOWLEDGEMENTS

I would like to acknowledge Dr. Charles Elkan for his support as the chair of my committee. He taught me how to do research, helped me in writing several papers listed below, and went through the draft of this dissertation.

I would also like to thank Dr. Lucila Ohno-Machado for her support as the co-chair in my committee. Without her financial support, it would be much harder for me to earn this degree.

I would especially like to thank Dr. Xiaoqian Jiang for his advice with my research. His mentorship has led to the publication of many papers including those listed below.

I would like to thank all those who participated in this research; I truly appreciate the time you gave to this project.

Chapter 2 is based on the survey paper *Differential Privacy and Machine Learning: a Survey and Review.* Zhanglong Ji, Zack C Lipton, Charles Elkan arXiv preprint arXiv:1412.7584, 2014. The dissertation author was the primary author of the paper.

Section 3.1.3 comes from the paper *Scalable Privacy-Preserving Data Sharing Methodology for Genome-Wide Association Studies: An Application to iDASH Healthcare Privacy Protection Challenge.* Fei Yu, Zhanglong Ji. BMC Medical Informatics and Decision Making 2014, 14(Suppl 1):S3. The dissertation author was the primary author of the part included in this dissertation.

In Section 3.1.4, we use the introduction of the Transmission Disequilibrium Test in the paper *Privacy-Preserving Mechanisms for Transmission Disequilibrium Test in Genome-Wide Association Studies.* Meng Wang, Zhanglong Ji, Shuang Wang, Jihoon Kim, Hai Yang, Lucila Ohno-Machado, and Xiaoqian Jiang. TBC 2015. The dissertation author contributed to the writing of the part used here.

Section 3.2 comes from the paper *Differentially Private Empirical Risk Mini-*

mization with Feature Selection. Zhanglong Ji, Xiaoqian Jiang, Lucila Ohno-Machado, which is in submission right now. The dissertation author was the primary author of the paper.

Section 4.1 comes from the paper *Differential Privacy Based on Importance Weighting.* Zhanglong Ji, Charles Elkan. Machine Learning, June 2013. The dissertation author was the primary author of the paper.

Section 4.2 comes from the paper *Select and Label (S&L): a Task-Driven Privacy-Preserving Data Synthesization Algorithm.* Zhanglong Ji, Xiaoqian Jiang, Haoran Li, Li Xiong, Lucila Ohno-Machado, which is in submission right now. The dissertation author was the primary author of the paper.

VITA

2006–2011 Bachelor of Science, Peking University

2011–2014 Master of Science, University of California, San Diego

2011–2017 Doctor of Philosophy, University of California, San Diego

ABSTRACT OF THE DISSERTATION

Learning Information from Data while Preserving Differential Privacy

by

Zhanglong Ji

Doctor of Philosophy in Computer Science

University of California, San Diego, 2017

Professor Charles Elkan, Chair
Professor Lucila Ohno-Machado, Co-Chair

In this dissertation, I introduce my work on differentially private data mining. There are usually two types of privacy preserving data mining algorithms: query answering algorithms and data publishing algorithms.

I present three algorithms on two differentially private query answering problems in Chapter 3. The first problem is to return Single-Nucleotide Polymorphisms (SNPs) most correlated to a disease, and I provide two efficient algorithms that make an accurate but previously inefficient algorithm feasible. The second problem is to learn a model minimizing empirical risk while selecting features. Our algorithm beats state-of-

the-art algorithms.

Then I present three algorithms on differentially private data publishing under two scenarios in Chapter 4. The first algorithm assumes the existence of public data, and assigns weights to public data so they are statistically similar to the private data. Analysis on the weighted public data is more accurate than analysis on the private data. The two following algorithms assume that published data are for supervised learning (one algorithm for classification and the other for regression), and that the prediction rules are continuous with respect to predictors. The data these algorithms publish perform very well on learning tasks.

Chapter 1

Overview

In today's world, data is being collected at unprecedented speed, and it has become common knowledge that data mining can benefit individuals, companies, and society. For example, Netflix uses user rating data to recommend movies/videos to other users, and Google uses deep learning to train its translate service. Other applications include credit scoring, search engine optimization, etc. Data mining has benefited both companies and society.

However, the benefit of data mining is limited to some fields because of privacy concerns. For example, medical data and genome data have to be kept private when used, as revealing these data may lead to an increase in life insurance premiums. Researchers found attackers can know whether a patient is in a database with high confidence given genome data statistics. Therefore, for genome data, even statistics are no longer safe to publish. These risks severely prevent researchers from making full use of private data, thus there is demand for privacy-preserving data mining algorithms.

One solution is to ensure data are used in a privacy-preserving way. This solution requires two steps. First, we need to find a mathematical definition of the word 'privacy', and then we can develop algorithms satisfying the definition. Different privacy definitions correspond to different assumptions on attackers' knowledge on data and definitions of successful attacks, and thus different protection strengths.

For example, differential privacy [14, 15] assumes attackers may already have some knowledge of the private data, such as, all information except one sample, or even except one value, and can change their guess of private information by issuing queries and looking at answers from the data holder. An attack is successful if the attacker's guess on the true private information can be altered a lot after viewing output information, e.g. answers to some outside queries. Informally, since differential privacy is against privacy leakage in any circumstance, and assumes the attackers may know almost all of the private information, its protection is against risk in extreme cases. There are also some other definitions, such as k-anonymity [56] or l-diversity [41], however my work focuses on differential privacy only.

Given the definitions of privacy, and some 'budgets' measuring how much private information can be leaked, there are still two ways to protect data in data mining. The first method is to release privacy-preserving answers. As we only release answers to the queries we receive in this situation, the randomness we add, which is informally proportional to information users need, is much smaller than information in the whole data set, thus the output can have small information to noise ratio. However, if we answer multiple queries, attackers may combine all answers to guess individual information. Therefore, our 'budget' could be exhausted at a point and then the data could never be used again.

The other method is to release a privacy-preserving version of synthetic data. The data could be used to answer as many queries as people want, and the attacker cannot infer more information than the data themselves. However, as the information released here is much more than what is needed to answer any specific query, sometimes the randomness/perturbation required is more than that required by the first method.

In this dissertation, I discuss my research in developing privacy-preserving versions of data mining algorithms. As most of the work uses the definition of differential

privacy, I will introduce some background knowledge of differential privacy in the second chapter. Chapter 3 presents three algorithms on two differentially private query answering problems. The first problem is to return SNPs most correlated to a disease, and I provide two efficient algorithms that make an accurate but previously inefficient algorithm feasible. The second problem is to learn a model minimizing empirical risk while selecting features. Our algorithm beats state-of-the-art algorithms. Chapter 4 presents three algorithms on differentially private data publishing. The first algorithm assumes existence of public data, and assigns weights to the public data so they are statistically similar to the private data. Analysis on the weighted public data is more accurate than analysis on the private data. Then I present two algorithms that assume the published data is for supervised learning (one for classification and one for regression), and the prediction rules are continuous with respect to predictors. The data these algorithms publish perform very well on learning tasks.

Chapter 2

Differential Privacy

Differential privacy [14, 15] is one of the most popular definitions of privacy today. Intuitively, it requires that the algorithm outputting information about an underlying dataset is robust to any change of one sample, thus protecting privacy. This chapter mathematically defines differential privacy and introduces some related methods, as well as some other differentially private machine learning algorithms. In the last part of this chapter, I will give some examples on how differential privacy is used in practice.

2.1 Definition of Differential Privacy

The definition of neighboring datasets reflects 'change of one sample'.

Definition 2.1.1: the *distance* between two datasets D and D' is the smallest number of sample changes needed to change one into another, and D and D' are called *neighbors* if the distance between them is 1. The neighboring relationship is denoted $D \sim D'$.

The original definition of differential privacy given rise to three different understandings of sample changes. Some interpret this as replacement $D - \{x\} = D' - \{x'\}$, some only consider addition/deletion $D = D' \cup \{x\}$, while others consider both. The last interpretation gives the shortest distance, and the distance by the second interpretation lower bounds the last one by the first by a factor of 2.

Most of the differentially private algorithms preserve privacy by injecting randomness, and the required randomness usually does not vary much if we switch from one definition of distance to another. Thus they usually work for all definitions. Therefore, I do not distinguish the distance functions unless we go through details of algorithms.

Definition 2.1.2: a random algorithm $\tilde{f}$ satisfies *(ε, δ)-differential privacy* [14, 15] for two non-negative numbers ε and δ iff for all neighbors $D \sim D'$, and all subsets S in $\tilde{f}$'s range, as long as the following probabilities are well-defined, there holds

$$P(\tilde{f}(D) \in S) \leq \delta + e^{\varepsilon} P(\tilde{f}(D') \in S)$$

The numbers ε and δ are called the *privacy budgets*. Intuitively, the number δ represents the probability that an algorithm's output varies by more than a factor of e^{ε} when applied to a dataset and any one of its neighbors. A lower value of δ signifies greater confidence and a smaller value of ε tightens the standard for privacy protection. The smaller ε and δ are, the closer $P(\tilde{f}(D) \in S)$ and $P(\tilde{f}(D') \in S)$ are, and the stronger the protection is.

Generally, there are two reasons to introduce δ in this definition. First, some algorithms depend on a small δ, such as the smooth sensitivity framework and the sample and aggregate framework in [48]. Second, some algorithms require infinite computational resources to ensure $\delta = 0$, while in practice we can only provide finite resources. Therefore, the output is approximate and a small δ must be introduced. In the first case, δ is a fixed number; in the second case, given more and more computational power, δ may tend to 0.

There is also a commonly used heuristic to choose δ[24]: when there are n samples in the dataset, $\delta \in o(1/n)$. This is because we can construct an algorithm satisfying $(0, \delta)$-differential privacy with high probability of breach privacy when $\delta >$

$1/n$. The algorithm releases each sample in the dataset with probability δ, and the release of different samples is independent. It is easy to prove that the algorithm is differentially private. However, the algorithm releases $n\delta > 1$ samples from the dataset. To prevent such leakage, δ must be significantly smaller than $1/n$.

Typically, $(\varepsilon, 0)$-differential privacy is simplified in notation to ε-differential privacy. With (ε, δ)-differential privacy, when $\delta > 0$, there is still a small chance that some information is leaked. When $\delta = 0$, the guarantee is not probabilistic. Reference [12] shows that, in terms of mutual information, ε-differential privacy is much stronger than (ε, δ)-differential privacy, even if $\delta = o(e^{-n})$.

According to the definition of differential privacy, as long as $\tilde{f}(D)$ depends on D, it must be random.

2.2 Differentially Private Algorithms

This section first introduces concepts of queries, then describes some basic algorithms to answer queries in a differentially private manner. Although there are other algorithms, such as smooth sensitivity framework, or sample and aggregate framework as in [48], I am not going to introduce all of them here. To design differentially private algorithms to answer specific queries, we can either use the basic algorithms in Section 2.2.2 and Section 2.2.3, or combine them as in 2.2.4, or apply them to equivalent/similar queries as in 2.2.5.

2.2.1 Query

Usually, we want the output of an algorithm to be both differentially private and useful. By 'useful', we mean the output accurately answers some queries on the dataset. Definition 2.2.1 defines query and, in the following subsections, some algorithms that guarantee differential privacy are introduced.

Definition 2.2.1: a *query* f is a function that takes a dataset as input. The answer to the query f is denoted $f(D)$.

For example, if D is a medical dataset, then 'how many patients were successfully cured?' is a query, since it takes D as input and outputs a number. The output of a query is not necessarily a number. A more sophisticated query can be 'the vector corresponding to the logistic regression model trained from the dataset'. However, some algorithms, notably the Laplace mechanism, assume answers to queries are numerical, or vectors $f(D) \in \mathbb{R}^p$ and hence not categorical.

2.2.2 The Laplace Mechanism

The Laplace mechanism[17] is a popular ε-differentially private algorithm for queries f with answers $f(D) \in \mathbb{R}^p$, in which sensitivity (Definition 2.2.2) plays an important role.

Definition 2.2.2: given a query f and a norm function $\|.\|$ over the range of f, the *sensitivity* $s(f, \|.\|)$ is defined as

$$s(f, \|.\|) = \max_{D \sim D'} \|f(D) - f(D')\|$$

Usually, the norm function $\|.\|$ is either L_1 or L_2 norm.

The Laplace mechanism[17]: given a query f and a norm function over the range of f, the random function $\tilde{f}(D) = f(D) + \eta$ satisfies ε-differential privacy. Here η is a random vector in $\mathbb{R}^p$ whose probability density function is $p(\eta) \propto e^{-\varepsilon \|\eta\| / s(f, \|.\|)}$.

2.2.3 The Exponential Mechanism

The exponential mechanism[43] is an ε-differentially private method to select one element from a set. Suppose the set to select from is A, and there exists a score function H whose input is a dataset D and a potential answer $a \in A$, and whose output is

a real number. Given a dataset D, the exponential mechanism selects the element $a \in A$ corresponding to the largest score $H(D,a)$ with randomness.

Definition 2.2.3: the *sensitivity* of the score function H is defined as

$$s(H, \|.\|) = \max_{D \sim D', a \in A} \|H(D,a) - H(D',a)\|$$

The exponential mechanism: given a dataset D and a set of possible answers A, if a random algorithm selects an answer based on the following probability, then the algorithm is ε-differentially private:

$$P(a \in A \text{ is selected}) \propto e^{\varepsilon H(D,a)/2s(H,\|.\|)}$$

The Laplace mechanism is related to the exponential mechanism. If $f(D)$ is a vector in $\mathbb{R}^p$, and $H(D,a) = -\|a - f(D)\|, \forall a \in \mathbb{R}^p$, then the exponential mechanism with privacy budget ε is equivalent to the Laplace mechanism with privacy budget $\varepsilon/2$.

According to [45], given a score function H, the exponential mechanism can provide the best output among all ε-differentially private algorithms in the sense that it has the smallest PAC Bayesian bound (an upper bound on expected loss in terms of H). This conclusion means the exponential mechanism is close to the optimal in some sense.

2.2.4 Combination of Differentially Private Algorithm

Sometimes we need to combine several differentially private algorithms in data processing, thus we need to know how combination affects the privacy protection. In this subsection, $\{\tilde{f}_i\}$ represents a set of algorithms differentially private w.r.t. its first input, D is a private dataset, the set of data sets $\{D_i\}$ is a partition of D, and $g()$ represents any function. Reference [42] provides the following two theorems.

Sequential Theorem[17, 15, 16]: if algorithms $\{\tilde{f}_i\}$ are $(\varepsilon_i, \delta_i)$-differentially pri-

vate, and randomness in these algorithms are independent, then $\tilde{f}(D) = g(\tilde{f}_1(D), \tilde{f}_2(D, \tilde{f}_1(D)), ..., \tilde{f}_n(D, \tilde{f}_1(D), \tilde{f}_2(D), ..., \tilde{f}_{n-1}(D)))$ is $(\sum_{i=1}^{n} \varepsilon_i, \sum_{i=1}^{n} \delta_i)$-differentially private.

Intuitively, it means that we can split privacy budget among a sequence of differentially private algorithms and allow an algorithm in the sequence to use both the dataset and outputs of the previous algorithms, while the final output is still differentially private. Some more sophisticated forms of this theorem can be found in [18, 49].

If a query f maps datasets to a vector of length p, it could be treated as p sub-queries, $\{f_1, ..., f_p\}$. Based on the sequential theorem, we can prove that, it is non-trivial to get a better answer by simply splitting f into p sub-queries. By using the Laplace mechanism as an example, based on the definition of sensitivity, we first have $\sum_{i=1}^{p} s_{f_i,\|.\|} \geq s(f, \|.\|)$. If we answer f directly, for each element of the answer vector, the noise added is from $Lap(s(f, \|.\|)/\varepsilon)$. When we answer p sub-queries separately, only if $\sum_{i=1}^{p} s_{f_i,\|.\|} = s(f, \|.\|)$, and each sub-query f_i is assigned $\varepsilon s(f_i, \|.\|)/s(f, \|.\|)$ privacy budget, the noise added to the answer of f_i is from the same distribution $Lap(s(f, \|.\|)/\varepsilon)$. Under these conditions, answering sub-queries separately provides the same utility as answering f directly. Otherwise, if the privacy budget is assigned in a different way, or $\sum_{i=1}^{p} s_{f_i,\|.\|} > s(f, \|.\|)$, there must be at least one element in the answer query that has larger noise than $Lap(s(f, \|.\|)/\varepsilon)$. Therefore, we cannot improve accuracy of all elements by simply splitting a query into many sub-queries.

Parallel Theorem: if $\{\tilde{f}_i\}$ are ε-differentially private and have independent randomness, given a partition $\{D_i\}$ of the dataset D, the addition/deletion distance in Definition 2.1.2, then $\tilde{f}(D = \cup D_i) = g(\tilde{f}_1(D_1), \tilde{f}_2(D_2), ..., \tilde{f}_n(D_n))$ is ε-differentially private.

If we apply ε-differentially private algorithms to each partition of the dataset, the combined output is still ε-differentially private. The partitioning here can be either independent of private data or based on output of another differentially private algorithm. If the third distance in Definition 2.1.2 is used, then the whole algorithm $\tilde{f}$ is

2ε-differentially private.

Both the sequential theorem and the parallel theorem study combinations of multiple algorithms. A natural question is, whether it is always beneficial to the utility to run the same algorithms many times and to average the outputs. The answer is no. In the sequential scenario, the privacy budget has to be split among several runs; in the parallel scenario, each partition has less samples than D does. In both cases, the ratio between the amount of noise applied and the accurate answer is larger than the corresponding ratio for the original query. Therefore, simply averaging them doesn't necessarily lead to better performance.

2.2.5 Properties of Differentially Private Algorithm

In differentially private learning, we make changes to query answering algorithms. These changes are not limited to adding randomness to the output directly, but include replacing the original query with equivalent or similar queries. People take these changes for better utility.

Here we design two examples to show utility improvement. The first example illustrates how equivalent queries can help. Suppose the original query is the average age of male patients, the Laplace mechanism is used to achieve 1-differential privacy, the ages are between 0 and 60, and there are 100 people in the dataset. Furthermore, we assume the true average age is 30. The sensitivity of average age is 60, and this maximum change can be achieved when D has only one man aged 0, and D' has only one man aged 60. When the Laplace mechanism is applied to this query, the noise we add to the true answer is from $Lap(60/\varepsilon)$, and its standard deviation is $60\sqrt{2} \simeq 85$ due to the property of the Laplace distribution. When such large noise is added to the true answer 30, the output is rather useless. However, if the Laplace mechanism is applied to both count of males and sum of male ages, each with privacy budget $1/2$, and use the noisy

answers to compute the average, then the noise can be much smaller. The sensitivity of count of males is 1, thus the noisy count has mean 100 and standard deviation 1.4. The sensitivity of sum of male ages is 60, thus the noisy count has mean 3000 and standard deviation 85. The output "noisy sum of male ages/noisy count of males", has mean 30 and standard deviation 0.9 by simulation. Thus the equivalent query "compute sum of male ages and count of males separately and compute average accordingly" is much better than the original query asking for average age, in terms of accuracy, even if the same privacy budget is used.

A second example shows that approximate queries can improve performance, too. Suppose the original query asks the median of patient ages (between 0 and 60) from a data set with $n = 100$ patients, given privacy budget $\varepsilon = 1$, and we already know those ages in such a dataset are from a Gaussian distribution, but we do not know the true distribution $N(30, 100)$ when we issue the query. We may choose to either issue the original median query, or an approximation query asking about the average (using the trick in the previous paragraph). For the original query, the sensitivity is 60, which can be achieved when D has only one patient aged 0, and D' has only one patient aged 60. Same as the analysis above, the Laplace mechanism adds noise whose standard deviation is 85, to the true median which is around 30, to make the output useless. However, if we use the trick to compute average above, then use the average as median, the difference between the output and the true median is around 0.3 by simulation, which is much smaller than the noise added corresponding to the original query. To summarize, equivalent or approximate queries can improve utility on original queries.

Besides, the second example also shows that, some conclusions in statistics are no longer true when differential privacy is required, for example, median is no longer robust. The reason is, while we usually focus on properties in an "average" sense, for a "regular" dataset, differential privacy cares about extreme cases only, such as only one

sample in the dataset, all samples are on the boundary of sample space, etc. Thus we need to double check conclusions from other machine learning algorithms before using them when differential privacy is required.

2.3 Differentially Private Learning Algorithms

I will explain some existing differentially private learning algorithms in this section. As the introduction states, there are two general ways to differentially private data mining. The first is to answer a specific query, while the second is to publish a synthetic dataset for data mining. Here I discuss them separately. Algorithms proposed in this dissertation are not included here.

Usually these algorithms have some assumptions on the data. Informally speaking, stronger assumptions lead to smaller randomness inserted, and thus better utility.

2.3.1 Query Answering Algorithm

There are many differentially private query answering learning algorithms, and they can be grouped according to the basic approaches they use to compute a privacy-preserving answer.

Some approaches use either the exponential mechanism or the Laplace mechanism to perturb the information learned from data. For example, [60, 46, 52] use the Laplace mechanism, while [9, 61] use the exponential mechanism. For some other approaches that have many iterations or multiple steps, the Laplace mechanism and the exponential mechanism are applied to output parameters of each iteration/step. Such approaches include [27, 35, 29, 23, 30, 47, 63].

Some algorithms add noise to the target function and derive information from the noisy function as the answer. This technique is called objective perturbation. Some examples include [68, 7, 8, 51].

Some algorithms use the idea of the sample and aggregate framework. The framework is specially designed for queries that can be measured with a small number of samples. First, they split the dataset into many small subsets and run original algorithms on each subset. Next, they combine the results from all subsets to come up with an answer, adding noise in this aggregation step. Algorithms that employ this idea include [48, 58]. The linear regression in [16] is partially based on this idea.

Some algorithms explore other ideas. For example, [37] partitions the sample space and uses counts in each partition to estimate the density function. Reference [31] interprets a model learned by regular learning algorithms as a function, and uses another function to approximate it by iteratively minimizing the largest distance in a differentially private way.

Most output perturbation and objective perturbation algorithms require a bounded sample space. This is because unbounded sample space usually leads to unbounded sensitivity. Algorithms based on the sample and aggregate framework don't have this limitation, however most of them use (ε, δ)-differential privacy. In practice, if the sample space is unbounded and we want to use ε-differential privacy, we can simply truncate the values in pre-processing. If the rule to truncate is independent of the private data, then the truncation is privacy-safe.

2.3.2 Data-publishing Algorithms

Many differentially private data release algorithms have been described, such as [6, 5, 22, 11, 26]. In this subsection, we focus on data release algorithms that are either useful for machine learning or based on machine learning algorithms.

Many papers use partition based algorithms to release data. Reference [63] assumes the density function is smooth, References [10, 69] assume the data's format permits it to be organized in a tree, and [47] assumes partitions can preserve most im-

portant information for further data mining. All these assumptions motivate partitioning the sample space and publishing counts in each partition. With respect to the algorithm design, the algorithms can be divided into two groups. Reference [47] first partitions the sample space using the exponential mechanism and then adds noise to the counts using the Laplace mechanism. The others ([10, 69, 63]) generate noisy counts with the Laplace mechanism for each cell and then partition according to the noisy counts.

Other data release algorithms don't depend on partitioning. Some of them assume the private data is fit well by some family of models. They select an optimal model from that family privately and then synthesize new data according to the selected model. Some others assume some property (like sparsity) of the data, and propose algorithms which make use of that property.

References [46, 52] publish graph generator models based on the assumption that the private data is fit well by some parametrized generative model. Though the two algorithms use different generative models, they both train a model first, add Laplace noise to the parameters of the model, and then use the noisy model to generate a new graph.

Reference [62] represents the network structure by using a statistical hierarchical random graph model. Unlike the two models above, the number of parameters in this model is proportional to the number of nodes in the graph. Thus we may introduce too much noise if we use the Laplace mechanism to publish the model. As the parameter space is very large and no score function exists, which is both simple and meaningful, it is not easy to use the exponential mechanism directly. To overcome this difficulty, the authors propose an algorithm based on the Markov Chain Monte Carlo procedure. It uses the Metropolis–Hastings algorithm to draw a model from the distribution in the exponential mechanism given the score function is the likelihood function. Though the likelihood function is complicated w.r.t. the whole space of parameters, it is simple w.r.t.

one parameter if all the others are fixed. Thus the Markov Chain Monte Carlo (MCMC) procedure is possible. The graph can be reconstructed from the output model after many iterations.

[39] assumes the data matrix is sparse, thus the algorithm can make use of results from compressive sensing research. Informally speaking, if we randomly project a high-dimensional data matrix to a low-dimensional space, and then attempt to recover the high-dimensional matrix using the low-dimensional embedding and the constraint that the recovered matrix is sparse, there is high probability the recovered matrix exactly matches the original one. [39] first randomly projects the data matrix, then adds noise to the compressed information, and reconstructs data from the noisy compressed information. As the dimension of compressed information is much smaller than that of the original data matrix, the scale of Laplace noise needed to preserve ε-differential privacy can be reduced from $O(\sqrt{n}/\varepsilon)$ to $O(\log n/\varepsilon)$ given n samples.

2.4 Practical Use of Differential Privacy

Though the research community has developed many algorithms satisfying differential privacy, the industry has just started to use differential privacy to collect data for learning. These algorithms are all in the category of differentially private data publishing, which create perturbed versions of data, and run usual learning algorithms on those data.

For example, Google collects software statistics [2] and traffic information [4] with the Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR) technique [21]. Apple, since June 2016, uses differential privacy to discover usage patterns of iOS users[1]. Used by these Internet giants, differential privacy has proven itself to be reliable and up to industrial privacy-protecting standards.

In all these cases, companies guarantee users that no devices except theirs are

trusted. Even encryption is not considered reliable (otherwise they can upload encrypted values and store statistics in their servers only). This is consistent with the 'worst case' intuition in differential privacy, in case the attacker happens to correctly decrypt. In accordance with this concern, at least in my work, when differential privacy is used, encryption is not seen as safe.

Chapter 2 is based on the survey paper *Differential Privacy and Machine Learning: a Survey and Review.* Zhanglong Ji, Zack C Lipton, Charles Elkan arXiv preprint arXiv:1412.7584, 2014. The dissertation author was the primary author of the paper.

Chapter 3

Differentially Private Query Answering Algorithms

This chapter presents my work on differentially private query answering algorithms, namely differentially private genome-wide association studies (GWAS) and differentially private empirical risk minimization with feature selection.

3.1 Differentially Private Genome-Wide Association Studies

In this section, I discuss how to combine differential privacy with genome-wide association studies, in order to find SNPs most correlated to a disease.

3.1.1 Background

Rapid developments in whole-genome sequencing technologies in recent years have made the collection of high quality genetic data faster and more economically feasible. Many types of genetic research can benefit from having a large amount of genetic data. For example, in genome-wide association studies (GWAS), which are usually used to examine correlations between single-nucleotide polymorphisms (SNPs) and a disease, increasing the number of DNA samples available for analysis allows researchers to make more accurate statistical inference and improve the overall quality

of the analysis.

Encouraging data sharing among researchers is the first step towards taking advantage of the benefits brought about by the rapid growth in genetic data collection. However, being able to share genetic data without compromising the study participants' privacy remains one of the biggest challenges in genetic research. While it is clear that individual level genetic data deserve a high level of protection, for many years it was widely considered safe to release to the public aggregate genetic data pooled from thousands of individuals without compromising genetic study participants' privacy. However, [28] in 2008 demonstrated that one can use publicly available aggregate genetic data, such as SNP data from the International HapMap Project (it has been brought down now [3]), to infer whether an individual has participated in a study. If a study involves only patients of a disease, or under a condition, then leaking one's presence means leaking his health information. Cautious about the potential breach of genetic study participants' privacy, the National Institute of Health (NIH) quickly responded to the attack by mandating an elaborate approval process that every researcher has to go through in order to gain access to aggregate genetic data. This NIH policy remains in effect today.

