A Study of Stability in Data Privacy

by

Xi Wu

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2016

Date of final oral examination: 07/11/2016

The dissertation is approved by the following members of the Final Oral Committee:

Somesh Jha, Professor, Computer Sciences
Jeffrey F. Naughton, Google
Anhai Doan, Professor, Computer Sciences
C. David Page Jr., Professor, Biostatistics and Medical Informatics
Stphen J. Wright, Computer Sciences

To my wife Zichen, daughter Naomi and my parents.

ACKNOWLEDGMENTS

I would not have been able to complete this dissertation without the guidance and encouragement of Professor Jeffrey Naughton. Jeff is always patient and supportive and perhaps most importantly, he taught me a lesson that would probably benefit my entire life: *"Always remember how you reached there."* It is by retrospection and learning from it, either it is now a success or failure, that one can grow and perhaps seizes opportunities.

I am also indebted to my coadvisor Professor Somesh Jha, who has provided me invaluable guidance in identifying research projects and connecting me to the right people. Somesh is very generous in helping his students, and especially in sharing his philosophy on research and life. Having discussions with him is purely enlightening. I owe him my deep gratitude.

It is very fortunate for me to be able to talk to and work with several prominent professors: Professor Dieter van Melkebeek taught me a lot of theory of computing in the first two years of my PhD study, laying a strong foundation for my research. Professor Kamalika Chadhrui (from UCSD) taught me a lot of machine learning and differential privacy. I had some illuminating and insightful discussions with Professor Jin-yi Cai and Professor Stephen J. Wright. Finally, thanks to Anhai Doan and David Page for sitting in my defense committee and provided me extensive and valuable feedback.

Over the years I have also been fortunate to work with a group of prominent students at UW-Madison, and I benefit significantly from talking to them. A partial list (listed alphabetically, and apologies ahead if I did not mention your name here) includes Matthew Anderson, Siddarth Barman, Matthew, Fredrikson, Xixuan Feng, Heng Guo, Yeye He, Arun Kumar, Jiexing Li, Yinan Li, Conghan Lim, Lanyue Lu, Xiang Peng, David Malec, Gautam Prakriya, Linhai Song, Sanketh Nalli, William Umboh,

Wentao Wu, Tyson Williams, Ce Zhang. Special thanks goes to Wentao, with whom we shared an office, had many intensive and exciting discussions, and wrote papers together.

Last but not least, I would like to thank my family. My wife Zichen Qiu has provided me love and support throughout all these years, without which I do not think I can finally arrive at this point. Also, whenever I felt frustrated, my daughter Naomi can always manage bringing me out of it, to whom I give special thanks here. My parents (Zhenmin Wu and Meng Wan) and aunt (Zhi Wan) have also always supported me all these years, though I am thousands of miles away from them. I love you all.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This thesis is about *stability*. Technically, it is about a tradeoff between computing something interesting and keeping the output "stable." More philosophically, it is about a life tradeoff that I have been struggling to balance – doing something exciting yet maintaining a "stable" life. Since life is difficult, it suggests that the technical tradeoff that I am facing must be challenging as well.

More precisely, this thesis looks at the tradeoff from the angle of data privacy, where one wants to compute interesting statistics from the data collected from a group of individuals, while protecting the "privacy" of each participant. This problem has attracted tremendous attention in both academia and industry, as the rapidly growing capability of collecting and analyzing individual data has brought significant concerns about privacy.

As "privacy" is inherently social and personal, it is not surprising that there are many different data privacy notions, and probably there will not be an end in defining new ones. A *thesis* we put forth is that, though different as they may appear, *data privacy notions are essentially all about stability (though possibly different forms of it).* In fact, the concept of stability comes in so naturally that somehow it has to be true – the "sensitive" data of a participant should not cause abrupt changes to the statistics computed, otherwise his sensitive data is somehow "leaked" and his privacy is broken.

We support this thesis by studying data privacy in the setting of machine learning. In the first part, we study *differentially private stochastic gradient descent*. Differential privacy has become the gold standard for protecting data privacy since the last decade, and its core is about stability of an algorithm. Stochastic gradient descent (SGD), on the other hand, is a fundamental optimization algorithm that powers various machine learning tasks. In this part, we improve the state-of-the-art differentially

private SGD: Not only our algorithms yield substantially better test accuracy, but also they are easier to implement and run faster. In fact, the analysis of our new algorithms is also simple and modular. All of these improvements come from a novel analysis about the "global" stability of SGD, in contrast to previous work which analyze the "local" stability of each update and compose them at then end. Our analysis leverages well-known expansion properties of gradient updates.

In the second part of this thesis *we move from the core differential privacy to its boundary*. We note that an intriguing work by Fredrikson et al. (USENIX Security 2014) studied *model inversion attacks*, which try to back out sensitive data of an individual using a machine learning model. To counter these attacks, the authors used, naturally, differential privacy. We, however, demonstrate that differential privacy is orthogonal to model inversion attacks. In fact, a simple thought experiment already reveals that differential privacy is not even designed to protect model inversion attacks. We go on to quantify this "orthogonality:" By studying the *stability of empirical risk minimization*, on one hand we significantly improve the privacy-utility tradeoff compared to the functional mechanism, which is used to train regression models by Fredrikson et al. On the other hand, we quantitatively show that as the privacy-utility tradeoff improves, the models become more susceptible to model inversion attacks.

Finally, we study model inversion attacks. Interestingly as it turns out, the additional invertibility from being able to access a model originates from a different form of stability – the "stability" of a model with respect to "sensitive features." Note that this supports our thesis again: Since machine learning models are statistical, the predictions made by evaluating these models (as functions) are also "statistical;" and abrupt change of the statistical predictions with respect to change of the sensitive features indicates privacy losses, which are "model inversions." More

technically, we connect model inversion attacks to the influence theory[1] in the analysis of Boolean functions. Our study also unveils intriguing new phenomena and questions.

[1] Note that the influence of an object is about the "instability" of the object.

1 INTRODUCTION

The first antonym of "stability:" insecurity.

— Merriam-Webster Thesaurus

The ability to collect and process massive amounts of data has significantly transformed our society. Large-scale data analyses, ranging from computing simple statistics to constructing highly sophisticated machine learning models, are widely applied to extract valuable patterns from data in domains such as personalized medicine, finance, web search histories and social networks. For these domains, there has also been an ever growing concern about individuals' privacy. A booming scientific field – *data privacy* – whose goal is to provide methodologies, models and techniques to release useful information while preserving individuals' privacy, has attracted significant interest in both academia and industry.

A basic setting of studying data privacy is the following: Let $S = \{z_1, \ldots, z_m\}$ be a dataset where each z_i is collected from an individual and f be a function one wants to compute. The goal of data privacy is to design a mechanism $\mathcal{M}$ that computes $f(S)$ "accurately," while "protecting privacy" of the participants. A natural and important question now is: *What do we mean by "protecting privacy?"*

One of the most important achievements in answering this question in the past decade is the notion of *differential privacy*, proposed in the seminal work of Dwork, McSherry, Nissim and Smith Dwork et al. (2006). Differential privacy is a *definition*: It *defines a certain desirable property* that $\mathcal{M}$ (i.e. the mechanism) should satisfy – $\mathcal{M}$ is differentially private if for any individual z_i, $\mathcal{M}(S)$ and $\mathcal{M}(S \setminus \{z_i\})$ "look the same." That is, from each individual's point of view, his own data will not affect the behavior of $\mathcal{M}$.[1] Intuitively, this suggests that individual i is "anonymized" and

[1] Clearly, "absolute equality" between $\mathcal{M}(S)$ and $\mathcal{M}(S \setminus \{z_i\})$ forces $\mathcal{M}$ to be a constant

his privacy is protected. Since its first appearance, a large body of theory on differential privacy has been developed (see the monograph of Dwork and Roth Dwork and Roth (2014)) and differential privacy has become the gold standard for protecting data privacy.

At the heart of differential privacy there lies an idea of "stability" – the behavior of $\mathcal{M}$ is stable with respect to any single individual's participation. A concrete way to conceive this stability is to think about a "constant" mechanism: That is, no matter what S is, $\mathcal{M}$ "stably" outputs some constant C. While such an $\mathcal{M}$ is uninteresting in view of computing f – it however achieves perfect differential privacy: It is so stable that no individual can ever affect its behavior.

The above example also illustrates a fundamental tradeoff, the *privacy-utility* tradeoff studied in differential privacy. That is, one needs to balance between designing a stable $\mathcal{M}$ with respect to any individual's participation, and making $\mathcal{M}$ compute f accurately. In fact, a common way to achieve differential privacy is to start with $f(S)$, and then inject random noise to get the desired degree of stability. The privacy-utility tradeoff is thus, more precisely, the *stability-accuracy* tradeoff. Therefore, in a strong sense, differential privacy is nothing but a stability notion.

Differential privacy is only one way to explore the idea of stability in data privacy. Taking the perspective of stability, in this thesis we study data privacy both *within* the realm of differential privacy and *beyond*. We make contributions in three directions: (i) *Advancing the theory and practice of differentially private optimization*, (ii) *Identifying misapplications of differential privacy*, and (iii) *Exploring privacy concerns beyond the scope of differential privacy*. Importantly, for all these three directions it turns out that some appropriate form of stability is the key to our development.

function, therefore in reality only some form of "approximately the same" is required.

1.1 Contributions

Advancing the theory and practice of differentially private optimization. The past decade has seen significant interest in integrating machine learning techniques into RDBMS systems (for example, MADlib Hellerstein et al. (2012), Bismarck Feng et al. (2012)) for large-scale enterprise applications. A fundamental optimization algorithm that powers various machine learning techniques in an RDBMS is *stochastic gradient descent* (SGD). SGD is attractive as it is simple to implement, easily parallelizable Zinkevich et al. (2010) and is known to have strong robustness properties.

It is thus desirable to have effective SGD algorithms with strong differential privacy guarantee. Unfortunately, while differentially private SGD have been studied in theory (SCS13 Song et al. (2013), BST14 Bassily et al. (2014)), each of the solutions has characteristics that render them unattractive for implementation in an RDBMS.

In the first part of this thesis we improve the state of the art SGD algorithms for in-RDBMS machine learning. We show, perhaps somewhat surprisingly, that the *output perturbation method*, one of the most basic paradigms for achieving differential privacy, can be applied to achieve private SGD with small noise. Specifically, we propose running SGD for a constant number of passes and then adding noise to the resulting output. *Key to our improvement is a careful and improved analysis of the L_2-sensitivity, which is a stability measure, of SGD*. Using well-known expansion properties of gradient operators, which date back to Polyak Polyak (1987), we prove that *SGD is indeed very stable* under the L_2-sensitivity measure.

To validate our approach, we provide a comprehensive empirical study comparing our method with SCS13 and BST14. First, we demonstrate practically that our algorithms can be integrated more easily with little effort and also run faster than the others. Second, using standard benchmark datasets of different scales, we analyze the effects of different

parameters for running private SGD, and demonstrate that our method yields substantially better test accuracy than SCS13 or BST14 for the same number of passes over the data. We present details of this work in Chapter 3.

Identifying areas where misapplications of differential privacy lead to claims that differential privacy is flawed. Differential privacy is ultimately a mathematical construct, and how it maps to the real world requires care and interpretation. Indeed, over the past decade, there have been several works that have tried to identify flaws in some intuitive (and popular) claims about differential privacy. For example, in Kifer and Machanavajjhala (2011), the authors argued that it is false to claim that "*differential privacy's guarantee is independent of the adversary's prior knowledge*", and that one must actually impose assumptions on the adversary's prior. This work has spawned a line of research Kifer and Machanavajjhala (2014); Li et al. (2013) that tries to identify the relative power of different priors.

Interestingly, however, it can be shown that the examples Kifer and Machanavajjhala (2011) used to correct the "misconception" about differential privacy are indeed themselves "misinterpretations." This point indeed has been clarified by some recent theoretical work Kasiviswanathan and Smith (2008) which gives a Bayesian interpretation of differential privacy: The guarantee of differential privacy is "posterior-to-posterior" in some properly defined two-world game, while the examples used to challenge differential privacy in Kifer and Machanavajjhala (2011) are "prior-to-posterior" privacy attacks. In fact, Kasiviswanathan and Smith (2008) proved that the posterior-to-posterior guarantee of differential privacy holds *independent of the prior.*

Such misinterpretations are not isolated. We observe that people (even accomplished privacy researchers) sometimes attempt to use differential privacy to counter privacy breaches that differential privacy was never

designed to prevent. An interesting example to this end is *model inversion attacks* (MI attacks), proposed in Fredrikson et al. (2014), which have received attention shortly after their discovery Fredrikson et al. (2015); Wang et al. (2015). Let us briefly describe MI attacks. As a simple example, consider a machine learning model that takes features $x_1,\ldots,x_d$ and produces a prediction value y. An MI attack takes input $x_1,\ldots,x_{d-1}$ and a value y' that is related to y, and tries to predict x_d (hence "inverting the model"). For example, in Fredrikson et al. (2014), the authors consider the case where x_d is some genetic marker, y is the warfarin dosage and $x_1,\ldots,x_{d-1}$ are some general background information (height, weight, etc.). They used an MI attack to predict an individual's *genetic markers* based on his or her *warfarin dosage*, thus breaking individual privacy.

In the work of Fredrikson et al. (2014), the authors applied differential privacy to prevent MI attacks. Specifically, they used the *functional mechanism* of Zhang et al. (2012) to build differentially private machine learning models and evaluated their resistance to MI attacks. Unfortunately, they found that (i) For reasonable differential privacy guarantee the accuracy of models produced by the functional mechanism is too low to be usable, and (ii) For models with enough noise (injected however for the purpose of differential privacy) to resist MI attacks, the accuracy is again too low to be usable.

In the second part of the thesis we address both of these issues. Our approach is to examine *stability of the empirical risk minimization* (recall that in part one, we examined stability of a particular algorithm, namely SGD). Our results are both positive and negative:

- For (i), we demonstrate that a substantially better privacy-utility tradeoff can be achieved compared to the functional mechanism. Specifically, we give a new analysis of the *L_2-sensitivity of the empirical risk minimization*. Our analysis relaxes the technical conditions required by previous work Chaudhuri et al. (2011). Moreover, it

reveals that output perturbation can be used to obtain a much better privacy-utility tradeoff than the functional mechanism. Under the same technical conditions, we achieve the *same tradeoff* between differential privacy and generalization risk as that recently proved by Bassily et al. Bassily et al. (2014), while avoiding use of the exponential mechanism McSherry and Talwar (2007) and the sophisticated sampling sub-procedure used by Bassily et al. Bassily et al. (2014). This makes our approach widely-applicable in practical settings.

- For (ii), however, we demonstrate that MI attacks and differential privacy are indeed orthogonal. To see this abstractly, it suffices to do a simple *thought experiment regarding stability*: Consider a case where somehow we obtain a perfect model without referencing any training data set. (Perhaps "nature" presents us with such a model, that is, by analysis from first principles we can construct the model.) Using this model achieves perfect differential privacy, as it is constant independent of any training data set. However, this perfect differential privacy clearly says nothing about susceptibility to MI attacks. Applying this thought experiment in a more realistic setting, we see that as long as the model our learning algorithm converges to is invertible, improving the privacy-utility tradeoff can only increase the susceptibility of the model to MI attacks. We formalize this intuition and quantify to what degree this holds in practice.

We present details of this work in Chapter 4.

Exploring privacy concerns beyond the scope of differential privacy. Our results in the previous part demonstrate that differential privacy is not designed to protect MI attacks. Simply put, what differential privacy protects is *the stability of computing a model*, while MI attacks are about models themselves.

However, the fact that MI attacks are outside of the cocoon of differential privacy does not mean that they should not be considered as privacy concerns. In particular, we note that MI attacks have received attention shortly after their discovery Fredrikson et al. (2015); Wang et al. (2015). Perhaps more importantly, in our interviews with medical doctors, they also raised MI attacks as serious privacy concerns. Therefore, it is a sensible question to ask: *What properties of a model characterize the invertibility of a model under MI attacks?*

With this question, in the last part of this thesis we take a first step to explore the theoretical foundation of the MI attacks. The main result of our study reveals that *a different form of stability plays a key role in model invertibility.* More precisely, it is the *influence* of a feature that affects the invertibility under MI attacks. That is, while differential privacy studies *the stability of the procedure producing the model*, influence is about *the stability of the model output when varying a feature*.

Using techniques from the theory of Boolean analysis, we characterize the model invertibility using *influence* (of a variable) in the situation where the adversary has precise knowledge (except for the target feature) of a victim. For the situation where the background knowledge of an adversary is noisy, we show that *stable influence* is an important indicator of invertibility. We also study the phenomenon of *invertibility interference*, where "unstable features" interfere with each other and render the model "stable" when small noise present. We give details of this work in Chapter 5.

1.2 Related Work

Differential privacy was proposed in the seminal work of Dwork, McSherry, Nissim and Smith Dwork et al. (2006) and has become the de-facto standard for data privacy. Since its introduction, there has been a large

body of theory developed; we refer readers to Dwork and Roth (2014) for an in-depth survey.

A topic that is not quite covered in Dwork and Roth (2014), but has received significant attention is differentially private convex optimization. There are three main styles of algorithms – output perturbation Bassily et al. (2014); Chaudhuri et al. (2011); Jain et al. (2012); Rubinstein et al. (2009), objective perturbation Chaudhuri et al. (2011); Kifer et al. (2012) and online algorithms Bassily et al. (2014); Duchi et al. (2013); Jain et al. (2012); Song et al. (2013). Output perturbation works by finding the exact convex minimizer and then adding noise to it, while objective perturbation *exactly* solves a randomly perturbed optimization problem. Unfortunately, the privacy guarantees provided by both styles often assume that the exact convex minimizer can be found, which usually does not hold in practice.

There are also a number of online approaches. Jain et al. (2012) provides an online algorithm for strongly convex optimization based on a *proximal algorithm* (see for example Parikh and Boyd Parikh and Boyd (2014)), which is more difficult to implement than SGD. They also provide an offline version (Algorithm 6) for the strongly convex case that is similar to our approach. Finally, SGD-style algorithms were provided by Bassily et al. (2014); Duchi et al. (2013); Jain et al. (2012); Song et al. (2013).

On the other hand, the application of these private convex optimization algorithms in practice seems slow-paced. One work in this direction is the functional mechanism by Zhang et al. Zhang et al. (2012), where their experiments show promising test accuracy on several benchmark datasets. We note that several work Aono et al. (2015); Wang et al. (2015); Winslett et al. (2012) have also adopted the functional mechanism as a basic building block in their studies.

The recent work by Fredrikson et al. Fredrikson et al. (2014) highlights the unfortunate low model accuracy even for weak differential privacy guarantees when using the functional mechanism to train differentially

private pharmacogenetic models. In the same paper, Fredrikson et al. also introduced the model inversion attacks. Shortly after their discovery, more instances of effective MI attacks have been discovered, further stimulating interest in this class of attack. For example, Fredrikson et al. (2015) considered a white-box MI attack on models used to classify images. They demonstrated that by exploiting the additional confidence information provided by such models, one can significantly improve both the effectiveness and efficiency of an MI attack. Interestingly, we note that these attacks are reminiscent of privacy attacks discussed in the context of inverting highly compressed image features, which were explored previously Daneshi and Guo (2011); d'Angelo et al. (2012); Kato and Harada (2014).

2 PRELIMINARIES

In this chapter we review preliminary knowledge for our technical development later. Specifically we collect definitions and useful technical tools that will be used throughout this thesis.

2.1 Machine Learning

Let X be a feature space, Y be an output space, and $Z = X \times Y$ be a sample space. For example, Y is a set of labels for classification, or an interval in $\mathbb{R}$ for regression. Let $\mathcal{W} \subseteq \mathbb{R}^d$ be a hypothesis space equipped with the standard inner product and 2-norm $\|\cdot\|$.[1] We are given a loss function $\ell : \mathcal{W} \times Z \mapsto \mathbb{R}$ which measures the how well a w classifies an example $z \in Z$, so that given a hypothesis $w \in \mathcal{W}$ and a sample $z \in Z$, we have a loss $\ell(w, z)$. We list the following definitions.

Definition 2.1 (Empirical Risk)**.** *Let $S = \{z_1, \ldots, z_m\}$ be a training set, the* empirical risk *over the training set S is defined to be*

$$L_S(w) = \frac{1}{m} \sum_{i=1}^{m} \ell(w, z_i).$$

For fixed S, we can think of $\ell_i(w) = \ell(w, z_i)$ as a function of w. The problem of minimizing L_S is called *empirical risk minimization* (ERM).

Definition 2.2 (Generalization Risk)**.** *Let $\mathcal{D}$ be a distribution over Z, the* generalization risk *is defined to be*

$$L_{\mathcal{D}}(w) = \mathop{\mathbb{E}}_{z \sim \mathcal{D}}[\ell(w, z)].$$

[1] Using standard results in machine learning, our results easily extend to the case when w lies in a Hilbert Space.