Reference [28]'s attack and NIH's subsequent reaction spurred research interest in privacy-preserving methodologies for GWAS data. When differential privacy has shown great promise as a basis for privacy-preserving methodologies, we have seen privacy-preserving methods based on differential privacy applied to real human GWAS data in recent studies (e.g., [59, 32, 65]). However, if we apply the Laplace mechanism to statistics, such as those in [59], the performance may not satisfy researchers. An algorithm based on Hamming distance (number of minimum sample changes to change a set to the other) is proposed in [32], and proven much better than those algorithms based on statistics and p-values. But the brute-force algorithm to compute the distance

is inefficient, and the approximate algorithm in it is wrong as it does not consider difference of orders of SNPs before and after a sample change. Reference [65] proposed a greedy algorithm to compute the distance for the allelic test, however we can still construct cases where the algorithm fails. Therefore, there were no efficient algorithms to compute Hamming distance based scores.

My contribution is to find efficient accurate algorithms for the allelic test, and the Transmission Disequilibrium Test (TDT). Only with these algorithms can we make use of the Hamming distance based scores.

In this section, I first introduce previous algorithms and the problem to find the Hamming distance, then introduce our findings.

3.1.2 Existing Methods for Genome Researches

All differentially private algorithms applied to genome researches share the same framework. Informally speaking, for all the SNPs, they first derive scores describing SNPs' correlation to the disease, and then either select SNPs with highest scores according to the exponential mechanism or the Laplace mechanism. The scores could be either negative p-value, the statistics, or based on Hamming distance. Negative p-value is used here to ensure that larger scores always correspond to desired SNPs.

While scores like p-values or statistics are easy to understand, we need to explain the score based on Hamming distance here. Given a threshold of significance on p-value, say p, a SNP is significant by the test, if its p-value is smaller than the threshold p, and insignificant otherwise. For significant SNPs, the score is the least number of sample changes to make a SNP insignificant, d_+. If a SNP is insignificant, we first compute the least number of sample changes to make the SNP significant, d_-, then use $1 - d_-$ as the score. By intuition, this score has the following properties: first, the more correlated a SNP is to the disease, the more significant that SNP is by this test, and the larger the

score is likely to be; second, change of one sample leads to at most change 1 on this score on a given SNP.

The score based on Hamming distance performs better than those on statistics or p-values, as it only requires necessary randomness. To explain this conclusion in details, we introduce a new term, local sensitivity. Local sensitivity means the maximum change of score if we change one sample in the actual private data set. Since the randomness added is expected to cover such effect of one sample change, the local sensitivity in fact means "necessary randomness". The sensitivity we defined in Section 2.2, however, determines the actual randomness added to the output. The closer these two sensitivities are, the smaller unnecessary randomness there is. Therefore, when the sensitivity and the local sensitivities are the same, as in this algorithm, only necessary randomness is added, which means good utility. Comparison of these scores can be found in [32, 66], etc.

However, the Hamming distance is not easy to compute. Theoretically, we can build a large graph in which each node is a state of dataset, and two nodes are connected only if they are neighbors. Then we denote which datasets are significant/insignificant, and run a shortest path algorithm to determine the distances. However, this algorithm is inefficient. The point is, the number of nodes/states could be so large that the correct score cannot be computed within reasonable time. [32] proposed an approximate algorithm, but the algorithm is wrong as it did not consider difference of orders of SNPs before and after a sample change. [65] proposed a greedy algorithm, but it was wrong again as there is no guarantee that its sensitivity is 1.

3.1.3 Efficient Accurate Algorithm for the Allelic Test

In this section, I introduce what is allelic test, what kind of data it uses, how it computes statistics, properties of statistics, how we visualize the optimization problem,

and what our algorithm looks like.

The allelic test is also known as the Cochran-Armitage trend test for the additive model. It aims to find the SNPs that are most correlated to a disease. The intuition behind this test is: if a variant does cause a disease, then it must be highly correlated to the disease. In the test, there are two groups: case and control. The case group usually consists of people with the disease, while the control group consists of healthy people. The two groups' sizes are public, but the two sizes may or may not be the same. For each person in the study, his/her genome information on, say, 100000 SNPs, are included. If there is no privacy concern, researchers usually use p-value of statistical tests on independence to see how likely a SNP is correlated to the disease. The smaller the p-value is, the more correlated a SNP is to the disease.

The data it uses are as follows. Let's refer to the case group's data and the control group's data collectively as a database, and call the data for an individual a record. The number of people in the case group is R, and the number of people in the control group is S. Given a SNP, there are three counts of people with 0, 1, 2 minor variants in the control group, s_0, s_1, s_2, and three counts for the case group, r_0, r_1, r_2. Due to the definition above, there are two constraints $S = s_0 + s_1 + s_2$, $R = r_0 + r_1 + r_2$. All the numbers are shown in the genotype table (See Table 3.1).

Usually only information of the case group needs protection, as inferring a person in the control group means s/he is healthy. Besides, the numbers of people in both groups, R and S, are usually public. Therefore, only numbers r_0, r_1, r_2 need to be protected. In the following part of this section, we only protect counts in case group with differential privacy, while other numbers are public and fixed.

Since only S, R, s_0, s_1, s_2 are public, and $r_2 = R - r_0 - r_1$, two counts from the private data, r_0 and r_1, can represent the private data. Change of private data can thus be reflected by change on r_0 and r_1. In Figure 3.1, we show if the original dataset cor-

Table 3.1. Genotype table for a SNP

	# of minor alleles			
	0	1	2	Total
Case	r_0	r_1	r_2	R
Control	s_0	s_1	s_2	S

responds to (r_0, r_1) at a certain SNP, what pairs of (r_0, r_1) may its neighbor corresponds to. There are only seven possibilities, $(r_0 \to r_0+1, r_1 \to r_1)$, $(r_0 \to r_0+1, r_1 \to r_1-1)$, $(r_0 \to r_0, r_1 \to r_1+1)$, $(r_0 \to r_0, r_1 \to r_1)$, $(r_0 \to r_0, r_1 \to r_1-1)$, $(r_0 \to r_0-1, r_1 \to r_1)$, and $(r_0 \to r_0-1, r_1 \to r_1+1)$. Therefore, samples changes can be expressed by $r_0 \sim r_1$ plot in Figure 3.1.

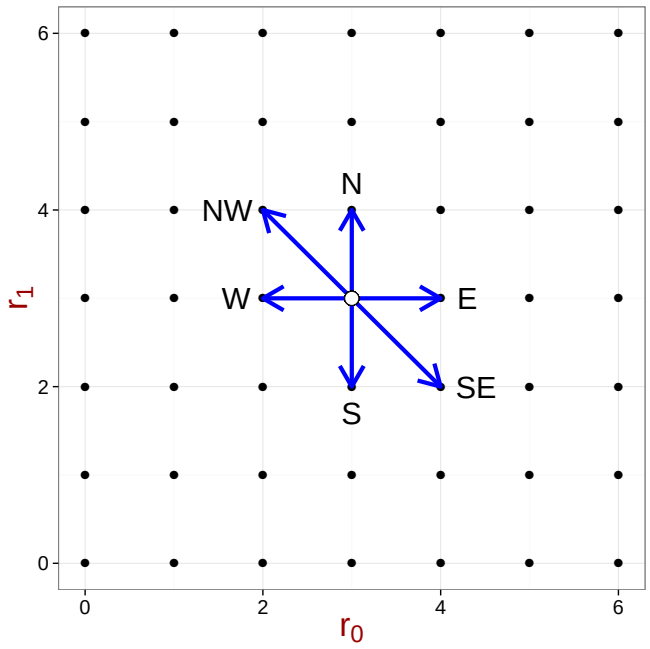


Figure 3.1. Legal moves in the space of genotype tables with fixed R, S, s_0, s_1, and s_2.

The allelic test statistic based on a genotype contingency table (Table 3.1) is equivalent to the χ^2-statistic based on the corresponding allelic contingency table (Table

Table 3.2. Allelic table

	Allele type		
	Minor	Major	Total
Case	r_1+2r_2	$2r_0+r_1$	2R
Control	s_1+2s_2	$2s_0+s_1$	2S

3.2). The allelic test statistic is

$$Y_A = \frac{2(R+S)\left[(2r_0+r_1)S-(2s_0+s_1)R\right]^2}{RS(2r_0+r_1+2s_0+s_1)(r_1+2r_2+s_1+2s_2)}$$

Given the null hypothesis (the disease is independent of the SNP), Y_A satisfies the $\chi^2(1)$ distribution. We can thus either compute p-values of the SNP accordingly, and select SNPs with smallest p-values, or simply select SNPs with largest statistics.

The statistic Y_A has the following properties:

Lemma 3.1.1: Given all the public information, Y_A depends on $2r_0+r_1$ only, and Y_A is monotonically decreasing when $2r_0+r_1 < \frac{R}{S}(2s_0+s_1)$, and monotonically increasing when $2r_0+r_1 > \frac{R}{S}(2s_0+s_1)$.

Proof. Let's denote $r = 2r_0+r_1$ and $s = 2s_0+s_1$. Note $s_1+2s_2 = 2S-s$, we have

$$Y_A = \frac{2(R+S)(rS-sR)^2}{RS(r+s)(2R+2S-r-s)}$$

Since numbers S, R, s are all public, Y_A depends on $r = 2r_0+r_1$ only.

Furthermore,

$$\begin{aligned}
\frac{dY_A}{dr} &= \frac{2(R+S)}{RS}\frac{d}{dr}\frac{(rS-sR)^2}{(r+s)(2R+2S-r-s)} \\
&= \frac{2(R+S)}{RS(r+s)^2(2R+2S-r-s)^2}\Big[(r+s)(2R+2S-r-s)\frac{d}{dr}(rS-sR)^2 \\
&\quad -(rS-sR)^2\frac{d}{dr}(r+s)(2R+2S-r-s)]\Big] \\
&= \frac{2(R+S)}{RS(r+s)^2(2R+2S-r-s)^2}[2(r+s)(2R+2S-r-s)(rS-sR)S \\
&\quad -(rS-sR)^2(2R+2S-r-s-r-s)] \\
&= \frac{4(R+S)(rS-sR)}{RS(r+s)^2(2R+2S-r-s)^2}[(r+s)(2R+2S-r-s)S \\
&\quad -(rS-sR)(R+S-r-s)] \\
&= \frac{4(R+S)(rS-sR)}{RS(r+s)^2(2R+2S-r-s)^2}\big[sR^2-s^2R+rSR+rS^2+3sRS-rsS-srR \\
&\quad +2sS^2-s^2S\big] \\
&= \frac{4(R+S)(rS-sR)}{RS(r+s)^2(2R+2S-r-s)^2}[sS(2S-s)+sR(2S-s)+sRR+sRS-srR \\
&\quad +rSR+rSS-rsS]
\end{aligned}$$

Given that $sRR+sRS-srR+rSR+rSS-rsS$ is a linear function of r, when $0 \le r \le 2R$ it must be between $sRR+sRS$ and $(2S-s)(R+S)R$. When both bounds are non-negative, we have $sRR+sRS-srR+rSR+rSS-rsS \ge 0$ for all $0 \le r \le 2R$. As $0 \le s \le 2S$, we also have $sS(2S-s)+sR(2S-s) \ge 0$. Thus $sS(2S-s)+sR(2S-s)+sRR+sRS-srR+rSR+rSS-rsS$ is non-negative. Obviously, $\frac{4(R+S)}{RS(r+s)^2(2R+2S-r-s)^2}$ is positive, therefore $\frac{dY_A}{dr}$ has the same signal as $rS-sR$. When $r < Rs/S$, we have $\frac{dY_A}{dr} \le 0$, and when $r > Rs/S$, we have $\frac{dY_A}{dr} \ge 0$. □

Given Lemma 3.1.1, we plot the relation between Y_A and r_0, r_1 as in Figure 3.2. Each dot represents a pair of (r_0, r_1), and dots on the same green dash line (slope=-1/2)

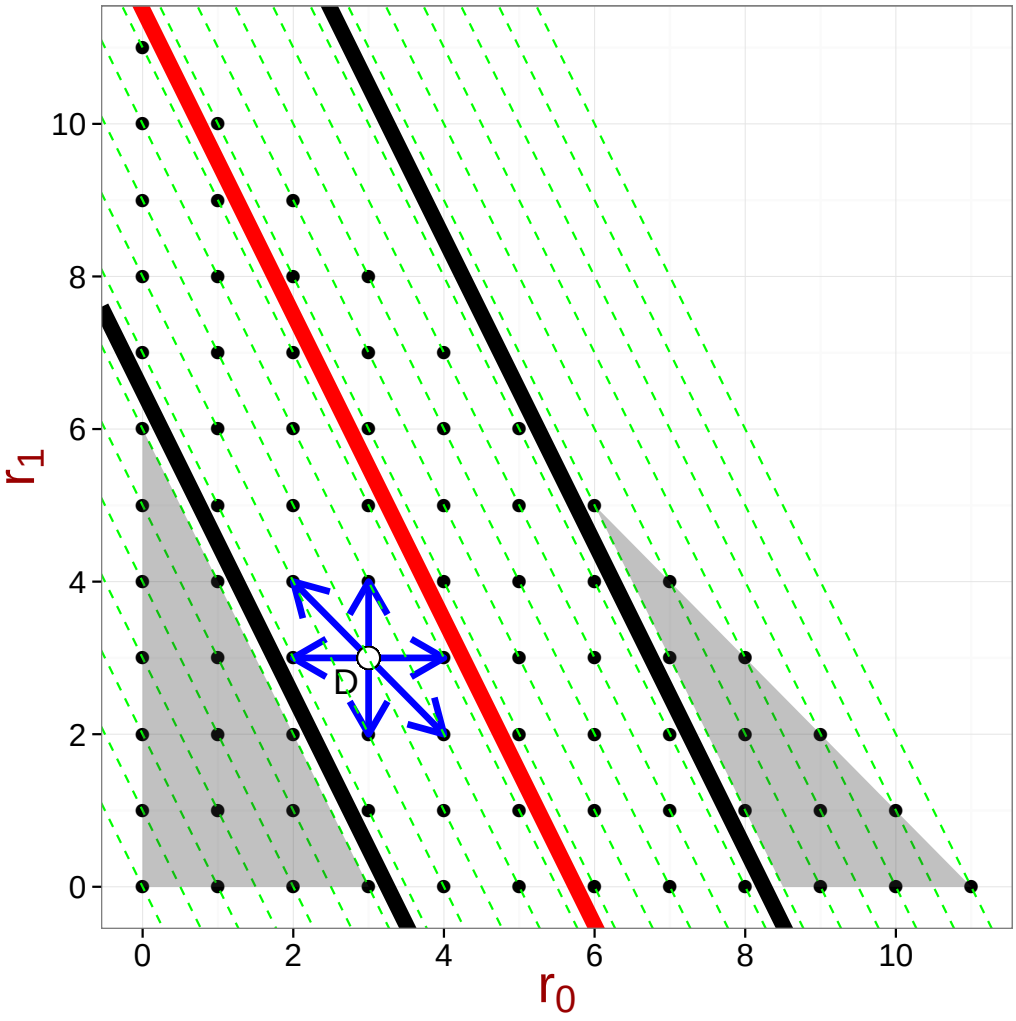


Figure 3.2. Relation between Y_A and r_0, r_1 given R, S, s_0, s_1, and s_2.

have the same value of Y_A. The shaded areas correspond to significant SNPs, while the area without shade corresponds to insignificant SNPs. The black lines represent borders of significant/insignificant areas and the red line represents the position where Y_A is minimized. The blue arrows show effect of a sample change on the pair (r_0, r_1).

The area of legal pairs of (r_0, r_1) is an isosceles right triangle, whose left edge is determined by $r_0 \geq 0$ and $r_0 + s_0 > 0$, lower edge determined by $r_1 \geq 0$ and $r_1 + s_1 > 0$, and the longer edge by $r_0 + r_1 \leq R$, $r_0 + r_1 < R + s_2$. We simplify these constraints to

$r_0 \geq I(s_0 = 0)$, $r_1 \geq I(s_1 = 0)$, and $r_0 + r_1 \leq R - I(s_2 = 0)$. All these constraints are irrelevant to private data.

The position of the black lines are determined by the threshold of significance we choose. Say, if we choose p-value< 0.05 as definition of significance, then $Y_A < 3.84$ should be the shaded area. Besides, based on Lemma 3.1.1, we know that $Y_A = 3.84$ has at most two solutions $2r_0 + r_1 = C_1$ and $2r_0 + r_1 = C_2$, each of which corresponds to a black line. Therefore, there are at most two black lines. When the threshold of p-value is too small, the threshold for Y_A could be too large, thus there could be only one black line, or none in the graph.

Now the score based on Hamming distance turns out to be:

1. For a point in the shaded area, how many steps are required to reach the area without shade if each step is one of the six blue steps?

2. For a point in the area without shade, how many steps are required to reach the shaded area if each step is one of the six blue steps?

Since the boundary between the shaded and non-shaded areas is always a straight line with slope -1/2, the Hamming distance can be solved if we can compute the following two distance.

1. For an arbitrary point, how many steps are required to cross a line $2r_0 + r_1 = C$ to its left.

2. For an arbitrary point, how many steps are required to cross a line $2r_0 + r_1 = C$ to its right.

Algorithm 1 computes the two numbers above, and its correctness is proven in Theorem 3.1.2 and Theorem 3.1.3.

Algorithm 1. Distance from a point (r'_0, r'_1) to a line $2r_0 + r_1 = C$

INPUT: a point (r'_0, r'_1) and a number C indicating position of the line.
OUTPUT: the smallest number of steps required to cross the line from (r'_0, r'_1).

1: If $2r'_0 + r'_1 > C$, go to Step 2; otherwise go to Step 3.
2: If $r'_1 + 2I(s_0 = 0) \le C$, return $\lceil r'_0 - (C - r'_1)/2 \rceil$; otherwise return $\lceil r'_0 + I(s_0 = 0) + r'_1 - C \rceil$.
3: If $2(R - I(s_2 = 0)) - r'_1 \ge C$, return $\lceil (C - r'_1)/2 - r_0 \rceil$; otherwise return $\lceil C - r'_0 + I(s_2 = 0) - R \rceil$.

Theorem 3.1.2: the distance between a point (r'_0, r'_1) and a straight line $2r_0 + r_1 = C$ to its left is $\lceil r'_0 - (C - r'_1)/2 \rceil$ if $r'_1 + 2I(s_0 = 0) \le C$, and $\lceil r'_0 + I(s_0 = 0) + r'_1 - C \rceil$ otherwise. Here $\lceil \rceil$ means ceiling function and $\lfloor \rfloor$ means floor function.

Proof. When the line $2r_0 + r_1 = C$ is to the left of (r'_0, r'_1), it means $2r'_0 + r'_1 > C$. Suppose a path starts from (r'_0, r'_1), and takes n_E steps to the east, n_N steps to the north, n_W steps to the west, n_S steps to the south, n_{NW} steps to the northwest, and n_{SE} steps to the southest, to $(r'_0 + n_E - n_W - n_{NW} + n_{SE}, r'_1 + n_N - n_S + n_{NW} - n_{SE})$ satisfying $2r_0 + r_1 \le C$. Then

$$\begin{aligned} 2(r'_0 + n_E - n_W - n_{NW} + n_{SE}) + r'_1 + n_N - n_S + n_{NW} - n_{SE} &\le C \\ r'_0 + n_E - n_W - n_{NW} + n_{SE} &\ge I(s_0 = 0) \\ n_E + n_W + n_N + n_S + n_{NW} + n_{SE} &= \#steps \end{aligned}$$

So we conclude

$$\begin{aligned} 2r'_0 + r'_1 - C &\le 2(-n_E + n_W + n_{NW} - n_{SE}) - n_N + n_S - n_{NW} + n_{SE} \\ &= -2n_E + 2n_W - n_N + n_S + n_{NW} - n_{SE} \\ &\le 2(n_E + n_W + n_N + n_S + n_{NW} + n_{SE}) \\ &= 2\#steps \\ \#steps &\ge \lceil r'_0 - (C - r'_1)/2 \rceil \end{aligned}$$

and

$$\begin{aligned}
r'_0 + r'_1 - C &\leq 2(-n_E + n_W + n_{NW} - n_{SE}) - n_N + n_S - n_{NW} + n_{SE} - r'_0 \\
&= -2n_E + 2n_W - n_N + n_S + n_{NW} - n_{SE} - r'_0 \\
&\leq -2n_E + 2n_W - n_N + n_S + n_{NW} - n_{SE} - r'_0 + (r'_0 + n_E - n_W - n_{NW} + n_{SE} \\
&\quad -I(s_0 = 0)) \\
&\leq -n_E + n_W - n_N + n_S - I(s_0 = 0) \\
&\leq n_E + n_W + n_N + n_S + n_{NW} + n_{SE} - I(s_0 = 0) \\
&= \#steps - I(s_0 = 0) \\
\#steps &\geq \lceil r'_0 + I(s_0 = 0) + r'_1 - C \rceil
\end{aligned}$$

Since the first bound is larger when $r'_1 + 2I(s_0 = 0) \leq C$, and the second bound is larger when $r'_1 + 2I(s_0 = 0) > C$, $\#steps$ is at least $\lceil r'_0 - (C - r'_1)/2 \rceil$ if $r'_1 + 2I(s_0 = 0) \leq C$, and $\lceil r'_0 - I(s_0 = 0) + r'_1 - C \rceil$ otherwise.

Besides, it is easy to verify that these bounds can be achieved. When $r'_1 + 2I(s_0 = 0) \leq C$, we can go west in the figure for $\lceil r'_0 - (C - r'_1)/2 \rceil$ steps to $(\lfloor (C - r'_1)/2 \rfloor, r'_1)$ which is still legal, and satisfies $2(\lfloor (C - r'_1)/2 \rfloor) + r'_1 \leq C$. When $r'_1 + 2I(s_0 = 0) > C$, we can go west in the figure for $r'_0 - I(s_0 = 0)$ steps first, then go south for $\lceil r'_1 + 2I(s_0 = 0) - C \rceil$ steps, to $(I(s_0 = 0), \lfloor C \rfloor - 2I(s_0 = 0))$ which is still legal, and satisfies $2I(s_0 = 0) + \lfloor C \rfloor - 2I(s_0 = 0) \leq C$. Therefore, this theorem provides the Hamming distance between a point (r'_0, r'_1) and a straight line $2r_0 + r_1 = C$ to its left. □

Theorem 3.1.3: the distance between a point (r'_0, r'_1) and a straight line $2r_0 + r_1 = C$ to its right is $\lceil (C - r'_1)/2 - r_0 \rceil$ if $2(R - I(s_2 = 0)) - r'_1 \geq C$, and $\lceil C - r'_0 + I(s_2 = 0) - R \rceil$ otherwise.

Proof. When the line $2r_0 + r_1 = C$ is to the right of (r'_0, r'_1), it means $2r'_0 + r'_1 < C$. We

still define n_E to n_{SE} as in the proof of Theorem 3.1.2, and now the end of the path is still $(r'_0+n_E-n_W-n_{NW}+n_{SE}, r'_1+n_N-n_S+n_{NW}-n_{SE})$ satisfying $2r_0+r_1 \geq C$. Then we have

$$\begin{aligned} 2(r'_0+n_E-n_W-n_{NW}+n_{SE})+r'_1+n_N-n_S+n_{NW}-n_{SE} &\geq C \\ r'_0+n_E-n_W-n_{NW}+n_{SE}+r'_1+n_N-n_S+n_{NW}-n_{SE} &\leq R-I(s_2=0) \\ n_E+n_W+n_N+n_S+n_{NW}+n_{SE} &= \#steps \end{aligned}$$

So we conclude

$$\begin{aligned} C-2r'_0-r'_1 &\leq 2(n_E-n_W-n_{NW}+n_{SE})+n_N-n_S+n_{NW}-n_{SE} \\ &= 2n_E-2n_W+n_N-n_S-n_{NW}+n_{SE} \\ &\leq 2(n_E+n_W+n_N+n_S+n_{NW}+n_{SE}) \\ &= 2\#steps \\ \#steps &\geq \lceil (C-r'_1)/2-r'_0 \rceil \end{aligned}$$

and

$$\begin{aligned}
C - r'_0 &\leq 2(n_E - n_W - n_{NW} + n_{SE}) + n_N - n_S + n_{NW} - n_{SE} + r'_0 + r'_1 \\
&= 2n_E - 2n_W + n_N - n_S - n_{NW} + n_{SE} + r'_0 + r'_1 \\
&= 2n_E - 2n_W + n_N - n_S - n_{NW} + n_{SE} + r'_0 + r'_1 + R - I(s_2 = 0) \\
&\quad -(r'_0 + n_E - n_W - n_{NW} + n_{SE} + r'_1 + n_N - n_S + n_{NW} - n_{SE}) \\
&= n_E - n_W - n_{NW} + n_{SE} + R - I(s_2 = 0) \\
&= n_E + n_W + n_N + n_S + n_{NW} + n_{SE} + R - I(s_2 = 0) \\
&= \#steps + R - I(s_2 = 0) \\
\#steps &\geq \lceil C - r'_0 + I(s_2 = 0) - R \rceil
\end{aligned}$$

Since the first bound is larger when $2(R - I(s_2 = 0)) - r'_1 \geq C$, and the second bound is larger when $2(R - I(s_2 = 0)) - r'_1 < C$, $\#steps$ is at least $\lceil (C - r'_1)/2 - r'_0 \rceil$ if $2(R - I(s_2 = 0)) - r'_1 \geq C$, and $\lceil C - r'_0 + I(s_2 = 0) - R \rceil$ otherwise.

Besides, it is easy to verify that these bounds can be achieved. When $2(R - I(s_2 = 0)) - r'_1 \geq C$, we can go east in the figure for $\lceil (C - r'_1)/2 - r'_0 \rceil$ steps to $(\lceil (C - r'_1)/2 \rceil, r'_1)$ which is still legal, and satisfies $2(\lceil (C - r'_1)/2 \rceil) + r'_1 \geq C$. When $2(R - I(s_2 = 0)) - r'_1 < C$, we can go east in the figure for $R - I(s_2 = 0) - r'_1 - r'_0$ steps first, then go southeast for $\lceil C + 2I(s_2 = 0) - 2R + r_1 \rceil$ steps, to $(\lceil C + I(s_2 = 0) - R \rceil, 2r'_1 - \lfloor C \rfloor - 2I(s_2 = 0) + 2R)$ which is still legal, and satisfies $2(C + I(s_2 = 0) - R) + 2r'_1 - \lfloor C \rfloor - 2I(s_2 = 0) + 2R \geq C$. Therefore, this theorem provides the Hamming distance between a point (r'_0, r'_1) and a straight line $2r_0 + r_1 = C$ to its right. □

Given the Algorithm 1, we have Algorithm 2 to compute Hamming distance based scores.

Algorithm 2. Algorithm to compute Hamming distance based score

INPUT: (r_0, r_1) corresponding to a given SNP, a given threshold p on p-value, counts S, R, s_0, s_1, s_2.

OUTPUT: Hamming distance based score of the SNP.

1: Find the $1-p$ percentile of $\chi^2(1)$ distribution, denote it c.
2: Find the values of $2r_0+r_1$ such that $Y_A = c$. Denote the one larger than $R(2s_0+s_1)/S$ as C_2, and the one smaller than $R(2s_0+s_1)/S$ as C_1, if they exist.
3: Call Algorithm 1 with $r_0' = r_0, r_1' = r_1$ and $C = C_1$.
4: Call Algorithm 1 with $r_0' = r_0, r_1' = r_1$ and $C = C_2$.
5: Return the smaller output value between the two above.

3.1.4 Effcient Accurate Algorithm for the Transmission Disequilibrium Test

This part discusses our algorithm for the Transmission Disequilibrium Test(TDT). It starts with an introduction to the TDT, then the statistic of the TDT, and our algorithm to compute Hamming distance based scores for TDT.