The minimum generalization error achievable over $\mathcal{W}$ is denoted as $L^*_{\mathcal{D}} = \min_{w \in \mathcal{W}} L_{\mathcal{D}}(w)$.

In machine learning, $\mathcal{D}$ is *unknown* but we are given a training set $S = \{z_1, \ldots, z_n\}$ drawn i.i.d. from $\mathcal{D}$. We are ready to define *learnability*. Our definition follows Shalev-Shwartz et al. Shalev-Shwartz et al. (2010) which defines learnability in the Generalized Learning Setting considered by Haussler Haussler (1992). This definition also directly generalizes PAC learnability Valiant (1984).

Definition 2.3 (Learnability)**.** *A problem is called* agnostically learnable with rate $\varepsilon(n, \delta) : \mathbb{N} \times (0,1) \mapsto (0,1)$ *if there is a learning rule* $A : Z^n \mapsto \mathcal{H}$ *such that for* any *distribution* $\mathcal{D}$ *over* Z*, given* $n \in \mathbb{N}$ *and* $\delta \in (0,1)$*, with probability* $1 - \delta$ *over* $S \sim \mathcal{D}^n$*,* $L_{\mathcal{D}}(A(S)) \leq L^*_{\mathcal{D}} + \varepsilon(n, \delta)$*. Moreover, we say that the problem is* agnostically learnable *if for any* $\delta \in (0,1)$*,* $\varepsilon(n, \delta)$ *vanishes to* 0 *as* n *tends to infinity.*

We stress that in this definition, the guarantee of generalization risk holds *simultaneously* for *every* distribution $\mathcal{D}$ on the data. Intuitively, this definition says that given a confidence parameter δ and a sample size n, the learned hypothesis $A(S)$ is $\varepsilon(n, \delta)$ close to the best achievable. Note that $\varepsilon(n, \delta)$ measures the rate we converge to the optimal.

In the above, the learning rule A is *deterministic* in the sense that it maps a training set deterministically to a hypothesis in $\mathcal{H}$. We will also talk about *randomized* learning rules, which maps a training set to a distribution over $\mathcal{H}$. More formally, a randomized learning rule $\widetilde{A}$ takes the form $Z^n \mapsto \mathcal{D}(\mathcal{H})$, where $\mathcal{D}(\mathcal{H})$ is the set of probability distributions over $\mathcal{H}$. The empirical risk of $\widetilde{A}$ on a training item z is defined as $\ell(\widetilde{A}(S), z) = \mathbb{E}_{w \sim \widetilde{A}(S)}[\ell(w, z)]$. The empirical risk of $\widetilde{A}$ on a training set S is defined as $L_S(\widetilde{A}(S)) = \mathbb{E}_{w \sim \widetilde{A}(S)}[L_S(w)]$. Finally, we define the generalization error of $\widetilde{A}$ as $L_{\mathcal{D}}(\widetilde{A}(S)) = \mathbb{E}_{w \sim \widetilde{A}(S)}[L_{\mathcal{D}}(w)]$. In short, we take expectation over the randomness of $\widetilde{A}$.

Stability Theory in Machine Learning

Stability theory is a sub-theory in machine learning (see, for example, Chapter 13 of Shalev-Shwartz and Ben-David (2014) for a gentle survey of this area). As we will see, the more stable a learning rule is, the better DP-utility tradeoff we can achieve in a private learning. To discuss stability, we need to define "change of input data set." We will use the following definition.

Definition 2.4 (Replace-One Operation)**.** *For a training set S, $i \in [n]$ and $z' \in Z$, we define $S^{(i,z')}$ to be the training set obtained by replacing z_i by z'. In other words,*

$$S = \{z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_n\},$$
$$S^{(i,z')} = \{z_1, \ldots, z_{i-1}, z', z_{i+1}, \ldots, z_n\}.$$

Moreover, we write $S^{(i)}$ instead of $S^{(i,z')}$ if z' is clear from the context.

Informally, a learning rule A is stable if $A(S)$ and $A(S^{(i)})$ are "close" to each other. There are many possible ways to formulate what does it mean by "close." We will discuss the following definition, which is the *strongest* stability notion defined by Shalev-Shwartz et al. Shalev-Shwartz et al. (2010).

Definition 2.5 (Strongly-Uniform-RO Stability Shalev-Shwartz et al. (2010))**.** *A (possibly randomized) learning rule A is* strongly-uniform-RO stable *with rate $\varepsilon_{\text{stable}}(n)$, if for all training sets S of size n, for all $i \in [n]$, and all $z', \hat{z} \in Z$, it holds that $|\ell(A(S^{(i)}), \hat{z}) - \ell(A(S), \hat{z})| \leq \varepsilon_{\text{stable}}(n)$.*

Intuitively, this definition captures the following property of a good learning algorithm A: if one changes *any one training item* in the training set S to get S', the two hypotheses computed by A from S and S', namely $A(S)$ and $A(S')$, will be "close" to each other (here "close" means that for any instance $\hat{z}$ sampled from $\mathcal{D}$, the loss of $A(S)$ and $A(S')$ on $\hat{z}$ are close).

A fundamental result on learnability and stability, proved recently by Shalev-Shwartz et al. Shalev-Shwartz et al. (2010), states the following,

Theorem 2.6 (Shalev-Shwartz et al. (2010), informal)**.** *Consider any learning problem in the generalized learning setting as proposed by Vapnik Vapnik (1998). If the problem is learnable, then it can be learned by a (randomized) rule that is strongly-uniform-RO stable.*

Qualitatively, this theorem says that *learnability and stability are equivalent*. That is, every problem that is learnable can be learned *stably* (under strongly-uniform-RO stability). In the context of differential privacy, this indicates that for any learnable problem one might hope to achieve a good DP-utility tradeoff. However, *quantitatively* the situation is much more delicate: as we will see later, strongly-uniform-RO stability is somewhat too weak to lead to a differential privacy guarantee without additional assumptions.

2.2 Optimization

Convexity

We list the following definitions:

Definition 2.7. *Let* $f : \mathcal{W} \mapsto \mathbb{R}$ *be a function:*

- f *is* L-Lipschitz *if for any* $u, v \in \mathcal{W}$,

$$\|f(u) - f(v)\| \leq L\|u - v\|.$$

- f *is* convex *if for any* $u, v \in \mathcal{W}$,

$$f(u) \geq f(v) + \langle \nabla f(v), u - v \rangle.$$

- f *is* γ-strongly convex *if*

$$f(u) \geq f(v) + \langle \nabla f(v), u - v \rangle + \frac{\gamma}{2}\|u - v\|^2.$$

- f *is* β-smooth *if*

$$\|\nabla f(u) - \nabla f(v)\| \leq \beta \|u - v\|.$$

Stochastic Gradient Descent

Stochastic gradient descent (SGD) is one of the most basic algorithms in optimization. In a nutshell, SGD performs a series of gradient updates: given time t, iterate w_t, and an individual data point (x_t, y_t), the update rule is as follows:

$$w_{t+1} = G_{\ell_t, \eta_t}(w_t) = w_t - \eta_t \ell_t'(w_t) \tag{2.1}$$

where $\ell_t(\cdot) = \ell(\cdot; (x_t, y_t))$ is the loss function and $\eta_t \in \mathbb{R}$ is a parameter called the *learning rate*, or *step size*. We will denote G_{ℓ_t, η_t} as G_t. A form of SGD that is commonly used in practice is permutation-based SGD (PSGD): first sample a random permutation τ of $[m]$ (m is the size of the training set S), and then repeatedly apply (2.1) by cycling through S according to τ. In particular, if we cycle through the dataset k times, we say that it is a k-pass PSGD. We will need two important properties of SGD, *expansiveness* and *boundedness*, which are described in Nesterov (2004); Polyak (1987).

Definition 2.8 (Expansiveness)**.** *Let $G : \mathcal{W} \mapsto \mathcal{W}$ be an operator that maps a hypothesis to another hypothesis. G is said to be ρ-expansive if*

$$\sup_{w, w'} \frac{\|G(w) - G(w')\|}{\|w - w'\|} \leq \rho.$$

Definition 2.9 (Boundedness). *Let $G : \mathcal{W} \mapsto \mathcal{W}$ be an operator that maps a hypothesis to another hypothesis. G is said to be σ-bounded if* $\sup_{w \in \mathcal{W}} \|G(w) - w\| \leq \sigma$.

Lemma 2.10 (Expansiveness (Nesterov (2004); Polyak (1987))). *Assume that ℓ is β-smooth. Then, the following hold.*

1. *If ℓ is convex, then for any $\eta \leq 2/\beta$, $G_{\ell,\eta}$ is 1-expansive.*
2. *If ℓ is γ-strongly convex, then for $\eta \leq \frac{2}{\beta+\gamma}$, $G_{\ell,\eta}$ is $(1 - \frac{2\eta\beta\gamma}{\beta+\gamma})$-expansive.*

In particular we use the following simplification due to Hardt et al. (2015).

Lemma 2.11 (Hardt et al. (2015)). *Suppose that ℓ is β-smooth and γ-strongly convex. If $\eta \leq \frac{1}{\beta}$, then $G_{\ell,\eta}$ is $(1 - \eta\gamma)$-expansive.*

Lemma 2.12 (Boundedness). *Assume that ℓ is L-Lipschitz. Then the gradient update $G_{\ell,\eta}$ is (ηL)-bounded.*

We are ready to describe a key quantity studied in this paper.

Definition 2.13 (δ_t). *Let $w_0, w_1, \ldots, w_T$, and $w'_0, w'_1, \ldots, w'_T$ be two sequences in $\mathcal{W}$. We define δ_t as $\|w_t - w'_t\|$.*

The following lemma bounds δ_t using expansiveness and boundedness properties (Lemma 2.10 and 2.12).

Lemma 2.14 (Growth Recursion Hardt et al. (2015)). *Fix any two sequences of updates $G_1, \ldots, G_T$ and $G'_1, \ldots, G'_T$. Let $w_0 = w'_0$ and $w_t = G_t(w_{t-1})$ and*

$w'_t = G'_t(w'_{t-1})$ *for* $t = 1, 2, \ldots, T$. *Then*

$$\delta_0 = 0, \text{ and for } 0 < t \le T$$

$$\delta_t \le \begin{cases} \rho\delta_{t-1} & G_t = G'_t \text{ is } \rho\text{-expansive.} \\ \min(\rho, 1)\delta_{t-1} + 2\sigma_t & G_t \text{ and } G'_t \text{ are } \sigma_t\text{-bounded,} \\ & G_t \text{ is } \rho\text{-expansive.} \end{cases}$$

2.3 Differential Privacy

We need some basic background from the theory of differential privacy. We say that two datasets S, S' are *neighboring*, denoted as $S \sim S'$, if they differ on a single individual's private value. Recall the following definition:

Definition 2.15 ((ε, δ)-differential privacy)**.** *A (randomized) algorithm A is said to be* ε-differentially private *if for any neighboring datasets S, S', and any event $E \subseteq \mathrm{Range}(A)$, $\Pr[A(S) \in E] \le e^{\varepsilon} \Pr[A(S') \in E] + \delta$.*

In particular, if $\delta = 0$, we will use ε-differential privacy instead of $(\varepsilon, 0)$-differential privacy. A basic paradigm to achieve ε-differential privacy is to examine its L_2-sensitivity,

Definition 2.16 (L_2-sensitivity)**.** *Let f be a deterministic query that maps a dataset to a vector in $\mathbb{R}^d$. The L_2-sensitivity of f is defined to be $\Delta_2(f) = \max_{S \sim S'} \|f(S) - f(S')\|$.*

The following theorem gives the "output perturbation" method for ensuring differential privacy. Essentially it relates ε-differential privacy with L_2-sensitivity.

Theorem 2.17 (Dwork et al. (2006)). *Let f be a deterministic query that maps a database to a vector in $\mathbb{R}^d$. Then publishing $f(D) + \kappa$ where κ is sampled from the distribution with density*

$$p(\kappa) \propto \exp\left(-\frac{\varepsilon\|\kappa\|}{\Delta_2(f)}\right) \tag{2.2}$$

ensures ε-differential privacy.

Importantly, the L_2-norm of the noise vector, $\|\kappa\|$, is distributed according to the Gamma distribution $\Gamma\left(d, \frac{\Delta_2(q)}{\varepsilon}\right)$. We have the following fact about Gamma distributions:

Theorem 2.18 (Chaudhuri et al. (2011)). *For the noise vector κ, we have that with probability at least $1-\gamma$, $\|\kappa\| \leq \frac{d\ln(d/\gamma)\Delta_2(q)}{\varepsilon}$.*

Finally, by changing the noise to Gaussian noise, one obtains (ε, δ)-differential privacy.

Theorem 2.19 (Dwork and Roth (2014)). *Let f be a deterministic query that maps a database to a vector in $\mathbb{R}^d$. Let $\varepsilon \in (0,1)$ be arbitrary. For $c^2 > 2\ln(1.25/\delta)$, adding Gaussian noise sampled according to*

$$\mathcal{N}(0, \sigma^2); \quad \sigma \geq \frac{c\Delta}{\varepsilon}, \quad c^2 > 2\ln\left(\frac{1.25}{\delta}\right) \tag{2.3}$$

ensures (ε, δ)-differentially privacy.

2.4 Boolean Analysis

We need some elementary concepts from Boolean analysis. More details regarding these concepts can be found in O'Donnell O'Donnell (2014). In Boolean analysis, a Boolean function $f : \{-1,1\}^n \mapsto \{-1,1\}$ is viewed as a 2^n-dimensional real vector, and we consider an inner product space of these

vectors, where the inner product is defined as $\langle f, g \rangle = \mathbb{E}_{x \sim \{-1,1\}^n}[f(x)g(x)]$. A central concept of Boolean analysis is its Fourier expansion, where the Fourier basis is the set of all parity functions $\Omega = \{\chi_S : S \subseteq [n]\}$ where $\chi_S(x) = \prod_{i \in S} x_i$ is the parity function of bits in S. Any function f can be represented as $f = \sum_{S \subseteq [n]} \widehat{f}(S)\chi_S$ where $\widehat{f}(S)$ is called the Fourier coefficient of f at S.

Definition 2.20 (Influence)**.** *Let $f : \{-1,1\}^n \mapsto \{-1,1\}$ and $i \in [n]$. The influence of i-th coordinate of f is $\mathbf{Inf}_i(f) = \Pr_{x \sim \{-1,1\}^n}[f(x) \neq f(x^{\oplus i})]$ where $x^{\oplus i}$ means to flip the i-th bit of x.*

Influence is related to the *difference operator* D_i.

Definition 2.21. *D_i is a linear operator applied to a Boolean function such that $(\mathrm{D}_i f)(x) = \frac{f(x^{i \to 1}) - f(x^{i \to -1})}{2}$. Here $x^{i \to 1}$ means we set the i-th bit of x to 1.*

Definition 2.22. *Let $b \in \{-1,1\}$ and $-1 \leq \rho \leq 1$. A random bit b' is ρ-correlated with b if*

$$b' = \begin{cases} b & w.p.\ \frac{1}{2} + \frac{\rho}{2} \\ -b & w.p.\ \frac{1}{2} - \frac{\rho}{2} \end{cases}$$

We write it as $b' \sim N_\rho(b)$. As ρ tends to 1, b' is more likely to be b.

We say that z and x are ρ-correlated if each z_i is drawn independently from $N_\rho(x_i)$, for $i \in [n]$. In such a case, we write it as $z \sim N_\rho(x)$.

Definition 2.23 (Noise Stability)**.** *Let $-1 \leq \rho \leq 1$. The ρ-noise stability of f, denoted as $\mathbf{Stab}_\rho[f]$, is defined to be $\mathbf{Stab}_\rho[f] = \mathbb{E}_{\substack{x \sim \{-1,1\}^n \\ y \sim N_\rho(x)}}[f(x)f(y)]$.*

Definition 2.24 (Stable Influence)**.** *Let $0 \leq \rho \leq 1$. The ρ-stable influence of f at i, denoted as $\mathbf{Inf}_i^{(\rho)}[f]$, is defined to be $\mathbf{Inf}_i^{(\rho)}[f] = \mathbf{Stab}_\rho[\mathrm{D}_i f] = \mathbb{E}_{\substack{x \sim \{-1,1\}^n \\ y \sim N_\rho(x)}}[\mathrm{D}_i f(x) \mathrm{D}_i f(y)]$. Note that when $\rho = 1$, this reduces to $\mathbf{Inf}_i[f]$.*

Definition 2.25 (Noise Operator)**.** *Let* $-1 \leq \rho \leq 1$*. The noise operator* T_ρ *is defined as* $\mathrm{T}_\rho f(x) = \mathbb{E}_{y \sim N_\rho(x)}[f(y)]$*.*

The following lemma gives some elementary properties of the noise operator and stable influence.

Lemma 2.26 (O'DonnellO'Donnell (2014))**.** *We have the following*

- T_ρ *is a linear operator.*
- $\mathrm{T}_\rho f = \sum_{S \subseteq [n]} \rho^{|S|} \widehat{f}(S) \chi_S$*.*
- $\mathbf{Stab}_\rho[f] = \langle f, \mathrm{T}_\rho f \rangle = \sum_{S \subseteq [n]} \rho^{|S|} \widehat{f}(S)^2$*.*

Model Inversion and Boolean Analysis

Because our functions are models learned from collected data, we will make the following assumption on the models:

Definition 2.27 (No Trivial Feature Assumption)**.** *Let* $f : \{\text{-}1, 1\}^n \mapsto \mathbb{R}$ *be a model learned from data. The* no trivial feature assumption *states that every feature has nontrivial influence. That is, for any* $i \in [n]$*,* $\mathbf{Inf}_i[f] > 0$*.*

The following simple proposition shows that if $\mathbf{Inf}_i[f] = 0$, then one can obtain an "equivalent" function over the Boolean cube without x_i.

Lemma 2.28. *Consider any* $f : \{\text{-}1, 1\}^n \mapsto \{\text{-}1, 1\}$*. Suppose that* $\mathbf{Inf}_i[f] = 0$*, then there exists another function* g *which maps* $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ *to* $\{\text{-}1, 1\}$*, such that* $\mathbf{Inf}_j[g] = \mathbf{Inf}_j[f]$ *for any* $j \neq i$*.*

Proof. Because $\mathbf{Inf}_i[f] = \sum_{S : i \in S} \widehat{f}(S)^2$, so if $\mathbf{Inf}_i[f] = 0$, this means $\widehat{f}(S) = 0$ for any $i \in S$. In particular, we can set now g to be the following function:

$$g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \sum_{S \not\ni i} \widehat{f}(S) x^S.$$

$f(x) = g(x_{-i})$ and $\mathbf{Inf}_j[g] = \mathbf{Inf}_j[f]$ for any $j \in [n]$, $j \neq i$. □

3 DIFFERENTIALLY PRIVATE STOCHASTIC GRADIENT DESCENT FOR IN-RDBMS ANALYTICS

> *"Truth is ever to be found in the simplicity, and not in the multiplicity and confusion of things."*
>
> — Issac Newton

Stochastic gradient descent (SGD) is one of the most basic, as well as popular, algorithms in optimization. SGD is widely applied in practice to power optimization tasks, such as those arise in machine learning tasks, in various domains.

In this chapter we consider differentially private SGD. State-of-the-art private algorithms, SCS13 Song et al. (2013), BST14 Bassily et al. (2014) adopt a paradigm that injects noise at every iteration of SGD. SCS13 requires a significant amount of noise at each iteration, resulting in a noisy solution. While BST14 reduces the noise per iteration, their analysis is subtle and complicated, and can only guarantee (ε, δ)-differential privacy (due to the use of advanced composition for (ε, δ)-differential privacy). From a practitioner's point of view, the fact that they require modifying gradient update steps of a non-private SGD algorithms and inject noise for each step makes it more difficult to integrate them into existing systems.

Our main results are better differentially private SGD algorithms. Our algorithms are simpler to implement, give better test accuracy and the analysis is simple and modular. Specifically, we propose running SGD for a constant number of passes and then adding noise to the resulting output. Somewhat surprisingly, we show that this simple paradigm works pretty well for SGD. Key to our improvement is *a careful and improved analysis of the L_2-sensitivity of SGD*, which leverages the well-known *expansion properties* of gradient operators Nesterov (2004); Polyak (1987).

3.1 Introduction

The past decade has seen significant interest in integrating complex data analytics into RDBMS systems (for example, MADlib Hellerstein et al. (2012), Bismarck Feng et al. (2012)) for large-scale enterprise applications. A core type of data analytics supported by these systems is *machine learning*, which is widely used to extract valuable patterns from data in domains such as personalized medicine, finance, web search histories and social networks. For these domains, there has also been an ever growing concern about individuals' privacy. To this end, *differential privacy*, a cryptographically motivated privacy notion, has emerged as the gold standard for protecting data privacy. *Differentially private learning* has been intensively studied in recent years by researchers from the database, machine learning and theoretical computer science communities Bassily et al. (2014); Chaudhuri et al. (2011); Duchi et al. (2013); Jain and Thakurta (2013); Kifer et al. (2012); Zhang et al. (2013, 2012).