The TDT is a family-based association test for the presence of genetic linkage between a genetic marker and a disease. It measures the over-transmission of an allele from heterozygous parents to an affected offspring. Say, if a parent has two alleles, M and m, this test checks whether the two alleles have the same probabilities to be transmitted to his/her affected offspring. Data of people without the disease are not needed in the TDT.

What is the intuition behind the TDT? Usually both variants for a SNP, say M and m, are transmitted with equal probabilities, then there should be equal numbers of $Mm \times mm \to Mm$ and $Mm \times mm \to mm$ in the whole population. If the variant M leads to the disease, then only families with type $Mm \times mm \to Mm$ are included in the patient family data set; if genotype is mm leads to the disease, then only families with type $Mm \times mm \to mm$ are included in the patient family data set. In both cases, we observe transmission disequilibrium in the data set. Thus the TDT can identify SNPs related to

a disease.

Though the TDT and the allelic test have similar goals, it achieves higher statistical power for rare diseases such as Kawasaki Disease [50]. TDT has also been applied to whole exome sequencing studies in autism [38] and Kashin-Beck disease [64].

Suppose a SNP has two alleles, M and m, based on the genotype of family members, there are 10 types of families in total, namely

$$\begin{aligned}
MM \times MM &\rightarrow MM \\
MM \times mm &\rightarrow Mm \\
mm \times mm &\rightarrow mm \\
MM \times Mm &\rightarrow MM \\
mm \times Mm &\rightarrow Mm \\
MM \times Mm &\rightarrow Mm \\
mm \times Mm &\rightarrow mm \\
Mm \times Mm &\rightarrow MM \\
Mm \times Mm &\rightarrow Mm \\
Mm \times Mm &\rightarrow mm
\end{aligned}$$

Since the TDT cares whether M and m have the same probabilities to get passed to a child, it counts how many times M is passed to a child from a parent with Mm alleles, and how many times m is passed to a child from a parent with Mm alleles. The first number is denoted b and the second is denoted c, and the two numbers are called count statistics in this section. For example, in a family of type $MM \times Mm \rightarrow Mm$, the first parent passes an M to the child while omits another M, and the second parent passes an m to the child while omits an M. The first parent does not contribute to either b or

c, but the second one contributes 1 to c. Thus the contribution of this family to (b,c) is $(0,1)$.

If b and c are different in statistical sense, then the probabilities are different; otherwise the probabilities are the same. According to contribution to b and c, the 10 types of families can be classified into 6 groups:

1. $(0,0)$: $MM \times MM \to MM$, $MM \times mm \to Mm$, $mm \times mm \to mm$
2. $(1,0)$: $MM \times Mm \to MM$, $mm \times Mm \to Mm$
3. $(0,1)$: $MM \times Mm \to Mm$, $mm \times Mm \to mm$
4. $(2,0)$: $Mm \times Mm \to MM$
5. $(1,1)$: $Mm \times Mm \to Mm$
6. $(0,2)$: $Mm \times Mm \to mm$

and counts of families in these groups, $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$, can be used to represent a data set.

Given that the probabilities of passing M and m are the same, b should be from the binomial distribution $B(b+c, 0.5)$. When $b+c$ is not very small, then b is approximately from the normal distribution $N((b+c)/2, (b+c)/4)$. Therefore, $T = \frac{(b-(b+c)/2)^2}{(b+c)/4} = \frac{(b-c)^2}{b+c}$ should be from $\chi^2(1)$ distribution under the null hypothesis. We call $T = \frac{(b-c)^2}{b+c}$ test statistic in this section.

To pick the genes with the most transmission disequilibrium, we construct scores similar to that in the allelic test. A p-value to threshold significance is set first, say 0.05. Then for SNPs with significant (b,c) count statistics, the least number of sample changes to make them insignificant, d_+, is computed, and used as the SNP's score. For SNPs with insignificant (b,c) count statistics, the least number of sample changes to make

them significant, d_-, is computed, and $1-d_-$ is used as the SNP's score. Readers may refer to Section 3.1.2 for more explanations supporting the Hamming distance based scores.

We can still construct a graph for the TDT as we did for the allelic test, but its dimensionality is higher here so that I can only describe it but cannot plot it. Each node corresponds to a tuple $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ subject to $N = n_{00} + n_{10} + n_{01} + n_{20} + n_{11} + n_{02}$. The count statistics (b, c) can be inferred as $b = n_{10} + 2n_{20} + n_{11}, c = n_{01} + 2n_{02} + n_{11}$. Two nodes are connected only if one can change to the other by one sample change. For each tuple, there are at most $6 \times 5 = 30$ kinds of sample changes, which are listed below based on influence on (b, c) (excluding replacing a family with another one in the same group, as such replacement does not lead to any change on (b, c)).

1. $(b, c) \to (b-2, c+2)$:

   $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}, n_{01}, n_{20}-1, n_{11}, n_{02}+1)$

2. $(b, c) \to (b-1, c+2)$:

   $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}-1, n_{01}, n_{20}, n_{11}, n_{02}+1)$

3. $(b, c) \to (b, c+2)$:

   $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}-1, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}+1)$

4. $(b, c) \to (b-2, c+1)$:

   $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}, n_{01}+1, n_{20}-1, n_{11}, n_{02})$

5. $(b, c) \to (b-1, c+1)$:

   $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}-1, n_{01}+1, n_{20}, n_{11}, n_{02})$,

   $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}, n_{01}, n_{20}-1, n_{11}+1, n_{02})$,

   $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}, n_{01}, n_{20}, n_{11}-1, n_{02}+1)$

6. $(b,c) \to (b,c+1)$:

   $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00}-1,n_{10},n_{01}+1,n_{20},n_{11},n_{02})$,

   $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00},n_{10}-1,n_{01},n_{20},n_{11}+1,n_{02})$,

   $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00},n_{10},n_{01}-1,n_{20},n_{11},n_{02}+1)$

7. $(b,c) \to (b+1,c+1)$:

   $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00}-1,n_{10},n_{01},n_{20},n_{11}+1,n_{02})$

8. $(b,c) \to (b-2,c)$:

   $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00}+1,n_{10},n_{01},n_{20}-1,n_{11},n_{02})$

9. $(b,c) \to (b-1,c)$:

   $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00}+1,n_{10}-1,n_{01},n_{20},n_{11},n_{02})$,

   $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00},n_{10}+1,n_{01},n_{20}-1,n_{11},n_{02})$,

   $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00},n_{10},n_{01}+1,n_{20},n_{11}-1,n_{02})$

10. $(b,c) \to (b+1,c)$:

    $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00}-1,n_{10}+1,n_{01},n_{20},n_{11},n_{02})$,

    $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00},n_{10}-1,n_{01},n_{20}+1,n_{11},n_{02})$,

    $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00},n_{10},n_{01}-1,n_{20},n_{11}+1,n_{02})$

11. $(b,c) \to (b+2,c)$:

    $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00}-1,n_{10},n_{01},n_{20}+1,n_{11},n_{02})$

12. $(b,c) \to (b-1,c-1)$:

    $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00}+1,n_{10},n_{01},n_{20},n_{11}-1,n_{02})$

13. $(b,c) \to (b,c-1)$:

    $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02}) \to (n_{00}+1,n_{10},n_{01}-1,n_{20},n_{11},n_{02})$,

$(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}+1, n_{01}, n_{20}, n_{11}-1, n_{02})$,

$(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}, n_{01}+1, n_{20}, n_{11}, n_{02}-1)$

14. $(b,c) \to (b+1, c-1)$:

$(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}+1, n_{01}-1, n_{20}, n_{11}, n_{02})$,

$(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}, n_{01}, n_{20}+1, n_{11}-1, n_{02})$,

$(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}, n_{01}, n_{20}, n_{11}+1, n_{02}-1)$

15. $(b,c) \to (b+2, c-1)$:

$(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}, n_{01}-1, n_{20}+1, n_{11}, n_{02})$

16. $(b,c) \to (b, c-2)$:

$(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}+1, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}-1)$

17. $(b,c) \to (b+1, c-2)$:

$(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}+1, n_{01}, n_{20}, n_{11}, n_{02}-1)$

18. $(b,c) \to (b+2, c-2)$:

$(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02}) \to (n_{00}, n_{10}, n_{01}, n_{20}+1, n_{11}, n_{02}-1)$

Still, the computation of Hamming distance based on shortest path in a graph is difficult in the TDT, as both time and space complexity increases fast as with N. Therefore, I propose Algorithm 3, which computes the Hamming distance based scores within $O(1)$ time for each SNP.

Due to symmetry between the two alleles, the score of $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ is the same as $(n_{00}, n_{01}, n_{10}, n_{02}, n_{11}, n_{20})$. W.l.o.g, we assume the test statistics (b,c) satisfy $b \geq c$. If it happens $b < c$ for $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$, we compute the score of $(n_{00}, n_{01}, n_{10}, n_{02}, n_{11}, n_{20})$ instead, which satisfies $b \geq c$.

Lemma 3.1.4: for $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ with $T \leq t$, Steps 3$\sim$7 give the Hamming distance to all datasets with $T > t, b > c$.

Proof. Informally, Steps 3∼7 remove families with count statistics $b = 0, c = 2$ first, then $b = 0, c = 1$, then $b = 1, c = 1$, then $b = 0, c = 0$, then $b = 1, c = 0$. Each time a family is removed, a family with $b = 2, c = 0$ is added. It is easy to verify that the data set got by this method always keeps the smallest c and largest $b - c$ among all datasets that can be achieved from $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ within the same steps.

Suppose Steps 3∼7 output a score s_1, and we denote the test statistics corresponding to $s_1 - 1$ sample changes by Steps 3∼7 as (b^*, c^*). For any data set $(n'_{00}, n'_{10}, n'_{01}, n'_{20}, n'_{11}, n'_{02})$ that can be achieved from $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ within $s' \leq s_1 - 1$ steps, and satisfying $b' \geq c'$, there are

$$\begin{aligned} 0 \leq b' - c' &\leq b^* - c^* \\ c' &\geq c^* \end{aligned}$$

According to the definition of s_1, there is also $T(b^*, c^*) \leq t$. Therefore, if we can prove that $T(b', c') \leq T(b^*, c^*)$ when $b' \geq c'$, the area $T > t, b > c$ cannot be reached within

$s_1 - 1$ steps.

$$
\begin{aligned}
b^* - c^* \geq b' - c' & \\
\Rightarrow (b^* - c^*)^2 c^* &\geq (b' - c')^2 c^* \\
\text{and } \Rightarrow b^* - c^* - b' + c' &\geq 0 \\
\Rightarrow (b^* - c^*)(b' - c')(b^* - c^* - b' + c') &\geq 0 \\
\Rightarrow (b^* - c^*)^2 (b' - c') - (b^* - c^*)(b' - c')^2 &\geq 0 \\
c^* &\leq c' \\
\Rightarrow (b^* - c^*)^2 c' &\geq (b^* - c^*)^2 c^* \\
\Rightarrow (b^* - c^*)^2 c' &\geq (b' - c')^2 c^* \\
\Rightarrow (b^* - c^*)^2 (b' + c') - (b^* + c^*)(b' - c')^2 &\geq 0 \\
\Rightarrow \frac{(b^* - c^*)^2 (b' + c') - (b^* + c^*)(b' - c')^2}{(b^* + c^*)(b' + c')} &\geq 0 \\
\Rightarrow \frac{(b^* - c^*)^2}{b^* + c^*} - \frac{(b' - c')^2}{b' + c'} &\geq 0 \\
\Rightarrow T(b^*, c^*) - T(b', c') &\geq 0
\end{aligned}
$$

Furthermore, as Step 3∼7 construct s_1 sample changes that can change $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ to another dataset with $T > t, b > c$, therefore the Hamming distance from $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ to the set of datasets satisfying $T > t, b > c$ is s_1.

After all replacements in Step 3∼7, all families have count statistics $b = 2, c = 0$, thus the maximum of the test statistics is achieved. Therefore, if the threshold of p-value can be achieved, Step 3∼7 always give the correct answer. □

Lemma 3.1.5: for $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ with $T \leq t$, Steps 8∼12 give the Hamming distance to all datasets with $T > t, b \leq c$.

Proof. Due to the symmetry between two alleles, the proof is similar to that of Lemma

3.1.4. □

Combining Lemma 3.1.4 and 3.1.5, and the definition of Hamming distance based score, we come to the following corollary.

Corollary 3.1.6: for $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ with $T < t$, Steps 3∼13 gives the Hamming distance based score.

Then let's look at the cases where $T > t$ for $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$.

Lemma 3.1.7: for $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ with $T > t, b > c$, Steps 14∼15 gives the Hamming distance to all datasets with $T \leq t$.

Proof. Informally, Steps 14∼15 remove families with count statistics $b = 2, c = 0$ first, then $b = 1, c = 0$. Each time a family is removed, a family with $b = 0, c = 2$ is added. It is easy to verify that the data set got by this method always keeps the smallest b and largest c among all datasets that can be achieved from $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ within the same steps.

Suppose Steps 14∼15 output a score s, and we denote the test statistics corresponding to $s-1$ sample changes by Steps 14∼15 as (b^*, c^*). For any dataset $(n'_{00}, n'_{10}, n'_{01}, n'_{20}, n'_{11}, n'_{02})$ that can be achieved from $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$ within $s' \leq s-1$ steps, and satisfying (b', c'), there are

$$\begin{aligned} b' &\geq b^* \\ c' &\leq c^* \\ b^* &> c^* \end{aligned}$$

The last inequility is due to the fact that b^*, c^* is still on the right side of the curve $T = t, b > c$. According to the definition of s, there is also $T(b^*, c^*) > t$. Therefore, if we can prove that $T(b', c') \geq T(b^*, c^*)$, then the area $T \leq t$ cannot be reached within

$s-1$ steps.

$$\begin{aligned} b'-c' \geq b^*-c^* &> 0 \\ \frac{dT}{db} &= \frac{d}{db}\frac{(b-c)^2}{b+c} \\ &= \frac{2(b-c)(b+c)-(b-c)^2}{(b+c)^2} \\ &= \frac{(b-c)(b+3c)}{(b+c)^2} \\ \frac{dT}{dc} &= \frac{d}{dc}\frac{(b-c)^2}{b+c} \\ &= \frac{-2(b-c)(b+c)-(b-c)^2}{(b+c)^2} \\ &= \frac{-(b-c)(3b+c)}{(b+c)^2} \end{aligned}$$

Since $b'-c'>0, b' \geq b^*, c' \leq c^*$, and $\frac{dT}{db}>0, \frac{dT}{dc}<0$ when $b-c>0$, we conclude $T(b',c')>T(b^*,c^*)$. Thus the area $T \leq t$ cannot be reached with less than s steps. In other word, the Hamming distance cannot be smaller than s.

Furthermore, if the path constructed in Step 14$\sim$15, which consists of s sample changes, ends in the area $T \leq t$, then we have found a path with length s. Thus the Hamming distance is s. If the path ends in the area $T>t, b<c$, we can make change to the last inserted sample to make the path ending in the area $b=c$ such that $T \leq t$ must hold. we replace the last inserted sample as follows:

1. if finally $b+3=c$, then use $b=1, c=0$ as the last inserted family.
2. if finally $b+2=c$, then use $b=0, c=0$ as the last inserted family.
3. if finally $b+1=c$, then use $b=0, c=1$ as the last inserted family.

Noting that after $s-1$ steps, there is still $b>c$ (which means $b \geq c+1$), and one sample

change result in at most a change of 4 in $b-c$, the ending point of path output by Algorithm 5 must have $b-c \in \{-3,-2,-1\}$ if $b<c$. Thus only the three cases above need to be considered. No matter which case the ending point is in, simply replacing the last inserted sample can lead to $b=c, T=0 \leq t$ after s sample changes. Therefore, we have constructed a path from $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02})$ to the area $T \leq t$ with length s. To summarize, the Hamming distance is still s.

Last of all, we need to prove that the s output by Step 16 must be no larger than n_{10}, which means the changes are feasible. After all replacements in Step 15$\sim$16, there are no families with $b=2,c=0$ and $b=1,c=0$ left in the dataset. In this case, all families left have $c \geq b$, thus $c \geq b$ for the whole data set, which means the path has already passed the threshold $T=t, b>c$. Therefore, regardless of the threshold of p-value, Step 14$\sim$15 always give the correct answer. □

Lemma 3.1.8: for $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02})$ with $T<t, b<c$, Steps 16$\sim$17 give the Hamming distance to all datasets with $T>t$.

Proof. Due to the symmetry between two alleles, the proof is similar to that of Lemma 3.1.7. □

Combining Lemma 3.1.7 and 3.1.8, and the definition of Hamming distance based score, we come to the following corollary.

Corollary 3.1.9: for $(n_{00},n_{10},n_{01},n_{20},n_{11},n_{02})$ with $T<t$, Steps 14$\sim$17 give the Hamming distance based score.

Then combining Corollary 3.1.6 and 3.1.9, we have the following Theorem:

Theorem 3.1.10: Algorithm 3 computes accurate Hamming distance based score.

3.2 Differentially Private Empirical Risk Minimization with Feature Selection

Empirical risk minimization with feature selection is a widely studied problem in data mining, for example, LASSO, elasticNet, logistic LASSO, etc. There have been some differentially private algorithms that combine feature selection and empirical risk minimization, such as [58, 36, 57]. However, our experiment results show their performance can be significantly improved. To achieve better performance, we apply the exponential mechanism to empirical risk minimization with feature selection, and our algorithm outperforms those state-of-the-art competitors by large margins in most cases. The intuition is, as we mentioned in Section 2.2.3, the exponential mechanism is optimal in some sense.

Although the idea of applying the exponential mechanism to empirical risk minimization with feature selection has been mentioned in [57], that paper does not come up with a computationally feasible algorithm. The Markov Chain Monte Carlo (MCMC) method has been proven useful in implementing the exponential mechanism, as [9, 54, 62] have done for other tasks, such as PCA, frequent pattern mining, network data release, etc. Though those algorithms have good performance, they either assume convergence of MCMC granted, or provide incomplete proofs. One type of incomplete proofs is in [54], which uses a statistical test to ensure convergence of MCMC. There are two problems with this method: first, private information is used in the test without protection, thus the whole method is not fully differentially private; second, the statistical test only guarantees that a chain which hasn't converged cannot pass with a certain probability, which means the privacy leakage is still possible. When privacy leakage from the second problem above can be controlled, that from the first cannot. Another type of incomplete proofs is in [62], which mentions that the privacy risk for early stop decreases

exponentially, but falls short of computing the constants in the exponential. Thus it is still difficult to tell how much privacy is preserved in practice.

In this paper, we use MCMC to implement the exponential mechanism on empirical risk minimization, and provide a detailed proof on convergence of MCMC, and a thorough analysis on convergence speed. To our knowledge, this is the first complete convergence analysis of using MCMC to implement the exponential mechanism.

To summarize, my main contributions in this paper include:

1. I propose a MCMC based differentially private empirical risk minimization algorithm with feature selection, which outperforms other competitors.

2. I am the first to give solid privacy guarantee for MCMC-based differentially private algorithms.

In the following sections, I first introduce some background knowledge on empirical risk minimization and MCMC, then introduce our algorithm and analyze what privacy guarantee it can achieve. Experiments comparing its performance with other differentially private algorithms follow.

3.2.1 Background

In this section, I introduce some basic concepts in empirical risk minimization, feature selection, and stochastic process.

Empirical Risk Minimization with Feature Selection

Both empirical risk minimization and feature selections are machine learning tasks. Empirical risk minimization, as the name suggests, selects a model by minimizing risk (or loss) over training samples. Say private data set D consists of n samples $\{(x_i, y_i)\}_{i=1}^{n}$, and risk function is $l(w, x_i, y_i)$ while w is the parameter to learn, empirical

risk minimization outputs the w minimizing overall risk $\sum_{i=1}^{n} l(w, x_i, y_i)$. In this paper, we only consider linear models, which means $l(w, x_i, y_i) = l(w^T x_i, y_i)$ and includes popular models such as linear regression and logistic regression.

Feature selection means to pick out a subset of influential features from a large set. For linear models, unselected features always correspond to zero components in w.

Sometimes people want to complete empirical risk minimization and feature selection in one algorithm, which outputs a model depending on a few features with low risk. Mathematically speaking, the algorithm returns

$$w = \arg\min_{w} \sum_{i=1}^{n} l(w^T x_i, y_i)$$

s.t.

$$\|w\| \leq k_0$$

for a constant k_0 and norm function $\|.\|$.

There are some choices for the norm $\|\|$ of the model parameters w here. Ideally, people want to use as few parameters as possible, thus L_0 norm, which means the number of non-zero elements in w, is preferred. However, in practice, optimization based on L_0 norm is intractable, as it requires computational resources exponential w.r.t. the number of features to find the exact solution. Thus people replace L_0 norm with L_1 norm. L_1 norm ensures the convexity of the optimization problem, so that the problem can be solved efficiently.

Noticing that using L_1 norm is for computing exact solutions, in cases where exact solutions are not demanded, approximate solutions based on L_0 norm is acceptable. This is exactly the case in this section. Since differential privacy requires randomness added to the output, the exact solution is not returned anyway. Thus it might be better to use L_0 norm instead, for two advantages over L_1 norm: L_0 norm is closer to the nature

of the problem, and it makes the design and analysis of our algorithm easier.

Concepts in Stochastic Process

In this part, I mainly introduce MCMC algorithm and Harris Chain.

MCMC algorithm draws samples from a distribution $p_1(w)$. Starting from any state w^* in the sample space, it iteratively selects another sample w' with probability $p_2(w^*, w')$, and replace w^* with w' with probability $p_3(w^*, w')$. If $p_1(w^*)p_2(w^*, w')p_3(w^*, w') = p_1(w')p_2(w', w^*)p_3(w', w^*)$ for any two states w^* and w', and the stochastic process is both aperiodic and irreducible, then the distribution of w^* will converge to $p_1(w)$ finally. For different types of Markov chain, the definitions of aperiodicity and irreducibility vary. In this paper, we focus the Harris chain (Definition 3.2.1), and the corresponding definitions of aperiodicity and irreducibility are in Definitions 3.2.2 to 3.2.4.

MCMC is often used when it is difficult to draw samples from $p_1(w)$ directly. The probabilistic distributions suggested by differential privacy often fall in this category, therefore, MCMC can be used to draw a noisy output, as in [9, 54, 62].

Harris chain (See Chapter 6.8 in [13]) is a special case of Markov chain.

Definition 3.2.1: A Markov chain is a *Harris chain* if there exist subsets A, B of the state space W, $\eta > 0$, and a probability measure ρ on B such that

1. Starting from any point in W, the process can reach A with non-zero probability within finite steps.

2. If $w \in A$ and $B' \subset B$, then $p(w, B') \geq \eta\rho(B')$.

If a Harris chain is recurrent, it has a stationary measure; if it is also aperiodic, then this stationary measure is unique. Informally, recurrence here means the chain will always return to the set A, and aperiodic means there is no periodicity in the stochastic

process. I leave the details of these two concepts to Definition 3.2.2, 3.2.3 and 3.2.4 in Section 3.2.3. If a Harris chain has a unique stationary measure, the distribution of state output by MCMC algorithm will converge to the measure.

3.2.2 Methodology

Here are notations I use in this paper, the private data set D consists of n samples $\{(x_i, y_i)\}_{i=1}^{n}$, while x_i is used to predict y_i, and each x_i has k features. The learning parameters are in a vector w of length k. The number of selected parameters is denoted k_0. The set of all selected parameters is F_u, while F_u's complementary set is called F_c.

We have a bound M on the loss function, e.g., $|l(w^T x_i, y_i)| \leq M$ for a constant M. This assumption is used to ensure bounded sensitivity of loss. If the private data does not satisfy our assumption, we can either rescale data, rescale the loss function l, or restrict space of parameters. For example, in linear regression or logistic regression, we can change the scale of x and y and only search for w whose components are bounded to, say, [-1, 1]. For some complex loss function, we can even replace l with a truncated version, like $\tilde{l} = \max\{-1, \min\{1, l\}\}$, which truncates l to $[-1, 1]$.

As long as the rescaling or truncation are based on our prior knowledge instead of the private data, this step doesn't involve any privacy issues. The prior knowledge is usually accessible. For example, when blood pressure is one of the features, without looking at any private data, we can guess its range. Then we can rescale and truncate according to this range.

Our algorithm (Algorithm 4) is MCMC style. The algorithm starts from k_0 randomly selected features in F_u. Then in each iteration, it first selects an index u from F_u and another index c from F_c. Then it gets new weights by replacing w_u with 0, and randomly sample a weight w_c for feature indexed c. Based on whether changes of weights can reduce loss, it takes or refuses the new weights. Based on this algorithm, there are

always k_0 elements in F_u after each iteration, which means the L_0 norm of w is always k_0.

3.2.3 Analysis

In this section, I first prove that the distribution of our algorithm output will converge to the distribution of the exponential mechanism. Therefore, if we can iterate until the Markov chain converges, our algorithm can be seen as ε-differentially private.

Then I prove that whenever we stop the process, the output is (ε, δ)-differentially private. The second proof is necessary because we cannot iterate forever to ensure convergence of the algorithm. It is better to have privacy guarantee in this situation when the algorithm stops early.

In all the proofs below, I assume that $k_0 + 2 \leq k$, which at least two features are left out.

Table 3.3 includes all notations in this section, to make it easy for readers to track them.

Proof of Convergence to the Exponential Mechanism

In Algorithm 4, the weights at any time can be seen as a state, and all the states form a stochastic process $\{\{w_1, ..., w_k\} : \|\{w_1, ..., w_k\}\|_0 = k_0\}$. As the transition rule depends only on the current state, this process is a Markov Chain. This subsection proves the distribution of states converges to the distribution in the exponential mechanism.

The guild line of this section is, I first prove some properties of the *original Markov chain* (Lemma 3.2.2$\sim$3.2.4), then extract the $2k_0, 4k_0, ..., 2k_0 l$-th states from the original Markov chain to form an *extracted Markov chain*, and prove the extracted Markov chain is a Harris chain (Lemma 3.2.5) with recurrence (Lemma 3.2.8) and aperiodicity (Lemma 3.2.10). Such a Harris chain converges to the distribution introduced

Table 3.3. Explanations of notations. Notations outside this table are local.

Notation	Description
ε, δ	Privacy budgets.
D	Private data set
D'	A neighbor data set of D
k	Length of learned weights
k_0	Number of non-zero elements in weights. We assume $k_0 + 2 \leq k$.
n	Number of samples in D
$w \in W$	Weights to learn. W is the space of all weights such that $\|w\|_0 = k_0$.
$l(w,x,y)$	Loss function on a single sample (x,y). Between 0 and M.
M	Upper bound of $l(w,x,y)$
$L(w,D)$	Loss function on the data set D. $L(w,D) = \sum_{(x_i,y_i)\in D} l(w,x_i,y_i)$
s	Sensitivity of $L(w,D)$. $s = \max_{w,D\sim D'} \lvert L(w,D) - L(w,D')\rvert$
$R(w)$	A function related to w. $-R(w)$ could be interpreted as a regularization function. $R(w) = \sum_{w_i \in w, w_i \neq 0} \log q(w_i)$
$\pi(w)$	Distribution in the exponential mechanism.
A, B, η, ρ	Notations used in the definition of a Harris chain. See Definition 3.2.1. Their values are specified in Lemma 3.2.5 below.
$q(w)$	A probability density from which to sample non-zero element of w.
α	A special state in the constructed Markov chain in Definition 3.2.6.
τ	The first time that the constructed Markov chain goes back to state α after starting from α. Defined in Definition 3.2.7.
π_T	The distribution of the T-th state.
θ_T	$\theta_T = 1 - \min_w \pi_T(w)/\pi(w)$ measures the difference between π_T and π. It is a number between 0 and 1, and the smaller θ_T is, the closer π_T and π are.
γ	Linear mapping based on the transition function. We have $\gamma(\pi_{T-1}) = \pi_T$ and $\gamma(\pi) = \pi$.
C	$C = \frac{\binom{k}{k_0}}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon(k_0+1)/2)$ is a constant determining convergence speed.
π^T	$\pi^T(w) = (\pi_{2k_0(T-1)}(w) - (1 - \theta_{2k_0(T-1)}\pi(w)))/\theta_{2k_0(T-1)}$ is a distribution describing the difference between π_T and π.

by the exponential mechanism (Lemma 3.2.11). At last I prove that the original Markov chain also converges to this distribution (Theorem 3.2.12).

There are three Markov chains in the part in total. The first is the original Markov chain, the second is the extracted Markov chain, which composes of selected states from the original Markov chain, and the last is a *constructed Markov chain* as described in Definition 3.2.6. I mention all of them here to prevent confusion.

Lemma 3.2.2: given a private data set $D = \{(x_i, y_i)\}_{i=1}^{n}$, the loss function L must be in $[0, nM]$, and the sensitivity of the loss function is M.

Proof.

$$\begin{aligned} 0 &\leq l(w, x_i, y_i) \leq M \\ 0 &\leq L(w, D) = \sum_{i=1}^{n} l(w, x_i, y_i) \leq nM \end{aligned}$$

Besides, changing one sample leads to change in at most one item among the n items in the sum. Since we replace a loss between $[0, M]$ with another loss in the same range, the change of overall loss is bounded by M. Thus the sensitivity of $L(w, D)$ is at most M. □

Lemma 3.2.3: if $\{w_1, ..., w_k\}$ and $\{w'_1, ..., w'_k\}$ differ on only i-th and j-th elements, and $w_j = 0$, $w'_i = 0$, then there is probability density

$$\frac{q(w'_j)}{k_0(k - k_0)} \exp(-n\varepsilon/2)$$

that $\{w_1, ..., w_k\}$ changes to $\{w'_1, ..., w'_k\}$ in one step in the original Markov chain.

Proof. As $w_i \neq w'_i = 0$ and $w_j = 0$, when the process has state $\{w_1, ..., w_k\}$, $i \in F_u$ and $j \in F_c$. Therefore, there is probability $1/k_0(k - k_0)$ that $u = i$ and $c = j$ are selected

in Step 3 in Algorithm 4. According to Step 4, there is probability $q(w'_j)$ that w'_j is sampled. Due to Lemma 3.2.2 and non-zero property of loss function, $L_{c'} - L_{u'} \leq nM$. Thus the probability that the change (in Step 6 of Algorithm 4) is taken is at least $\min \exp\left(\frac{(L_{c'}-L_{u'})\varepsilon}{2s}\right) = \exp\left(\frac{-nM\varepsilon}{2M}\right) = \exp(-n\varepsilon/2)$. If all the three incidents above happen, then $\{w_1,...,w_k\}$ changes to $\{w'_1,...,w'_k\}$ in this iteration. This change has probability density at least $\frac{q(w'_j)}{k_0(k-k_0)}\exp(-n\varepsilon/2)$. □

Lemma 3.2.4: starting from any state $\{w_1,...,w_k\}$, the original Markov chain could reach any other state $\{w'_1,...,w'_k\}$ with probability density at least $\frac{\prod_{1\leq i\leq k, w'_i\neq 0} q(w'_i)}{k_0^{2k_0}(k-k_0)^{2k_0}}$ $\exp(-n\varepsilon k_0/2)$ after $2k_0$ steps, while q is in input of Algorithm 4.

Proof. We can construct a path as follows.

1. Iterate the following for all k_0 non-zero elements in $\{w'_1,...,w'_k\}$.

2. For $w'_i \neq 0$, check whether $w_i = 0$. If yes, go to Step 3; otherwise go to Step 5.

3. Find $j_1, j_2 \neq i$ such that $w'_{j_1} = 0, w_{j_1} \neq 0$ and $w_{j_2} = 0$. Since $\{w_1,...,w_k\}$ and $\{w'_1,...,w'_k\}$ both have exactly k_0 non-zero elements, and $w'_i \neq 0, w_i = 0$, such j_1 must exist. Besides, as we assume $k_0 + 2 \leq k$ at the beginning of this section, there must be some $j_2 \neq j_1$ such that $w'_{j_2} = 0$.

4. Add the following two changes to the constructed path: $w_{j_1} \to 0, w_{j_2} \to$ some non-zero value, and $w_i \to w'_i, w_{j_2} \to 0$. Return to Step 1 for the next iteration.

5. Find $j \neq i$ such that $w_j \neq 0$. Since we assume $k_0 + 2 \leq k$ at the beginning of this section, we can always find j.

6. Add the following two changes to the constructed path: $w_i \to 0, w_j \to$ some non-zero value, and $w_i \to w'_i, w_j \to 0$. Return to Step 1 for the next iteration.

First let's prove the path does change $\{w_1,...,w_k\}$ to $\{w'_1,...,w'_k\}$ in $2k_0$ steps. This method consists of k_0 iterations, and in each iteration, two changes are added the path to inserted a non-zero element w'_i into w_i. If the corresponding $w_i = 0$, then Steps 3 and 4 insert w'_i; otherwise Steps 5 and 6 insert w'_i. If later iterations do not affect weights that have been set in previous iterations, then the constructed path changes $\{w_1,...,w_k\}$ to $\{w'_1,...,w'_k\}$.

It is not hard to check. If Steps 3 and 4 are taken, the two changes only affect w_{j_1} and w_{j_2} besides w_i. Since $w'_{j_1} = 0$, w_{j_1} cannot be a non-zero element from $\{w'_1,...,w'_k\}$ in previous iterations. Since $w_{j_2} = 0$, it is not a non-zero element from $\{w'_1,...,w'_k\}$ either. Therefore, Steps 3 and 4 do not change non-zero elements that have been added from $\{w'_1,...,w'_k\}$ previously.

If Steps 5 and 6 are taken, the two changes only affect w_j besides w_i. Since $w_j = 0$ before the changes, it is not a non-zero element from $\{w'_1,...,w'_k\}$ either. Therefore, the changes that inserted w'_i into $\{w_1,...,w_k\}$ do not affect previously inserted elements.

The second part of this proof is on the lower bound of the probability. In Step 4 and 6, the first change's probability mass is lower bounded by the product of the following three probabilities: with probability $1/k_0$, the non-zero element $w_{j_1}(w_i)$ is selected; with probability $1/(k-k_0)$, the zero element $w_{j_2}(w_j)$ is selected; no matter what is the new value selected for $w_{j_2}(w_j)$, with probability at least $\exp(-n\varepsilon/2)$, this change will be taken. The second change's probability density is lower bounded by and the probability density for the second change is at least $\frac{q(w'_i)}{k_0(k-k_0)}\exp(-n\varepsilon/2)$ due to Lemma 3.2.3. Therefore, the probability density for this iteration is at least $\frac{q_{w'_i}}{k_0^2(k-k_0)^2}\exp(-n\varepsilon)$ no matter whether $w'_i = 0$. The whole probability density for all k_0 iterations is thus at least $\frac{\prod_{1\le i\le k, w'_i\neq 0} q(w'_i)}{k_0^{2k_0}(k-k_0)^{2k_0}}\exp(-n\varepsilon k_0)$. □

Then I prove the extracted Markov chain, which consists of the $2k_0, 4k_0, ..., 2k_0 l$-

th state of the original Markov chain, is a Harris chain. For the extracted Markov chain, Lemma 3.2.4 means the state can transform from any state $\{w_1,...,w_k\}$ to any other state $\{w'_1,...,w'_k\}$ within one step, and the transition probability density is at least $\frac{\prod_{1\le i\le k, w'_i\neq 0} q(w'_i)}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon k_0)$.

Lemma 3.2.5: the extracted Markov chain is a Harris chain.

Proof. Let's take $A = \{w \in W$: only the first k_0 components are non-zero$\}$, $B = \{w \in W$: only the first $k_0 - 1$ and the (k_0+1)-th components are non-zero$\}$, $\rho(w_1,...,w_k) = \prod_{i\in\{1,...,k-1,k+1\}} q(w_i)$, and $\eta = \frac{1}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon k_0)$. We want to prove the following two conditions in Definition 3.2.1:

1. Starting from any point in W, the process can reach A with non-zero probability within finite steps.

2. If $w \in A$ and $B' \subset B$, then $p(w,B') \ge \eta\rho(B')$.

Due to Lemma 3.2.4, starting from any state in W, the probability density to arrive a point $\{w_1,...,w_{k_0},...,0\} \in A$ in the next step is at least $\frac{\prod_{1\le i\le k_0} q(w_i)}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon k_0)$. Therefore, the probability to arrive at A is at least

$$\begin{aligned} & \int_{w_1,...,w_{k_0}} \frac{\prod_{1\le i\le k_0} q(w_i)}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon k_0)\, dw_1...dw_{k_0} \\ = & \frac{1}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon k_0) \prod_{i=1}^{k_0} \int_{w_i} q(w_i) dw_i \\ = & \frac{1}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon k_0) \\ > & 0 \end{aligned}$$

Therefore, the first condition is satisfied.

Then for the second condition, the transition probability from any point in A to another point $w' = \{w_1,...,w_{k_0-1},0,w_{k_0+1},...,0\} \in B$ is at least $\frac{\prod_{i=1\sim k_0-1,k_0+1} q(w'_i)}{k_0^{2k_0}(k-k_0)^{2k_0}}$

$\exp(-n\varepsilon k_0) = \eta\rho(\{w_1, ..., w_{k_0-1}, 0, w_{k_0+1}, ..., 0\})$. We can take integral over B' for both sides to prove it.

Thus this extracted Markov chain is a Harris chain. □

To prove this extracted Markov chain has a unique stationary distribution, I need to prove recurrence and aperiodicity of the following constructed Markov chain.

Definition 3.2.6: For a Harris chain, we can construct the following Markov chain by introducing a new element α to the state space, and modifying the transition probability accordingly: state space $\bar{W} = W \cup \alpha$ and transition probability $\bar{p}$ is

$$\begin{aligned}
&\text{If} \quad w \in W - A, \quad \text{then } \bar{p}(w, w') = p(w, w') \\
&\text{If} \quad w \in A, \quad \text{then } \bar{p}(w, \alpha) = \eta, \bar{p}(w, w') = p(w, w') - \eta\rho(w') \\
&\text{If} \quad w = \alpha, \quad \text{then } \bar{p}(w, w') = \int_B p(y, w')\rho(dy)
\end{aligned}$$

while A, B, η, ρ are as in the definition of this Harris chain (specified in proof of Lemma 3.2.5). This Markov chain is called a *constructed Markov chain*.

The definition of recurrence and aperiodicity of a Harris chain depends on its constructed Markov chain.

Definition 3.2.7: let the first state of the constructed Markov chain $\bar{X}_1$ be α, if $\tau = inf\{n > 1 : \bar{X}_n = \alpha\}$ is the first time it returns to the state α, and $P(\tau < +\infty) = 1$, then the chain is called *recurrent*.

Lemma 3.2.8: this extracted Markov chain is a recurrent Harris chain.

Proof. Lemma 3.2.4 means $P(\tau > l) - P(\tau > l+1) = P(\tau = l+1) \geq P(\tau > l))$ $\frac{\prod_{1 \leq i \leq k, w_i' \neq 0} q(w_i')}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon k_0)$. Therefore, we have $P(\tau > l+1)/P(\tau > l) = 1 - \frac{\prod_{1 \leq i \leq k, w_i' \neq 0} q(w_i')}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon k_0)$, and $P(\tau > l) = P(\tau > 1)(1 - \frac{\prod_{1 \leq i \leq k, w_i' \neq 0} q(w_i')}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon k_0))^{l-1}$.

Since $P(\tau > 1) = 1$,

$$\begin{aligned} P(\tau < +\infty) &= \lim_{l \to +\infty} P(\tau < l) \\ &= 1 - \lim_{l \to +\infty} (1 - \frac{\prod_{1 \le i \le k, w_i' \neq 0} q(w_i')}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp(-n\varepsilon k_0))^{l-1} \\ &= 1 - 0 = 1 \end{aligned}$$

According to Definition 3.2.7, the extracted Markov chain is also a recurrent Harris chain. □

Definition 3.2.9: a Harris chain is aperiodic if the greatest common divisor (g.c.d.) $\{n : p^n(\alpha, \alpha) > 0\} = 1$, while $p^n(\alpha, \alpha)$ is the probability that a process starting from α returns to α after n transmissions.

Lemma 3.2.10: the extracted Markov chain is an aperiodic Harris chain.

Proof. Lemma 3.2.4 means $p^1(\alpha, \alpha) > 0$. Therefore, the set $\{n : p^n(\alpha, \alpha) > 0\}$ has number 1 in it. In that case, the greatest common divider must be 1. So the extracted Markov chain is an aperiodic Harris chain. □

Lemma 3.2.11: this extracted Markov chain converges to

$$\pi(w) \propto \exp(-L(w, D)\varepsilon/2M) \prod_{w_i \in w, w_i \neq 0} q(w_i)$$

.

Proof. Combining Lemma 3.2.8 and Theorem 6.8.5 in [13], there exists a stationary measure for the extracted Markov chain. Furthermore, Theorem 6.8.8 in [13] tells us that the stationary measure is unique given Lemma 3.2.10 and the existence of the stationary measure. Thus I only need to prove that the unique measure is $\pi(w) \propto \exp(-L(w, D)\varepsilon/2M) \prod_{w_i \in w, w_i \neq 0} q(w_i)$.

The idea is to validate $\pi(w)$ is a stationary distribution of the original Markov chain. Since the extracted Markov chain consists of states from the original Markov chain, the stationary distribution of the original Markov chain must be a stationary distribution of the extracted Markov chain. Since the extracted Markov chain has a unique stationary distribution, then $\pi(w)$ must be the unique distribution.

Especially, I need to validate that $\pi(w')p(w,w') = \pi(w)p(w',w)$ for any two states w and w' in the original Markov chain, while p is the transition probability. As both probabilities on the two sides are 0 if w can only be changed to w' with more than one sample change, only neighboring w and w' need to be considered. W.l.o.g, we assume $L(w',D) > L(w,D)$, $w = \{w_1, 0, w_3, ..., w_{k_0+1}, 0, ..., 0\}$, and $w' = \{0, w_2, w_3, ..., w_{k_0+1}, 0, ..., 0\}$. In Algorithm 4,

$$\begin{aligned}
p(w,w') &= \frac{q(w_2)}{k_0(k-k_0)} \exp(-(L(w',D) - L(w,D))/2M) \\
p(w',w) &= \frac{q(w_1)}{k_0(k-k_0)} \\
\frac{\pi(w)}{\pi(w')} &= \frac{\exp(-L(w,D)\varepsilon/2M) q(w_1) \prod_{i=3}^{k_0+1} q(w_i)}{\exp(-L(w',D)\varepsilon/2M) q(w_2) \prod_{i=3}^{k_0+1} q(w_i)} \\
&= \exp((L(w',D) - L(w,D))/2M) q(w_1) \big/ q(w_2) \\
&= \frac{p(w',w)}{p(w,w')}
\end{aligned}$$

Therefore, the original Markov chain takes $\pi(w)$ as a stationary distribution, so does the extracted Markov chain. Since the extracted chain has only one unique stationary distribution, $\pi(w)$ is the distribution, and the extracted chain will converge to $\pi(w)$.

If we set $R(w) = \prod_{w_i \in w, w_i \neq 0} q(w_i)$, $\pi(w)$ is the distribution in the exponential mechanism. □

Theorem 3.2.12: the original Markov chain converges to the distribution $\pi(w)$, thus finally its state satisfies ε differential privacy.

Proof. First of all, we notice that if in the original Markov chain, the distribution of the n-th state is $\pi_n(w)$, then the distribution of the $n+1$-th state $\pi_{n+1}(w)$ is

$$\pi_{n+1}(w) = \int_{w' \in W, w' \neq w} \pi_n(w') p(w', w) dw' + \pi_n(w) \left(1 - \int_{w' \in W, w' \neq w} p(w, w') dw'\right)$$

, while the first item is the probability that w is transferred from other states, and the rest measures the probability that previous state w fails to transfer to other states. As $\pi(w)$ is a stationary distribution of the original Markov chain, there is

$$\pi(w) = \int_{w' \in W, w' \neq w} \pi(w') p(w', w) dw' + \pi(w) \left(1 - \int_{w' \in W, w' \neq w} p(w, w') dw'\right)$$

.

The extracted Markov chain converges to $\pi(w)$ means for any number $\gamma > 0$, there is a number N such that when $l > N$, the distribution of the l-th state of the extracted Markov chain is γ close to $\pi(w)$. If we denote the distribution $\pi_{2k_0 l}(w)$ here, as it is the distribution of the $2k_0 l$-th state in the original distribution, we have

$$\int_{w \in W} |\pi_{2k_0 l}(w) - \pi(w)| dw \leq \gamma$$

Then for the distribution of the next state in the original Markov chain, π_{2k_0l+1}, there is

$$
\begin{aligned}
& \int_w |\pi_{2k_0l+1}(w) - \pi(w)| dw \\
= & \int_w \left| \int_{w' \neq w} \pi_{2k_0l}(w') p(w', w) dw' + \pi_{2k_0l}(w) \left(1 - \int_{w' \neq w} p(w, w') dw' \right) \right. \\
& \left. - \int_{w' \neq w} \pi(w') p(w', w) dw' - \pi(w) \left(1 - \int_{w' \neq w} p(w, w') dw' \right) \right| dw \\
= & \int_w \left| \int_{w' \neq w} (\pi_{2k_0l}(w') - \pi(w')) p(w', w) dw' \right. \\
& \left. - (\pi_{2k_0l}(w) - \pi(w)) \left(1 - \int_{w' \neq w} p(w, w') dw' \right) \right| dw \\
\leq & \int_w \int_{w' \neq w} |\pi_{2k_0l}(w') - \pi(w')| p(w', w) dw' dw \\
& + \int_w |\pi_{2k_0l}(w) - \pi(w)| \left(1 - \int_{w' \neq w} p(w, w') dw' dw \right) \\
= & \int_w \int_{w' \neq w} |\pi_{2k_0l}(w) - \pi(w)| p(w, w') dw' dw \\
& + \int_w |\pi_{2k_0l}(w) - \pi(w)| \left(1 - \int_{w' \neq w} p(w, w') dw' dw \right) \\
= & \int_w |\pi_{2k_0l}(w) - \pi(w)| dw \\
\leq & \gamma
\end{aligned}
$$

Similarly, there is $\int_w |\pi_{2k_0l+l'}(w) - \pi(w)| dw \leq \gamma$, etc. For all integer $2k_0l + l'$ between $2k_0l$ and $2k_0(l+1)$, we can bound the distance between $\pi(w)$ and $\pi_{2k_0l+l'}$ to γ.

To summarize, for all $\gamma > 0$, there is an N such that when $n > 2k_0N$, $\int_w |\pi^n(w) - \pi(w)| dw \leq \gamma$. Thus the distribution of states in the original Markov chain converges to $\pi(w)$. □

Proof on Output after Finite Changes

In this part, I prove Algorithm 4 satisfies (ε, δ)-differential privacy if we stop after finite changes. The proofs here are only bounds for privacy budgets. These bounds

can be further improved given more complicated computation.

In the following part, I use π to denote the stationary distribution of the chain, and π_T to denote the distribution of states after T iterations. The number $\theta_T = 1 - \min_w \frac{\pi_T(w)}{\pi(w)}$ denotes the difference between π and π^T, θ_T is between 0 and 1, and the smaller it is, the more closer two distributions are. The transition function is denoted γ. It's easy to see γ is a linear function, and $\gamma(\pi) = \pi$, $\gamma(\pi_{T-1}) = \pi_T$.

Lemma 3.2.13:

$$\pi(w) \leq \frac{1}{\binom{k}{k_0}} \exp(n\varepsilon/2 - L(w,D)\varepsilon/2M) \prod_{w_i \in w, w_i \neq 0} q(w_i)$$

Proof.

$$\pi(w) \propto \exp(-L(w,D)\varepsilon/2M) \prod_{w_i \in w, w_i \neq 0} q(w_i)$$

means

$$\pi(w) = \frac{1}{Z} \exp(-L(w,D)\varepsilon/2M) \prod_{w_i \in w, w_i \neq 0} q(w_i)$$

for some constant Z. Here we can lower bound Z given $L(w,D) \leq nM$ in Lemma 3.2.2.

$$\begin{aligned} Z &= \sum_{F_u : |F_u| = k_0} \int_w \exp(-L(w,D)\varepsilon/2M) \prod_{w_i \in w, w_i \neq 0} q(w_i) dw \\ &\geq \sum_{F_u : |F_u| = k_0} \exp(-nM\varepsilon/2M) \int_w \prod_{w_i \in w, w_i \neq 0} q(w_i) dw \\ &= \sum_{F_u : |F_u| = k_0} \exp(-n\varepsilon/2) \\ &= \binom{k}{k_0} \exp(-n\varepsilon/2) \end{aligned}$$

thus,

$$\pi(w) \leq \frac{1}{\binom{k}{k_0}} \exp(n\varepsilon/2 - L(w,D)\varepsilon/2M) \prod_{w_i \in w, w_i \neq 0} q(w_i)$$

□

Lemma 3.2.14: $\theta_{2k_0 t} \leq (1-C)^t$ while

$$C = \frac{\binom{k}{k_0}}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp\left(-n\varepsilon(k_0+1)/2\right)$$

Proof. Lemma 3.2.4 means that for all distributions π^* and all states w, there is

$$\gamma^{2k_0}(\pi^*)(w) \quad \geq \quad \frac{\prod_{w_i \in w, w_i \neq 0} q(w_i)}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp\left(-n\varepsilon k_0/2\right)$$

. With Lemma 3.2.13, we further have

$$\begin{aligned}
\gamma^{2k_0}(\pi^*)(w) &\geq \frac{\prod_{w_i \in w, w_i \neq 0} q(w_i)}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp\left(-n\varepsilon k_0/2\right) \\
&\geq \frac{\prod_{w_i \in w, w_i \neq 0} q(w_i)}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp\left(-n\varepsilon k_0/2\right) \\
&\quad \frac{\pi(w)\binom{k}{k_0}}{\prod_{w_i \in w, w_i \neq 0} q(w_i)} \exp(L(w,D)\varepsilon/2M - n\varepsilon/2) \\
&= \frac{\pi(w)\binom{k}{k_0}}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp\left(-n\varepsilon(k_0+1)/2\right)
\end{aligned}$$

Therefore, for all distribution π^*, there is

$$\begin{aligned}
&\min_w(\gamma^{2k_0}(\pi^*)(w)/\pi(w)) \\
\geq \quad & \frac{\binom{k}{k_0}}{k_0^{2k_0}(k-k_0)^{2k_0}} \exp\left(-n\varepsilon(k_0+1)/2\right) \\
= \quad & C
\end{aligned}$$

.

For all $t \geq 1$, given the definition $\theta_{2k_0(t-1)} = \min_w(1-\pi_{2k_0(t-1)}(w)/\pi(w))$, we

define the following function $\pi^t(w) = (\pi_{2k_0(t-1)}(w) - (1-\theta_{2k_0(t-1)})\pi(w))/\theta_{2k_0(t-1)}$. As

$$\begin{aligned}
\pi^t(w) &= (\pi_{2k_0(t-1)}(w) - (1-\theta_{2k_0(t-1)})\pi(w))/\theta_{2k_0(t-1)} \\
&= (\pi_{2k_0(t-1)}(w) - \max_{w'}(\pi_{2k_0(t-1)}(w')/\pi(w'))\pi(w))/\theta_{2k_0(t-1)} \\
&\geq (\pi_{2k_0(t-1)}(w) - (\pi_{2k_0(t-1)}(w)/\pi(w))\pi(w))/\theta_{2k_0(t-1)} \\
&\geq (\pi_{2k_0(t-1)}(w) - \pi_{2k_0(t-1)}(w))/\theta_{2k_0(t-1)} \\
&= 0 \\
\int_w \pi^t(w)dw &= \int_w (\pi_{2k_0(t-1)}(w) - (1-\theta_{2k_0(t-1)})\pi(w))/\theta_{2k_0(t-1)}dw \\
&= (\int_w \pi_{2k_0(t-1)}(w)dw - (1-\theta_{2k_0(t-1)})\int_w \pi(w)dw)/\theta_{2k_0(t-1)} \\
&= (1-(1-\theta_{2k_0(t-1)}))/\theta_{2k_0(t-1)} \\
&= \theta_{2k_0(t-1)}/\theta_{2k_0(t-1)} \\
&= 1
\end{aligned}$$

So $\pi^t(w)$ is a probability function. According to the logic in the first part of this proof, there is

$$\min_w(\gamma^{2k_0}(\pi^t)(w)/\pi(w)) \geq C$$

combining them together, as well as the fact $\gamma(\pi) = \pi$, we conclude

$$\begin{aligned}
\pi_{2k_0(t-1)} &= (1-\theta_{2k_0(t-1)})\pi + \theta_{2k_0(t-1)}\pi^t \\
1-\theta_{2k_0t} &= \min \pi_{2k_0t}(w)/\pi(w) \\
&= \min_w \gamma^{2k_0}\pi_{2k_0(t-1)}(w)/\pi(w) \\
&= \min_w \gamma^{2k_0}((1-\theta_{2k_0(t-1)})\pi(w) + \theta_{2k_0(t-1)}\pi^t)(w)/\pi(w) \\
&= \min_w [(1-\theta_{2k_0(t-1)})\pi(w) + \theta_{2k_0(t-1)}\gamma^{2k_0}(\pi^t)(w)]/\pi(w) \\
&= (1-\theta_{2k_0(t-1)}) + \theta_{2k_0(t-1)} \min_w \gamma^{2k_0}(\pi^t)(w)/\pi(w) \\
&\geq (1-\theta_{2k_0(t-1)}) + \theta_{2k_0(t-1)}C \\
\theta_{2k_0t} &\leq (1-C)\theta_{2k_0(t-1)} \\
&\leq (1-C)^t\theta_0 \\
&\leq (1-C)^t
\end{aligned}$$

□

Lemma 3.2.15: $\theta_T \leq (1-C)^t$, while t is the largest integer satisfying $2k_0t \leq T$.

Proof. First of all, θ_T is decreasing as T goes up, as

$$\begin{aligned}
1-\theta_T &= \min_w \pi_T(w)/\pi(w) \\
&= \min_w (\gamma\pi_{T-1})(w)/\pi(w) \\
&\geq \min_w (\gamma((1-\theta_{T-1})\pi))(w)/\pi(w) \\
&= \min_w (1-\theta_{T-1})\pi(w)/\pi(w) \\
&= 1-\theta_{T-1} \\
\theta_T &\leq \theta_{T-1}
\end{aligned}$$

Therefore, $\theta_T \leq \theta_{2k_0 t}$. Then according to Lemma 3.2.14, we have $\theta_T \leq (1-C)^t$. □

Theorem 3.2.16: After T iterations, the output of Algorithm 4 satisfies (ε', δ') differential privacy while $\varepsilon' = \varepsilon - \ln(1-(1-C)^t)$, $\delta = (1-C)^t$, and t, C are defined as in Lemma 3.2.15 and 3.2.14.