Interestingly, while in-RDBMS machine learning and differentially private learning have been separately studied, to the best of our knowledge little previous work has examined private learning in an RDBMS system. We observe that an ideal solution would satisfy three properties: (1) *High accuracy*. The algorithm must learn private models of accuracy competitive with the non-private models. (2) *Low overhead*. Ideally, the private learning algorithm should only incur a small runtime overhead compared to non-private algorithms. (3) *Ease of integration*. The private algorithm must fit naturally into an RDBMS code base and should not require a significant modification to the RDBMS.

This work considers a *specific* algorithm – *stochastic gradient descent (SGD)* for differentially private machine learning in an in-RDBMS system. There are two main reasons to specifically consider SGD. First, it is a fundamental machine learning algorithm that lies at the heart of many in-RDBMS data analytics systems, as it is simple to implement, easily par-

allelizable Zinkevich et al. (2010) and is known to have strong robustness properties. For example, a key feature of Bismarck Feng et al. (2012) is a single framework to implement all convex analysis techniques available in RDBMSes based on a highly efficient implementation of SGD using User Defined Aggregates (UDA), a widely-available RDBMS abstraction. As a result, integrating a private version of SGD is relatively simple, and it automatically provides private in-RDBMS versions of all these convex analysis techniques.

The second reason to consider SGD has to do with a core problem for both in-RDBMS learning and private learning: *empirical risk minimization*. A long line of work, that includes the algorithms in Chaudhuri et al. (2011); Kifer et al. (2012); Rubinstein et al. (2009), and a subset of algorithms in Bassily et al. (2014); Jain et al. (2012); Jain and Thakurta (2013), has studied how to privately solve this problem assuming that the *exact* empirical risk minimizer can be computed. However, as also observed by Jain, Kothari and Thakurta Jain et al. (2012), this assumption often does not hold in reality, and it is not clear if the privacy guarantees based on it still hold if only approximate solutions can be computed. By directly using private SGD, we do not have to rely on the "exact minimizer" assumption any more, and can provide a more general and practical solution.

While previous work has considered differentially private versions of SGD, each of the solutions has characteristics that render them unattractive for implementation in an RDBMS. Song, Chaudhuri and Sarwate Song et al. (2013) adds a large amount of noise at each iteration of SGD to make it differentially private; this results in a noisy solution, which is partially mitigated in practice by mini-batching. Bassily, Smith and Thakurta Bassily et al. (2014) provides a second solution following the same paradigm, but with less noise per iteration. This is achieved by first, using a novel subsampling technique and second, relaxing the privacy guarantee to

(ε, δ)-differential privacy for $\delta > 0$. This relaxation is necessary as they need to use advanced composition results for (ε, δ)-differential privacy. Therefore, both of these solutions require modifying gradient update steps of a non-private SGD algorithm and inject noise for each step. This makes it more difficult to integrate them with an in-RDBMS analytics system. In fact, when we integrated them with Bismarck, it required a somewhat deep modification of the existing code base.

Main Contributions. We make the following contributions in improving private SGD for better private in-RDBMS machine learning.

- We show, perhaps somewhat surprisingly, that the *output perturbation method*, one of the most basic paradigms for achieving differential privacy, can be applied to achieve private SGD with small noise. Specifically, we propose running SGD for a constant number of passes and then adding noise to the resulting output. We provide guarantees on the privacy properties of this approach, which are based on a careful and improved analysis of the L_2-sensitivity of SGD. Our analysis leverages the well-known *expansion properties* of gradient operators Nesterov (2004); Polyak (1987).

- Because output perturbation assumes black-box access to non-private algorithms, our method thus directly inherits many of the nice properties of SGD, while allowing easier integration. We integrate our private SGD algorithms, as well as SCS13 Song et al. (2013) and BST14 Bassily et al. (2014) into Bismarck Feng et al. (2012), an in-RDBMS data analytics platform. We demonstrate practically that our algorithms can be integrated more easily with little effort and also runs faster than the others.

- We provide a comprehensive empirical study comparing our method with SCS13 and BST14 . Using standard benchmark datasets of different scales, we analyze the effects of different parameters for

running private SGD, and demonstrate that our method yields substantially better test accuracy than SCS13 or BST14 for the same number of passes over the data.

The rest of this chapter is organized as follows: In Section 3.2, we present our private SGD algorithms and analyze their privacy and convergence guarantees. Along the way, we extend our main algorithms in various ways to incorporate common practices of SGD. We then provide a comprehensive empirical study in Section 3.3 to evaluate private SGD with respect to the three aforementioned properties for in-RDBMS analytics: High accuracy, lower overhead, and ease of integration.

3.2 Private SGD

In this section we present differentially private PSGD algorithms and analyze their privacy and convergence guarantees. Specifically, we present a new analysis of the output perturbation method for PSGD. Our new analysis shows that very little noise is needed to achieve differential privacy. In fact, the resulting private algorithms have good convergence rates with even *one pass* through the data. Since output perturbation also uses standard PSGD algorithm as a black-box, this makes our algorithms attractive for in-RDBMS scenarios.

This section is structured accordingly in two parts. In Section 3.2 we give two main differentially private algorithms for convex and strongly convex optimization. In Section 3.2 we first prove that these two algorithms are differentially private (Section 3.2 and 3.2), then extend them in various ways (Section 3.2), and finally prove their convergence (Section 3.2).

Algorithms

As we mentioned before, our differentially private PSGD algorithms use one of the most basic paradigms for achieving differential privacy – the *output perturbation* method Dwork et al. (2006) based on L_2-*sensitivity* (Definition 2.16). Specifically, our algorithms are "instantiations" of the output perturbation method where the L_2-sensitivity parameter Δ_2 comes out of our new analysis. To describe the algorithms, we assume a standard permutation-based SGD procedure (denoted as PSGD) which can be invoked as a black-box. To facilitate the presentation, Table 3.1 summarizes the parameters and their meaning.

Parameter	Meaning
L	Lipschitz constant.
γ	Strong convexity.
β	Smoothness.
ε, δ	Privacy parameters.
η_t	Learning rate or step size at iteration t.
$\mathcal{W}$	Hypothesis space.
$\ell(w, z)$	$w \in \mathcal{W}$, z is a single individual's data.
S	Training set.
m	$\lvert S \rvert$, the size of S.
R	Radius of the convex set.

Table 3.1: Parameters.

Algorithm 1 Private Convex Permutation-based SGD

Require: $\ell(\cdot, z)$ is convex for every z, $\eta \leq 2/\beta$.

Input: Data S, parameters k, η, ε

1: **function** PrivateConvexPSGD(S, k, ε, η)
2: $w \leftarrow \mathsf{PSGD}(S)$ with k passes and $\eta_t = \eta$
3: $\Delta_2 \leftarrow 2kL\eta$
4: Sample noise vector κ according to (2.2).
5: **return** $w + \kappa$

Algorithm 2 Private Strongly Convex Permutation-based SGD

Require: $\ell(\cdot, z)$ is γ-strongly convex for every z

Input: Data S, parameters k, ε

1: **function** PrivateStronglyConvexPSGD(S, k, ε)
2: $w \leftarrow \mathsf{PSGD}(S)$ with k passes and $\eta_t = \min(\frac{1}{\beta}, \frac{1}{\gamma t})$
3: $\Delta_2 \leftarrow \frac{2L}{\gamma m}$
4: Sample noise vector κ according to (2.2).
5: **return** $w + \kappa$

Algorithms 1 and 2 give our private SGD algorithms for convex and strongly convex cases, respectively. A key difference between these two algorithms is at line 3 where different L_2-sensitivities are used to sample the noise κ. Note that different learning rates are used: In the convex case, a constant rate is used, while a decreasing rate $\frac{1}{\gamma t}$ is used in the strongly convex case. Finally, note that the standard PSGD is invoked as a black box at line 2.

Analysis

In this section we establish privacy and convergence guarantees of Algorithms 1 and 2. Along the way, we also describe extensions to accom-

modate common practices in running stochastic gradient descent. Most proofs in this section are deferred to the appendix.

We start with an overview of the privacy analysis. Let $A(r;S)$ denote a randomized non-private algorithm where r denotes the randomness (e.g., random permutations sampled by SGD) and S denotes the input training set. On a pair of neighboring datasets S, S' (i.e., S, S' differ on a single data point), we want to bound $\max_{r,r'} \|A(r;S) - A(r';S')\|$, where r, r' can be *different randomness sequences of* A in general. This can be complicated in general since $A(r;\cdot)$ and $A(r';\cdot)$ may access the data in vastly different patterns.

Fortunately, we observe that for non-adaptive randomized algorithms, one can consider randomness sequences one at a time, and it suffices to bound $\max_r \|A(r;S) - A(r;S')\|$. We have the following definition,

Definition 3.1 (Non-Adaptive Algorithms)**.** *A randomized algorithm A is* non-adaptive *if its random choices do not depend on the input data values.*

PSGD is clearly non-adaptive as a single random permutation is sampled at the very beginning of the algorithm. Another common SGD variant, where one independently and uniformly samples $i_t \sim [m]$ at iteration t and picks the i_t-th data point, is also non-adaptive. In fact, more modern SGD variants, such as Stochastic Variance Reduced Gradient (SVRG Johnson and Zhang (2013)) and Stochastic Average Gradient (SAG Roux et al. (2012)), are non-adaptive as well. Now we have the following lemma for non-adaptive algorithms and differential privacy.

Lemma 3.2. *Let $A(r;S)$ be a non-adaptive randomized algorithm where r denotes the randomness of the algorithm and S denotes the dataset A works on. Suppose that*

$$\sup_{S \sim S'} \sup_{r} \|A(r;S) - A(r;S')\| \leq \Delta.$$

Then publishing $A(r;S)+\kappa$ where κ is sampled with density $p(\kappa) \propto \exp\left(-\frac{\varepsilon\|\kappa\|_2}{\Delta}\right)$ ensures ε-differential privacy.

Proof. Let $\widetilde{A}$ denote the private version of A. $\widetilde{A}$ has two parts of randomness: One part is r, which is used to compute $A(r;S)$; the second part is κ, which is used for perturbation (i.e. $A(r;S)+\kappa$). Let R be the random variable corresponding to the randomness of A. Note that R does not depend on the input training set. Thus for any event E,

$$\begin{aligned} &\Pr[\widetilde{A}((r,\kappa);S) \in E] \\ &= \sum_r \Pr[R=r] \cdot \Pr_\kappa[A((r,\kappa);S) \in E \mid R=r]. \end{aligned} \tag{3.1}$$

Denote $\Pr_\kappa[A((r,\kappa);S) \in E \mid R=r]$ by $p_\kappa(A_r(S) \in E)$. Then similarly for S' we have that

$$\begin{aligned} &\Pr[\widetilde{A}((r,\kappa);S') \in E] \\ &= \sum_r \Pr[R=r] \cdot p_\kappa(A_r(S') \in E). \end{aligned} \tag{3.2}$$

Compare (3.1) and (3.2) term by term (for every r): the lemma then follows as we calibrate the noise κ so that $p_\kappa(A_r(S) \in E) \leq e^\varepsilon p_\kappa(A_r(S') \in E)$. □

From now on we denote PSGD by A. With Definition 2.13, our goal is thus to bound $\sup_{S\sim S'} \sup_r \delta_T$. Fortunately, one can now derive a good upper bound for it using the expansion and boundedness properties: In fact, because for PSGD in each pass the differing data point is only encountered once, one can easily accumulate their contributions and obtains the desired L_2-sensitivity bounds.

Remark 3.3. *We note that in an independent line of research on formally proving differential privacy, Barthe et al. Barthe et al. (2016) found a new useful proof principle which is related to "using the same randomness for privacy noise." This resembles our analysis here which relies on "same randomness of*

SGD." However, we note a significant difference: the "same randomness" in Barthe et al.'s work is related to the randomness for privacy, which in our case is about the Laplace noise injected at the end. However, the "same randomness" used in our analysis refers to the randomness of SGD (not for privacy).

Convex Optimization

In this section we prove privacy guarantee when $\ell(\cdot, z)$ is convex. Recall that for general convex optimization, we have 1-expansiveness by Lemma 1. We thus have the following lemma that bounds δ_T.

Lemma 3.4. *Consider k-passes PSGD for L-Lipschitz, convex and β-smooth optimization where $\eta_t \leq \frac{2}{\beta}$ for $t = 1, \ldots, T$. Let S, S^i be any neighboring datasets. Let r be a random permutation of $[m]$. Suppose that $r(i) = i^*$. Let $T = km$, then $\delta_T \leq 2L \sum_{j=0}^{k-1} \eta_{i^*+jm}$.*

Proof. Let $T = km$, so we have in total T updates. Applying Lemma 2.12, Growth Recursion Lemma (Lemma 2.14), and the fact that the gradient operators are 1-expansive, we have:

$$\delta_t \leq \begin{cases} \delta_{t-1} + 2L\eta_t & \text{if } t = i^* + jm, \\ & j = 0, \ldots, k-1 \\ \\ \delta_{t-1} & \text{otherwise.} \end{cases} \tag{3.3}$$

Unrolling the recursion completes the proof. □

We immediately have the following corollary on L_2-sensitivity with constant step size,

Corollary 3.5 (Constant Step Size)**.** *Consider k-passes PSGD for L-Lipschitz, convex and β-smooth optimization. Suppose further that we have constant learning rate $\eta_1 = \eta_2 = \cdots = \eta_T = \eta \leq \frac{2}{\beta}$. Then $\sup_{S \sim S'} \sup_r \delta_T \leq 2kL\eta$.*

This directly yields the following theorem,

Theorem 3.6. *Algorithm 1 is ε-differentially private.*

We now give L_2-sensitivity results for two different choices of step sizes, which are also common for convex optimization.

Corollary 3.7 (Decreasing Step Size)**.** *Let $c \in [0,1)$ be some constant. Consider k-passes PSGD for L-Lipschitz, convex and β-smooth optimization. Suppose further that we take decreasing step size $\eta_t = \frac{2}{\beta(t+m^c)}$ where m is the training set size. Then $\sup_{S\sim S'} \sup_r \delta_T = \frac{4L}{\beta}\left(\frac{1}{m^c} + \frac{\ln k}{m}\right)$.*

Proof. We have that

$$\sup_{S\sim S'} \sup_r \|A(r;S) - A(r;S')\| \leq \frac{4L}{\beta}\left(\sum_{j=0}^{k-1} \frac{1}{m^c + jm + 1}\right).$$

Therefore

$$\begin{aligned}
\frac{4L}{\beta}\left(\sum_{j=0}^{k-1} \frac{1}{m^c + jm + 1}\right) &= \frac{4L}{\beta}\left(\frac{1}{m^c+1} + \sum_{j=1}^{k-1} \frac{1}{m^c + jm + 1}\right) \\
&\leq \frac{4L}{\beta}\left(\frac{1}{m^c} + \frac{1}{m}\sum_{j=1}^{k-1} \frac{1}{j}\right) \\
&\leq \frac{4L}{\beta}\left(\frac{1}{m^c} + \frac{\ln k}{m}\right)
\end{aligned}$$

as desired. □

Corollary 3.8 (Square-Root Step Size)**.** *Let $c \in [0,1)$ be some constant. Consider k-passes PSGD for L-Lipschitz, convex and β-smooth optimization. Sup-*

pose further that we take square-root step size $\eta_t = \frac{2}{\beta(\sqrt{t}+m^c)}$. *Then*

$$\begin{aligned}\sup_{S\sim S'}\sup_{r}\delta_T \leq & \frac{4L}{\beta}\left(\sum_{j=0}^{k-1}\frac{1}{\sqrt{jm+1}+m^c}\right)\\ = & O\left(\frac{L}{\beta}\left(\frac{1}{m^c}+\min\left(\frac{k}{m^c},\sqrt{\frac{k}{m}}\right)\right)\right).\end{aligned}$$

Two remarks are in order: First, in Lemma 3.5 we use "constant" step size for the SGD. However, one should note that "constant step size" does not mean "constant noise" in a typical differential privacy sense. Constant step size for SGD can depend on the size of the training set, and in particular can vanish to zero as training set size increases. This, in particular, implies that our private PSGD algorithm is *consistent* (i.e., L_2-sensitivity vanishes to 0 as training set size increases). In fact, in typical convergence results of SGD (see, for example in Bubeck (2015); Nemirovsky and Yudin (1983)) the constant step size η is set to $1/T^{O(1)}$ where T is the total number of iterations. In such cases, T asymptotically depends on m, the size of the training set. For example, if the step size is $1/\sqrt{T}$, and we run a single epoch through the data, then the L_2-sensitivity is indeed $\frac{2L}{\sqrt{m}}$, and so the noise vanishes to 0 as m increases.

Second, in SGD we may want to approximately measure the "maximal progress" one can make, which is how far away one can walk from the starting point. Formally, this is the sum of all step sizes. We note that, for this to be large in the case of decreasing and square-root step sizes, c should not be set to be too large, while making it large is beneficial for reducing noise. Indeed, since our initial step size is $O(1/m^c)$, and the final step size is $O(1/km)$, the sum is roughly $\int_{m^c}^{km}\frac{1}{x}dx = O((1-c)\ln m)$.

Strongly Convex Optimization

Now we consider the case where $\ell(\cdot, z)$ is γ-strongly convex. In this case the sensitivity is smaller because the gradient operators are ρ-expansive for $\rho < 1$ so in particular they become contractions. We have the following lemmas.

Lemma 3.9 (Constant Step Size)**.** *Consider PSGD for L-Lipschitz, γ-strongly convex and β-smooth optimization with constant step sizes $\eta \leq \frac{1}{\beta}$. Let k be the number of passes. Let S, S' be two neighboring datasets differing at the i-th data point. Let r be a random permutation of $[m]$. Suppose that $r(i) = i^*$. Let $T = km$, then $\delta_T \leq 2L\eta \sum_{j=0}^{k-1} (1-\eta\gamma)^{(k-j)m-i^*}$. In particular,*

$$\sup_{S \sim S'} \sup_{r} \delta_T \leq \frac{2\eta L}{1-(1-\eta\gamma)^m}.$$

Proof. Let $T = km$, so we have in total T updates. We have the following recursion

$$\delta_t \leq \begin{cases} (1-\eta\gamma)\delta_{t-1} + 2\eta L & \text{if } t = i^* + jm, \\ & j = 0, 1, \ldots, k-1 \\ (1-\eta\gamma)\delta_{t-1} & \text{otherwise.} \end{cases} \tag{3.4}$$

This is because at each pass different gradient update operators are encountered only at position i^* (corresponding to the time step $t = i^* + jm$), and so the two inequalities directly follow from the growth recursion lemma (Lemma 2.14). Therefore, the contribution of the differing entry in the first pass contributes $2\eta L(1-\eta\gamma)^{T-i^*}$, and generalizing this, the differing entry in the $(j+1)$-th pass ($j = 0, 1, \ldots, k-1$) contributes $2\eta L(1-\eta\gamma)^{T-i^*-jm}$. Summing up gives the first claimed bound.

For sensitivity, we note that for $j = 1, 2, \ldots, k$, the j-th pass can only

contribute at most $2\eta L \cdot (1-\eta\gamma)^{(k-j)m}$ to δ_T. Summing up gives the desired result. □

Lemma 3.10 (Decreasing Step Size)**.** *Consider k-passes PSGD for L-Lipschitz, γ-strongly convex and β-smooth optimization. Suppose further that we use decreasing step length: $\eta_t = \min(\frac{1}{\gamma t}, \frac{1}{\beta})$. Let S, S' be two neighboring datasets differing at the i-th data point. Let r be a random permutation of $[m]$. Suppose that $r(i) = i^*$. Let $T = km$, then $\sup_{S\sim S'} \sup_r \delta_T \leq \frac{2L}{\gamma m}$.*

Proof. From the Growth Recursion Lemma (Lemma 2.14) we know that in the γ-strongly convex case, with appropriate step size, in each iteration either we have a contraction of δ_{t-1}, or, we have a contraction of δ_{t-1} plus an additional additive term. In PSGD, in each pass the differing data point will only be encountered once, introducing an additive term, and is contracted afterwards.

Formally, let T be the number of updates, the differing data point is at location i^*. Let $\rho_t < 1$ be the expansion factor at iteration t. Then the first pass contributes $\delta_1^* \prod_{t=i^*+1}^{T} \rho_t$ to δ_T, the second pass contributes $\delta_2^* \prod_{t=i^*+m+1}^{T} \rho_t$ to δ_T. In general pass j contributes $\delta_j^* \prod_{t=i^*+(j-1)m+1}^{T} \rho_t$ to δ_T.

Let $\iota_j = \delta_j^* \prod_{t=i^*+(j-1)m+1}^{T} \rho_t$ be the contribution of pass j to δ_T. We now figure out δ_j^* and ρ_t. Consider ι_1, we consider two cases. If $i^* \geq \frac{\beta}{\gamma}$, then $\eta_t \leq \frac{1}{\gamma t} \leq \frac{1}{\beta}$, and so G_t is $(1-\eta_t\gamma) = (1-\frac{1}{t})$ expansive. Thus if $i^* \geq \frac{\beta}{\gamma}$ then before i^* the gap is 0 and after i^* we can apply expansiveness such that

$$\frac{2L}{\gamma t} \cdot \prod_{i=t+1}^{km} \left(1 - \frac{1}{i}\right) = \frac{2L}{\gamma t} \cdot \prod_{i=t+1}^{km} \frac{i-1}{i} = \frac{2L}{\gamma km},$$

The remaining case is when $i^* \leq \frac{\beta}{\gamma} - 1$. In this case we first have 1-expansiveness due to convexity that the step size is bounded by $\frac{1}{\beta} < \frac{2}{\beta}$.

Moreover we have $(1-\frac{1}{t})$-expansiveness for G_t when $\frac{\beta}{\gamma} \leq t \leq m$. Thus

$$2L\eta_{i^*} \cdot \prod_{j=\frac{\beta}{\gamma}}^{km} \left(1 - \frac{1}{j}\right) \leq \frac{2L\eta_{i^*}\beta/\gamma}{km} = 2L \cdot \frac{1}{\beta} \cdot \frac{\beta}{\gamma km} = \frac{2L}{\gamma km},$$

Therefore $\iota_1 \leq \frac{2L}{\gamma km}$. Finally, for $j = 2, \ldots, k$,

$$\iota_j \leq \frac{2L}{\gamma((j-1)m + i^*)} \cdot \prod_{t=(j-1)m+i^*+1}^{km} \frac{t-1}{t} = \frac{2L}{\gamma km}.$$

Summing up gives the desired result. □

In particular, Lemma 3.10 yields the following theorem,

Theorem 3.11. *Algorithm 2 is ε-differentially private.*

One should contrast this theorem with Theorem 3.6: In the convex case we bound L_2-sensitivity by $2kL\eta$, while in the strongly convex case we bound it by $2L/\gamma m$.

Extensions

In this section we extend our main argument in several ways: (ε, δ)-differential privacy, mini-batching, model averaging, fresh permutation at each pass, and finally constrained optimization. These extensions can be easily incorporated to standard PSGD algorithm, as well as our private algorithms 1 and 2, and are used in our empirical study later. We close this section with a discussion on the limitations of our analysis.

(ε, δ)-Differential Privacy. We can also obtain (ε, δ)-differential privacy easily using Gaussian noise (see Theorem 2.19). We have that

Lemma 3.12. *Let $A(r; S)$ be a non-adaptive randomized algorithm where r denotes the randomness of the algorithm and S denote the dataset. Suppose*

that

$$\sup_{S \sim S'} \sup_{r} \|A(r;S) - A(r;S')\| \leq \Delta.$$

Then for any $\varepsilon \in (0,1)$, publishing $A(r;S) + \kappa$ where each component of κ is sampled using (2.3) ensures (ε, δ)-differential privacy.

In particular, combining this with our L_2-sensitivity results, we get the following two theorems,

Theorem 3.13 (Convex and Constant Step)**.** *Algorithm 1 is (ε, δ)-differentially private if each component of κ at line 3 is sampled according to equation (2.3).*

Theorem 3.14 (Strongly Convex and Decreasing Step)**.** *Algorithm 2 is (ε, δ)-differentially private if each component of κ at line 3 is sampled according to equation (2.3).*

Mini-batching. A popular way to do SGD is that at each step, instead of sampling a single data point z_t and do gradient update with respect to it, we randomly sample a batch $B \subseteq [m]$ of size b, and do

$$w_t = w_{t-1} - \eta_t \frac{1}{b} \left(\sum_{i \in B} \ell_i'(w_{t-1}) \right) = \frac{1}{b} \sum_{i \in B} G_i(w_{t-1}).$$

For permutation SGD, a natural way to employ mini-batch is to partition the m data points into mini-batches of size b (for simplicity let us assume that b divides m), and do gradient updates with respect to each chunk. In this case, we notice that mini-batch indeed improves the sensitivity by a factor of b. In fact, let us consider neighboring datasets S, S', and at step t, we have batches B, B' that differ in at most one data point. Without loss of generality, let us consider the case where B, B' differ at one data point, then on S we have $w_t = \frac{1}{b} \sum_{i \in B} G_i(w_{t-1})$, and on S' we have $w_t' = \frac{1}{b} \sum_{i \in B} G_i'(w_{t-1}')$,

and so

$$\begin{aligned}\delta_t &= \left\| \frac{1}{b} \sum_{i \in B} G_i(w_{t-1}) - G_i'(w_{t-1}') \right\| \\ &\leq \frac{1}{b} \sum_{i=1}^{B} \|G_i(w_{t-1}) - G_i'(w_{t-1}')\|.\end{aligned}$$

We note that for all i except one in B, $G_i = G_i'$, and so by the Growth Recursion Lemma 2.14, $\|G_i(w_{t-1}) - G_i'(w_{t-1}')\| \leq \rho\delta_{t-1}$ if G_i is ρ-expansive, and for the differing index i^*, $\|G_{i^*}(w_{t-1}) - G_{i^*}'(w_{t-1}')\| \leq \min(\rho, 1)\delta_{t-1} + 2\sigma_t$. Therefore, for a uniform bound ρ_t on expansiveness and σ_t on boundedness (for all $i \in B$, which is the case in our analysis), we have that $\delta_t \leq \rho_t \delta_{t-1} + \frac{2\sigma_t}{b}$. This implies a factor b improvement for all our sensitivity bounds.

Model Averaging. Model averaging is a popular technique for SGD. For example, given iterates $w_1, \ldots, w_T$, a common way to do model averaging is either to output $\frac{1}{T}\sum_{t=1}^{T} w_t$ or output the average of the last $\log T$ iterates. We show that model averaging will not affect our sensitivity result, and in fact it will give a constant-factor improvement when earlier iterates have smaller sensitivities. We have the following lemma.

Lemma 3.15 (Model Averaging)**.** *Suppose that instead of returning w_T at the end of the optimization, we return an averaged model $\bar{w} = \sum_{t=1}^{T} \alpha_t w_t$, where α_t is a sequence of coefficients that only depend on t, T. Then,*

$$\sup_{S \sim S'} \sup_{r} \|\bar{w} - \bar{w}'\| \leq \sum_{t=1}^{T} \alpha_t \|w_t - w_t'\| = \sum_{t=1}^{T} \alpha_t \delta_t.$$

In particular, we notice that the δ_t's we derived before are non-decreasing, so the sensitivity is bounded by $(\sum_{t=1}^{T} \alpha_t)\delta_T$.

Fresh Permutation at Each Pass. We note that our analysis extends ver-

batim to the case where in each pass a new permutation is sampled. This is because our analysis applies to *any* fixed permutation.

Constrained Optimization. Until now, our SGD algorithm is for unconstrained optimization. That is, the hypothesis space $\mathcal{W}$ is the entire $\mathbb{R}^d$. Our results easily extend to constrained optimization where the hypothesis space $\mathcal{W}$ is a convex set $\mathcal{C}$. That is, our goal is to compute $\min_{w\in\mathcal{C}} L_S(w)$. In this case, we change the original gradient update rule 2.1 to the *projected gradient update rule*:

$$w_t = \prod_{\mathcal{C}}\Big(w_{t-1} - \eta_t \ell_t'(w_{t-1})\Big), \tag{3.5}$$

where $\prod_{\mathcal{C}}(w) = \arg\min_v \|v - w\|$ is the projection of w to $\mathcal{C}$. It is easy to see that our analysis carries over verbatim to the projected gradient descent. In fact, our analysis works as long as the optimization is carried over a Hilbert space (i.e., the $\|\cdot\|$ is induced by some inner product). The essential reason is that projection will not increase the distance ($\|\prod u - \prod v\| \le \|u - v\|$), and thus will not affect our sensitivity argument.

Limitations. Our analysis so far applies to *permutation-based* stochastic gradient descent and variants where the randomness of the SGD is *non-adaptive*. We note that while PSGD is commonly used in practice, in theoretical studies one often looks at *sampling-with-replacement* method, where in each iteration one samples uniformly and independently from $[m]$, instead of sampling a permutation at the start and cycling through it. The convergence behavior of sampling-with-replacement method is much better well-understood than sampling-without-replacement method (i.e. permutation-based). However, with sampling-with-replacement method one can only hope for (ε, δ)-differential privacy because with tiny probability one always encounters the differing data point and so a vast amount of noise is needed.

Our analysis so far also fails for adaptive randomized algorithms.

An important example to this end is the Nesterov's momentum method, which has fast convergence guarantee in theory. Extending our analysis to handle momentum method seems to be challenging and important.

Convergence of Optimization

We now bound the optimization error of our private PSGD algorithms. More specifically, we bound the *excess empirical risk* $L_S(w) - L_S^*$ where $L_S(w)$ is the loss of the output w of our private SGD algorithm and L_S^* is the minimum obtained by any w in the feasible set $\mathcal{W}$. Note that in PSGD we sample data points *without replacement*. While sampling without replacement benefits our L_2-sensitivity argument, its convergence behavior is poorly understood in theory. Our results are based on very recent advances by Shamir Shamir (2016) on the sampling-without-replacement SGD.

As in Shamir Shamir (2016), we assume that the loss function ℓ_i takes the form of $\ell_i(\langle w, x_i \rangle) + r(w)$ where r is some fixed function. Further we assume that the optimization is carried over a convex set $\mathcal{C}$ of radius R (i.e., $\|w\| \leq R$ for $w \in \mathcal{C}$). We use projected PSGD algorithm (i.e., we use the projected gradient update rule 3.5).

Finally, $R(T)$ is a regret bound if for any $w \in \mathcal{W}$ and convex-Lipschitz $\ell_1, \ldots, \ell_T$, $\sum_{t=1}^{T} \ell_t(w_t) - \sum_{t=1}^{T} \ell_t(w) \leq R(T)$ and $R(T)$ is sublinear in T. We use the following regret bound,

Theorem 3.16 (Zinkevich Zinkevich (2003))**.** *For SGD with constant step size $\eta_1 = \eta_2 = \cdots = \eta_T = \eta$, the regret bound $R(T)$ is bounded by $\frac{R^2}{2\eta} + \frac{L^2 T \eta}{2}$.*

Proof. The proof follows exactly the same argument as Theorem 1 of Zinkevich Zinkevich (2003), except we change the step size in the final accumulation of errors. □

We are now ready to bound the excess empirical risk. We start with the following simple lemma.

Lemma 3.17 (Error Introduced by Privacy)**.** *Consider L-Lipschitz and β-smooth optimization. Let w be the output of the non-private SGD algorithm, κ be the noise of the output perturbation, and $\widetilde{w} = w + \kappa$. Then $L_S(w) - L_S(\widetilde{w}) \leq L\|\kappa\|$.*

Convex Optimization. If $\ell(\cdot, z)$ is convex, we use the following theorem from Shamir Shamir (2016),

Theorem 3.18 (Corollary 1 of Shamir Shamir (2016))**.** *Let $T \leq m$ (that is we take at most 1-pass over the data). Suppose that each iterate w_t is chosen from $\mathcal{W}$, and the SGD algorithm has regret bound $R(T)$, and that $\sup_{t,w\in\mathcal{W}} |\ell_t(w)| \leq R$, and $\|w\| \leq R$ for all $w \in \mathcal{W}$. Finally, suppose that each loss function ℓ_t takes the form $\bar{\ell}(\langle w, x_t\rangle) + r(w)$ for some L-Lipschitz $\bar{\ell}(\cdot, x_t)$ and $\|x_t\| \leq 1$, and a fixed r, then*

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} L_S(w_t) - L_S(w^*)\right] \leq \frac{R(T)}{T} + \frac{2(12+\sqrt{2}L)R}{\sqrt{m}}.$$

Together with Theorem 3.16, we thus have the following lemma,

Lemma 3.19. *Consider the same setting as in Theorem 3.18, and 1-pass PSGD optimization defined according to rule (3.5). Suppose further that we have constant learning rate $\eta = \frac{R}{L\sqrt{m}}$. Finally, let $\bar{w}_m$ be the model averaging $\frac{1}{m}\sum_{t=1}^{T} w_t$. Then,*

$$\mathbb{E}[L_S(\bar{w}_T) - L_S^*] \leq \frac{(L + 2(12+\sqrt{L}))R}{\sqrt{m}}.$$

Now we can bound the excess empirical risk as follows,

Theorem 3.20 (Convex and Constant Step Size)**.** *Consider the same setting as in Lemma 3.19 where the step size is constant $\eta = \frac{R}{L\sqrt{m}}$. Let $\tilde{w} = \bar{w}_T + \kappa$ be*

the result of output perturbation. Then

$$\mathbb{E}[L_S(\tilde{w}) - L_S^*] \leq \frac{(L + (2(12 + \sqrt{L}))R}{\sqrt{m}} + \frac{2dLR}{\varepsilon\sqrt{m}}.$$

Proof. The output of the private PSGD algorithm is $\tilde{w} = \bar{w}_T + \kappa$, where κ is distributed according to a Gamma distribution $\Gamma(d, \frac{\Delta_2}{\varepsilon})$. By Lemma 3.5, $\Delta_2 \leq 2L\eta = \frac{2R}{\sqrt{m}}$. Therefore by Lemma 3.17, $\mathbb{E}_\kappa[L_S(\tilde{w}) - L_S(\bar{w}_m)] \leq \frac{2dR}{\varepsilon\sqrt{m}}$, where we use the fact that the expectation of the Gamma distribution is $\frac{d\Delta_2}{\varepsilon}$. Summing up gives the bound. □

Note that the term $\frac{2dLR}{\varepsilon\sqrt{m}}$ corresponds to the expectation of $L\|\kappa\|$.

Strongly Convex Optimization. If $\ell(\cdot, z)$ is γ-strongly convex, we instead use the following theorem,

Theorem 3.21 (Theorem 3 of Shamir Shamir (2016))**.** *Suppose $\mathcal{W}$ has diameter R, and $L_S(\cdot)$ is γ-strongly convex on $\mathcal{W}$. Assume that each loss function ℓ_t takes the for $\bar{\ell}(\langle w_t, x_t\rangle) + r(w)$ where $\|x_i\| \leq 1$, $r(\cdot)$ is possibly some regularization term, and each $\bar{\ell}(\cdot, x_t)$ is L-Lipschitz and β-smooth. Furthermore, suppose $\sup_{w\in\mathcal{W}} \|\ell_t'(w)\| \leq G$. Then for any $1 < T \leq m$, if we run SGD for T iterations with step size $\eta_t = 1/\gamma t$, we have*

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} L_S(w_t) - L_S(w^*)\right] \leq c \cdot \frac{((L + \beta R)^2 + G^2)\log T}{\gamma T},$$

where c is some universal positive constant.

Using the same argument as in the convex case, we immediately have the following theorem,

Theorem 3.22 (Strongly Convex and Decreasing Step Size)**.** *Consider the same setting as in Theorem 3.21 where the step size is $\eta_t = \frac{1}{\gamma t}$. Consider 1-pass PSGD. Let $\bar{w}_T$ be the result of model averaging and $\tilde{w} = \bar{w}_T + \kappa$ be the result of output perturbation. Then $\mathbb{E}[L_S(\tilde{w}) - L_S(w^*)] \leq c \cdot \frac{((L+\beta R)^2+G^2)\log m}{\gamma m} + \frac{2dG^2}{\varepsilon\gamma m}$.*

Several remarks are in order. (i) Our excess empirical risk bounds are weaker than those obtained in Bassily et al. Bassily et al. (2014), where in the convex case their bound is $O\left(\frac{LR\log^{3/2}(m/\delta)\sqrt{d\log(1/\delta)}}{\varepsilon m}\right)$, and in the strongly convex case their bound is $O\left(\frac{L^2\log^2(m/\delta)d\log(1/\delta)}{\gamma\varepsilon^2 m^2}\right)$. (ii) However, our bounds are for a *single pass* through the data, while theirs need m passes. (iii) Our bounds are for ε-differential privacy, while theirs are for (ε, δ)-differential privacy ($\delta > 0$). (iv) Finally, our bounds need more assumptions: Specifically, we assume that ℓ_t takes the special form of $\bar{\ell}(\langle w, x_t\rangle) + r(w)$.

3.3 Empirical Evaluation

In this section we conduct a comprehensive empirical study investigating three alternatives for private SGD in RDBMSes: Two previously proposed state-of-the-art private SGD algorithms, SCS13 Song et al. (2013) and BST14 Bassily et al. (2014), and our algorithms which are instantiations of the output perturbation method with our new analysis.

In our study we try to answer the following four main questions regarding the three issues for a good private in-RDBMS SGD: High accuracy, low overhead and ease of integration.

1. *What is the effort to integrate each algorithm into an in-RDBMS system?*
2. *How does the test accuracy of our algorithms compare to SCS13 and BST14?*
3. *How do various parameters, such as mini-batch sizes, number of passes (epochs), and privacy parameters, affect the test accuracy?*
4. *What is the runtime overhead when we deploy these algorithms in real data analytics systems?*

As a summary, our main findings are the following: **(i)** Our SGD algorithm requires almost no changes to Bismarck, while both SCS13 and BST14 require deeper changes. **(ii)** Under the same differential privacy guarantees, our private SGD algorithms yield substantially better accuracy than SCS13 and BST14, for all datasets and settings of parameters we test. **(iii)** As for the effect of parameters, our empirical results align well with the theory. For example, as one might expect, mini-batch sizes are important for reducing privacy noise. The number of passes is more subtle. For our algorithm, if the learning task is only convex, more passes result in larger noise (e.g., see Lemma 3.5), and so give rise to potentially worse test accuracy. On the other hand, if the learning task is strongly convex, the number of passes will not affect the noise magnitude (e.g., see Lemma 3.10). As a result, doing more passes may lead to better convergence and thus potentially better test accuracy. Interestingly, we note that slightly enlarging mini-batch size can reduce noise very effectively so it is affordable to run our private algorithms for more passes to get better convergence in the convex case. This corroborates the results of Song et al. (2013) that mini-batches are helpful in private SGD settings. **(iv)** Our algorithms incur virtually no overhead, while SCS13 and BST14 run much slower. The reason is that our algorithms only need to inject noise once at the end while SCS13 and BST14 need to inject noise at every gradient update step.

In the rest of this section we give more details of our empirical study. Our discussion is structured as follows: In Section 3.3 we first discuss the implemented algorithms. In particular, we discuss how we modify SCS13 and BST14 to make them better fit into our experiments. We also give some remarks on other relevant previous algorithms, and on parameter tuning. Then in Section 3.3 we discuss the effort of integrating different algorithms into Bismarck. Next in Section 3.3 we discuss the experimental design and datasets for testing accuracy and runtime. Then in Section 3.3,

we report the results on test accuracy for various datasets and parameter settings, and discuss the effects of parameters. Finally in Section 3.3, we report runtime overhead.

Implemented Algorithms

We first discuss implementations of our algorithms, SCS13 and BST14. Importantly, we extend both SCS13 and BST14 to make them better fit into our experiments. Among these extensions, probably most importantly, we extend BST14 to support a smaller number of iterations through the data and *reduce the amount of noise* needed for each iteration. Our extension makes BST14 more competitive in our experiments.

Our Algorithms. We implement Algorithms 1 and 2 with the extensions of mini-batching and constrained optimization (see Section 3.2). Note that our algorithms invoke a standard PSGD algorithm as a black-box and Bismarck already supports mini-batching and constrained optimization. Therefore the only change we need to make for Algorithms 1 and 2 is the setting of L_2-sensitivity parameter Δ_2 at line 3 of respective algorithms, which we divide by b if the mini-batch size is b.

BST14 Bassily et al. (2014). The original BST14 algorithm needs $O(m^2)$ iterations to finish, which is prohibitive for even moderate sized datasets. We extend it to support cm iterations for some constant c. Reducing the number of iterations means that potentially we can *reduce the amount of noise for privacy* because data is "less examined." This is indeed the case: One can go through the same proof in Bassily et al. (2014) with a smaller number of iterations, and show that each iteration only needs a smaller amount of noise than before (unfortunately this does not give convergence results). Our extension makes BST14 *more competitive*. In fact it yields significantly better test accuracy compared to the case where one naïvely stops BST14 after c passes, but the noise magnitude in each

iteration is the same as in the original paper Bassily et al. (2014) (which is for m passes). The extended BST14 algorithms are given in Algorithm 3 and 4. Finally, we also make straightforward extensions so that BST14 supports mini-batching.