Proof. Given any two neighboring data sets $D \sim D'$, and set S, Lemma 3.2.15 ensures that

$$\begin{aligned}
\pi_T(f(D) \notin S) &\geq \pi(f(D) \notin S)(1-(1-C)^t) \\
&\geq \pi(f(D) \notin S) - (1-C)^t \\
\pi_T(f(D') \in S) &\geq \pi(f(D') \in S)(1-(1-C)^t)
\end{aligned}$$

Therefore

$$\begin{aligned}
\pi_T(f(D) \in S) &\leq \pi(f(D) \in S) + (1-C)^t \\
\pi(f(D') \in S) &\leq \frac{1}{1-(1-C)^t}\pi_T(f(D') \in S)
\end{aligned}$$

Furthermore, Lemma 3.2.11 ensures that $\pi(f(D) \in S) \leq e^{\varepsilon}\pi(f(D') \in S)$. Thus

$$\begin{aligned}
\pi_T(f(D) \in S) &\leq \pi(f(D) \in S) + (1-C)^t \\
&\leq e^{\varepsilon}\pi(f(D') \in S) + (1-C)^t \\
&\leq \frac{e^{\varepsilon}}{1-(1-C)^t}\pi_T(f(D') \in S) + (1-C)^t \\
&= e^{\varepsilon'}\pi_T(f(D') \in S) + \delta'
\end{aligned}$$

Then Algorithm 4 satisfies (ε', δ')-differential privacy. □

3.2.4 Experiments

In this section, we empirically test our algorithm's performance on feature selection and prediction, and compare its performance to some state-of-the-art algorithms. The first part of this section introduces the data sets we use and the design of the experiments, while the second part presents results.

Experiment Design

I use both synthetic data sets and real data sets, as I can only identify truly effective predictors (features) for synthetic data. For real data sets, it is difficult to tell whether a predictor is useful or not with 100% confidence.

For the classification task, the synthetic data set has 1000 samples, each of which has 100 binary predictors and one response. At the beginning of each experiment, I sample values for all 1000 samples and 100 predictors from $Bernoulli(0.5)$ i.i.d., pick 10 predictors uniformly randomly from the 100 predictors, and set the possibility of a positive response as the sigmoid function of the sum of those selected 10 predictors. In different experiments, not only the data sets are different, but also the truly useful parameters are different.

Besides, from the medical database MIMIC III [33], I uniformly randomly select 2000 samples, 669 frequent binary predictors (the threshold for diagnoses and prescriptions predictors is 10% while that for abnormal lab results is 5%) and one response indicating whether a patient dies by the time the medical records are generated. In different experiments, I randomly reshuffle the samples before using cross validation.

Given a data set generated as described above, we first split the data set into a training set and a test set using 10-fold cross validation, then for a certain algorithm, we learn a classification model (logistic regression) from the training set, and use AUC on the test set to evaluate the model's performance. This measurement, AUC, is a number

between 0 and 1, and AUC=0.5 corresponds to a trivial random classifier. The larger AUC a model has, the better the model is, and the better the corresponding algorithm is. The whole process is repeated 100 times, and the average AUC is used as the performance of the algorithm.

For the regression task, we generate predictors in the synthetic data set in the same way we do for the classification task. However, as regression is usually more difficult compared to classification for differentially private algorithms, the number of samples increases from 1000 to 10000 here. Again, 10 truly useful predictors are randomly selected, and their average is used as the response, without any more random noise added.

The real data set for regression is the diabetes data set from [19], which has 442 samples with 64 predictors and one number indicating disease progression. The predictors, derived from age, sex, bmi, etc, are used to predict the disease progression. All predictors and the progression feature are normalized to range [0,1].

The use of cross validation and repetition of experiments are the same as that in the classification task. The differences are two-fold due to the nature of regression: first, we use linear regression to replace logistic regression; second, we use relative MSE and correlation to evaluate performance of models, as well as algorithms. Relative MSE here is the MSE of a model divided by the MSE of a trivial predictor, which predicts training average for all test samples. Relative MSE is a non-negative number and smaller value corresponds to better performance. Most algorithms can have relative MSE close to 1 by setting regularization strength on parameters to a very large number, thus it is meaningless to have a relative MSE larger than 1. Correlation between the predict and test response is a number between -1 and 1, and the trivial predictor mentioned above corresponds to correlation=0. The larger correlation a model has, the better the model is.

Though our algorithm does not converge to the distribution of the exponential mechanism within finite iterations, we can stop after a certain number of iterations. Figure 3.3 and 3.4 shows that the more iterations it uses, the better performance it has, and the performance tends to converge after 100,000 iterations. Therefore, we use the performance after 100,000 iterations in the following comparisons, which at least do not give additional advantage to our algorithm over competitors.

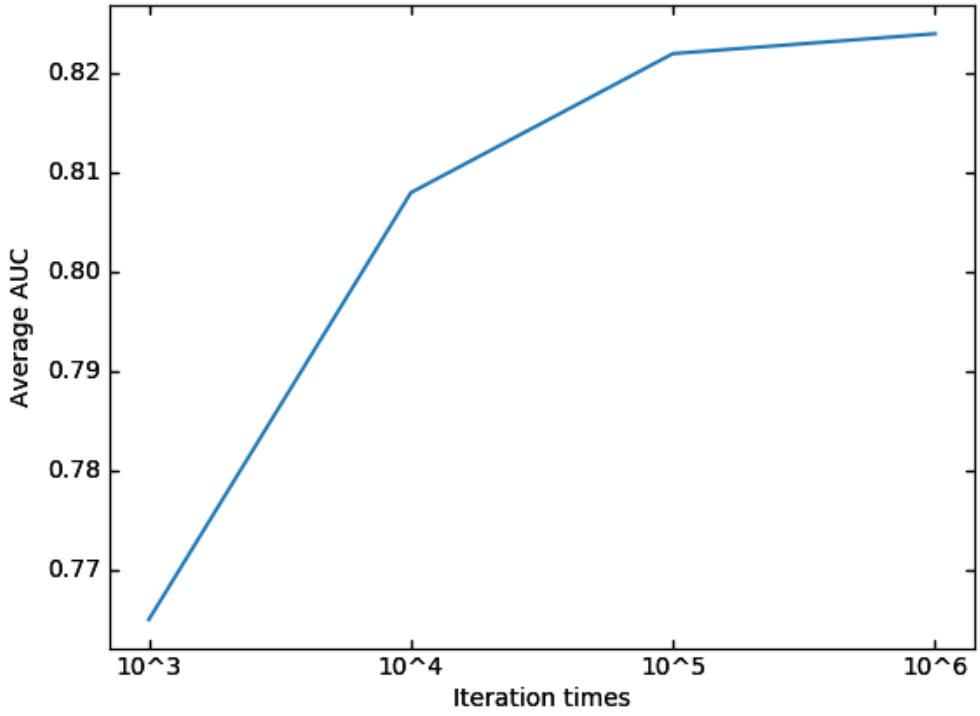


Figure 3.3. Classification performance given different iteration times on MIMIC III data set given $\varepsilon = 1, k_0 = 3$ (default values of them in Section 3.2.4).

The baselines for both classification and regression include sparse linear regression via Sample-and-Aggregate framework (SLRSA, Algorithm 3 in [36]), and the DPFW algorithm ([57]). The SLRSA algorithm is ε-differentially private, while the DPFW algorithm satisfies (ε, δ)-differential privacy. Thus DPFW in fact benefits a lit-

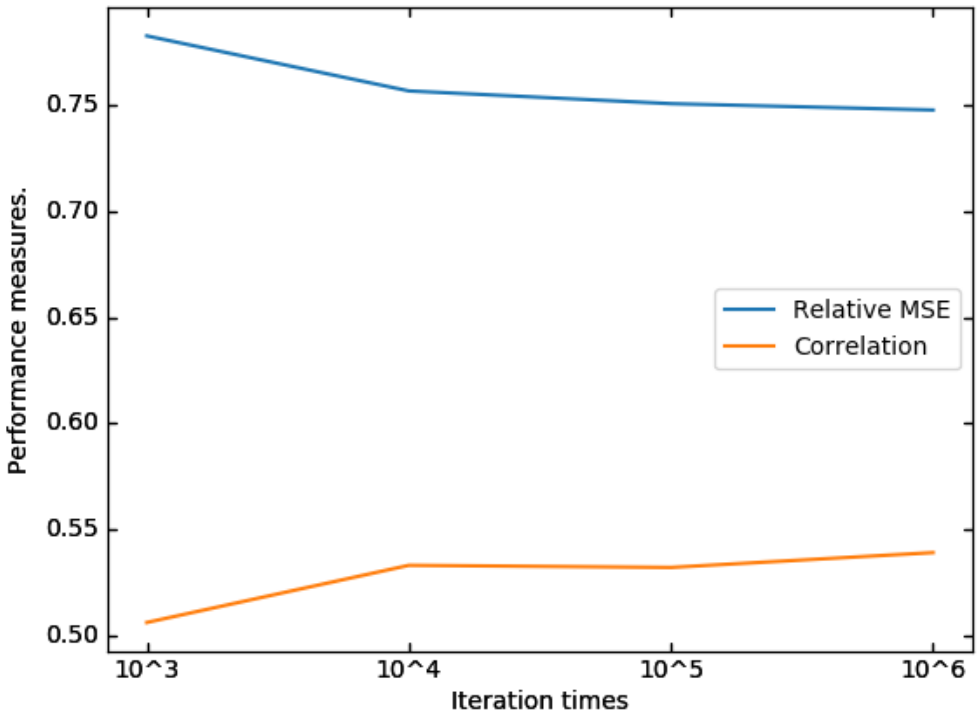


Figure 3.4. Regression performance given different iteration times on diabetes data set given $\varepsilon = 5, k_0 = 2$ (default values of them in Section 3.2.4).

tle from the definition of privacy. In this experiment section, we set $\delta = 1/n^2$, which is loose. We do not include algorithms in [58] in this section because in our experiments, its algorithms always return nothing.

In regression experiments, we replace square error by absolute error for the seek of performance for our algorithm. Similar use of absolute error can be seen in differentially private robust regression [16]. The use of absolute error is due to the fact that absolute error is most robust to outliers. In other words, an extreme sample change usually leads to a smaller change on output model, if we use absolute error instead of square error. Thus absolute error is preferred if we care about maximum effect of sample changes. To make the comparison fair, we use two versions of the DPFW algorithm, one with absolute error and the other with square error. As for the SLRSA algorithm, we cannot combine it with absolute error, thus it has only the version with square error.

Results

In the experiments section, except for privacy budget ε, all other parameters are set to the value that optimizes performance unless they are specified. For example, when we apply our algorithm to learn a logistic regression model from the MIMIC III data, and check how the AUC changes given different privacy budgets, as in Figure 3.5 (c), we set the number of selected parameters, k_0, to 3, because it corresponds to the highest AUC in Figure 3.5 (d). For different algorithms, the same parameter may take different values. Say, in Figure 3.5 (c), k_0 for SASLR/DPFW are 50/10 separately. Since our algorithm has the fewest parameters to tune, our algorithm cannot take any advantage from this setting.

The default privacy budget ε is 5 for regression on diabetes data, and 1 otherwise. We use larger ε for diabetes data because if smaller ε is used, all algorithms cannot beat the training average w.r.t. MSE. Thus to use the measurement relative MSE, $\varepsilon > 5$ is

required.

We first compare performance of algorithms in classification task. The results are listed in Figure 3.5. Our algorithm is proven better both at prediction and feature selection regardless of privacy budget on both data sets, except when large k_0 is used for the MIMIC III data.

There could be two reasons for worse performance given large k_0 in Figure 3.5 (c). First of all, the convergence is exponentially slower when k_0 is larger, according to the Theorem 3.2.16. Therefore, our algorithm is far from convergence after 100,000 iterations, and its performance is also far from the optimal performance. For alternatives, there are no such concerns about convergence. Second, our algorithm can find the truly effective parameters when k_0 is small, while the alternatives cannot, according to the experiments on synthetic data. Therefore, using a larger k_0 does not benefit our algorithm with more useful features, but helps alternatives by including those features. Combining the two factors together, large k_0 brings down performance of our algorithm, while improves performance of alternatives'. But anyway, the performance peak of our algorithm is much better than that of the alternatives'.

Occasionally, the performance of our algorithm degrades when the privacy budget becomes larger. The reason is similar to the one above: according to Theorem 3.2.16, convergence becomes slower when ε becomes larger. Thus performance gets worse.

We then compare performance of algorithms in regression task. The results on synthetic data are listed in Figure 3.6. our algorithm is proven much better on all measurements.

The results on diabetes data are listed in Figure 3.7. Our algorithm has much better correlation compared to alternatives. As for relative MSE, our algorithm is the only one that can beat training average, a trivial baseline.

The curse of large k_0 is observed again in the regression task, and it has almost

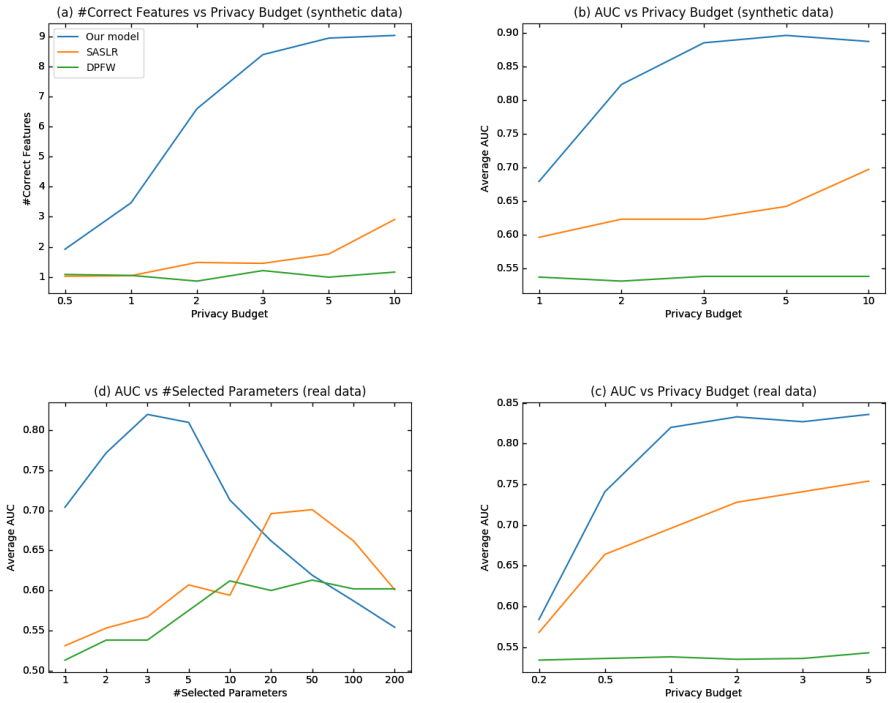


Figure 3.5. Classification results: (a) number of correctly picked features based on the synthetic data. (b) AUCs on synthetic data. (c) and (d) AUCs given different k_0 and ε on the MIMIC III data. Our algorithm is the best except given larger k_0 on MIMIC III data.

the same reason as that in the classification task.

Section 3.1.3 comes from the paper *Scalable Privacy-Preserving Data Sharing Methodology for Genome-Wide Association Studies: an Application to iDASH Healthcare Privacy Protection Challenge.* Fei Yu, Zhanglong Ji. BMC Medical Informatics and Decision Making 2014, 14(Suppl 1):S3. The dissertation author was the primary author of the part included in this dissertation.

In Section 3.1.4, we use the introduction of the Transmission Disequilibrium Test in the paper *Privacy-Preserving Mechanisms for Transmission Disequilibrium Test in Genome-Wide Association Studies.* Meng Wang, Zhanglong Ji, Shuang Wang, Jihoon Kim, Hai Yang, Lucila Ohno-Machado, and Xiaoqian Jiang. TBC 2015 The dissertation

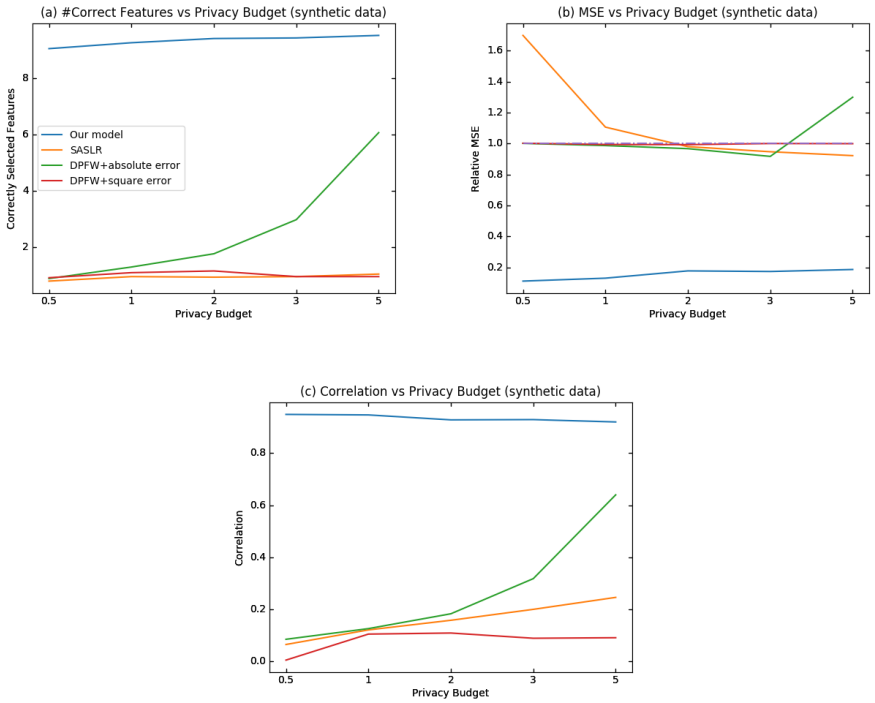


Figure 3.6. Regression results for synthetic data: the four figures show relative MSE and correlation given different privacy budgets ϵ and numbers of selected features k_0. Our algorithm always performs the best.

author contributed to the writing of the part.

Section 3.2 comes from the paper *Differentially Private Empirical Risk Minimization with Feature Selection.* Zhanglong Ji, Xiaoqian Jiang, Lucila Ohno-Machado, which is in submission right now. The dissertation author was the primary author of the paper.

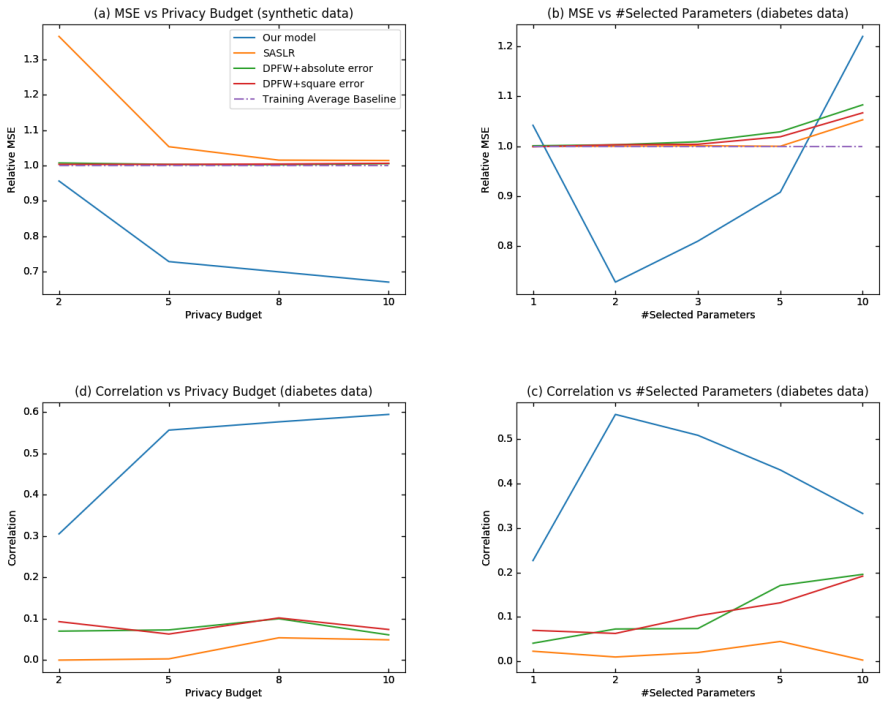


Figure 3.7. Regression results for diabetes data: relative MSE and correlation given different ϵ and k_0. Our algorithm always has the best correlation, and is the only algorithm that can beat baseline on relative MSE.

Algorithm 3. Hamming distance based score for a SNP

INPUT: tuple $(n_{00}, n_{10}, n_{01}, n_{20}, n_{11}, n_{02})$, threshold on p-value p.
OUTPUT: Hamming distance based score s.

1: Compute $t \geq 0$, which corresponds to the p-value p.
2: If test statistics $T > t$, go to Step 14 if $b \geq c$, or Step 16 if $b < c$.
3: If $b = n_{10} + 2(n_{20} + n_{02}) + n_{11}, c = n_{01} + n_{11}$ has $T > t$, then compute the smallest s_1 such that $b = n_{10} + n_{11} + 2(n_{20} + s_1), c = n_{01} + n_{11} + 2(n_{02} - s_1)$ has $T > t$. Then go to Step 8.
4: If $b = n_{10} + 2(n_{20} + n_{02} + n_{01}) + n_{11}, c = n_{11}$ has $T > t$, then compute the smallest s_1 such that $b = n_{10} + n_{11} + 2(n_{20} + s_1), c = n_{01} + n_{11} - s_1 + n_{02}$ is has $T > t$. Then go to Step 8.
5: If $b = n_{10} + 2(n_{20} + n_{02} + n_{01} + n_{11}), c = 0$ has $T > t$, then compute the smallest s_1 such that $b = n_{10} + (n_{01} - s_1 + n_{11} + n_{02}) + 2(n_{20} + s_1), c = n_{01} - s_1 + n_{11} + n_{02}$ has $T > t$. Then go to Step 8.
6: If $b = n_{10} + 2(n_{00} + n_{20} + n_{02} + n_{01} + n_{11}), c = 0$ has $T > t$, then compute the smallest s_1 such that $b = n_{10} + 2(n_{20} + s_1), c = 0$ has $T > t$. Then go to Step 8.
7: Compute the smallest s_1 such that $b = n_{10} - (s - n_{00} - n_{11} - n_{01} - n_{02}) + 2(n_{20} + s_1), c = 0$ has $T > t$. Then go to Step 8.
8: If $b = n_{10} + n_{11}, c = n_{01} + n_{11} + 2(n_{20} + n_{02})$ has $T > t$, then compute the smallest s_2 such that $b = n_{10} + n_{11} + 2(n_{20} - s_2), c = n_{01} + n_{11} + 2(n_{02} + s_2)$ has $T > t$. Then go to Step 13.
9: If $b = n_{11}, c = n_{11} + n_{01} + 2(n_{20} + n_{02} + n_{10})$ has $T > t$, then compute the smallest s_2 such that $b = n_{10} + n_{11} - s_2 + n_{20}, c = n_{01} + n_{11} + 2(n_{02} + s_2)$ is has $T > t$. Then go to Step 13.
10: If $b = 0, c = n_{01} + 2(n_{02} + n_{20} + n_{10} + n_{11})$ has $T > t$, then compute the smallest s_2 such that $b = n_{10} - s_2 + n_{11} + n_{20}, c = n_{01} + (n_{10} - s_2 + n_{11} + n_{20}) + 2(n_{02} + s_2)$ has $T > t$. Then go to Step 13.
11: If $b = 0, c = n_{01} + 2(n_{00} + n_{02} + n_{20} + n_{10} + n_{11})$ has $T > t$, then compute the smallest s_2 such that $b = 0, c = n_{01} + 2(n_{02} + s_2)$ has $T > t$. Then go to Step 13.
12: Compute the smallest s_2 such that $b = 0, c = n_{01} - (s - n_{00} - n_{11} - n_{10} - n_{20}) + 2(n_{02} + s_1)$ has $T > t$. Then go to Step 13.
13: Return $s = 1 - \min\{s_1, s_2\}$.
14: If $b = n_{10} + n_{11}, c = n_{01} + n_{11} + 2n_{20} + 2n_{02}$ has $T \leq t$ or $b \leq c$, then return the smallest s such that $b = n_{10} + n_{11} + 2(n_{20} - s), c = n_{01} + n_{11} + 2(n_{02} + s)$ has $T \leq t$ or $b \leq c$.
15: Return the smallest s such that $b = n_{10} - s + n_{20} + n_{11}, c = n_{01} + n_{11} + 2(n_{02} + s)$ has $T \leq t$ or $b \leq c$.
16: If $b = n_{10} + n_{11} + 2n_{20} + 2n_{02}, c = n_{01} + n_{11}$ has $T \leq t$ or $b \geq c$, then return the smallest s such that $b = n_{10} + n_{11} + 2(n_{20} + s), c = n_{01} + n_{11} + 2(n_{02} - s)$ has $T \leq t$ or $b \geq c$.
17: Return the smallest s such that $b = n_{10} + n_{11} + 2(n_{20} + s), c = n_{01} - s + n_{02} + n_{11}$ has $T \leq t$ or $b \geq c$.

Algorithm 4. Our Algorithm on Empirical Risk Minimization with Feature Selection

Require: Dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$. Number of groups to select $k_0 < k$, privacy budget ε, bound on loss function M, number of iterations T. A probability density to sample each non-zero component of w, $q(w)$.

Ensure: Linear regression weights $(w_1, ..., w_k)$.

1: Randomly choose k_0 features to form F_u, and randomly sample their weights from $q(w)$. Add all the other features to F_c and set their weights to 0. Now we have $\{w_1, ..., w_k\}$.

2: Randomly select one element u from F_u and another one c from F_c.

3: Get current weight w_u of feature u. Randomly, sample w'_c for c from $q(w)$.

4: Compute loss L_u for current model

$$L_u = \sum_{i=1}^{n} \left| y_i - \sum_{u \in F_u} w_u^T x_u \right|$$

5: Compute loss L_c for new model

$$L_c = \sum_{i=1}^{n} \left| y_i - \sum_{u \in F_u - \{u'\}} w_u^T x_u - w_c'^T x_c \right|$$

6: If $L_c \leq L_u$, change w_u to 0 and $w_c = 0$ to w'_c. Otherwise take the same change with probability $\exp\left(\frac{(L_u - L_c)\varepsilon}{2M}\right)$.

7: $T = T - 1$. If $T \geq 0$, go back to Step 3.

8: Output $\{w_1, ..., w_k\}$.

Chapter 4

Differentially Private Data Publishing Algorithms

In this Chapter, I introduce some of my work on publishing synthetic data in a differentially private way, namely a data publishing algorithm based on importance weighting and the Select and Label (S&L) algorithm publishing data for supervised learning.

We do not add independent randomness to each sample, as it may result in too much randomness even if the sample size is large (Section 2.4). Informally speaking, we first learn some properties from data in a noisy way, and then generate data from these noisy properties.

4.1 Differentially Private Data Publishing based on Importance Weighting

The first section introduces a data publishing algorithm based on importance weighting. In other words, assuming that we have a public dataset E which has the same data schema as private dataset D, this algorithm assigns weights for each sample in E so that the weighted dataset E would be statistically similar to D, especially the queries are in the form of $E_D[b(x)]$.

Many queries are in the form of, or based on, $E_D[b(x)]$, which asks the expectation of a function $b(x)$ on private data D. For example, if someone wants to learn a model from the private data, s/he may ask what the gradient vector or Hessian matrix of a loss function is. If s/he wants to study causation among variables in the dataset, s/he may ask what the values of correlation coefficients are. Generally, we suppose that the user wants to know the expectation of some function $b(x)$ over the distribution $p_D(\cdot)$ from which the private dataset D is drawn. That is, the goal is to know $E_D[b(x)] = E_{x\sim p_D(\cdot)}[b(x)]$. The function $b(x)$ is not limited to be an indicator function, as it is for counting queries. Note that E_D is an expectation over p_D, as opposed to over an empirical distribution defined by a specific dataset D.

Suppose that there exists another dataset E that is already public, whose samples are random from the distribution $p_E(\cdot)$. Since the samples in D have privacy concerns but those in E do not, we want to use E to help estimate $E_D[b(x)]$. Because D and E in general arise from different distributions, it is not reasonable to simply compute the average of $b(x)$ over E. Importance weighting varies the weights of the samples in E in order to improve accuracy. Let the cardinalities of E and D be N_E and N_D. The goal is to find a weight $w(x)$ for each x in E such that for any function $b(x)$ the following equation is approximately satisfied:

$$E_D[b(x)] = \frac{1}{N_E}\sum_{x\in E} b(x)w(x). \tag{4.1}$$

If E is already public and the owner of D publishes the weights $w(x)$ in a way that guarantees differential privacy, then outsiders can estimate $E_D[b(x)]$ without access to D, for any $b(x)$, without violating privacy, by computing $\frac{1}{N_E}\sum_{x\in E} b(x)w(x)$.

In general, no $w(x)$ can make Equation (4.1) be satisfied exactly for all possible $b(x)$ when the dataset E is finite. So, we explain here a differentially private algorithm

$\mathscr{K}$ based on logistic regression that yields weights that make the equation hold approximately. The output of the algorithm is the set of weights, that is $\mathscr{K}(D) = \{w(x) : x \in E\}$.

The so-called importance sampling identity is the equation

$$E_D[b(x)] = E_E\left[b(x)\frac{p_D(x)}{p_E(x)}\right].$$

To be valid, the support of the distribution p_E must contain the support of p_D, that is if $p_D(x) > 0$ then $p_E(x) > 0$ must be true also. Equation (4.1) and the identity make $p_D(x)/p_E(x)$ a natural choice for $w(x)$.

For a sample x, its importance weight $w(x)$ is the ratio of the probability density of x according to the two different distributions p_D and p_E. Both these distributions are in general high-dimensional densities, where the dimensionality is the length of the x vectors. Estimating high-dimensional densities is difficult at best, and often infeasible [53]. Fortunately, one can estimate the ratio $w(x)$ indirectly, without estimating p_D and p_E explicitly. Consider an equally balanced mixture of the distributions p_D and p_E, and suppose that samples from p_D are extended with the label $s = 1$ while those from p_E are extended with the label $s = 0$. A similar idea was used previously by [55] and by [20]. Then,

$$p(s = 1|x) = \frac{p(x|s = 1)p(s = 1)}{p(x)} = \frac{p_D(x)(1/2)}{p(x)}$$

by Bayes' rule. Therefore,

$$p(s = 1|x) = \frac{p_D(x)(1/2)}{p_D(x)(1/2) + p_E(x)(1/2)} = \frac{1}{1 + p_E(x)/p_D(x)}.$$

We can derive

$$w(x) = \frac{p_D(x)}{p_E(x)} = \frac{1}{1/p(s = 1|x) - 1}.$$

Algorithm 5. Importance weighting algorithm

Require: Private dataset D, public dataset E, privacy budget ε, regularization strength λ. Each sample x in D has d components that are in $[0,1]$.

Ensure: Weight $w(x)$ for each x in E.

1: Regularized logistic regression: Obtain β^* by solving

$$\begin{aligned}\beta^* &= \arg\min_{\beta} -\frac{1}{N_E}\sum_{x\in E}\log p(x\in E|x\in D\cup E) \\ &\quad -\frac{1}{N_D}\sum_{x\in D}\log p(x\in D|x\in D\cup E)+\frac{\lambda}{2}\|\beta\|^2\end{aligned}$$

where $p(x\in D|x\in D\cup E)=1-p(x\in E|x\in D\cup E)=1/(1+\exp(-\beta^T x))$.

2: Add high dimensional Laplace noise to β^* to get the final perturbed β:

$$\beta=\beta^*+\delta \text{ where } p(\delta)\propto\exp(-\frac{\varepsilon\|\delta\|_2 N_D\lambda}{\sqrt{d}}).$$

3: Output $w(x)=(N_E/Z)e^{\beta^T x}$ for each x in E, where $Z=\sum_{x\in E}e^{\beta^T x}$.

This equation lets us write each weight $w(x)$ as a deterministic transformation of $p(s=1|x)$. The equation is correct as a statement of probability theory. Its practical usefulness depends on having a good model for $p(s=1|x)$.

Concretely, we treat the datasets D and E as training sets for two classes $s=1$ and $s=0$. The logistic regression model

$$p(s=1|x)=p(x\in D|x\in D\cup E)=\frac{1}{1+e^{-\beta^T x}}$$

which yields $w(x)=e^{\beta^T x}$ is an obvious choice. However, it cannot ensure differential privacy directly, because there is no bound on the sensitivity of the logistic regression parameters β when D changes by one sample. If we use a strongly convex penalty function (see Definition 4.1.1), such as the sum of squared components of β in Step 1 of Algorithm 5, and if each sample x in D is a vector of length d with components that are in the range $[0,1]$, then Theorem 4.1.5 says that ε-differential privacy is achieved. The

parameter of the Laplace distribution in Algorithm 5 has denominator $\sqrt{d}$ because that is the maximum norm of any x. In general, $\sqrt{d}$ can be replaced by the upper bound over D of the L_2 norm of samples. The definition of strongly convex function and proof of differential privacy follows:

Definition 4.1.1: The function f is λ-strongly convex if and only if for every $x_1 < x_2$ and all $0 \leq \alpha \leq 1$

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2) - \frac{\lambda}{2}\alpha(1-\alpha)(x_1 - x_2)^2.$$

With a strongly convex loss function, such as the sum of squares of β in Step 1 of Algorithm 5, and if each sample x in D has d components that are in the interval $[0,1]$ then Algorithm 5 achieves differential privacy. In the following, $\|.\|$ always means L_2 norm.

Lemma 4.1.2: If $G(x)$ and $G(x)+g(x)$ are λ-strongly convex, continuous, and differentiable at all points, and the norm of the first derivative of $g(x)$ is at most c, then the points that minimize $G(x)$ and $G(x)+g(x)$ differ by at most c/λ.

Proof: This is Lemma 7 of [8].

Lemma 4.1.3: Let the dimension of each training example be d, let each example component be in $[0,1]$, and let the logistic regression parameters based on D_1 and D_2 be β_1^* and β_2^*. Then $\|\beta_1^* - \beta_2^*\|$ is bounded by $\sqrt{d}/N_D\lambda$ where $N_D = \max\{\#D_1, \#D_2\}$.

Proof. For deletion or addition, suppose $D_2 = D_1 \backslash \{x_0\}$ and $N_D = \#D_1$. Then the regu-

larized loss functions for training on D_1 and D_2 are

$$\begin{aligned} G_1(\beta) &= \frac{1}{N_E}\sum_{x\in E}\log(1+\exp(\beta^T x)) + \frac{1}{N_D}\sum_{x\in D_1}\log(1+\exp(-\beta^T x)) + \frac{\lambda}{2}\|\beta\|^2 \\ G_2(\beta) &= \frac{1}{N_E}\sum_{x\in E}\log(1+\exp(\beta^T x)) + \frac{1}{N_D-1}\sum_{x\in D_2}\log(1+\exp(-\beta^T x)) + \frac{\lambda}{2}\|\beta\|^2. \end{aligned}$$

Define $g_1(\beta)$ and $g_2(\beta)$ as

$$\begin{aligned} g_1(\beta) &= \frac{1}{N_D(N_D-1)}\sum_{x\in D_2}\log\frac{1}{1+\exp(-\beta^T x)} \\ g_2(\beta) &= \frac{1}{N_D}\log\frac{1}{1+\exp(-\beta^T x_0)}. \end{aligned}$$

The difference between G_1 and G_2 is

$$g(\beta) = G_1(\beta) - G_2(\beta) = g_2(\beta) - g_1(\beta).$$

Because the unregularized loss function in logistic regression is convex, $G_1(\beta)$ and $G_2(\beta)$ are both λ-strongly convex. In addition, because each partial derivative of the loss function is in $(0,1)$, all components of $g_1'(\beta)$ and $g_2'(\beta)$ are in $[0,1/N_D]$, and so are the absolute values of components of $g'(\beta) = g_1'(\beta) - g_2'(\beta)$. Therefore, $\|g'(\beta)\| \leq \sqrt{d}/N_D$, as there are at most d components. Then according to Lemma 4.1.2, $\|\beta_1^* - \beta_2^*\|$ is bounded by $\sqrt{d}/N_D\lambda$.

For replacement, suppose $D_2 = D_1\backslash\{x_1\}\cup\{x_2\}$ and $\#D_1 = \#D_2 = N_D$. Now $G_1(\beta)$ is the same as above but

$$G_2(\beta) = \frac{1}{N_E}\sum_{x\in E}\log(1+\exp(\beta^T x)) + \frac{1}{N_D}\sum_{x\in D_2}\log(1+\exp(-\beta^T x)) + \frac{\lambda}{2}\|\beta\|^2$$

so

$$g(\beta) \quad = \quad G_1(\beta) - G_2(\beta) = \frac{1}{N_D}\log(1+\exp(-\beta^T x_1)) - \frac{1}{N_D}\log(1+\exp(-\beta^T x_2)).$$

So again $\|g'(\beta)\| \leq \sqrt{d}/N_D$. Thus $\|\beta_1^* - \beta_2^*\|$ is bounded by $\sqrt{d}/N_D\lambda$. Therefore, $\|\beta_1^* - \beta_2^*\| \leq \sqrt{d}/N_D\lambda$ always holds. □

Lemma 4.1.4: The Laplacian noise algorithm yielding β in Step 2 of Algorithm 5 is ε-differentially private.

Proof: From Lemma 4.1.3 and Proposition 1 of [17], this algorithm is ε-differentially private.

Theorem 4.1.5: The algorithm $\mathscr{K}$ specified in Algorithm 5 is ε-differentially private.

Proof. Lemma 4.1.4 says that the algorithm $\mathscr{K}_2$ in Step 2 is ε-differentially private. That is, for all $S_\beta \subset Range(\mathscr{K}_2)$ and neighboring datasets D_1 and D_2

$$P(\mathscr{K}_2(D_1) \in S_\beta) \leq e^{\varepsilon} P(\mathscr{K}_2(D_2) \in S_\beta).$$

Furthermore, for all $S_w \subset Range(\mathscr{K})$, there is a set $S_\beta = \{\beta | w(x) \propto e^{\beta^T x} \in S_w\} \subset Range(\mathscr{K}_2)$ such that

$$P(w_1(x) \in S_w) = P(\mathscr{K}(D_1) \in S_w) = P(\mathscr{K}_2(D_1) \in S_\beta)$$

$$P(w_2(x) \in S_w) = P(\mathscr{K}(D_2) \in S_w) = P(\mathscr{K}_2(D_2) \in S_\beta).$$

To summarize,

$$\begin{aligned} P(w_1(x) \in S_w) &= P(\mathscr{K}_2(D_1) \in S_\beta) \\ &\leq e^\varepsilon P(\mathscr{K}_2(D_2) \in S_\beta) = e^\varepsilon P(w_2(x) \in S_w). \end{aligned}$$

So $\mathscr{K}$ is ε-differentially private. □

A common issue with importance weighting is that a few samples may have large weights, and these increase the variance of estimates based on the weights. There are various proposals using techniques such as softmax to make weights more uniform. Let τ be a constant. When $0 \leq \tau < 1$, the modified weights $w'(x) \propto w(x)^\tau \propto \exp(\tau\beta^T x)$ are less extreme. This is equivalent to replacing β by $\tau\beta$. Since softmax makes the norm of β smaller, its effect is similar to that of a larger penalty coefficient λ in Algorithm 5. We can use a larger λ to reduce the impact of individual samples in E on estimates, and introducing a separate constant τ is not necessary. [1]

As the strength of regularization λ increases, the learned coefficients β^* in Algorithm 5 tend towards zero, and the weights $w(x)$ tend towards one. This implies that estimates computed using Equation (4.1) increase in bias and tend towards the corresponding mean computed on the public dataset E. This property is evident in the statement of Theorem 4.1.10 below and in the experimental results (Figure 4.1). In practice,

[1] In standard regularized logistic regression, the loss function that is minimized is

$$-\frac{1}{N_E+N_D}[\sum_{x\in E}\log p(x\in E)+\sum_{x\in D}\log p(x\in D)]+\frac{\lambda}{2}\|\beta\|^2.$$

Instead, we use the balanced loss function

$$-\frac{1}{N_E}\sum_{x\in E}\log p(x\in E)-\frac{1}{N_D}\sum_{x\in D}\log p(x\in D)+\frac{\lambda}{2}\|\beta\|^2$$

which gives the log likelihoods for examples from D and E equal mass. In our scenarios, the samples in E are fixed, while the samples in D are random. With the usual form of logistic regression, the asymptotic convergence, in Step 1 of Algorithm 5, of β^* to the true parameter vector is not guaranteed.

solving the regularized optimization problem in Step 1 of the algorithm is computationally straightforward and fast regardless of the magnitude of λ.

Algorithm 5 adds noise to the coefficients β^* in order to protect privacy. An alternative approach to guarantee privacy with logistic regression is to perturb the objective function used for training [8]. Although we do not have theoretical results showing how well this alternative approach works, experiments indicate that its performance is similar to that of Algorithm 5.

4.1.1 Analysis

For a query function $b(x)$, the estimate of its true expectation $E_D[b(x)]$ obtained via the differentially private importance weighting algorithm is

$$\frac{1}{N_E}\sum_{x\in E} b(x)w(x).$$

Here we analyze the variance of this estimate. We assume that the public dataset E is fixed, so the variance of the estimate comes from the randomness of the dataset D and from the noise in Step 2 of Algorithm 5. Note that even in the absence of privacy concerns, there is variance in any estimate of $E_D[b(x)]$ due to randomness in D.

The weights are based on the logistic regression parametric model that $p_D(x)/p_E(x) = \exp(\beta^T x)$ for some β. The difference between the estimate and the true value may not converge to zero when this parametric assumption is not true, that is when logistic regression is not well-specified. However, in Theorem 4.1.10, we can give an upper bound on the variance of the estimate that converges to zero asymptotically, that is as the cardinality of D tends to infinity, regardless of whether logistic regression is well-specified.

In the following proofs, for square matrices A and B the expression $A \leq B$ means

$a^T Aa \leq a^T Ba$ for all vectors a. The vector x has length d and each of its components is in the range $[0,1]$.

Lemma 4.1.6: For any vector β that has the same length as x

$$Var_D\left[\frac{x}{1+\exp(\beta^T x)}\right] \leq E_D[xx^T] \leq dI.$$

Proof. For the first inequality, since $Var[y] = E[yy^T] - E[y]E[y^T]$, it is always true that $Var[y] \leq E[yy^T]$. Therefore, we just need to prove that

$$E_D\left[\frac{xx^T}{(1+\exp(\beta^T x))^2}\right] \leq E_D[xx^T]$$

As $\exp(\beta^T x)$ is always larger than 0, $\frac{xx^T}{(1+\exp(\beta^T x))^2} \leq xx^T$ always holds, thus this is true.

For the second inequality, since for all vectors a,

$$\begin{aligned} a^T E_D[xx^T]a &= E_D[a^T xx^T a] = E_D[\|a^T x\|^2] \\ &\leq E[\|a\|^2\|x\|^2] = \|a\|^2 E[\|x\|^2] \\ &\leq d\|a\|^2 = a^T(dI)a \end{aligned}$$

it follows that $E_D[xx^T] \leq dI$. □

For the next two lemmas, let

$$g(\beta) = \frac{1}{N_E}\sum_{x\in E}\log(1+\exp(\beta^T x)) - \frac{\lambda}{2}\|\beta\|^2$$

and let the vector β_0 optimize the loss function of logistic regression on fixed E and the true distribution of D:

$$\beta_0 = \arg\max_{\beta} g(\beta) + E_D[\log(1+\exp(-\beta^T x))].$$

Lemma 4.1.7: Let E be fixed and let D be random. The variance of the output parameters β^* of the regularized logistic regression is asymptotically

$$\frac{1}{N_D}\left(g''(\beta_0)+E_D\left[\frac{\exp(\beta_0^T x)xx^T}{(1+\exp(\beta_0^T x))^2}\right]\right)^{-1}$$
$$\cdot Var_D\left[\frac{x}{1+\exp(\beta_0^T x)}\right]\cdot\left(g''(\beta_0)+E_D\left[\frac{\exp(\beta_0^T x)xx^T}{(1+\exp(\beta_0^T x))^2}\right]\right)^{-1}$$

where g'' is the second derivative of g.

Proof. Note that all three factors in the variance of β^* are matrices, and that the first and third factors are the same. Since only the set D is random, $g(\beta)$ is a deterministic function of β. The solution β^* is

$$\beta^* = \arg\max_{\beta} g(\beta)+\frac{1}{N_D}\sum_{x\in D}\log(1+\exp(-\beta^T x)).$$

As D is drawn from an underlying distribution, β^* is a random variable.

When N_D is large, β^* is close to β_0 with high probability. Furthermore, all the functions here are infinitely differentiable. Thus we can use a Taylor expansion to express the target function using its first and second derivatives at β_0:

$$\begin{aligned}\beta^* &= \arg\max_{\beta} g(\beta_0)+(\beta-\beta_0)^T g'(\beta_0)+\frac{1}{2}(\beta-\beta_0)^T g''(\beta_0)(\beta-\beta_0)\\ &\quad+\frac{1}{N_D}\sum_{x\in D}\left[\log(1+\exp(-\beta_0^T x))-(\beta-\beta_0)^T\frac{x}{1+\exp(\beta_0^T x)}\right.\\ &\qquad\left.+(\beta-\beta_0)^T\frac{\exp(\beta_0^T x)xx^T}{2(1+\exp(\beta_0^T x))^2}(\beta-\beta_0)\right]+o((\beta-\beta_0)^T(\beta-\beta_0)).\end{aligned}$$

The maximization is an unconstrained optimization problem, so the first derivative of

this expression is zero at the maximum point:

$$\begin{aligned} 0 &= g'(\beta_0) + g''(\beta_0)(\beta^* - \beta_0) \\ &+ \frac{1}{N_D}\sum_{x\in D}\left[-\frac{x}{1+\exp(\beta_0^T x)} + \frac{\exp(\beta_0^T x)xx^T}{(1+\exp(\beta_0^T x))^2}(\beta^* - \beta_0)\right] + o(\beta^* - \beta_0). \end{aligned}$$

Omitting the asymptotically negligible term yields

$$\begin{aligned} \beta^* - \beta_0 &= -\left[g''(\beta_0) + \frac{1}{N_D}\sum_{x\in D}\frac{\exp(\beta_0^T x)xx^T}{(1+\exp(\beta_0^T x))^2}\right]^{-1} \\ &\cdot\left[g'(\beta_0) - \frac{1}{N_D}\sum_{x\in D}\frac{x}{1+\exp(\beta_0^T x)}\right]. \end{aligned}$$

The law of large numbers ensures that the expression inside the matrix inverse converges to $g''(\beta_0) + E_D\left[\exp(\beta_0^T x)xx^T/(1+\exp(\beta_0^T x))^2\right]$ as N_D increases.

Also, because β_0 minimizes $g(\beta) + E_D\log(1+\exp(-\beta^T x))$, and this minimization is unconstrained, $0 = g'(\beta_0) - E_D\frac{x}{1+\exp(\beta_0^T x)}$. Therefore, according to the central limit theorem, the second factor

$$g'(\beta_0) - \frac{1}{N_D}\sum_{x\in D}\frac{x}{1+\exp(\beta_0^T x)} \sim N\left(0, \frac{1}{N_D}Var\left[\frac{x}{1+\exp(\beta_0^T x)}\right]\right)$$

asymptotically. Finally, the asymptotic variance of β^* is

$$\begin{aligned} Var[\beta^*] &= Var[\beta^* - \beta_0] \\ &= \frac{1}{N_D}\left(g''(\beta_0) + E_D\left[\frac{\exp(\beta_0^T x)xx^T}{(1+\exp(\beta_0^T x))^2}\right]\right)^{-1} \\ &\cdot Var_D\left[\frac{x}{1+\exp(\beta_0^T x)}\right]\left(g''(\beta_0) + E_D\left[\frac{\exp(\beta_0^T x)xx^T}{(1+\exp(\beta_0^T x))^2}\right]\right)^{-1}. \end{aligned}$$

□

The previous lemma gives an exact asymptotic expression for $Var[\beta^*]$ when the cardinality of D tends to infinity. However, β_0 in the expression is unknown. The following lemma gives an upper bound for the variance that depends only on the underlying distribution of D and on λ.

Lemma 4.1.8: Let E be fixed and let D be random. The variance of the output parameters β^* of the regularized logistic regression is asymptotically less than $\frac{dI}{N_D\lambda^2}$.

Proof. Because $g(\beta)$ is the sum of a convex function and $\frac{\lambda}{2}\|\beta\|^2$, its second derivative is larger than λ. Also, $E_D\left[\frac{\exp(\beta_0^T x)xx^T}{(1+\exp(\beta_0^T x))^2}\right] \geq 0$. Therefore

$$\begin{aligned} Var[\beta^*] &= \frac{1}{N_D}\left(g''(\beta_0) + E_D\left[\frac{\exp(\beta_0^T x)xx^T}{(1+\exp(\beta_0^T x))^2}\right]\right)^{-1} \\ &\quad \cdot Var_D\left[\frac{x}{1+\exp(\beta_0^T x)}\right]\left(g''(\beta_0) + E_D\left[\frac{\exp(\beta_0^T x)xx^T}{(1+\exp(\beta_0^T x))^2}\right]\right)^{-1} \\ &< \frac{1}{N_D}\lambda^{-1} Var_D\left[\frac{x}{1+\exp(\beta_0^T x)}\right]\lambda^{-1} \\ &= \frac{1}{N_D\lambda^2} Var_D\left[\frac{x}{1+\exp(\beta_0^T x)}\right] \leq \frac{dI}{N_D\lambda^2} \end{aligned}$$

using Lemma 3.1.6 for the last inequality. □

The following lemma takes into account not just randomness from D, but also randomness from the noise added to protect privacy in Step 2 of Algorithm 5. Here, I is the identity matrix.

Lemma 4.1.9: The total variance of β is asymptotically less than

$$\frac{dI}{N_D\lambda^2} + \frac{d}{d+1}\frac{1}{(N_D\lambda\varepsilon)^2}I.$$

Proof. The noise $\delta = \{\delta_1, ..., \delta_d\}$ added to β^* is independent of β^*, so the variance of β is the variance of β^* plus the variance of the noise:

$$Var[\beta] = Var[\beta^*] + Var[\delta].$$

The probability density of δ is $p(\delta) \propto \exp(-\delta/\gamma)$ where $\gamma = S/\varepsilon = \sqrt{d}/N_D\lambda\varepsilon$.

Because of independence and symmetry among the elements of δ, its covariance matrix A is cI for some scalar

$$\begin{aligned} c &= \frac{1}{d}\sum_{i=1}^{d} A_{ii} = \frac{1}{d}\sum_{i=1}^{d} Var[\delta_i] \\ &= \frac{1}{d}\sum_{i=1}^{d} E[\delta_i^2] = \frac{1}{d}E[\delta^T\delta] \\ &= \frac{1}{d}\frac{\int_0^{+\infty} r^2\exp(-r/\gamma)r^{d-1}dr}{\int_0^{+\infty}\exp(-r/\gamma)r^{d-1}dr} \\ &= \frac{\gamma^2}{d}\frac{\int_0^{+\infty} t^2\exp(-t)t^{d-1}dt}{\int_0^{+\infty}\exp(-t)t^{d-1}dt} \\ &= \frac{\gamma^2}{d}\frac{\Gamma(d+2)}{\Gamma(d)} \\ &= \frac{(d+1)d}{(N_D\lambda\varepsilon)^2}. \end{aligned}$$

This result, with Lemma 4.1.8, gives the bound on the total variance of β. □

At last, we are in a position to prove the theorem about the asymptotic variance of the estimate of the expectation of a query function $b(x)$.

Theorem 4.1.10: The total variance of the estimate $\frac{1}{N_E}\sum_{x\in E} b(x)w(x)$ is asymptotically

$$\begin{aligned} Var\left[\frac{1}{N_E}\sum_{x\in E} b(x)w(x)\right] &= \alpha^T Var[\beta]\alpha \\ &< \alpha^T(dI/N_D\lambda^2 + d(d+1)I/(N_D\lambda\varepsilon)^2)\alpha \end{aligned}$$

where d is the dimensionality of data points x, I is the identity matrix, and

$$\alpha = \frac{\sum_{x_i,x_j \in E} e^{\beta_0^T (x_i+x_j)} (b(x_i) - b(x_j))(x_i - x_j)}{\sum_{x_i,x_j \in E} e^{\beta_0^T (x_i+x_j)}}.$$

The vector β_0 minimizes the loss function of logistic regression on E and the distribution p_D.

Proof. Using the definition of the weights $w(x)$, the variance is

$$Var\left[\frac{1}{N_E}\sum_{x\in E} b(x)w(x)\right] = Var\left[\sum_{x\in E} b(x)e^{\beta^T x} \Big/ \sum_{x\in E} e^{\beta^T x}\right] = f(\beta).$$

Since E is fixed and $b(x)$ is given, the variance arises only from β. As β asymptotically converges to β_0, $f(\beta)$ satisfies the following equations asymptotically:

$$\begin{aligned} f(\beta) &= f(\beta_0) + f'(\beta)(\beta - \beta_0) \\ Var[f(\beta)] &= (f'(\beta))^T Var[\beta] f'(\beta). \end{aligned}$$

The derivative of $\sum_{x\in E} b(x)e^{\beta^T x} \Big/ \sum_{x\in E} e^{\beta^T x}$ is

$$\alpha = \frac{\sum_{x_i,x_j \in E} e^{\beta_0^T (x_i+x_j)} (b(x_i) - b(x_j))(x_i - x_j)}{\sum_{x_i,x_j \in E} e^{\beta_0^T (x_i+x_j)}}.$$

Hence the variance of the estimate is

$$\alpha^T Var[\beta]\alpha < \alpha^T (dI/N_D\lambda^2 + d(d+1)I/(N_D\lambda\varepsilon)^2)\alpha.$$

□

Theorem 4.1.11: The bias of the estimate is asymptotically

$$\sum_{x \in E} b(x) \frac{\exp(\beta_0^T x)}{\sum_{y \in E} \exp(\beta_0^T y)} - E_D[b(x)]$$

where β_0 minimizes the loss function of regularized logistic regression on E and p_D, as in Theorem 4.1.10.

Proof. When the number of samples in D is large, the logistic regression parameter vector obtained in the first step of Algorithm 5 converges to β_0, and the noise added in the second step converges to 0. Therefore, the vector β used to compute the weights also converges to β_0. Since the weights and the estimate are both continuous with respect to β, the estimate converges to

$$\sum_{x \in E} b(x) \frac{\exp(\beta_0^T x)}{\sum_{y \in E} \exp(\beta_0^T y)}.$$

The bias is the difference between the convergence point and the true expectation. □

Theorem 3.1.10 provides a strict inequality. We write $Var[]$ and not $Var_D[]$ because the variance includes not only randomness from D, but also randomness from the noise in Step 2 of Algorithm 5. The factor α comes from the derivative with respect to β of the estimate $\frac{1}{N_E} \sum_{x \in E} b(x) w(x)$.

A large N_D ensures a decrease of the variance of β^* and of estimates, because more samples have less noise on average, and also because the noise needed for privacy is less due to smaller sensitivity of β^*. The rate of decrease $1/N_D$ is of the same order as for the variance of direct estimates $\frac{1}{N_D} \sum_{x \in D} b(x)$, which of course is $\frac{1}{N_D} Var_D[b(x)]$. Thus differential privacy can be achieved without slowing the convergence of estimates compared to the absence of privacy, that is using the dataset D directly.

A large λ can reduce the Laplacian noise significantly, but if it is too large, then

the bias in estimates can be large. A large privacy budget ε helps reduce the Laplacian noise, and hence reduces the variance of estimates. However, ε may be specified by policy, and making it larger harms privacy. Moreover, if $N_D\varepsilon^2 \gg d$, then the first term dominates and a smaller ε cannot help reduce the variance.

When the number of dimensions d increases, the variance gets larger for two reasons. First, the L_2 sensitivity of β^* increases. Second, the curse of dimensionality worsens the situation: if $p(\delta) \propto \exp(-\|\delta\|_2)$ with $\delta \in R^d$ then $E[\|\delta\|_2]$ increases linearly with d.