Algorithm 3 Convex BST14 Constant with Constant Epochs

Require: $\ell(\cdot, z)$ is convex for every z, $\eta \leq 2/\beta$.
Input: Data S, parameters $k, \varepsilon, \delta, d, L, R$

function ConvexBST14ConstNpass($S, k, \varepsilon, \delta, d, L, R$)
  $m \leftarrow |S|$
  $T \leftarrow km$
  $\delta_1 \leftarrow \delta/km$
  $\varepsilon_1 \leftarrow$ Solution of $\varepsilon = T\varepsilon_1(e_1^{\varepsilon} - 1) + \sqrt{2T\ln(1/\delta_1)}\varepsilon_1$
  $\varepsilon_2 \leftarrow \min(1, m\varepsilon_1/2)$
  $\sigma^2 \leftarrow 2\ln(1.25/\delta_1)/\varepsilon_2^2$
  $w \leftarrow 0$
  for $t = 1, 2, \ldots, T$ **do**
    $i_t \sim [m]$ and let (x_{i_t}, y_{i_t}) be the data point.
    $z \sim \mathcal{N}(0, \sigma^2 \iota I_d)$ ▷ $\iota = 1$ for logistic regression, and in general is the L_2-sensitivity localized to an iteration; I_d is d-dimensional identity matrix.
    $w \leftarrow \prod_{\mathcal{W}} \left(w - \eta_t(\nabla\ell(w; (x_{i_t}, y_{i_t}) + z)\right)$ where $\eta_t = \frac{2R}{G\sqrt{t}}$ and $G = \sqrt{d\sigma^2 + b^2L^2}$.
  return w_T

SCS13 Song et al. (2013). We also modify Song et al. (2013), which originally only supports one pass through the data, to support multi-passes over the data.

Other Related Work. We also note the work of Jain, Kothari and Thakurta Jain et al. (2012) which is related to our setting. In particular their Algorithm 6 is similar to our private SGD algorithm in the setting of strong convexity and (ε, δ)-differential privacy. However, we note that their algorithm uses Implicit Gradient Descent (IGD), which belongs to *proximal algorithms*

Algorithm 4 Strongly Convex BST14 with Constant Epochs

Input: Data S, parameters $k, \varepsilon, \delta, d, L, R$

function StronglyConvexBST14ConstNpass($S, k, \varepsilon, \delta, d, L, R$)
  $m \leftarrow |S|$
  $T \leftarrow km$
  $\delta_1 \leftarrow \delta/km$
  $\varepsilon_1 \leftarrow$ Solution of $\varepsilon = T\varepsilon_1(e_1^{\varepsilon} - 1) + \sqrt{2T\ln(1/\delta_1)}\varepsilon_1$
  $\varepsilon_2 \leftarrow \min(1, m\varepsilon_1/2)$
  $\sigma^2 \leftarrow 2\ln(1.25/\delta_1)/\varepsilon_2^2$
  $w \leftarrow 0$
  for $t = 1, 2, \ldots, T$ **do**
    $i_t \sim [m]$ and let (x_{i_t}, y_{i_t}) be the data point.
    $z \sim \mathcal{N}(0, \sigma^2 \iota I_d)$
    $w \leftarrow \prod_{\mathcal{W}} \left(w - \eta_t(\nabla \ell(w; (x_{i_t}, y_{i_t}) + z) \right)$, $\eta_t = \frac{1}{\gamma t}$.
  return w

(see for example Parikh and Boyd Parikh and Boyd (2014)) and is known to be more difficult to implement than stochastic gradient methods. Due to this consideration, in this study we will not compare empirically with this algorithm. Finally, we also note that Jain et al. (2012) also has an SGD-style algorithm (Algorithm 3) for strongly convex optimization and (ε, δ)-differential privacy. This algorithm adds noise comparable to our algorithm *at each step* of the optimization, and as a result, we do not compare with this algorithm either.

Parameter Tuning. We observe that for all the SGD algorithms we considered, multiple parameters need to be fine-tuned to achieve the best performance. In particular, the mini-batch sizes, number of passes, L_2-regularization parameter, and privacy parameters all have significant impact on the final performance. Therefore, a natural attempt is to show how one can *tune* these parameters in order to achieve best performance. However, one should note that this is an issue orthogonal to our main questions: We are interested in obtaining *relative* performance of the

algorithms for some same (sensible) setting of parameters, rather than seeking the best parameters for one particular algorithm. Moreover, we note that under differential privacy, tuning parameters must also be done *privately*, and there has been an independent line of research Chaudhuri et al. (2011); Chaudhuri and Vinterbo (2013) investigating it. As a result, we choose to compare performance of these algorithms under *various settings of parameters*, and leave tuning for the best performing parameters as separate future work.

Integration with Bismarck

We now explain how we integrate private SGD algorithms in RDBMS. To begin with, we note that the state-of-the-art way to do in-RDBMS is via the User Defined Aggregates (UDA) offered by almost all RDBMSes Gray et al. (1997). Using UDAs enables scaling to larger-than-memory datasets seamlessly while still being fast.[1] A well-known open source implementation of the UDAs required is Bismarck Feng et al. (2012). Bismarck achieves high performance and scalability through a unified architecture of in-RDBMS data analytics systems using the permutation-based stochastic gradient descent.

Therefore, we use Bismarck to experiment with private SGD inside RDBMS. Specifically, we use Bismarck on top of PostgreSQL, which implements the UDA for SGD in C to provide high runtime efficiency. Our results carry over naturally to any other UDA-based implementation of analytics in an RDBMS. The rest of this section is organized as follows. We first describe Bismarck's system architecture. We then compare the system extensions and the implementation effort needed for integrating our private PSGD algorithm as well as SCS13 and BST14.

[1] The MapReduce abstraction is similar to an RDBMS UDA mah. Thus our implementation ideas apply to MapReduce-based systems as well.

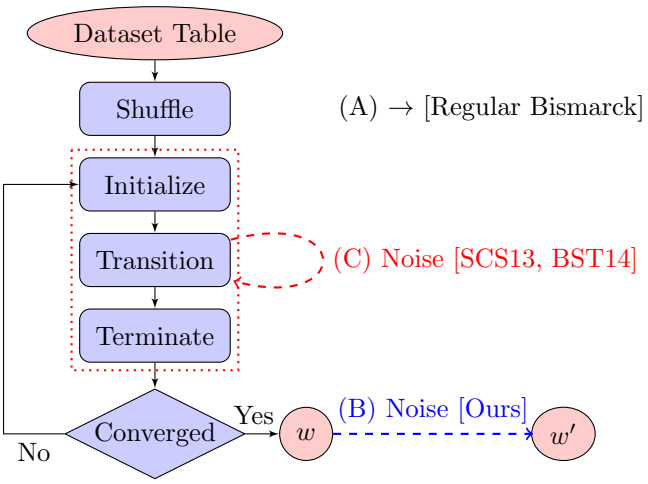


Figure 3.1: (A) System architecture of regular Bismarck. (B) Extension to implement our algorithms. (C) Extension to implement any of SCS13 and BST14.

Figure 3.1 (A) gives an overview of Bismarck's architecture. The dataset is stored as a table in PostgreSQL. Bismarck permutes the table using an SQL query with a shuffling clause, viz., `ORDER BY RANDOM()`. A pass (or epoch, which is used more often in practice) of SGD is implemented as a C UDA and this UDA is invoked with an SQL query for each epoch. A front-end controller in Python issues the SQL queries and also applies the convergence test for SGD after each epoch. The developer has to provide implementations of three functions in the UDA's C API: initialize, transition, and terminate, all of which operate on the aggregation `state`, which is the quantity being computed over the table.

To explain how this works, we compare SGD with a standard SQL aggregate: `AVG`. The state for `AVG` is the 2-tuple `(sum, count)`, while that for SGD is the model vector w. The function initialize sets `(sum, count)` = `(0, 0)` for `AVG`, while for SGD, it sets w to the value given by the Python controller (the previous epoch's output model). The function `transition` updates the state based on a single tuple (one example). For example, given a tuple

with value x, the state update for AVG is as follows: (sum, count) += (x, 1). For SGD, x is the feature vector and the update is the update rule for SGD with the gradient on x. If mini-batch SGD is used, the updates are made to a temporary accumulated gradient that is part of the aggregation state along with counters to track the number of examples and mini-batches seen so far. When a mini-batch is over, the transition function updates w using the accumulated gradient for that mini-batch using an appropriate step size. The function terminate computes sum/count and outputs it for AVG, while for SGD, it simply returns w at the end of that epoch.

It is easy to see that our private SGD algorithm requires almost no change to Bismarck – simply add noise to the final w output after all epochs, as illustrated in Figure 3.1 (B). Thus, our algorithm does not modify any of the RDBMS-related C UDA code. In fact, we were able to implement our algorithm in about 10 lines of code (LOC) in Python within the front-end Python controller. In contrast, both SCS13 and BST14 require deeper changes to the UDA's transition function because they need to add noise at the end of each mini-batch update. Thus, implementing them required adding dozens of LOC in C to implement their noise addition procedure within the transition function, as illustrated in Figure 3.1 (C). Furthermore, Python's scipy library already provides the sophisticated distributions needed for sampling the noise (gamma and multivariate normal), which our algorithm's implementation exploits. But for both SCS13 and BST14, we need to implement some of these distributions in C so that it can be used in the UDA.[2]

Experimental Method and Datasets

We now describe our experimental method and datasets used for testing accuracy and runtime overhead.

[2]One could use the Python-based UDAs in PostgreSQL but that incurs a significant runtime performance penalty compared to C UDAs.

Test Scenarios. We consider four main scenarios to evaluate the algorithms: (1) Convex, ε-differential privacy, (2) Convex, (ε, δ)-differential privacy, (3) Strongly Convex, ε-differential privacy, and finally (4) Strongly Convex, (ε, δ)-differential privacy. Note that BST14 only supports (ε, δ)-differential privacy. Thus for tests (1) and (3) we compare non-private algorithm, our algorithms, and SCS13. For tests (2) and (4), we compare non-private algorithm, our algorithms, SCS13 and BST14. For each of these scenarios, we train models on standard datasets used for evaluating SGD and measure the test accuracy of the resulting models on the test datasets. As is common in the literature for evaluating SGD in the context of convex optimization, we choose *logistic regression* models. We use the standard logistic regression for the convex case (Tests (1) and (2)), and logistic regression regularized by L_2-regularizer for the strongly convex case (Tests (1) and (3)). We now describe the datasets and the parameter spaces considered for each of them.

Dataset	Task	Train Size	Test Size	#Dimensions
MNIST	10 classes	60000	10000	784 (50) [*]
Protein	Binary	72876	72875	74
Forest	Binary	498010	83002	54

Table 3.2: Datasets. Each row gives the name of the dataset, number of classes in the classification task, sizes of training and test sets, and finally the number of dimensions. [*]: For MNIST, it originally has 784 dimensions, which is difficult for differential privacy as the noise scales linearly with the number of dimensions. Therefore we randomly project it to 50 dimensions. All data points are normalized to the unit sphere.

Datasets. We consider three standard benchmark datasets: MNIST[3], Protein[4], and Forest Covertype[5]. MNIST is a popular dataset used for

[3] http://yann.lecun.com/exdb/mnist/.
[4] http://osmot.cs.cornell.edu/kddcup/datasets.html.
[5] https://archive.ics.uci.edu/ml/datasets/Covertype.

image classification. MNIST poses a challenge to differential privacy for three reasons: First, it has 784 dimensions which is relatively more high dimensional than other datasets considered here. To get meaningful test accuracy we thus use Gaussian Random Projection, which is known to preserve differential privacy, to randomly project to 50 dimensions. This random projection only incurs very small loss in test accuracy, and thus the performance of non-private SGD on 50 dimensions will serve the baseline. Second, MNIST is of medium size and differential privacy is known to be more difficult for medium or small sized datasets. Finally, MNIST is a multiclass classification (there are 10 digits), we built "one-vs.-all" multiclass logistic regression models. Using the "one-vs.-all" method means that we need to construct 10 binary models (one for each digit). Thus for privacy, one needs to split the privacy budget across sub-models. We used the simplest composition theorem Dwork and Roth (2014), and divide the privacy budget evenly.

For Protein dataset, because its test dataset does not have labels, we randomly partition the training set into halves to form train and test datasets. Logistic regression models have very good test accuracy on it. Finally, Forest Covertype is a large dataset with 581012 data points, almost 6 times larger than previous ones. We split it to have 498010 training points and 83002 test points. We use this large dataset for two purposes: First, in this case, one may expect that privacy will follow more easily. We test to what degree this holds for different private algorithms. Second, since training on such large datasets is time consuming, it is desirable to use it to measure runtime overheads of various private algorithms.

Parameters. For MNIST we vary ε in $\{0.1, 0.2, 0.5, 1, 2, 4\}$. For Protein and Covertype, we vary ε in $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.4\}$ (as they are binary classification and we do not need to divide by 10). δ is set to be $1/m$ where m is the size of the training set size. We vary mini-batch sizes in $\{1, 10, 50\}$, and number of passes in $\{1, 10, 20\}$. The L_2-regularization parameter is

0.0001. Finally, Table 3.3 summarizes step sizes for different algorithms and tests.

	Non-private	Ours	SCS13	BST14
C + ε-DP	Constant	Constant	$\frac{1}{\sqrt{t}}$	×
C + (ε, δ)-DP	Constant	Constant	$\frac{1}{\sqrt{t}}$	Alg. 3
SC + ε-DP	$\frac{1}{\gamma t}$	$\min(\frac{1}{\beta}, \frac{1}{\gamma t})$	$\frac{1}{\sqrt{t}}$	×
SC + (ε, δ)-DP	$\frac{1}{\gamma t}$	$\min(\frac{1}{\beta}, \frac{1}{\gamma t})$	$\frac{1}{\sqrt{t}}$	Alg. 4

Table 3.3: Step Sizes for different test scenarios and algorithms. C: Convex, SC: Strongly Convex, DP: differential privacy. For SCS13 and strongly convex case we choose step size $1/\sqrt{t}$ because results inSong et al. (2013) suggest that this step size yields smaller variance with small $\lambda = 0.0001$. In our experiments, these two step sizes yield very similar accuracy.

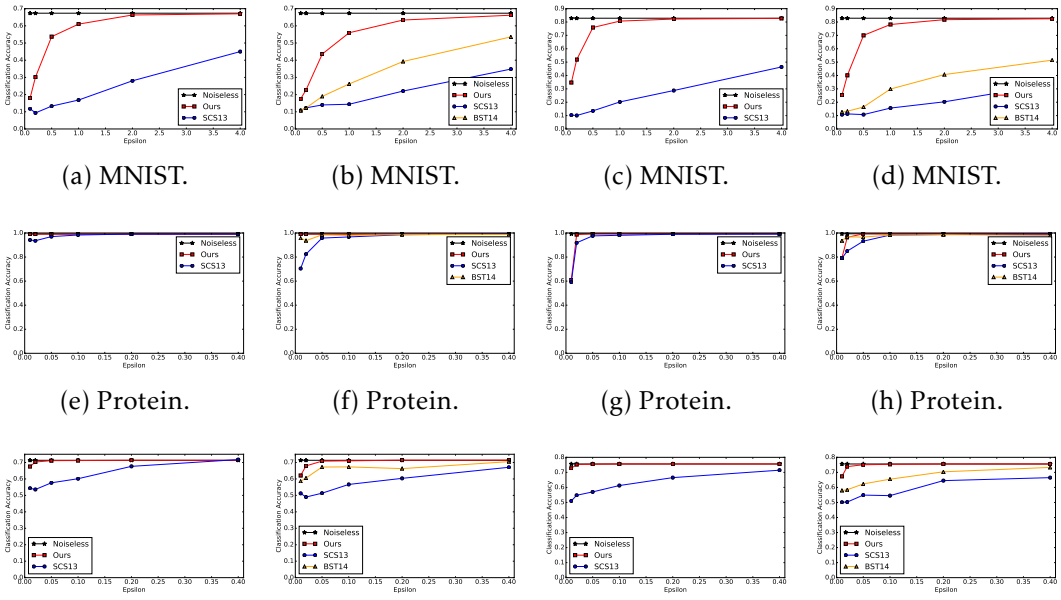
Experimental Environment. All the experiments were run on a machine with Intel Xeon E5-2680 2.50GHz CPUs (48-core) and 64GB RAM running Ubuntu 14.04.4.

Accuracy and Effects of Parameters

We now present results on test accuracy and analyze the effects of parameters. Due to lack of space, we only report partial results for representative parameter settings.

Test Accuracy. Figure 3.2 gives the test accuracy results of MNIST, Protein and Covertype for all 4 test scenarios with mini-batch size 50 and 10 passes over the data. *For all the four tests we see that our algorithms give significantly better accuracy, up to 4x better than SCS13 and up to 3.5x better than BST14.*

SCS13 and BST14 exhibit much better accuracy on Protein than on MNIST, since logistic regression fits well to the problem. Specifically, BST14 has very close accuracy as our algorithms, though our algorithms



(a) MNIST. (b) MNIST. (c) MNIST. (d) MNIST.

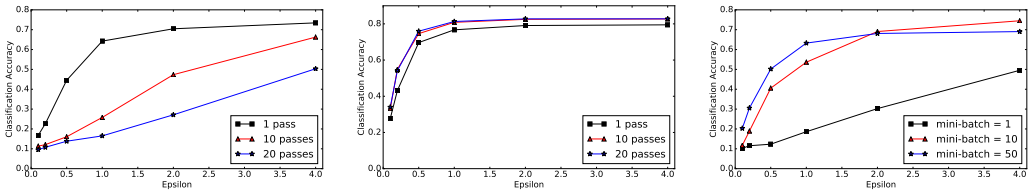
(e) Protein. (f) Protein. (g) Protein. (h) Protein.

(i) Forest Covertype. (j) Forest Covertype. (k) Forest Covertype. (l) Forest Covertype.

Figure 3.2: **Test accuracy** for all 4 tests with mini-batch size 50 and 10 passes. ε is varied in $\{0.1, 0.2, 0.5, 1.0, 2.0, 4.0\}$, and $\delta = 1/m$. For Test 3 and Test 4, the L_2-regularization parameter $\lambda = 0.0001$. For constrained optimization (SCS13 in the regularized case), the radius is set to $1/\lambda$, otherwise we report results of unconstrained optimization. Each row gives the results of 4 tests, where Test 1 is Convex, $(\varepsilon, 0)$-DP, Test 2 is Convex, (ε, δ)-DP, Test 3 is Strongly Convex, $(\varepsilon, 0)$-DP, and Test 4 is Strongly Convex, (ε, δ)-DP. For Test 1 and 3, we compare Noiseless, our algorithm and SCS13. For Test 2 and 4, we compare all four algorithms.

still consistently outperform BST14. The accuracy of SCS13 decreases significantly with smaller ε.

For Covertype, even on this large dataset, SCS13 and BST14 give much worse accuracy compared to ours. The accuracy of our algorithms is close to the baseline at around $\varepsilon = 0.05$. The accuracy of SCS13 and BST14 slowly improves with more passes over the data. Specifically, the accuracy of BST14 approaches the baseline only after $\varepsilon = 0.4$.



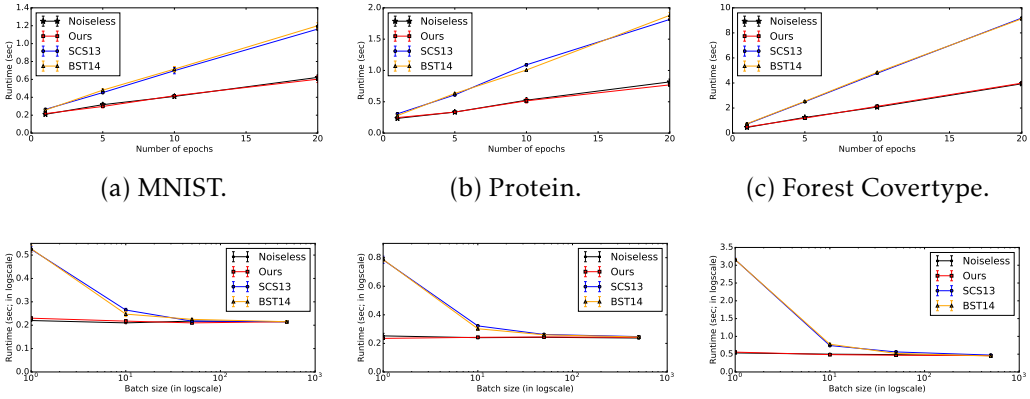
(a) Convex, Test 1 (b) Strongly Convex, Test 3 (c) Convex, Test 1

Figure 3.3: **(a), (b) The effect of number of passes**: We report the results on MNIST dataset. We contrast Test 1 (Convex ε-DP) using mini-batch size 1, with Test 3 (Strongly Convex ε-DP) using mini-batch size 50. In the former case, more passes through the data introduces more noise due to privacy and thus results in worse test accuracy. In the latter case, more passes improves the test accuracy as it helps convergence while no more noise is needed for privacy. **(c) The effect of mini-batch size**. We run again Test 1 (Convex, ε-DP) with 20 passes through the data, and vary mini-batch size in $\{1, 10, 50\}$. We note that as soon as we increase mini-batch size to 10 the test accuracy improves drastically from 0.45 to 0.71.

Finally, we observe that in some cases our private algorithms actually yield better test accuracy than the baseline. This is not surprising because test accuracy amounts to *generalization risk*, and it is known that deliberately introducing small noise can better stabilize the output hypothesis and helps generalization Shalev-Shwartz et al. (2010).

Number of Passes (Epochs). Intuitively, more passes through the data can improve the convergence of SGD and identify a better hypothesis. However, differential privacy brings in another dimension: More passes over the data implies that data is examined more thoroughly and so more noise is needed for privacy. As a result, we face a tradeoff here: Taking more passes can improve accuracy of the non-private solution, yet one may inject more noise for privacy, and thus harm the accuracy in the end.

This intuition is made precise in the case of convex optimization, where Lemma 3.5 shows that with k-passes the L_2-sensitivity goes up to



(a) MNIST. (b) Protein. (c) Forest Covertype.

(d) MNIST. (e) Protein. (f) Forest Covertype.

Figure 3.4: **Runtimes of the implementations on** Bismarck**. (a), (b), (c)**: Vary the number of epochs with mini-batch size = 10. **(d), (e), (f)**: Vary the mini-batch size for a single epoch. Only the results of Strongly Convex, (ε, δ)-DP are reported, and other settings have very similar trends. Noiseless is the regular mini-batch SGD in Bismarck. We fix $\varepsilon = 0.1$.

$O(k\eta)$. In fact, our experiments confirm that the larger noise due to more passes in fact harms the accuracy. Figure 3.3 (a) reports test accuracy in the convex case as we run our algorithm 1 pass, 10 passes and 20 passes through the MNIST data. The accuracy drops from 0.71 to 0.45 for $\varepsilon = 4.0$.

The case of strong convexity is different, as Lemma 3.10 shows that the L_2-sensitivity is independent of the number of passes. As a result no more noise will be needed for privacy in the strongly convex case, and more passes may help convergence. In fact, as Figure 3.3 (b) confirms that more passes does *improve* the test accuracy.

Mini-batch Sizes. We find that slightly enlarging the mini-batch size can effectively reduce the noise and thus allow the private algorithm to run more passes in the convex setting. This is useful since it is very common in practice to adopt a mini-batch size at around 10 to 50. To illustrate the effect of mini-batch size we consider the same test as we did above for

measuring the effect of number of passes: We run Test 1 with 20 passes through the data, but vary mini-batch sizes in $\{1, 10, 50\}$. Figure 3.3 (c) reports the test accuracy for this experiment: As soon as we increase mini-batch size to 10 the test accuracy already improves drastically from 0.45 to 0.71.

Runtime Overhead

We now compare the runtime overheads of our private SGD algorithm integrated into Bismarck against the regular, noiseless version, as well as the other algorithms. Unlike the accuracy experiments, the key parameters that affect runtimes are the number of epochs and the batch sizes. Thus, we vary each of these parameters, while fixing the others. The runtimes are the average of 4 warm-cache runs and all datasets fit in the buffer cache of PostgreSQL. The error bars represent 90% confidence intervals. The results are plotted in Figure 3.4 (a)–(c) and Figure 3.4 (d)–(f) (only the results of strongly convex, (ε, δ)-differential privacy are reported; the other results are similar and thus, we skip them here for brevity).

The first observation is that our algorithm incurs virtually no runtime overhead over noiseless Bismarck, which is as expected because our algorithm only adds noise once at the end of all epochs. In contrast, both SCS13 and BST14 incur significant runtime overheads in all settings and datasets. In terms of runtime performance for 20 epochs and a batch size of 10, both SCS13 and BST14 are between 2x and 3x slower than our algorithm. The gap grows larger as the batch size is reduced: for a batch size of 1 and 1 epoch, both SCS13 and BST14 are up to 6x slower than our algorithm. This is expected since these algorithms invoke expensive random sampling code from sophisticated distributions for each mini-batch. When the batch size is increased to 500, the runtime gap between these algorithms practically disappears as the random sampling code is invoked much less often. Overall, we find that not only is our algorithm

easier to integrate with existing popular RDBMSes, it can be significantly faster than the alternative algorithms.

4 DIFFERENTIAL PRIVACY AND MODEL INVERSION ATTACKS

> *"...There are various important parts of modern mathematics in which the empirical origin is untraceable, or, if traceable, so remote that it is clear that the subject has undergone a complete metamorphosis since it was cut off from its empirical roots... the only remedy seems to me to be the rejuvenating return to the source: the re-injection of more or less directly empirical ideas."*
>
> — John von Neumann: The Mathematician

Differential privacy is a behavioral definition in the sense that it defines a property of the process of private computation. However, the definition of differential privacy does not directly say what kind of privacy it protects. To better understand its implications to privacy, one has to define "semantic" models to reason about the meaning of differential privacy. In fact, ever since the first work of differential privacy Dwork et al. (2006), various work Kasiviswanathan and Smith (2008); Kifer and Machanavajjhala (2011); Li et al. (2013) have studied the semantics of differential privacy.

Unfortunately, even after much of this work and many debates about what differential privacy does and does not protect, we observe that people (even accomplished privacy researchers) sometimes attempt to use differential privacy to counter privacy breaches that differential privacy was never designed to prevent. This issue certainly goes beyond academic interests given that differential privacy has become the de-facto standard for protecting data privacy, and large data vendors have been using systems based on differential privacy to collect what they cannot collect before due to privacy concerns Erlingsson et al. (2014). In these situations, it can be disastrous to use differential privacy as a silver bullet

and release whatever has been computed under the differential privacy umbrella.

In this chapter we move from studying differential privacy to the boundary between differential privacy and other privacy concerns. Specifically, we examine the relationship between differential privacy and model inversion attacks (MI attacks), a privacy attack proposed in Fredrikson et al. (2014) where the authors applied differential privacy as a counter measure for MI attacks. Our approach is based on a careful study of *the stability of empirical risk minimization*. Our main results are twofold: (i) MI attacks and differential privacy are orthogonal to each other. In fact, improving the privacy-utility tradeoff of differential privacy will only worsen the problem of MI attacks. (ii) Along the way of investigating (i), we show that substantially better privacy-utility tradeoff can be achieved compared to the functional mechanism Zhang et al. (2012), which is used in Fredrikson et al. (2014) to train regression models.

4.1 Introduction

Regression models are widely used to extract valuable data patterns in "sensitive" domains. For example, in personalized medicine, linear regression models are commonly used to predict medication dosages International Warfarin Pharmacogenetic Consortium (2009) and other useful features Diaz and Yeh (2012); McKeague and Qian (2011). For such scenarios, individuals' privacy has become a major concern and learning with differential privacy (DP) has become the state of the art scheme to protect data privacy. As a result, *differentially private regression*, and more generally private convex learning, has been intensively studied in recent years by researchers from the security, theory and data management communities Bassily et al. (2014); Chaudhuri et al. (2011); Duchi et al. (2013); Fredrikson et al. (2014); Jain and Thakurta (2013); Zhang et al. (2012).

A notable contribution in the direction of private regression is the *functional mechanism*, proposed by Zhang et al. Zhang et al. (2012), which is a practical mechanism for training differentially-private regression models. We notice that the functional mechanism has become the state of the art method in the practice of private regression, and in particular several research groups Aono et al. (2015); Wang et al. (2015); Winslett et al. (2012) have adopted the functional mechanism as a basic building block in their study and initial empirical results are promising.

Our starting point is recent work by Fredrikson et al. Fredrikson et al. (2014), which used the functional mechanism to train differentially private linear regression models to predict doses of a medicine called *warfarin* from patients' genomic traits. Unfortunately, they found that the model accuracy was unacceptable with ε-DP — even for ε as high as 5. One of the main goals of this chapter is to develop a suitable theory which allows mechanisms based on simple techniques such as *output perturbation* to obtain accurate, differentially private models at reasonable private levels (e.g., $\varepsilon = 0.1$).

To this end, we start by presenting a precise connection between differential privacy and stability theory in machine learning. Our formalization makes a folklore connection between differential privacy and stability (Chaudhuri) explicit. Specifically, in the setting of learning, we show that differential privacy is essentially another stability definition, yet it is so strong that it implies previous stability definitions in the learning literature. Combining this observation with some machinery from stable learning theory, we give an analysis showing that simple mechanisms such as output perturbation can learn *every* convex-Lipschitz-bounded problem with *strong* differential privacy guarantees.

Our analysis has several advantages over previous work. First, it *relaxes* the technical conditions required by Chaudhuri et al. Chaudhuri

et al. (2011) for output perturbation[1]. In particular, we do not require the loss function to be smooth or differentiable. Second, our analysis reveals that output perturbation can be used to obtain a much better privacy/utility tradeoff than the functional mechanism. Under the same technical conditions, we achieve the *same tradeoff* between differential privacy and generalization error as that recently proved by Bassily et al. Bassily et al. (2014), while avoiding use of the exponential mechanism McSherry and Talwar (2007) and the sophisticated sampling sub-procedure used by Bassily et al. Bassily et al. (2014). This makes our approach widely-applicable in practical settings.

Following the analysis we derive algorithms for private convex optimization. Our algorithms are regularized versions of the empirical risk minimization using the Tikhonov regularizer. In contrast to the functional mechanism, where regularization is a heuristic, regularization is crucial for our theoretical guarantee.

Regularization adds additional parameters that need to be picked carefully in order to maintain DP. A standard approach used in related contexts is to employ a private parameter tuning algorithm Chaudhuri et al. (2011) to select a set of parameters that depends on the training data. We have implemented this approach to parameter tuning and refer to the mechanisms using it as the *privately-tuned mechanisms* in our empirical study. We also consider a different approach, based on our proof of generalization error, that picks the parameters in a *data independent* manner.

Compared to the functional mechanism, our theoretical analysis shows that much smaller noise is needed for the same level of differential privacy for all convex-Lipschitz-bounded learning problems. To further confirm the utility of our method in practice, we evaluated both of our approaches

[1] On the other hand, we note that for *objective perturbation* Kifer et al. Kifer et al. (2012) relaxed the technical conditions so that the loss function does not need to be differentiable.

using real-world datasets. To this end, we note that as the training set size increases, the magnitude of noise required by our method vanishes to 0, while the functional mechanism always requires constant noise. Thus for sufficiently large training sets, our method is guaranteed (theoretically) to be better than the functional mechanism. As a result, we choose to evaluate our method on *both small and medium-sized* datasets. Specifically, we construct logistic regression models on the diabetes dataset (768 samples, 8 dimensions) and construct linear regression models on the Warfarin dataset (~3000 samples, 14 dimensions).

The empirical results are encouraging — our approach produces substantially more accurate models than the functional mechanism of Zhang et al. Zhang et al. (2012). Specifically, for the logistic regression models, we obtain 60% – 80% classification accuracy when the privacy budget ε ranges from 0.1 to 4 (the baseline has 80% accuracy), while the functional mechanism only gives 40% – 60% accuracy. For the linear regression models, our mechanisms give up to **20x** smaller mean squared error and provides accurate models even for $\varepsilon = 0.1$, while the functional mechanism provide comparable models only after $\varepsilon \geq 5$.

Our results described thus far are positive, and follow in the tradition of a large body of research on differential privacy, which seeks to improve utility/privacy tradeoffs for various problems of interest. However, Fredrikson et al. Fredrikson et al. (2014) did not only consider differential privacy, they also considered an attack they term *model inversion*. As a simple example, consider a machine learning model w that takes features $x_1, \ldots, x_d$ and produces a prediction y. A model-inversion (MI) attack takes input $x_1, \ldots, x_{d-1}$ and a value y' that is related to y, and tries to predict x_d (thus "inverting the model"). For example, in Fredrikson et al. (2014), the authors consider the case where x_d is a genetic marker, y is the warfarin dosage, and $x_1, \ldots, x_{d-1}$ are general background information such as height and weight. They used an MI attack to predict an individual's

genetic markers based on his or her *warfarin dosage*, thus violating that individual's privacy.

In the final part of our chapter, we consider the impact of our improved privacy-utility tradeoff on MI attacks[2] . Here our results are less positive: in improving the privacy-utility tradeoff, we have increased the effectiveness of MI attacks. While unintended, upon deeper reflection this is not surprising: simply put, the improved privacy-utility tradeoff results in less noise added, and less noise added means the model is easier to invert. We formalize this discussion and prove that this phenomenon is quite general, and not an artifact of our approach or this specific learning task.

What this means for privacy is an open question. If one "only cares" about differential privacy, then the increased susceptibility to MI attacks is irrelevant. However, if one believes that MI attacks are significant (and anecdotal evidence suggests that some medical professionals are concerned about MI attacks), then the fact that improved differential privacy can mean worse MI exposure warrants further study. In this direction, our work indeed extends a long line of work that discusses the interaction of DP and *attribute privacy* Kifer and Machanavajjhala (2011); Lindell and Omri (2011); Reed et al. (2010), and gives a realistic application where misconceptions about DP can lead to unwanted disclosure. It is our hope that our work might highlight this issue and stimulate more discussion.

Our technical contributions can be summarized as follows:

[2] Note that specifically this is to measure the success rate of model inversion attacks over the training set. One may wonder that a more plausible privacy notion should compare the success rate of model inversion attacks on the training set with that on the validation training set. However, one should note that this falls exactly into the guarantees of differential privacy. In fact, the work of Fredrikson et al. (see Figure 1. "Disclosure, private LR") has also demonstrated that for $\varepsilon \leq 1$, the difference is not discernible. On the other hand, the work of Fredrikson et al. does consider using differential privacy to prevent MI attacks (under the theme of using differential privacy to protect "genomic privacy"). It is the purpose of this work to clarify that differential privacy is never designed to prevent such attacks.

- We continue the exploration about the connection between stability and differential privacy. By borrowing some machinery from stability theory, we prove that the simple output perturbation mechanism can learn *every* convex-Lipschitz-bounded learning problems with strong differential privacy. Our analysis relaxes the technical conditions required by Chaudhuri et al. Chaudhuri et al. (2011) for output perturbation, and achieves the same tradeoff between DP and generalization error as proved by Bassily et al. Bassily et al. (2014), and is much simpler than both.

- Since output perturbation is one of the simplest mechanisms to implement, it means that our method is widely applicable in practice. We go on to apply the theory to linear regression. We present and analyze regularized variants of linear regression, and give a detailed description on how to privately select parameters for regularization.

- We perform an empirical study on both small and medium-sized datasets, comparing the functional mechanism with our mechanisms to train logistic and linear regression models. Encouragingly, we observe a substantially better tradeoff between differential privacy and utility.

- Finally, we study the impact of the improved DP-utility tradeoff on model inversion attacks studied by Fredrikson et al. Fredrikson et al. (2014). We demonstrate that as we improve differentially-private mechanisms, MI attacks become more problematic. We provide a preliminary analysis of this intriguing phenomenon.

The rest of the chapter is organized as follows. We present the connection between differential privacy and stability theory in Section 4.2, and discuss the applications in Section 4.3. In Section 4.4, we compare output perturbation and the functional mechanism with respect to the privacy-utility or model-inversion efficacy tradeoff.

4.2 Differential Privacy and Stability Theory

In this section we present our results on the connection between differential privacy and stability theory. Our technical results can be summarized as follows:

DP implies Strongly-Uniform-RO Stability. In Section 4.2, we show that, in the setting of learning, differential privacy is a strong stability notion that implies strongly-uniform-RO stability. Strongly-Uniform-RO stability is the strongest stability notion proposed by Shalev-Shwartz et al. Shalev-Shwartz et al. (2010) from learning theory.

Norm Stability implies DP. In Section 4.2, we give a stability notion, ℓ_2-RO stability, that leads to differential privacy by injecting a small amount of noise. ℓ_2-RO stability is used implicitly in Shalev-Shwartz et al. (2009) to prove learnability of convex-Lipschitz-bounded problems under the condition that the instance loss function is smooth. Our analysis removes this requirement.

Simpler Mechanism with the Same Generalization Error. In recent work, Bassily et al. Bassily et al. (2014) give tight bounds (for both training and generalization errors) for differentially privately learning convex-Lipschitz-bounded problems. Their mechanisms require the exponential mechanism and a sophisticated sampling subprocedure. We show in Section 4.2 that the elementary output perturbation mechanism presented in Chaudhuri et al. (2011) can give the same tradeoff between differential privacy and generalization error (however with weaker training error) for *every* convex-Lipschitz-bounded learning problem. Our proof relaxes the technical requirements of Chaudhuri et al. (2011) (smoothness), and is significantly simpler than both Bassily et al. (2014); Chaudhuri et al. (2011).

Differential Privacy is a Stability Notion

If one writes out the definition of bounded differential privacy (Definition 2.15) in the language of learning, it becomes: a learning rule A is ε-differentially private if, for all training set S of size n, for all $i \in [n]$, and all $z' \in Z$, and *any event* E, it holds that $\Pr[A(S) \in E] \leq e^{\varepsilon} \Pr[A(S^{(i)}) \in E]$, where the probability is taken over the randomness of A. When we contrast this definition with strongly-uniform-RO stability (Definition 2.5), the only difference is that the latter considers a particular type of event, namely for $\bar{z} \in Z$, the magnitude of $\ell(\widetilde{A}(S), \bar{z})$. At this point, it is somewhat clear that differential privacy is essentially (yet another) stability notion. Nevertheless, it is so strong that it implies strongly-uniform-RO stability, as shown in the following:

Proposition 4.1. *Suppose that $|\ell(\cdot,\cdot)| \leq B$. Let $\varepsilon > 0$ and A be a randomized learning rule. If A is ε-differentially private, then it is strongly-uniform-RO stable with rate $\varepsilon_{\text{stable}} \leq B(e^{\varepsilon} - 1)$. Specifically, for $\varepsilon \in (0,1)$, this is approximately $B\varepsilon$.*

Proof. Let $f_S(w)$ be the probability density function of $A(S)$. Due to ε-differential privacy, then for any S, i, z', $f_S(w) \leq e^{\varepsilon} f_{S^{(i)}}(w)$. Therefore

$$\begin{aligned}
&\left|\mathbb{E}[\ell(\widetilde{A}(S), \bar{z})] - \mathbb{E}[\ell(\widetilde{A}(S^{(i)}), \bar{z})]\right| \\
=&\left|\int \ell(w, \bar{z})\Big(f_S(w) - f_{S^{(i)}}(w)\Big)dw\right| \\
\leq& \int |\ell(w, \bar{z})||f_S(w) - f_{S^{(i)}}(w)|dw \\
\leq& B \int |f_S(w) - f_{S^{(i)}}(w)|dw \qquad (1)
\end{aligned}$$

Note that $(e^{-\varepsilon} - 1)f_{S^{(i)}}(w) \leq f_S(w) - f_{S^{(i)}}(w) \leq (e^{\varepsilon} - 1)f_{S^{(i)}}(w)$, so $|f_S(w) - f_{S^{(i)}}(w)| \leq \max\{1 - e^{-\varepsilon}, e^{\varepsilon} - 1\} f_{S^{(i)}}(w)$. Plugging into (1) gives the first claimed

inequality. The second inequality follows from the observation that $e^{\varepsilon} + e^{-\varepsilon} \geq 2$. □

Two remarks are in order. First, this implication holds without assuming anything on the loss function ℓ except for boundedness. Second, one may note that the *converse* of this proposition, however, is not true in general. For example, consider the case where ℓ is a constant function and $A(S) = h_1 \neq h_2 = A(S^{(i)})$. Then A is strongly-uniform-RO stable (with rate 0!) yet it is clearly *not* differentially private. Moreover, this example indicates that, even if strongly-uniform-RO stability has been achieved, one cannot hope for differential privacy by adding a "small amount" of noise to the output of A. This is because h_1 and h_2 can be arbitrarily far away from each other so the sensitivity of A cannot be bounded. This motivates us to define another stability notion for the purpose of differential privacy.

Norm Stability and Noise for DP

In this section we present a different stability notion that does lead to differential privacy by injecting a small amount of noise. We then use some machinery from stability theory to quantify the amount of noise needed. Following our discussion above, a natural idea now is that $A(S)$ and $A(S^{(i)})$ shall be close *by themselves*, rather than being close *under the evaluation of some functions*. Because the output of A lies in $\mathcal{H}$, which is a normed space[3] as long as perturbation on), a "universal" notion for closeness is that $A(S)$ and $A(S^{(i)})$ are close in norm. This leads to the following definition.