The factor α is the most complicated among the factors that determine the variance of estimates. It is not controllable, because the function $b(x)$ and the public dataset E must be taken as fixed. However, the expression for α reveals which $b(x)$ can be estimated with smaller variance: if the values of $b(x)$ in E are close to each other, especially on the samples for which $w(x)$ is large, then α can be small.

Theorem 4.1.10 is useful not only for bounding the variance, but also for bounding the total error under some conditions. Specifically, suppose that logistic regression is well-specified and regularization is weak, meaning that λ is small and β_0 exists such that $p_D(x)/p_E(x) = \exp(\beta_0^T x)$. The existence of β_0 means that Equation (4.1) holds for any $b(x)$ highly accurately with $w(x) = \exp(\beta_0^T x)$. Small λ means that β^* is close to β_0 given large N_D, and that the β^* and β vectors are approximately unbiased. Hence, the estimate is approximately unbiased.

The argument about asymptotic unbiasedness is formalized in Theorem 4.1.11. Combining Theorems 4.1.10 and 4.1.11, variance and bias are both small, and hence total error is small, when the four following conditions hold: (i) there exists β such that $\frac{p_D(x)}{p_E(x)} \propto \exp(\beta^T x)$, (ii) the regularization strength λ is small so that β_0 is close to β and thus the bias is small, (iii) the number of samples in D is large so that the estimate has small variance, and (iv) the number of samples in E is large so that the weighted sum

over E converges to $E_E\left[b(x)\frac{p_D(x)}{p_E(x)}\right]$

4.1.2 Design of experiments

Here we investigate empirically the performance of the importance weighting method. We see how parameter values (the strength of regularization λ and the privacy budget ε) affect the accuracy of estimates obtained using the method, and how the method behaves with different target functions, that is queries.

The dataset we use is derived from the "adult" dataset in the UC Irvine repository [40]. The original dataset contains more than 40,000 records, each corresponding to a person. Each record has 15 features: sex, education level, race, national origin, job, etc. The first 14 features are often used to predict the last one, which is whether a person earns more than $50,000 per year. We use a processed version which has 63 binary variables obtained from 12 original features, taken from the R package named "arules" [25]. In general, preprocessing a dataset is a computation that must be taken into account in a privacy analysis, but here we assume that the private dataset is the preprocessed one as opposed to the original one. The preprocessing was done by other researchers for reasons unrelated to privacy, so the dataset was not created to favor any particular approach to privacy preservation.

Our approach needs a public dataset E. There is a test set that has the same schema as the original "adult" set, but it is from the same distribution, so we expect all weights to be approximately $1/N_E$, which is uninteresting (but does not violate privacy). To simulate the general situation where the public dataset is not from the same distribution as the private one, we split records from the pre-processed dataset by the feature sex. We place 90% of males and 10% of females in D, and the rest in E. The cardinalities of D and E are about 21,000 and 12,000 respectively. We then remove the feature sex, because in typical applications there is not any single feature that makes learning

the weights $w(x)$ easy. Splitting based on sex simulates, in an extreme way, situations where, for example, the public dataset consists of information from volunteers, while the private dataset consists of information from non-volunteers, who are quite different statistically from volunteers.

The experiments use $\lambda = 0.1$ and $\varepsilon = 0.1$ as default values. This value for the privacy budget ε is commonly used in research. We choose $\lambda = 0.1$ as a baseline because it is a good choice for training a conventional logistic regression classifier on the preprocessed "adult" dataset. We vary λ and ε to see how they affect the accuracy of estimates obtained using the importance weighting method. For each pair of λ and ε, we use bootstrap sampling to create randomness in the private dataset D: each time N_D samples are drawn from D with replacement to form a new private dataset D', and this D' is used with the importance weighting method to get an estimate. The results of 100 estimates from 100 experiments are shown. Note that records in D' are regarded as independent. Even if bootstrap sampling makes two records be copies of the same record in D, only one of the copies may change in the definition of differential privacy.

Alternative Algorithms

Some non-data-publishing algorithms can answer individual queries more accurately than the importance weighting method. In particular, the sensitivity of a count query is 1, so the Laplace mechanism can answer these queries, including the query $b(x) = I(income > \$50K)$ used later, directly with high accuracy on the "adult" dataset. For example, with $\varepsilon = 0.1$ and $|D| = 21,000$ as above, the answer is unbiased, with standard deviation approximately $10\sqrt{2}/21,000 \simeq 0.0007$. However, non-data-publishing algorithms must consume some of the available privacy budget for each query, leaving a smaller privacy budget for future queries. The point of this paper, in contrast, is to provide a once-and-for-all method of publishing data, after which an unlimited number

and range of queries can be answered without consuming any further privacy budget. Therefore, we compare experimentally only to other data publishing algorithms.

There are also some alternative data-publishing algorithms, such as [6, 26, 26, 63, 47, 39]. On the one hand, for the methods that require a predetermined query set Q [6, 26, 26], it is hard to find a reasonable choice for this set Q. It is too restrictive to make Q simply equal the specific test queries used below. On the other hand, most existing query-independent data-publishing algorithms either eliminate many features or feature values [63, 47], or place restrictions on the dataset [39], so they are not useful for this dataset.

The Laplace perturbation data-publishing algorithm adds noise to each feature in each sample in the dataset. This method is query-independent and does not eliminate any features. However, unfortunately, so much noise must be added that answers to queries are not useful. With 63 binary features obtained from 12 original categorical features, the L_1 sensitivity of the private dataset (viewed as a query) is at least 24. Given the privacy budget 0.1, noise from $Lap(240)$ must be added to each binary feature value in D. Suppose that we want to estimate the average value of a feature, a number between 0 and 1. The average of these noisy values is an unbiased estimate, but the standard deviation of the noisy average can be as large as $\sqrt{2 \cdot 240^2/21,000} \simeq 2.34$. This standard deviation is too large for the Laplace publishing algorithm to be practical.

In an alternative use of the Laplace publishing algorithm, the noisy features are trimmed to $[0,1]$. In this case the variance can be small, about $0.25/\sqrt{21,000} \simeq 0.002$. However, trimming causes large bias. When the noise-free true answer is 1, the expectation of the answer based on trimmed noisy values is 0.501. Similarly, when the true answer is 0, the expectation is 0.499. In both cases, the bias is 0.499.

In summary, for the first experiment we are not aware of an alternative method with which comparison would be appropriate. In the second experiment, we do compare

the importance weighting method to the non-data-publishing method of [8].

Queries and measures of success

The queries used in the two experiments are as follows. The first is a typical count query, namely the function $b(x) = I(income > \$50K)$. The second is a sequence of complex queries: all functions of the training data computed by the LIBLINEAR software while training a linear SVM. We investigate this because outsiders often want to use the dataset E and the published weights to learn a model that applies to the private dataset D, or to learn relationships between features within it. Linear SVMs are one of the most popular modeling methods. The outcome of SVM training depends on the gradients of the loss function, so training an SVM is equivalent to getting answers to queries concerning these. To evaluate success, we compare the SVM parameters β^D and β^E learned directly from D versus from the weighted E. As is standard, the linear SVM is trained to predict $income > \$50K$ from the other features.

For the count query, we plot the true empirical average on D and the estimates obtained using the importance weighting algorithm. To show the distribution of estimates, we plot the 95% confidence interval and quantiles at 1/4, 1/2 and 3/4. For the SVM, we plot the distribution of the Euclidean distance between the weight vectors β^D and β^E. We do not compare the prediction errors because the weight vectors are more informative, and because the relationship between prediction error and the gradient queries is not as close as the relationship between the parameters and the queries. Since the parameter corresponding to an unpredictive feature is close to 0, absolute Euclidean distance is more informative than relative distance $\sum_i (\beta_i^D - \beta_i^E)/\beta_i^D$ where i ranges over the components of β^D and β^E.

We compare SVM learning results with results from the method of [8], which outputs differentially private SVM parameters directly. Note that this comparison method

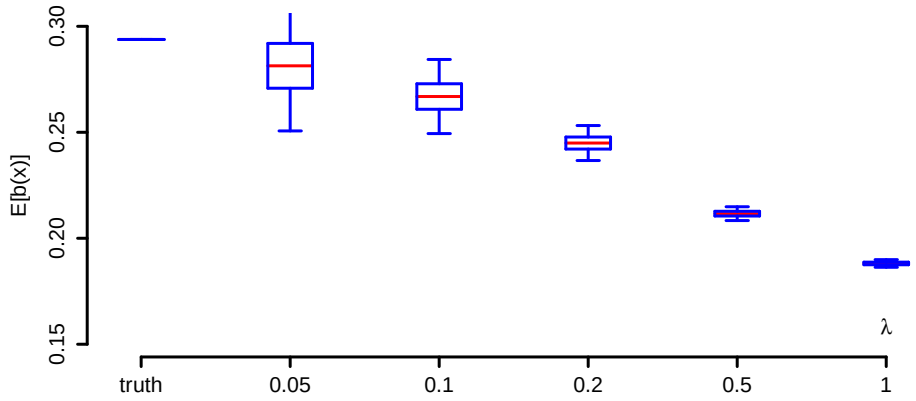


Figure 4.1. Performance of the importance weighting method as the strength of regularization λ varies. The query is $b(x) = I(income > \$50K)$ and the privacy budget is $\varepsilon = 0.1$.

is more specialized than the importance weighting method, which is general for all queries and all learning algorithms, linear and nonlinear.

4.1.3 Results of experiments

The unweighted average of $b(x) = I(income > \$50K)$ on E is around 0.15, which is far from the true value of $E_D[b(x)]$, which is approximately 0.3. However, in most of the experiments below, the estimates from the importance weighting method are close to 0.3. This shows the method is successful on a typical query, for a real-world dataset of limited size and a realistic privacy budget.

Figure 4.1 shows that the variance decreases as λ gets larger, while the bias increases and the estimate tends towards $E_E[b(x)] = 0.15$. This happens because when regularization becomes stronger, the β^* from the logistic regression is closer to the zero vector, and all the weights are closer to 1. Then $E_E[b(x)w(x)]$ tends to $E_E[b(x)]$. Note that privacy is guaranteed by setting $\varepsilon = 0.1$ regardless of λ.

Figure 4.2 shows that changing ε has a large effect on the variance of the es-

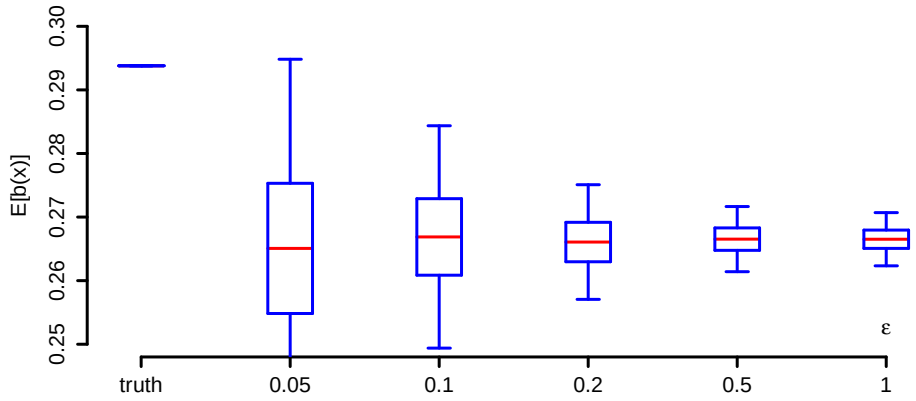


Figure 4.2. Performance of the importance weighting method as the privacy budget ε varies. The query is $b(x) = I(income > \$50K)$ and $\lambda = 0.1$.

timate, but little effect on its mean. This means that a smaller privacy budget causes greater noise in estimates, but does not make these estimates more biased. This behavior is the best that we can hope for from any method that preserves privacy.

Figure 4.3 shows the Euclidean distance between the parameters of the SVM model trained on D and the parameters of the model trained on E using weights. The norm of the parameters learned from D is 7.17, so distances around 1 indicate successful SVM training. As expected, the variance and bias both become smaller when the privacy requirement is less strict, that is when ε is larger. Regardless of how relaxed the privacy requirement is, distances remain above 0.8. Increasing ε cannot reduce the distance to zero mainly because $p_D(x)/p_E(x) \propto \exp(\beta^T x)$ is not satisfied exactly. With a better-specified model for the importance weights, the proposed method would perform even better.

We also compare our result with that of the differentially private SVM derived by [8]. We use the first algorithm of that paper, which adds noise to the true SVM coefficients. Fortunately, the scale of noise in the algorithm can be computed explicitly. The sensitivity stated in the paper is $2/n\Lambda$, under the assumption that $\|x\|_2 \leq 1$, where

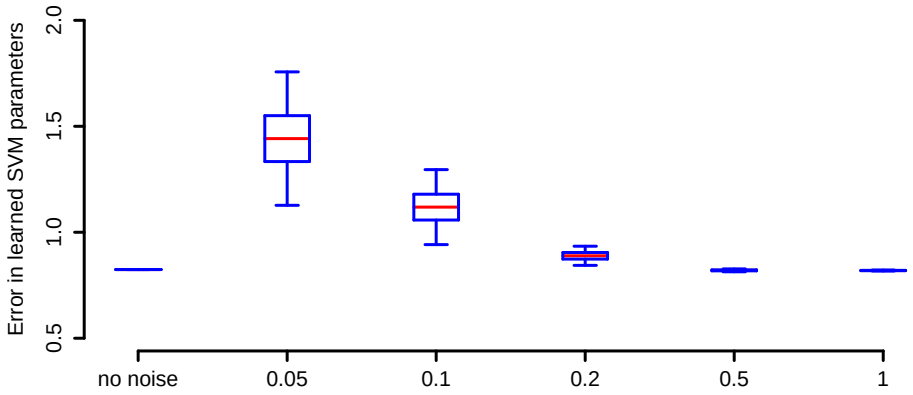


Figure 4.3. Euclidean distance between linear SVM parameter vectors learned from D and from E, with $\lambda = 0.1$ and regularization strength $\Lambda = 0.1$ for the SVM.

n is the cardinality of the training set and Λ is the regularization strength of the SVM. Because $\|x\|_2 \leq \sqrt{d}$ for the "adult" dataset, the sensitivity for it is $2\sqrt{d}/N_D\Lambda\varepsilon$ and the density function of noise b in the algorithm is $v(b) \propto \exp(-\frac{N_D\Lambda\varepsilon}{2\sqrt{d}}\|b\|_2)$.

The distribution of noise is symmetric around zero and $b \in \mathbb{R}^{+d} = [0,+\infty]^d$, so

$$\begin{aligned}
E[\|b\|_2^2] &= \int_{\mathbb{R}^{+d}} v(b)\|b\|_2^2 db \\
&= \frac{\int_{\mathbb{R}^{+d}} \exp(-\frac{N_D\Lambda\varepsilon\|b\|_2}{2\sqrt{d}})\|b\|_2^2 db}{\int_{\mathbb{R}^{+d}} \exp(-\frac{N_D\Lambda\varepsilon\|b\|_2}{2\sqrt{d}})db} \\
&= \frac{4d}{N_D^2\Lambda^2\varepsilon^2}\frac{\int_{\mathbb{R}^{+d}} \exp(-\|s\|_2)\|s\|_2^2 ds}{\int_{\mathbb{R}^{+d}} \exp(-\|s\|_2)ds} \\
&= \frac{4d}{N_D^2\Lambda^2\varepsilon^2}\frac{\int_{\mathbb{R}^{+}} t^2\exp(-t)d(t^d)}{\int_{\mathbb{R}^{+}} \exp(-t)d(t^d)} \\
&= \frac{4d}{N_D^2\Lambda^2\varepsilon^2}\frac{\int_{\mathbb{R}^{+}} t^{d+1}\exp(-t)dt}{\int_{\mathbb{R}^{+}} t^{d-1}\exp(-t)dt} \\
&= \frac{4d}{N_D^2\Lambda^2\varepsilon^2}\frac{\Gamma(d+2)}{\Gamma(d)} = \frac{4d^2(d+1)}{N_D^2\Lambda^2\varepsilon^2}.
\end{aligned}$$

Thus the expected L_2 norm of the noise is $\frac{2d\sqrt{d+1}}{N_D\Lambda\varepsilon} \simeq 4.8$ given dimensionality $d = 63$.

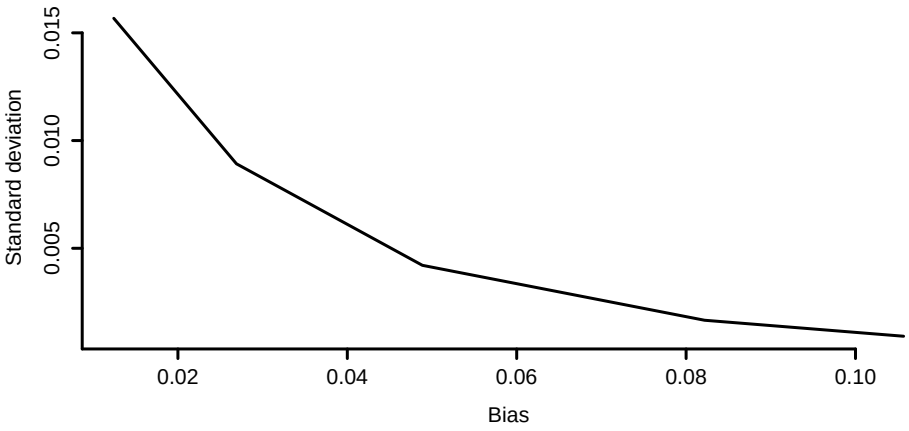


Figure 4.4. Trade-off between bias (horizontal axis) and standard deviation (vertical axis) when the strength of regularization λ varies, for the query $b(x) = I(income > \$50K)$ and with privacy budget $\varepsilon = 0.1$.

The importance weighting method has smaller error, less than 1.5.

Another experimental question is the effect of λ on the accuracy of estimates. We know theoretically that larger λ brings smaller standard deviation and larger bias, and vice versa. Figure 4.4 shows this trade-off between bias and standard deviation.

Last but not least, we would like to know how the importance weighting algorithm performs in extreme cases. One such case occurs when the public dataset and the private dataset are the same. Another extreme case is when the public dataset is uniformly drawn from the sample space. Results for these cases are shown in Figures 4.5 and 4.6. As before, $\varepsilon = 0.1$ and $\lambda = 0.1$, and the same two queries from before are used, so previous experimental results are shown. Not surprisingly, for both queries the best performance is when E is identical to D. Performance with the skewed E used previously is not much worse. Performance with the uniformly drawn E is worst, but in particular the trained SVM classifier (Figure 4.6) is still useful.

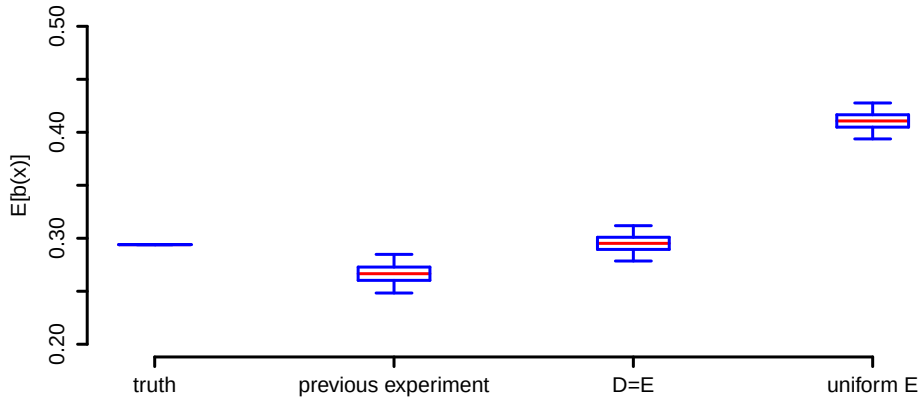


Figure 4.5. Performance of the importance weighting algorithm in extreme cases, for the query $b(x) = I(income > \$50K)$. The closer an output is to the truth, the better.

4.1.4 Discussion

The experimental results in Section 4.1.3 show that the differential privacy algorithm proposed in this paper is useful in practice, both for answering individual queries and for training supervised learning models. The theoretical results in Section 4.1.1 show that if the private dataset is large, then privacy can be preserved while still allowing queries to be answered with variance asymptotically similar to the variance that stems from the private dataset itself being random.

Naturally, variations on the importance weighting approach are possible. One idea is to draw a new dataset from E using the computed weights, instead of publishing the weights. However, this increases the variance of estimates without changing their expectation. Thus publishing the weights explicitly is preferable. Algorithm 5 ensures the weights are limited in magnitude and have enough noise to protect privacy.

The regularized logistic regression approach of Algorithm 5 is not the only possible way to obtain privacy-preserving importance weights. As mentioned earlier, the approach to privacy-preserving logistic regression of [8] could be applied also. Other

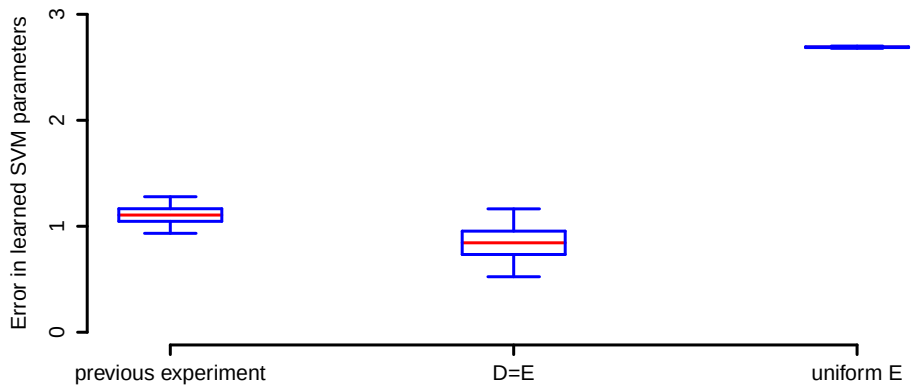


Figure 4.6. Performance of the importance weighting algorithm in extreme cases, for training an SVM. The smaller the distance is, the better. The norm of the true SVM parameter vector is about 7, so the algorithm provides useful information in all three cases.

methods of estimating well-calibrated conditional probabilities [67, 34, 44] can be used also, if modified to guarantee differential privacy.

The theory of importance weighting says that the closer the two distributions p_D and p_E are, the better the estimates based on E are. Thus, not surprisingly, the more similar the distribution of E is to that of D, the better. However, the experiments above use sets D and E with quite different distributions, and results are still good. Specifically, the set D is 90% male, while the set E is 90% female.

An obvious issue is where the public dataset E can come from. This question has no universal answer, but it does have several possible answers. First, E may be synthetic. The experiments section shows that even if E is uniformly drawn from the sample space, the importance algorithm can still provide useful output. Second, E may be the result of a previous breach of privacy. Any such event is regrettable, but if it does happen, using E as suggested above does not worsen the breach. Third, E may be a subset of examples from the original dataset for which privacy is not a concern. In a medical

scenario, E may contain the records of volunteers who have agreed to let their data be used for scientific benefit. In the U.S., laws on the privacy of health information are less restrictive when a patient is deceased, and such records have already been released for research by some hospitals.

Another issue is how to define E if more than one public dataset is available. If we know which public dataset was sampled from a distribution most similar to that of the private dataset D, then is is natural to select that dataset as E. Otherwise, in particular if all the public datasets follow the same distribution or if their distributions are unknown, then it is natural to take their union as E. However, if the public datasets follow varying distributions, then logistic regression is likely to be mis-specified for representing the contrast between D and the union of the public datasets, so it can be preferable to select just one of these datasets, for example the one with highest cardinality.

The schemas of D and E may be different. In this case, only the features that appear in both datasets can be used. However, if prior knowledge is available, disparate features can be used after pre-processing. For example, D may include patients' diseases, while E records patients' medications. If a probabilistic model relating diseases and medications is known, and this model is independent of the datasets D and E, then the two features can still contribute to the ratio of probability densities.

The usefulness of the method proposed in this paper is not restricted to medical domains. For example, consider a social network such as Facebook or Linkedin, and an advertiser such as Toyota. Let the profiles of all users be the dataset D. For privacy reasons, the network cannot give the advertiser direct access to D. However, suppose that some users have opted-in to allowing the advertiser access to their profiles. The profiles of these users can be the dataset E. The social network can compute privacy-protecting weights that make the dataset E reflect the entire population D, and let the advertiser use these weights. Note that both in medical and other domains, an advantage

of the importance weighting method is that all analysis is performed on genuine data, that is on the records of E. In contrast, other data-publishing methods require analyses to be done on synthetic or perturbed data.

4.2 Select and Label Algorithm

In this section, we aim at developing a task-driven privacy-preserving data release algorithm. Task-driven here means the published data specially targets at future supervised learning tasks, though other applications are also possible. The basic idea is to select a set of points E from the predictor space based on a non-private [2] dataset D, so that E includes the most critical points to the supervised learning task, followed by labeling (re-weighting) E by their relative importance determined by the private data. Figure 4.7 illustrates a high level overview for the S&L algorithm, while the four bold arrows represent the S&L iterations. The process of iteratively selecting and labeling is the same for both regression and classification.

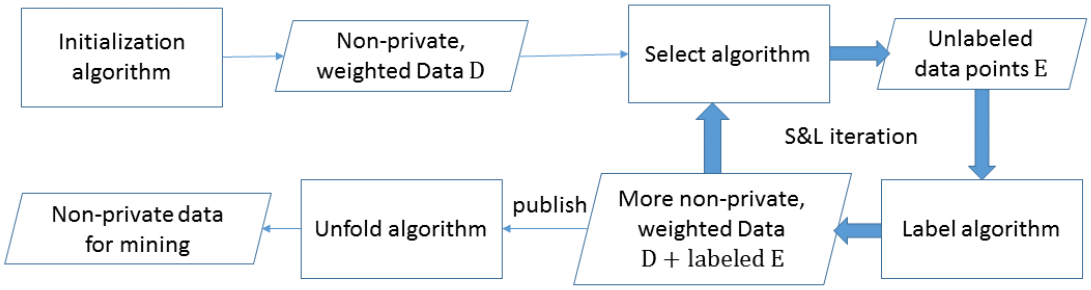


Figure 4.7. Overview of the S&L algorithm. Both classification and regression have the same 'select and label' iteration.

The contributions of our work are:

[2] We need to clarify the meaning of word *non-private*. A non-private model is a traditional machine learning model that does not take privacy into account. A non-private data set can be one of the following: a data set that is randomly drawn from the sample space, and independent of any private data; a synthetic data set generated by a differentially private algorithm. Arbitrary use of these non-private data sets do not leak to any privacy risk.

1. S&L is a task-driven differential private data synthesization algorithm that reduces unnecessary randomness by other general purpose privacy-preserving data release methods.

2. S&L releases data for all supervised learning algorithms, therefore there are no limitations on which learning models can be used.

3. S&L satisfies ε-differential privacy, which is the stronger version of differential privacy.

Though classification and regression have the same iterations, their difference in nature determines that different algorithms have to be designed for them. In the following sections, I will introduce algorithms for them separately.

4.2.1 Methodology for classification

I will introduce S&L algorithm for classification in this section. I first introduce initialization step, then select and label steps separately.

Basic Idea

Our basic idea is to use private data to generate new samples aimed at improving model performance. This idea is similar to active learning. In active learning, we start from a dataset $\hat{D}$ and iteratively select (in our case, synthesize) a set from the sample space based on $\hat{D}$, ask for their labels, and add the selected points to $\hat{D}$. In S&L, we first initialize a non-private dataset E, then iteratively select a set of points from the sample space and use the private data D to label the selected points and add them to E. As the algorithm mainly consists of two steps: select and label, we name the algorithm S&L. An overview of these steps for classification tasks are as follows.