Definition 4.2 (ℓ_2-RO Stability)**.** *A learning rule A is* ℓ_2-RO stable *with rate $\varepsilon(n)$, if for* any *$S \sim \mathcal{D}^n$, $z' \sim \mathcal{D}$ and $i \in [n]$, $\|A(S^{(i)}) - A(S)\|_2 \leq \varepsilon(n)$.*

[3] For simplicity, $\|\cdot\|$ refers to ℓ_2-norm in the rest of the chapter. Our results are applicable to other settings as long as perturbation is properly defined on the normed space.

Astute readers may realize that this is nothing more than a *rephrasing of the ℓ_2-sensitivity* of a query (Definition 2.16). Thus, in the spirit of the output perturbation method mentioned in Section 2.3, if one can bound ℓ_2-RO stability, then we only need to inject a small amount of noise for differential privacy.

If A is ℓ_2-RO stable, then adding a small amount of noise to its output ensures differential privacy, and thus strongly-uniform-RO stability. However, without the Lipschitz condition, the resulting hypothesis might be useless. This is because a small distance (in ℓ_2-norm) to $A(S)$ could give significant change in loss. This presents a barrier for proving learnability for the private mechanism. Thus in the following, we will restrict ourselves back to the setting where we assume that ℓ is convex and Lipschitz (in w).

We now move on to quantifying the amount of noise needed for differential privacy. Specifically, we will show that for strongly-convex learning tasks, the "scale of the noise" we need is roughly only $O_{d,\varepsilon}(1/n)$ where n is the training set size (the big-O notation hides a constant that depends on the number of features d, and the DP parameter ε.). This means that as training set size increases, the noise we need vanishes to zero for a fixed model and ε-DP. By contrast, for the functional mechanism, the "scale of the noise" is $O_{d,\varepsilon}(1)$. The following two lemmas are due to Shalev-Shwartz et al. Shalev-Shwartz et al. (2009). We include their proofs in the appendix for completeness.

Lemma 4.3 (Exchanging Lemma)**.** *Let A be a learning rule such that $A(S) = \arg\min_w \vartheta_S(w)$, where $\vartheta_S(w) = L_S(w) + \rho(w)$ and $\rho(w)$ is a regularizer. For any $S \sim \mathcal{D}^n$, $i \in [n]$ and $z' \sim \mathcal{D}$,*

$$\vartheta_S(u) - \vartheta_S(v) \leq \frac{\ell(v, z') - \ell(u, z')}{n} + \frac{\ell(u, z_i) - \ell(v, z_i)}{n},$$

where $u = A(S^{(i)})$ and $v = A(S)$.

Proof. By the definition of ϑ_S,

$$\begin{aligned}
&\vartheta_S(u) - \vartheta_S(v) \\
&= \left(L_{S^{(i)}}(u) + \rho(u) - \frac{1}{n}\ell(u, z') + \frac{1}{n}\ell(u, z_i)\right) \\
&\qquad - \left(L_{S^{(i)}}(v) + \rho(v) - \frac{1}{n}\ell(v, z') + \frac{1}{n}\ell(v, z_i)\right) \\
&= \vartheta_{S^{(i)}}(u) - \vartheta_{S^{(i)}}(v) + \frac{\ell(v, z') - \ell(u, z')}{n} + \frac{\ell(u, z_i) - \ell(v, z_i)}{n}.
\end{aligned}$$

Since u minimizes $\vartheta_{S^{(i)}}$ so $\vartheta_{S^{(i)}}(u) - \vartheta_{S^{(i)}}(v) \leq 0$, so

$$\vartheta_S(u) - \vartheta_S(v) \leq \frac{\ell(v, z') - \ell(u, z')}{n} + \frac{\ell(u, z_i) - \ell(v, z_i)}{n}$$

completing the proof. □

Intuitively, this lemma concerns about the behavior of a learning rule A on neighboring training sets. A is a regularized learning rule: Its objective function is in the form of empirical risk $L_S(w)$ plus regularization error $\rho(w)$ ($\rho(\cdot)$ is called a regularizer). More specifically, this lemma upper bounds *the difference between the objective values of u and v*, that is $\vartheta_S(u)$ and $\vartheta_S(v)$, in terms of instance losses on the specific two instances that get exchanged.

Recall that our goal is to upper bound $\|u - v\|$. The following lemma accomplishes this task by upper bounding the norm of the difference by the difference of the objective values of u and v.

Lemma 4.4. *Let A be a rule where $A(S) = \arg\min_w \vartheta_S(w)$ and $\vartheta_S(w)$ is λ-strongly convex in w. Then for any $S \sim \mathcal{D}^n$, $i \in [n]$ and $z' \sim \mathcal{D}$, $\frac{\lambda}{2}\|u - v\|^2 \leq \vartheta_S(u) - \vartheta_S(v)$, where $u = A(S^{(i)})$, $v = A(S)$.*

Proof. We have that for any $\alpha \in (0,1)$,

$$\begin{aligned}\vartheta_S(v) &\leq \vartheta_S(\alpha v + (1-\alpha)u) \\ &\leq \alpha\vartheta_S(v) + (1-\alpha)\vartheta_S(u) - \frac{\lambda}{2}\alpha(1-\alpha)\|v-u\|^2\end{aligned}$$

where the first inequality is because v is the minimizer of ϑ_S and the second inequality is by the definition of λ-strong convexity. By elementary algebra, this give that $\frac{\lambda}{2}\alpha\|v-u\|^2 \leq \vartheta_S(u) - \vartheta_S(v)$. Tending α to 1 gives the claim. □

To see this Lemma, we note that v is a *minimizer* of ϑ_S by the definition of the learning rule. Therefore by the definition of strong convexity, we have that for any $\alpha > 0$,

$$\begin{aligned}\vartheta_S(v) &\leq \vartheta_S((1-\alpha)v + \alpha u) \\ &\leq \alpha\vartheta(v) + (1-\alpha)\vartheta(u) - \frac{\lambda}{2}\alpha(1-\alpha)\|u-v\|^2\end{aligned}$$

The lemma is then proved by rearranging and tending α to 1. Intuitively, this lemma says that as long as the objective function $\vartheta_S(w)$ of A is good (that is, strongly convex), then one can upper bound the difference between u and v *in norm* by the difference between the *objective values* of u and v. Combining these two lemmas, we can prove the following main theorem in this section.

Theorem 4.5. *Let A be a learning rule with a λ-strongly convex objective loss function $\vartheta_S(w) = L_S(w) + \rho(w)$ where $\rho(w)$ is a regularizer. Assume further that for any $z \in Z$, $\ell(\cdot, z)$ is ρ-Lipschitz. Then A is $\frac{4\rho}{\lambda n}$ ℓ_2-RO stable.*

Proof. Let $u = A(S^{(i)})$, $v = A(S)$. By Lemma 4.4,

$$\frac{\lambda}{2}\|u-v\|^2 \leq \vartheta_S(u) - \vartheta_S(v) \tag{1}$$

By Lemma 4.3,

$$\vartheta_S(u) - \vartheta_S(v) \leq \frac{\ell(v,z') - \ell(u,z')}{n} + \frac{\ell(u,z_i) - \ell(v,z_i)}{n} \tag{2}$$

Now because $\ell(\cdot, z)$ is ρ-Lipschitz, we have that $\ell(v,z') - \ell(u,z') \leq \rho\|v-u\|$, and $\ell(u,z_i) - \ell(v,z_i) \leq \rho\|u-v\|$. Plugging these two inequalities to (2), we have that $\vartheta_S(u) - \vartheta_S(v) \leq \frac{2\rho}{n}\|u-v\|$. Plugging this to (1) and rearranging completes the proof. □

This theorem says that if both the *instance loss function* and the *objective function* are well behaved (ℓ is ρ-Lipschitz and ϑ is strongly convex), then the learning rule A is roughly $(1/n)$-norm stable (in other words, the sensitivity is $1/n$, which vanishes to 0 as training set size n grows). In the case when ℓ is already λ-strongly convex, we do not need a regularizer so one can set $\rho(w) = 0$, hence a natural algorithm to ensure differential privacy in this case is to directly perturb the empirical risk minimizer with noise calibrated to its norm stability. Our discussion so far thus leads to Algorithm 5

Algorithm 5 Output Perturbation for strongly-convex loss function: $\ell(w,z)$ is λ-strongly convex and ρ-Lipschitz in w, for every $z \in Z$.

Input: Privacy budget: $\varepsilon_p > 0$. Training set $S = \{(x_i, y_i)\}_{i=1}^n$.

1: **function** OutputPerturbationStronglyConvex(S, ε_p)
2: Solve the empirical risk minimization $\bar{w} = \arg\min_w L_S(w)$.
3: Draw a noise vector $\kappa \in \mathbb{R}^d$ according to a distribution with density function $p(\kappa) \propto \exp\left(-\frac{\lambda n \varepsilon_p \|\kappa\|_2}{4\rho}\right)$.
4: **return** $\bar{w} + \kappa$

Theorem 4.6. *Let $(Z, \mathcal{H}, \ell)$ be a learning problem where ℓ is λ-strongly convex and ρ-Lipschitz. Then Algorithm 5 is ε-differentially private.*

To see this, we note that $L_S(w)$ is λ-strongly convex because ℓ is, so we can set the objective function $\vartheta(w) = L_S(w)$. $\vartheta(w)$ is λ-strongly convex and ρ-Lipschitz, so Theorem 4.5 bounds its ℓ_2-norm stability. The proof is then completed by plugging the stability bound into Theorem 2.17.

If the loss function is only convex (instead of being strongly convex), then the idea is to use a strongly convex regularizer to make the objective function strongly convex. Specifically, we use the Tikhonov regularizer $\rho(w) = \lambda\|w\|^2/2$ (indeed, any strongly convex regularizer applies). This gives Algorithm 6.

Algorithm 6 Output Perturbation for general-convex loss function: $\ell(w,z)$ is convex and ρ-Lipschitz in w, for every $z \in Z$.

Input: Privacy budget: $\varepsilon_p > 0$. Regularization parameter: $\lambda > 0$. Boundedness parameter: $R > 0$. Training data: $S = \{(x_i, y_i)\}_{i=1}^n$.

1: **function** OutputPerturbationConvex($S, \lambda, \varepsilon_p, R$)

2: Solve the regularized empirical risk minimization problem $\bar{w} = \arg\min_{\|w\|\leq R}\left(L_S(w) + \frac{\lambda}{2}\|w\|^2\right)$.

3: Draw a noise vector $\kappa \in \mathbb{R}^d$ according to a distribution where $p(\kappa) \propto \exp\left(-\frac{\lambda n \varepsilon_p \|\kappa\|_2}{4(\rho+\lambda R)}\right)$.

4: **return** $\bar{w} + \kappa$

Theorem 4.7. *Let $(Z, \mathcal{H}, \ell)$ be a learning problem where ℓ is convex and ρ-Lipschitz. Suppose further that the hypothesis space is $\mathcal{H}$ is R-bounded. Then Algorithm 6 is ε-differentially private.*

The easiest way to see this theorem is to define a new loss function $\bar{\ell}(w,z) = \ell(w,z) + (\lambda\|w\|^2)/2$. Then $\bar{\ell}$ is λ-strongly convex and $(\rho + \lambda R)$-Lipschitz over $\mathcal{H}$, and the theorem directly follows from Theorem 4.6. We remark that Algorithm 6 and Theorem 4.7 can be strengthened based on Theorem 4.5 and specifically do not rely on the boundedness condition. However, since our analysis of generalization error for this case (in the

next section) critically relies on the boundedness condition, the algorithm and its privacy guarantee are stated in the current form.

Generalization Error

Until now, we have only talked about ensuring differential privacy with a small amount of noise, and have not said anything about whether this small amount of noise will lead to a model with "good utility", which is usually measured by *generalization error* in learning theory. We accomplish this in this section. At a high level, we will show that for strongly convex learning tasks, output perturbation produces hypotheses that are roughly $(1/n)$-away from the optimal. For general convex learning tasks, this degrades to roughly $1/\sqrt{n}$.

For notational convenience, throughout this section we let A denote a deterministic learning mechanism, and $\widetilde{A}$ be its output perturbation counterpart. We begin with a general lemma.

Lemma 4.8. *Suppose that for any $z \sim \mathcal{D}$, $\ell(w, z)$ is ρ-Lipschitz in w. If with probability at least $1 - \gamma$ over $w \sim \widetilde{A}(S)$,*

$$\|w - A(S)\|_2 \leq \kappa(n, \gamma),$$[4]

then with probability at least $1 - \gamma$ over $w \sim \widetilde{A}(S)$,

$$|L_{\mathcal{D}}(w) - L_{\mathcal{D}}(A(S))| \leq \rho\kappa(n, \gamma).$$

[4] We abuse the notation κ to remind readers that this quantity is related to the noise vector.

Proof. We have

$$\begin{aligned} |L_{\mathcal{D}}(w) - L_{\mathcal{D}}(A(S))| &= |\mathop{\mathbb{E}}_{z\sim\mathcal{D}}[\ell(w,z) - \ell(A(S),z)]| \\ &\leq \mathop{\mathbb{E}}_{z\sim\mathcal{D}}[|\ell(w,z) - \ell(A(S),z)|] \end{aligned}$$

For every $z \in Z$, $\ell(\cdot, z)$ is ρ-Lipschitz, so $|\ell(w,z) - \ell(A(S),z)| \leq \rho\|w - A(S)\|_2$. Therefore

$$\mathop{\mathbb{E}}_{z\sim\mathcal{D}}[|\ell(w,z) - \ell(A(S),z)|] \leq \rho\|w - A(S)\|_2.$$

The proof is complete by observing that with probability at least $1-\gamma$ over $w \sim \widetilde{A}(S)$, $\|w - A(S)\|_2 \leq \kappa(n,\gamma)$. □

This lemma translates the closeness between two hypotheses *in norm* to the closeness in *generalization error*. More specifically, note that the randomized learning rule $\widetilde{A}$ induces a *distribution* $\widetilde{A}(S)$ over the hypothesis space. This lemma says as long as a hypothesis sampled from $\widetilde{A}(S)$ is close to $A(S)$ (a single hypothesis) *in norm*, then these two hypotheses are close in their generalization error. Note that this closeness is controlled by the Lipschitz constant of the instance loss function ℓ.

Combing this lemma with Theorem 4.5 from the last section, which bounds the norm stability, we have the following general theorem upper bounding the generalization error of our method.

Theorem 4.9. *Let $(\mathcal{H}, Z, \ell)$ be a learning problem that is agnostically learnable by a deterministic learning algorithm A with rate $\varepsilon(n,\delta)$. Suppose that $\ell(w,z)$ is ρ-Lipschitz in w for any $z \in Z$. Let $\mathcal{D}$ be a distribution over Z. Finally, suppose that for any $S \sim \mathcal{D}^n$, with probability at least $1-\gamma$ over $w \sim \widetilde{A}(S)$,*

$$\|w - A(S)\|_2 \leq \kappa(n,\gamma).$$

Then with probability at least $1-\delta-\gamma$ *over* $S \sim \mathcal{D}^n$ *and* $w \sim \widetilde{A}(S)$*, we have* $L_{\mathcal{D}}(w) - L^*_{\mathcal{D}} \leq \varepsilon(n,\delta) + \rho\kappa(n,\gamma)$.

Proof. For any $S \sim \mathcal{D}^n$, we have that $L_{\mathcal{D}}(w) - L^*_{\mathcal{D}} = \Big(L_{\mathcal{D}}(w) - L_{\mathcal{D}}(A(S))\Big) + \Big(L_{\mathcal{D}}(A(S)) - L^*_{\mathcal{D}}\Big)$. For $L_{\mathcal{D}}(A(S)) - L^*_{\mathcal{D}}$, we know that $\Pr_{S\sim\mathcal{D}^n}\Big[L_{\mathcal{D}}(A(S)) - L^*_{\mathcal{D}} > \varepsilon(n,\delta)\Big] < \delta$. Further, for every $S \sim \mathcal{D}^n$, from Lemma 4.8, $\Pr_{w\sim\widetilde{A}(S)}\Big[L_{\mathcal{D}}(w) - L_{\mathcal{D}}(A(S)) > \rho\kappa(n,\gamma)\Big] < \gamma$. The proof is complete by a union bound. □

In the rest of this section we give concrete bounds for different types of instance loss function.

Strongly Convex Loss. We now bound generalization error of our method for the case where the instance loss function ℓ is strongly convex. We will use the following theorem from stability theory:

Theorem 4.10 (Theorem 6, Shalev-Shwartz et al. (2009))**.** *Consider a learning problem such that* $\ell(w,z)$ *is* λ*-strongly convex and* ρ*-Lipschitz in* w*. Then for any distribution* $\mathcal{D}$ *over* Z *and any* $\delta > 0$*, with probability at least* $1-\delta$ *over* $S \sim \mathcal{D}^n$*,*

$$L_{\mathcal{D}}(\mathsf{ERM}(S)) - L^*_{\mathcal{D}} \leq \frac{4\rho^2}{\delta\lambda n}.$$

where $\mathsf{ERM}(S)$ *is the empirical risk minimizer, i.e.* $\mathsf{ERM}(S) = \arg\min_{w\in\mathcal{H}} L_S(w)$.

We are ready to prove the following theorem on generalization error. Our bound matches the bound obtained by Bassily et al. for the same setting (Theorem F.2, Bassily et al. (2014)).

Theorem 4.11. *Consider a learning problem such that* $\ell(w,z)$ *is* λ*-strongly convex and* ρ*-Lipschitz in* w*. Let* $\varepsilon_p > 0$ *be a privacy parameter for differential privacy. Let* $\widetilde{A}$ *denote Algorithm 5. Then for any* $\delta \in (0,1)$*, with probability*

$1-\delta$ *over* $S \sim \mathcal{D}^n$ *and* $w \sim \widetilde{A}(S)$,

$$L_{\mathcal{D}}(w) - L^*_{\mathcal{D}} \leq O\Big(\frac{\rho^2 d \ln(d/\delta)}{\lambda n \delta \varepsilon_p}\Big).$$

Proof. Let A denote the rule of empirical risk minimization, and $\widetilde{A}$ be its output-perturbation counter-part which ensures ε_p-differential privacy. Then by Theorem 4.5, and corollary 2.18, with probability at least $1-\gamma$ over $w \sim \widetilde{A}(S)$, $\|w - A(S)\|_2 \leq \frac{4d\ln(d/\gamma)\rho}{\lambda n \varepsilon_p}$.

Together with Theorem 4.10, it follows that with probability at least $1-\delta'-\gamma$ over $S \sim \mathcal{D}^n$ and $w \sim \widetilde{A}(S)$,

$$L_{\mathcal{D}}(w) - L^*_{\mathcal{D}} \leq \frac{4\rho^2}{\delta' \lambda n} + \frac{4d\ln(d/\gamma)\rho^2}{\lambda n \varepsilon_p}.$$

Put $\delta' = \gamma = \delta/2$, thus with probability at least $1-\delta$,

$$L_{\mathcal{D}}(w) - L^*_{\mathcal{D}} \leq \frac{4\rho^2}{\lambda n}\Big(\frac{2}{\delta} + \frac{4d\ln(2d/\delta)}{\varepsilon_p}\Big).$$

Asymptotically this is $O\Big(\frac{\rho^2 d \ln(d/\delta)}{\lambda n \delta \varepsilon_p}\Big)$. The proof is complete. □

Basically, this theorem says that if the instance loss function is strongly convex, then with high probability, the generalization error of a hypothesis sampled using our method is only roughly $1/n$-away from the "optimal" $(L^*_{\mathcal{D}})$.

General Convex Loss. We now consider the general case where ℓ is only convex. We have the following theorem on the generalization error, which matches the bound obtained in Bassily et al. (2014) (Theorem F.3),

Theorem 4.12. *Consider a convex, ρ-Lipschitz learning problem that is also R-bounded. Let $\varepsilon_p > 0$ be a privacy parameter for differential privacy. Let $\widetilde{A}$ denote Algorithm 6. Then for any $\delta \in (0,1)$, with probability $1-\delta$ over $S \sim \mathcal{D}^n$*

and $w \sim \widetilde{A}(S)$,

$$L_{\mathcal{D}}(w) - L^*_{\mathcal{D}} \leq O\left(\frac{\rho R\sqrt{d\ln(d/\delta)}}{\sqrt{n\delta\varepsilon_p}}\right).$$

Proof. Let $\bar{\ell}$ be defined as $\bar{\ell}(w,z) = \ell(w,z) + \frac{\lambda}{2}\|w\|^2$. $\bar{\ell}$ is λ-strongly convex and $(\rho + \lambda R)$-Lipschitz. Let $\bar{L}_S$ and $\bar{L}_{\mathcal{D}}$ be the empirical loss and true loss functions with respect to $\bar{\ell}$. Note that for any $w \in \mathcal{H}$, $\bar{L}_{\mathcal{D}}(w) = L_{\mathcal{D}}(w) + \frac{\lambda}{2}\|w\|^2$.