To initialize, we try to find some points that can represent as many samples in the private dataset as possible. Besides, our implementation of select and label steps below

work on the convex hull of initial datasets, therefore we want the initial set to include vertices of the sample space of predictors so more area could be covered by this convex hull, or the space in which our algorithm works. To combine the two ideas together, we want to pick vertices with the most samples around to form the initial set E. The samples in E are labeled with the label step introduced below.

In the select step, we want to pick points from the space of predictors that can improve model performance the most. By intuition, for classification, the points around the boundary are the most important, and if a positive and a negative samples are linked, the line linking them must cross the boundary. If the two samples are very far away, then it would be difficult to find the boundary; but if they are close to each other, the middle point of them should not be far away from the boundary. Therefore, in *select* step, for each sample in the non-private dataset E, we search for the closest sample with the opposite label. The middle point between the sample and the nearest opposite is selected as by intuition it is close to the classification boundary. Note that the select step does not use private data D, thus it does not have any privacy concerns (i.e., no privacy budget consumption).

In *label* step, the unlabeled points selected in the select step are labeled and weighted using private data D. Then, the non-private data set is updated to include these new samples. Our only assumption is that closer points have larger probability to have the same label, therefore, a nearest neighbor method is used.

In the rest of this section, I will walk you through the details of S&L implementation.

Notation

There are two datasets in our algorithm, one private dataset $D = \{(x_i, y_i)\}$ and a non-private dataset we iteratively expand, $E = \{(x_i, w_{i+}, w_{i-})\}$. x_i in both sets means

a predictor, while $y_i \in \{-1, 1\}$ in D means a binary label. By intuition, w_{i+} means the number of positive samples in D whose nearest neighbors in E is x_i, and w_{i-} means the number of negative samples. When E is released for learning, and learning algorithms to use can run on weighted samples, then we turn a tuple (x_i, w_{i+}, w_{i-}) into two weighted samples $(x_i, 1)$ with weight w_{i+} and $(x_i, -1)$ with weight w_{i-}. If learning algorithms to use cannot take weighted samples as input, we can round w_{i+} and w_{i-} to nearest integers and copy $(x_i, 1)$ w_{i+} times and $(x_i, -1)$ w_{i-} times.

Unless specified, same subscript in different sets does not refer to the same sample, e.g., there is no relation between x_i in D and E.

When we talk about *samples*, we mean a tuple (x, y). When we talk about *points*, we mean x only.

Implementation of Initialization Step

As we discussed above, we want the initial set satisfying two conditions:

1. Close to many private samples.

2. Be vertices of sample spaces.

The detailed algorithm is in Algorithm 6. It ensures the initial set includes at most M samples. According to the number of vertices, we may have three ways to initialize E. If the number of vertices is smaller than M, then we simply add all vertices to E without consuming any privacy budget. If the number of vertices is larger than M, but not a huge number, then we count how many private samples are around each vertex in a differentially private way, and select those vertices with most private samples around to form E. If the number of vertices is really large, then we randomly draw, say, 1000 vertices from all vertices, and select M vertices with most private samples around. Usually we set $M = 10$. The first case does not consumes differential privacy at all,

Algorithm 6. Initialization Step for Classification

Require: Private data D, privacy budget ε, the upper bound of number of samples to select in this step, M.

Ensure: Initial set E.

1: If the number of vertices of the sample space is smaller than M, go to Step 2; if the number of vertices of the sample space is larger than M but smaller than 1000, go to Step 3. Otherwise go to Step 4.
2: Set E to all the vertices of predictors' sample space with $w_{i+} = w_{i-} = 0$ for these vertices, and return E.
3: Let S be the set of all vertices of predictors' sample space and go to Step 5.
4: Let S be the set of 1000 random vertices of predictors' sample space and go to Step 5.
5: For each point in S, count how many samples in D take it as nearest neighbor in S.
6: Add i.i.d. noise from $Lap(2/\varepsilon)$ to each count.
7: Let E be the M vertices in S with highest count, set their $w_{i+} = w_{i-} = 0$, and return E.

while the later two cases consume ε.

Implementation of Select Step

The selection rule should have the following properties:

1. Selects points around classification boundary.

2. The number of selected points should not be too large as too many points introduce too much noise in the label step, which results in too much randomness in the released data.

3. The number of selected points should not be too small (say only 1) as too few samples in E means worse coverage of samples in D.

Detailed implementation is in Algorithm 7.

At this step, no information from private data is used. Therefore, this step does not consume any privacy budget.

Algorithm 7. Select Step for Classification

Require: Non-private dataset E.
Ensure: Updated set E.
1: Let $E' = \emptyset$.
2: For each positive(negative) sample in E, find its nearest negative(positive) sample in E and add the middle point of the two to $E' = \{x_i\}$.
3: Label E' with $w_{i+} = w_{i-} = 0$.
4: Add E' to E.
5: Merge duplicated x in E: $(x_i, w_{i+}, w_{i-}), (x_j, w_{j+}, w_{j-}) \to (x_i, w_{i+} + w_{j+}, w_{i-} + w_{j-})$ if $x_i = x_j$.
6: Return E.

Algorithm 8. Label Step for Classification

Require: Current non-private data E, private data D, and privacy budget ε.
Ensure: Updated w_{i+} and w_{i-} in E.
1: For each sample (x_i, y_i) in D, find its nearest neighbor (x_j, w_{j+}, w_{j-}) in E according to the distance of x_i and x_j.
2: If $y_i = 1$, then update $w_{j+} = w_{j+} + 1$; otherwise update $w_{j-} = w_{j-} + 1$.
3: After iterating over all samples in D, add i.i.d. noise from $Lap(2/\varepsilon)$ to each w_{j+} and w_{j-}.
4: Return updated E.

Implementation of Label Step

We hope the label step has the following properties:

1. We want both weights and labels of a sample, as most loss functions in learning process not only depend on labels, but also on weights.

2. To make as few constraints to learning algorithms as possible, the label step should not depend on too many or demanding assumptions.

By only assuming that closer points tends to have closer labels, we design Algorithm 8. In this algorithm, weights from the latest iterations are added to weights from previous iterations, rather than overwriting them. This is a reuse of previous private information, which leads to smaller variance as well as larger bias.

Algorithm 9. S&L Algorithm for Classification

Require: A private data set D, privacy budget ε, number of iterations T, the upper bound of number of samples to select in this step, M.

Ensure: A synthetic, non-private dataset E for classification.

1: Initialize E with Algorithm 6, D and $\varepsilon/(T+2)$.
2: If Algorithm 6 consumes privacy budget, set $\varepsilon' = \varepsilon/(T+2)$; otherwise set $\varepsilon' = \varepsilon/(T+1)$.
3: Call Algorithm 8 to label points in E with privacy budget ε'.
4: Repeat the following two steps for T times.
5: Call Algorithm 7 to insert new points to E.
6: Call Algorithm 8 to label points in E with privacy budget ε'.
7: Release E.

Full Algorithm

The Algorithm 9 introduces the S&L algorithm. It first generates an initial set, then iteratively executes select and label steps.

4.2.2 Methodology for Regression

The two largest differences between S&L on regression and S&L on classification are:

1. Since we cannot represent weights of samples by counts of positive and negative samples, we replace tuples (x_i, w_{i+}, w_{i-}) in E with $(x_i, w_{i0}, w_{i1}, w_{i2})$. The first element w_{i0} means how many points in D take x_i as nearest neighbor in E, while w_{i1} and w_{i2} are sums of y and y^2 for these points separately.

2. The points that improve current model most are no longer the points around boundary, as there is no boundary in regression. Instead, the points to select should be those with largest uncertainty.

Besides, we assume that all y are between 0 and 1. If y is not, we can either do a linear transformation or truncation to enforce this assumption.

Algorithm 10. Initialization Step for Regression

Require: Private data D, privacy budget ε, the upper bound of number of samples to select in this step, M.

Ensure: Initial set E.

1: If the number of vertices of the sample space is smaller than M, go to Step 2; if the number of vertices of the sample space is larger than M but smaller than 1000, go to Step 3. Otherwise go to Step 4.
2: Set E to all the vertices of predictors' sample space with $w_{i0} = w_{i1} = w_{i2} = 0$ for these vertices, and return E.
3: Let S be the set of all vertices of predictors' sample space and go to Step 5.
4: Let S be he set of 1000 random vertices of predictors' sample space and go to Step 5.
5: For each point in S, count how many samples in D take it as nearest neighbor in S.
6: Add i.i.d. noise from $Lap(2/\varepsilon)$ to each count.
7: Let E be the M vertices in S with highest count, set their $w_{i0} = w_{i1} = w_{i2} = 0$, and return E.

Compared to the method for classification, the overall algorithm is the same. Therefore, it is not repeated again here. However, one additional algorithm is required to turn $E = \{(x_i, w_{i0}, w_{i1}, w_{i2})\}$ into a dataset that other learning algorithms can use. Therefore, we give out an unfold algorithm here. Not only is it used before releasing E, but also it is called during select step.

Implementation of Initialization Step

The initialization step is very similar to that for classification. The only difference is that we change w_{i+} and w_{i-} to w_{i0}, w_{i1}, w_{i2}. There are still three cases and the first one does not consume privacy budget.

Unfold Algorithm

Different from classification tasks, we now use tuples like $(x_i, w_{i0}, w_{i1}, w_{i2})$ to represent a weighted sample. However, such tuples cannot be fed into any existing algorithm directly. Therefore, we use the following unfold algorithm (Algorithm 11) to generate regular samples for learning algorithms. For each tuple $(x_i, w_{i0}, w_{i1}, w_{i2})$, this

Algorithm 11. Unfold Algorithm for Regression

Require: Non-private dataset E.
Ensure: Unfolded set E'', which can be used by other regression algorithms.
1: Set $E'' = \emptyset$.
2: For each tuple $(x_i, w_{i0}, w_{i1}, w_{i2})$ in E, do the following.
3: Round w_{i0} to nearest non-negative integers.
4: Compute $E[y_i] = w_{i1}/w_{i0}$ and $Var[y_i] = \max\{w_{i2}/w_{i0} - E^2[y_i], 10^{-4}\}$.
5: Insert w_{i0} samples (x_i, y_{ij}), while $j = 1, ..., w_{i0}$ into E''. Those y_{ij} are i.i.d. drawn from normal distribution $N(E[y_i], Var[y_i])$.
6: Return E'' after iterating over all tuples in E.

algorithm tries to generate w_{i0} samples whose predictors are x_i and whose responses are from the normal distribution $N(w_{i1}/w_{i0}, w_{i2}/w_{i0} - (w_{i1}/w_{i0})^2)$. Therefore, the generated samples correspond to $(x_i, w_{i0}, w_{i1}, w_{i2})$ approximately. The generated samples are then used in Algorithm 12.

In Step 4, we lower bound $Var[y_i]$ by 10^{-4} because the random noise we add may lead to $w_{i2}/w_{i0} - E^2[y_i] < 0$. We use this smoothing to avoid such cases.

Implementation of Select Step

In this select step, we first build a candidate set, then select points from the candidate set with largest uncertainty. The candidate set consists of middle points of close pairs, and the computation of uncertainty is based on the Gaussian process regression model. Detailed implementation is in Algorithm 12. This step is still privacy-free.

In Step 5 of Algorithm 12, we only select half of points from E' with largest variances. This is due to the intuition behind the select step: we want to add points with largest uncertainty, which in regression means largest variances. Therefore, points that have small variances are removed from E'.

Algorithm 12. Select Step for Regression

Require: Non-private dataset E.
Ensure: Updated set E.
1: Let $E' = \emptyset$.
2: For each point in E, find its nearest neighbor in E and add the middle point of the two to $E' = \{x_i\}$.
3: Unfold E by Algorithm 11 to get a dataset E''.
4: Train a Gaussian Process model based on E'', then use it to find variance of each point in E'.
5: Select half of points from E' with largest variances, and add them to E with $w_{i0} = 0, w_{i1} = 0, w_{i2} = 0$.
6: Merge duplicated x in E: $(x_i, w_{i0}, w_{i1}, w_{i2}), (x_j, w_{j0}, w_{j1}, w_{j2}) \to (x_i, w_{i0} + w_{j0}, w_{i1} + w_{j1}, w_{i2} + w_{j2})$ if $x_i = x_j$.
7: Return E.

Algorithm 13. Label step for Regression

Require: Current non-private data E, private data D, and privacy budget ε.
Ensure: Updated w_{i0}, w_{i1}, w_{i2} in E.
1: For each sample (x_i, y_i) in D, find its nearest neighbor $(x_j, w_{j0}, w_{j1}, w_{j2})$ in E according to the distance of x_i and x_j.
2: Update $w_{j0} = w_{j0} + 1, w_{j1} = w_{j1} + y_i, w_{j2} = w_{j2} + y_i^2$.
3: After iterating over all samples in D, add i.i.d. noise from $Lap(6/\varepsilon)$ to each w_{j0}, w_{j1}, w_{j2}.
4: Return updated E.

Implementation of Label Step

The difference between label step in classification and regression results from different w in E. The other part of this step is generally the same.

Full Algorithm

The full algorithm for regression is generally the same as the one for classification (Algorithm 9), therefore we choose not to repeat it here. The only difference is in the last step, E must be unfolded by Algorithm 11 before released.

4.2.3 Experiments

In this section, we examine performance of S&L's output on both classification and regression tasks. The settings for experiments on classification are:

Models: given S&L outputs, we train a linear classifier (regularized logistic regression) and a non-linear one (kernel SVM).

Baselines: we use existing differetially private versions of the models above, in [8] and [51] respectively.

Performance measurement: we use area under curve (AUC) to measure performance of classifiers. The AUC of a classifier is a number between 0 and 1, and larger AUC means better performance. A random trivial classifier gets AUC=0.5, while a perfect classifier gets AUC=1. The reported AUC is averaged over 100 repeated experiments.

Dataset: we use complete samples in the Adult data set [40] as processed in the R package arules. The dataset has 14 predictors (including age, gender, etc.) and 1 label (when his/her income is more than 50K per year) originally, and after turning predictors to dummy variables, there are 64 predictors in total. The data set has two subsets, one training set and one test set. The training set has 30162 samples while the test set has 15060 samples. We apply S&L to the training set and then train several models based on the output. The model is then evaluated on the test set. The whole process is repeated 100 times.

As for regression task, the settings are:

Models: given S&L outputs, we train a linear regression model (ridge regression) and a non-linear one (nu-SVR).

Baselines: we use existing differetially private versions of ridge regression, one based on [8] and the other on [68] respectively.

Performance measurement: we use mean square error (MSE) and correlation between predicts and true response to measure performance of classifiers. The MSE here is divided by the variance of response, ensuring that a trivial model, which returns training average for all test samples, has MSE=1. Better models correspond to smaller MSE and larger correlation. Again, the average of 100 experiments is reported.

Dataset: we use the Brazil census dataset (Minnesota Population Center, Integrated public use microdata seriesinternational: Version 5.0., 2009, which has 14 features. We use the features such as age, gender to predict a feature income. As it consists of only one set, 10 cross validation is applied to split training and test sets.

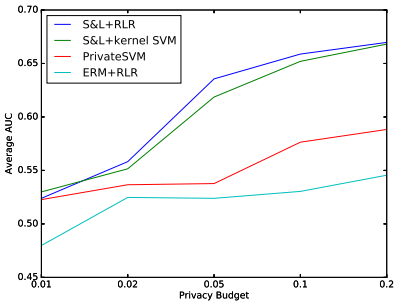
The comparison results are shown in Figure 4.8(a). S&L is better than the objective perturbation algorithm regardless of the training model.

The comparison results are shown in Figure 4.8(b) and Figure 4.8(c). S&L is better than the baselines regardless of the training model.

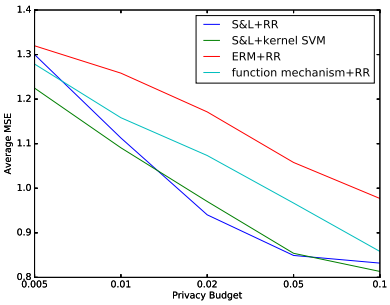
To conclude, the S&L algorithm outperforms baseline models in both classification and regression tasks.

4.2.4 Discussion

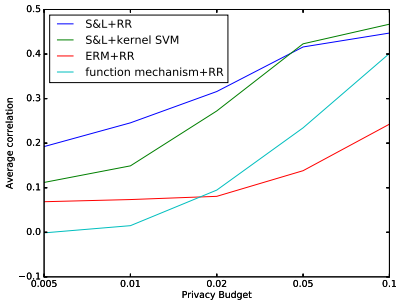
The S&L algorithm can be improved and simplified, if there are public data available. We can use the public data as the initial set and skip the initialization step. However, it requires that the distributions of predictors in the public data and private



(a) Average AUC in classification task given different privacy budgets. Larger is better.



(b) Average MSE in regression task given different privacy budgets. To make it easier to understand, we divide the MSE by variance of y. Smaller is better.



(c) Average correlation in regression task given different privacy budgets. Larger is better.

Figure 4.8. Performance given different privacy budgets.

data to be similar, otherwise, the selected points cannot be labeled well, and the private information cannot be used well either.

Section 4.1 comes from the paper *Differential Privacy Based on Importance Weighting.* Zhanglong Ji, Charles Elkan. Machine Learning, June 2013. The dissertation author was the primary author of the paper.

Section 4.2 comes from the paper *Select and Label (S&L): a Task-Driven Privacy-Preserving Data Synthesization Algorithm.* Zhanglong Ji, Xiaoqian Jiang, Haoran Li, Li Xiong, Lucila Ohno-Machado, which is in submission right now. The dissertation author was the primary author of the paper.

Bibliography

[1] Apple Previews iOS 10, the Biggest iOS Release Ever. https://www.apple.com/pr/library/2016/06/13Apple-Previews-iOS-10-The-Biggest-iOS-Release-Ever.html.

[2] Learning Statistics with Privacy, Aided by the Flip of a Coin. https://security.googleblog.com/2014/10/learning-statistics-with-privacy-aided.html.

[3] NCBI Retiring HapMap Resource. https://www.ncbi.nlm.nih.gov/variation/news/NCBI_retiring_HapMap/.

[4] Tackling Urban Mobility with Technology. https://europe.googleblog.com/2015/11/tackling-urban-mobility-with-technology.html.

[5] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, Accuracy, and Consistency too: a Holistic Solution to Contingency Table Release. In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 273–282, 2007.

[6] Avrim Blum, Katrina Ligett, and Aaron Roth. A Learning Theory Approach to Non-interactive Database Privacy. In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 609–618, 2008.

[7] Kamalika Chaudhuri and Claire Monteleoni. Privacy-Preserving Logistic Regression. In *NIPS*, pages 289–296, 2008.

[8] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially Private Empirical Risk Minimization. In *Journal of Machine Learning Research*, pages 1069–1109, 2011.

[9] Kamalika Chaudhuri, Anand D Sarwate, and Kaushik Sinha. Near-optimal Differentially Private Principal Components. In *NIPS*, pages 998–1006, 2012.

[10] Rui Chen, Noman Mohammed, Benjamin C M Fung, Bipin C Desai, and Li Xiong. Publishing Set-Valued Data via Differential Privacy. In *PVLDB*, pages 1087–1098, 2011.

[11] Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Thanh T L Tran. Differentially Private Summaries for Sparse Data. In *ICDT*, pages 299–311, 2012.

[12] Anindya De. Lower Bounds in Differential Privacy. In *Theory of Cryptography*, pages 321–338, 2012.

[13] Rick Durrett. Probability: Theory and Examples. 2013.

[14] Cynthia Dwork. Differential Privacy. In *Encyclopedia of Cryptography and Security (2nd Ed.)*, pages 338–340, 2011.

[15] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our Data, Ourselves: Privacy via Distributed Noise Generation. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006.

[16] Cynthia Dwork and Jing Lei. Differential Privacy and Robust Statistics. In *STOC*, pages 371–380, 2009.

[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, pages 265–284, 2006.

[18] Cynthia Dwork, Guy N Rothblum, and Salil P Vadhan. Boosting and Differential Privacy. In *FOCS*, pages 51–60, 2010.

[19] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least Angle Regression. *The Annals of statistics*, 32(2):407–499, 2004.

[20] Charles Elkan and Keith Noto. Learning Classifiers from only Positive and Unlabeled Data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM, 2008.

[21] 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.

[22] Chengfang Fang and Ee-Chien Chang. Adaptive Differentially Private Histogram of Low-Dimensional Data. In *Privacy Enhancing Technologies*, pages 160–179, 2012.

[23] Arik Friedman and Assaf Schuster. Data Mining with Differential Privacy. In *KDD*, pages 493–502, 2010.

[24] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition Attacks and Auxiliary Information in Data Privacy. In *KDD*, pages 265–273, 2008.

[25] Michael Hahsler, Christian Buchta, Bettina Gruen, and Kurt Hornik. *arules: Mining Association Rules and Frequent Itemsets.*, 2017. R package version 1.5-2.

[26] Moritz Hardt, Katrina Ligett, and Frank McSherry. A Simple and Practical Algorithm for Differentially Private Data Release. In Peter L. Bartlett, Fernando C N Pereira, Christopher J C Burges, Léon Bottou, and Kilian Q Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 2348–2356, 2012.

[27] Moritz Hardt and Aaron Roth. Beyond Worst-Case Analysis in Private Singular Vector Computation. In *Computing Research Repository*, 2012.

[28] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures using High-density SNP Genotyping Microarrays. *PLoS Genet*, 4(8):e1000167, 2008.

[29] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N. Wright. A Practical Differentially Private Random Decision Tree Classifier. In *International Conference on Data Mining Workshops*, pages 114–121, 2009.

[30] Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially Private Online Learning. In *COLT*, pages 24.1–24.34, 2012.

[31] Prateek Jain and Abhradeep Thakurta. Differentially Private Learning with Kernels. In *ICML (3)*, pages 118–126, 2013.

[32] Aaron Johnson and Vitaly Shmatikov. Privacy-preserving Data Exploration in Genome-wide Association Studies. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1079–1087. ACM, 2013.

[33] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a Freely Accessible Critical Care Database. *Scientific data*, 3, 2016.

[34] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A Least-squares Approach to Direct Importance Estimation. *Journal of Machine Learning Research*, 10(Jul):1391–1445, 2009.

[35] Michael Kapralov and Kunal Talwar. On Differentially Private Low Rank Approximation. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1395–1414, 2013.

[36] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private Convex Empirical Risk Minimization and High-dimensional Regression. *Journal of Machine Learning Research*, 1:41, 2012.

[37] Jing Lei. Differentially Private M-Estimators. In *Advances in Neural Information Processing Systems*, pages 361–369, 2011.

[38] Tal Levin-Decanini, Nell Maltman, Sunday M Francis, Steve Guter, George M Anderson, Edwin H Cook, and Suma Jacob. Parental Broader Autism Subphenotypes in ASD Affected Families: Relationship to Gender, Child's Symptoms, SSRI Treatment, and Platelet Serotonin,. *Autism Research*, 6(6):621–630, 2013.

[39] Yang D Li, Zhenjie Zhang, Marianne Winslett, and Yin Yang. Compressive Mechanism: Utilizing Sparse Representation in Differential Privacy. In *WPES*, pages 177–182, 2011.

[40] M. Lichman. UCI Machine Learning Repository., 2013.

[41] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *ICDE*, page 24, 2006.

[42] Frank McSherry. Privacy Integrated Queries: an Extensible Platform for Privacy-Preserving Data Analysis. In *SIGMOD*, pages 19–30, 2009.

[43] Frank McSherry and Kunal Talwar. Mechanism Design via Differential Privacy. In *FOCS*, pages 94–103, 2007.

[44] Aditya Krishna Menon, Xiaoqian J Jiang, Shankar Vembu, Charles Elkan, and Lucila Ohno-Machado. Predicting Accurate Probabilities with a Ranking Loss. In *Proceedings of the... International Conference on Machine Learning. International Conference on Machine Learning*, volume 2012, page 703. NIH Public Access, 2012.

[45] Darakhshan J Mir. Differentially-Private Learning and Information Theory. In *EDBT/ICDT Workshops*, pages 206–210, 2012.

[46] Darakhshan J. Mir and Rebecca N. Wright. A Differentially Private Graph Estimator. In *ICDM Workshops*, pages 122–129, 2009.

[47] Noman Mohammed, Rui Chen, Benjamin C M Fung, and Philip S Yu. Differentially Private Data Release for Data Mining. In *KDD*, pages 493–501, 2011.

[48] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth Sensitivity and Sampling in Private Data Analysis. In *STOC*, pages 75–84, 2007.

[49] Sewoong Oh and Pramod Viswanath. The Composition Theorem for Differential Privacy. In *Computing Research Repository*, 2013.

[50] Jurg Ott, Yoichiro Kamatani, and Mark Lathrop. Family-based Designs for Genome-wide Association Studies. *Nature Reviews Genetics*, 12(7):465–474, 2011.

[51] Benjamin I P Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. Learning in a Large Function Space: Privacy-Preserving Mechanisms for SVM Learning. 2009.

[52] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Sharing Graphs using Differentially Private Graph Models. In *Internet Measurement Conference*, pages 81–98, 2011.

[53] David W Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization.* John Wiley & Sons, 2015.

[54] Entong Shen and Ting Yu. Mining Frequent Graph Patterns with Differential Privacy. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 545–553. ACM, 2013.

[55] Andrew Smith and Charles Elkan. A Bayesian Network Framework for Reject Inference. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 286–295. ACM, 2004.

[56] Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. pages 557–570, 2002.

[57] Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Nearly Optimal Private LASSO. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3025–3033. Curran Associates, Inc., 2015.

[58] Abhradeep Thakurta and Adam Smith. Differentially Private Feature Selection via Stability Arguments, and the Robustness of the Lasso. In *COLT*, pages 819–850, 2013.

[59] Caroline Uhlerop, Aleksandra Slavković, and Stephen E Fienberg. Privacy-preserving Data Sharing for Genome-wide Association Studies. *The Journal of privacy and confidentiality*, 5(1):137, 2013.

[60] Jaideep Vaidya, Basit Shafiq, Anirban Basu, and Yuan Hong. Differentially Private Naive Bayes Classification. In *Web Intelligence*, pages 571–576, 2013.

[61] Staal A. Vinterbo. Differentially Private Projected Histograms: Construction and Use for Prediction. In *European Conference on Machine Learning (ECML) and Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 19–34, 2012.

[62] Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially Private Network Data Release via Structural Inference. In *KDD*, pages 911–920, 2014.

[63] Yonghui Xiao, Li Xiong, and Chun Yuan. Differentially Private Data Release through Multidimensional Partitioning. In *Secure Data Management*, pages 150–168, 2010.

[64] Zhenxing Yang, Yu Xu, Hongrong Luo, Xiaohong Ma, Qiang Wang, Yingcheng Wang, Wei Deng, Tao Jiang, Guangqing Sun, Tingting He, Jingchu Hu, Yingrui Li, Jun Wang, Tao Li, and Xun Hu. Whole-exome Sequencing for the Identification of Susceptibility Genes of Kashin-Beck Disease. *PloS one*, 9(4):e92298, 2014.

[65] Fei Yu, Stephen E Fienberg, Aleksandra B Slavković, and Caroline Uhler. Scalable Privacy-preserving Data Sharing Methodology for Genome-wide Association Studies. *Journal of biomedical informatics*, 50:133–141, 2014.

[66] Fei Yu and Zhanglong Ji. Scalable Privacy-preserving Data Sharing Methodology for Genome-wide Association Studies: an Application to iDASH Healthcare Privacy Protection Challenge. *BMC Med. Inf. & Decision Making*, 14(S-1):S3, 2014.

[67] Bianca Zadrozny and Charles Elkan. Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers. In *ICML*, volume 1, pages 609–616. Citeseer, 2001.

[68] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. Functional Mechanism: Regression Analysis under Differential Privacy. In *PVLDB*, pages 1364–1375, 2012.

[69] Xiaojian Zhang, Xiaofeng Meng, and Rui Chen. Differentially Private Set-Valued Data Release against Incremental Updates. In *DASFAA (1)*, pages 392–406, 2013.