By Theorem 4.11, there is an ε_p-differentially private mechanism $\widetilde{A}$ such that with probability $1-\delta$ over $S \sim \mathcal{D}^n$ and $w \sim \widetilde{A}(S)$,

$$\bar{L}_{\mathcal{D}}(w) - \bar{L}^*_{\mathcal{D}} \leq O\left(\frac{(\rho + \lambda R)^2 d\ln(d/\delta)}{\lambda n\delta\varepsilon_p}\right) \tag{1}$$

Let $\bar{w} \in \mathcal{H}$ such that $\bar{L}_{\mathcal{D}}(\bar{w}) = \bar{L}^*_{\mathcal{D}}$ and $w^* \in \mathcal{H}$ such that $L^*_{\mathcal{D}} = L_{\mathcal{D}}(w^*)$. Because $\bar{w}$ is the minimizer of $\bar{L}_{\mathcal{D}}(\cdot)$, so

$$L_{\mathcal{D}}(w^*) + \frac{\lambda}{2}\|w^*\|^2 \geq L_{\mathcal{D}}(\bar{w}) + \frac{\lambda}{2}\|\bar{w}\|^2 \tag{2}$$

Combining (1) and (2) we have

$$\begin{aligned} L_{\mathcal{D}}(w) - L^*_{\mathcal{D}} &\leq O\left(\frac{(\rho + \lambda R)^2 d\ln(d/\delta)}{\lambda n\delta\varepsilon_p}\right) + \frac{\lambda}{2}\left(\|w^*\|^2 - \|w\|^2\right) \\ &\leq O\left(\frac{(\rho + \lambda R)^2 d\ln(d/\delta)}{\lambda n\delta\varepsilon_p}\right) + \frac{\lambda R^2}{2} \\ &\leq O\left(\frac{2\rho R d\ln(d/\delta)}{n\delta\varepsilon_p} + \frac{\rho^2 d\ln(d/\delta)}{\lambda n\delta\varepsilon_p} + \lambda R^2\right) \end{aligned}$$

where the second inequality is because the hypothesis space is R-bounded. Putting $\lambda = \frac{\rho}{R}\sqrt{\frac{d\ln(d/\delta)}{n\delta\varepsilon_p}}$ gives the claimed bound. □

Note that for convex loss functions (instead of strongly convex ones),

we are only roughly $1/\sqrt{n}$-away from the optimal.

We notice that regularization plays a vital role in this theoretical guarantee. Indeed, the key to prove Theorem 4.12 is the use of the Tikhonov regularizer ($\lambda\|w\|^2/2$) to gain *stability*. By contrast, regularization is only used as a heuristic in the analysis of the functional mechanism Zhang et al. (2012).

4.3 On Applications and Previous Work

We now give concrete applications of our results. Due to lack of space, we only describe two important examples, Support Vector Machine (SVM) and Logistic Regression. As we will see, our analysis enables us to train an SVM *without* approximating the loss function using a smooth loss function that gives higher or point-wise equal loss.

Support Vector Machine (SVM). In SVM we use the *hinge loss function*. Specifically, given hypothesis space $\mathcal{H} \subseteq \mathbb{R}^d$, feature space $X \subseteq \mathbb{R}^d$, and output space $Y = \{0, 1\}$, then for $(x, y) \in X \times Y$, the hinge loss is defined as

$$\ell^{\text{hinge}}(w, x, y) = \max\Big\{0, y(1 - \langle w, x\rangle)\Big\}.$$

In the common setting where X is scaled to the unit sphere, it is straightforward to verify that ℓ^{hinge} is convex 1-Lipschitz. Therefore our Algorithm 6 and Theorem 4.12 directly apply to provide differential privacy with a small generalization error. We note that, however, hinge loss is *not* differentiable. Therefore Chaudhuri et al.'s analysis for output perturbation does not directly apply. Indeed, in order to use their method, they need to use a smooth loss function to approximate the hinge loss by giving higher or equal point-wise loss.

Logistic Regression. For Logistic Regression we have $\mathcal{H} \subseteq \mathbb{R}^d$, $X \subseteq \mathbb{R}^d$ and $Y = \{\pm 1\}$. Then for $y \in Y$ and $x \in X$, the loss of w on (x, y) is defined

to be

$$\ell^{\log}(w, x, y) = \log\left(1 + e^{-y\langle w,x\rangle}\right).$$

It is not hard to verify that $\ell^{\log}$ is convex and differentiable with bounded derivatives. Therefore both our analysis and Chaudhuri et al.'s analysis directly apply.

More Comparison with Chaudhuri et al. Chaudhuri et al. (2011). We note that Chaudhuri et al. Chaudhuri et al. (2011) have given an output perturbation mechanism that is essentially the same as our Algorithm 6. Moreover, they give an *objective perturbation* algorithm. The main difference is that the noise is added to the objective function, instead of the output. We refer interested readers to their work for more details.

Our work differs in analysis and applicability. Specifically, to get their claimed generalization bounds Chaudhuri et al. require differentiability and smoothness for their output perturbation, and require twice differentiability with bounded derivatives for their objective perturbation (see Theorem 6 and Theorem 9 in Chaudhuri et al. (2011)). Our analysis removes all these differentiability conditions, and demonstrates that *output perturbation* works well for *all* convex-Lipschitz-bounded learning problems. This is by far the largest class of convex learning problems that are known to be learnable (see Shalev-Shwartz and Ben-David (2014)).

More Comparison with Bassily et al. Bassily et al. (2014). We observe that Bassily et al. (2014); Chaudhuri et al. (2011), as well as our work, are all based on regularized learning using the Tikhonov regularizer ($\lambda\|w\|^2/2$, where λ is the regularization parameter). For fixed λ we note that Bassily et al. Bassily et al. (2014) achieve better training error (the generalization error remains the same). However, their mechanism is substantially more complicated. Specifically, their mechanism is the exponential mechanism with a sophisticated sampling procedure to give polynomial running time. We note that there have also been practical

concerns in implementing the exponential mechanism Hardt (2015). Also, the sampling procedure Applegate and Kannan (1991) runs in time $O(n^3)$, where n is the training set size. This can be prohibitive for reasonably large data sets. For these two reasons, in this chapter we will not empirically compare with Bassily et al..

4.4 Empirical Study

In this section we empirically evaluate our method. Our main empirical question is the following:

"How does the accuracy of models produced using our mechanisms compare to the functional mechanism?"

To answer this question, we note that as the training set size increases the magnitude of noise using our method vanishes to 0, while the functional mechanism always requires constant noise. Thus for sufficiently large training set, our method is guaranteed (theoretically) to be better than the functional mechanism. However, it is still possible that for small datasets the functional mechanism can be better than ours.

Therefore, in our empirical study we choose to evaluate on *on both small and medium-sized* datasets. For small datasets, we use the diabetes[5], which only has 768 training samples. For medium-sized datasets, we use the same Warfarin dataset that is used by Fredrikson et al. Fredrikson et al. (2014), which has about 3000 patient records. We build logistic regression models to do binary classification on the diabetes dataset and build linear regression models to predict dosage on the Warfarin dataset.

A Summary. We find that our mechanisms provided significantly better accuracy than the functional mechanism for a given ϵ setting. Specifically,

[5] We use a preprocssed version of the diabetes dataset (https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#diabetes). The original dataset can be found at UCI (https://archive.ics.uci.edu/ml/datasets/Diabetes).

for logistic regression, we obtain 60% – 80% classification accuracy for ε ranging from 0.1 to 4 (the baseline has 80% accuracy), while the functional mechanism only gives 40% – 60% accuracy (note that a random guessing classifier gives 50% accuracy in expectation). For linear regression, we obtain accurate models with ε-DP even for $\varepsilon = 0.1$, whereas the functional mechanism does not provide comparable models until $\varepsilon \geq 5$.

The rest of the section is organized as follows. We first describe the experimental methodology in Section 4.4. Then we present our results on model accuracy in Section 4.4. Finally, in Section 4.4 we consider the relationship between DP and model inversion attacks, a new privacy attack proposed in Fredrikson et al. Fredrikson et al. (2014).

Experimental Methodology

We now give more details on our experimental methodology. This section has four parts. We first describe algorithms considered in our experiments and how to set parameters for them. Then we describe the datasets we use. Finally, we give more details on how we evaluate the algorithms. We consider two goals: (i) *model accuracy*, for both linear regression and logistic regression; (ii) *model inversion attacks*, a new privacy attack proposed in Fredrikson et al. (2014), for linear regression.

We now describe algorithms considered in our experiments, To this end, because we use L_2-regularization, we have to restrict the hypothesis space to obtain learnability. For both logistic regression and linear regression, we restrict the hypothesis space to be a ball of radius R centered at the origin. Therefore we have to set two parameters, λ and R, for these two learning tasks.

Oracle Output Perturbation. This is the *idealized* version of the output perturbation mechanism, where we exhaustively search for the best parameters using the training data and then use them in as if they were

"constants". We consider this variant because it shows the best possible result one can obtain using output perturbation.

We search R exhaustively within the space $\{0.25, 0.5, 1, 2\}$. To search for a good λ, we consider an arithmetic progression starting at initial value $(d/n\varepsilon_p')^{1/2}$ and ending at $(d/n\varepsilon_p^*)^{1/2}$, where $\varepsilon_p' = 100$ and $\varepsilon_p^* = 0.1$. The start and end values are determined by a computation based on our proof of Theorem 4.12.

Data-Independent Output Perturbation. Based on the proof of Theorem 4.12, We determine λ, R through a computation so that they are independent of the training data.

We put $R = 1$. This is because we scale the data to the unit sphere, and the coefficients of the linear regression model are all small when trained over the scaled data. For λ, the proof of Theorem 4.12 indicates that λ is approximately $\sqrt{d/n\varepsilon_p}$.

Privately-Tuned Output/Objective Perturbation. Chaudhuri et al. Chaudhuri et al. (2011) used private parameter tuning to pick data-dependent parameters for both of their output/objective perturbation. Fortunately, the private tuning algorithm only makes black-box use of the output/objective perturbation. Algorithm 7 describes the private-tuning framework. Given ε_p, the tuning algorithm ensures ε_p-DP.

Let $\mathcal{R}$ be a set of l choices of R and Λ be a set of k choices of λ. Let $m = l \cdot k$, and suppose the pairs of parameters can be listed as $\{(R_1, \lambda_1), \ldots, (R_m, \lambda_m)\}$. The more settings of parameters to try, larger the m is, and so we have smaller chunks for training, thus more entropy in the probability in picking the final hypothesis. Therefore, we want to have a small set of good parameters. For Λ, instead of using an arithmetic progression as in the oracle variant, we now use a geometric progression of ratio 2. For R, we try two sets $\mathcal{R}_1 = \{.25, .5, 1\}$, and $\mathcal{R}_2 = \{.5, 1, 2\}$.

Datasets. As we mentioned before, we consider two datasets: diabetes,

Algorithm 7 Parameter Tuning Algorithm: it accesses a privacy-preserving training algorithm as a black box.

Input: Privacy budget: $\varepsilon_p > 0$. Training data: $S = \{(x_i, y_i)\}_{i=1}^n$, where $\|x_i\| \leq 1$, $|y_i| \leq 1$. Parameter space: $\mathcal{R} = \{R_1, \ldots, R_l\}$, where $\max_{1 \leq i \leq l} R_i \leq R$, $\Lambda = \{\lambda_1, \ldots, \lambda_k\}$.

1: **function** ParameterTuning($S, \mathcal{R}, \Lambda$)

2: $\mathcal{R} \times \Lambda \leftarrow (R_i, \lambda_i)\}_{i=1}^m$

3: Divide the training data into $m+1$ chunks, $S_1, \ldots, S_m, S'$. $S_1, \ldots, S_m$ are used for training, and S' is used for validation.

4: For each $i = 1, 2, \ldots, m$, apply a privacy-preserving algorithm to train w_i (for example output or objective perturbation) with parameters $\varepsilon_p, \lambda_i, R_i, S_i$. Evaluate w_i on S' to get utility $u_{S'}(w_i) = -L_{S'}(w_i)$.

5: Pick a w_i in $\{w_1, \ldots, w_m\}$ using the exponential mechanism with privacy budget ε_p and utility function $u_{S'}(w_i)$. Note that the sensitivity of u is

$$\Delta(u) = \max_{w \in \mathcal{H}} \max_{S,i,z'} |u_S(w) - u_{S^{(i)}}(w)| \leq R^2.$$

Thus this amounts to sampling w_i with probability

$$p(w_i) \propto \exp\left(-\frac{L_{S'}(w_i)\varepsilon_p}{2R^2}\right).$$

a small dataset from the UCI repository, and the Warfarin dataset, a medium-sized dataset considered in Fredrikson et al. Fredrikson et al. (2014). For the Warfarin dataset, the data was collected by the International Warfarin Pharmacogenetics Consortium (IWPC), and contains information pertaining to the age, height, weight, race, partial medical history, and two genomic SNPs: VKORC1 and CYP2C9. The outcome variable corresponds to the stable therepeutic dose of warfarin, defined as the steady-state dose that led to stable anticoagulation levels. At the time of collection, this was the most expansive database of information relevant to pharmacogenomic warfarin dosing. We refer the reader to the original IWPC paper International Warfarin Pharmacogenetic Consor-

tium (2009) and the paper by Fredrikson et al. Fredrikson et al. (2014) for more information about the data and how it was preprocessed.

Model Accuracy. For both diabetes and Warfarin datasets we consider the question of model accuracy. Specifically, we build logistic regression models to classify whether a patient has diabetes or not. For the Warfarin dataset, we examine linear warfarin dosing models trained on the data. For binary classification, we measure the model accuracy by measuring its *classification accuracy*. For linear regression, we consider the *mean squared error* of the regression.

Model Inversion Attacks. While our main empirical question is about model accuracy, we note that for the Warfarin dataset and linear regression models, Fredrikson et al. Fredrikson et al. (2014) also conducted a detailed evaluation on model inversion attacks (MI attacks), a new kind of privacy attacks, on the resulting model. In short, Fredrikson et al. used a model inversion algorithm to predict VKORC1 (a genetic feature) for each patient based on the Warfarin dosage and partial information of other features. Since we improve the tradeoff between differential privacy and utility, it is thus a sensible goal to consider the impact of the improved tradeoff on MI attacks.

Model Accuracy

Diabetes Dataset. Figure 4.1 gives the results. For the diabetes dataset, we only report the results for the data independent output perturbation algorithm, because all other private variants have very similar performance. (except with the functional mechanism). Our private algorithm gives significantly better accuracy than the functional mechanism, for all ε considered.

Warfarin Dataset. Figure 4.2 compares Functional Mechanism with all the private output perturbation mechanisms, which includes Tuned Out-

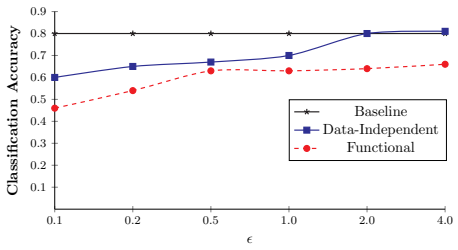


Figure 4.1: Classification accuracy on the diabetes dataset (768 samples, 8 features). It is a binary classification task. We consider the classification accuracy (#correctly classified samples)/(#samples in the test set). Thus, larger accuracy is better. In this figure, the baseline has accuracy 80%. We range the privacy budget ε in $\{0.1, 0.2, 0.5, 1.0, 2.0, 4.0\}$. Our method outperforms the functional mechanism significantly – with accuracy from 60% – 80%, while the functional mechanism only gives 40% – 60%. Note that a random guessing algorithm gives 50% accuracy in expectation.

put Perturbation (Out), Data-Independent Output Perturbation (Data Independent), and Oracle Output Perturbation (Oracle). We also include the model accuracy of the non-private algorithm (Non-Private). We observe that all the output perturbation give much better model accuracy compared to the functional mechanism, especially for small ε. Specifically, Tuned Output Perturbation obtains accurate models at $\varepsilon = 0.3$. Data-Independent and the Oracle Mechanisms give much the same model accuracy, and provide accurate models even for $\varepsilon = 0.1$. In particular, the accuracy is very close to the models produced by the non-private algorithm.

Figure 4.3 further compares the performance of Data-Independent and Oracle mechanisms and demonstrate their closeness. For the entire parameter range considered, the maximum MSE gap we observed is only 0.1. Figure 4.4 further compares the risk of mortality, hemorrhage, and stroke using Functional Mechanism, Tuned Output Perturbation and Data-Independent Output Perturbation. unsurprisingly, Data-Independent Output Perturbation gives the best result, and in particular much smaller

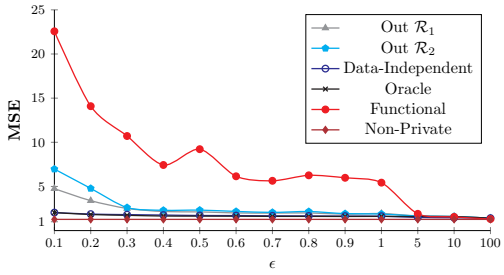


Figure 4.2: Model Accuracy: compare the functional mechanism with our output perturbation method, as well as the non-private mechanism. $\mathcal{R}_1$ stands for the Private-Tuned mechanism where we try the bounded hypothesis space with radius R in $\{.25, .5, 1\}$. $\mathcal{R}_2$ stands for the same mechanism with radius in $\{.5, 1, 2\}$. Out indicates Privately Tuned Output Perturbation methods. Oracle (Data-Independent) stands for the Oracle (Data-Independent) Output Perturbation.

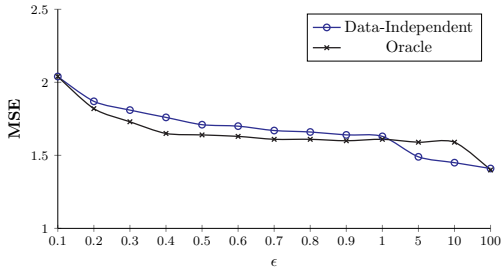
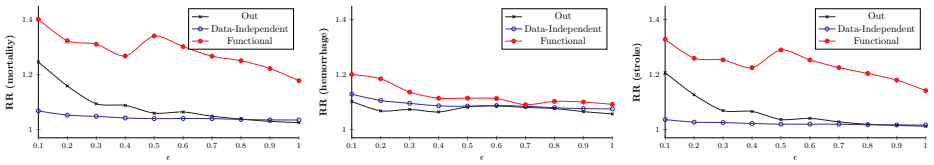


Figure 4.3: Model Accuracy: Further comparison between Data-Independent Output Perturbation and Oracle Output Perturbation. The maximum MSE gap we observed is only 0.1.

risk than Functional Mechanism.

Finally, we also compare with two other private algorithms.

Projected Histogram. We notice that in the previous work of Fredrikson et al. Fredrikson et al. (2014), they have implemented private projected histogram mechanism and compared it with the functional mechanism on linear regression. Specifically, as Figure 6 (Section 5.2) of their paper shows, the projected-histogram algorithm indeed has similar model accu-



(a) Mortality (b) Bleeding (c) Stroke

Figure 4.4: We estimate the risk of mortality, hemorrhage, and stroke using the approach described by Fredrikson et al.

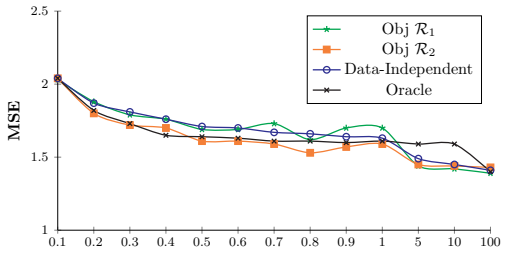


Figure 4.5: Model Accuracy: Objective Perturbation, Data-Independent Output Perturbation and Oracle Output Perturbation. $\mathcal{R}_1$ stands for the Private-Tuned mechanism where we try the bounded hypothesis space with radius R in $\{.25, .5, 1\}$. $\mathcal{R}_2$ stands for the same mechanism with radius in $\{.5, 1, 2\}$.

racy compared to the functional mechanism, which is much worse than our Data-Independent algorithm (of which the accuracy is close to that of the non-private algorithm).

Objective Perturbation. We have so far mainly compared with the function mechanism, which we believe is the most important task, because the functional mechanism has become the recognized state of the art for training regression models and has been adopted by many research teams, including Aono et al. (2015); Wang et al. (2015); Winslett et al. (2012). On the other hand, we notice that under very restricted conditions, it is known that Chaudhuri et al.'s objective perturbation method Chaudhuri et al. (2011) can provide very good model accuracy. Interestingly, because we impose boundedness condition for linear regression, the technical con-

ditions of objective perturbation are satisfied. Therefore we also compare it with our method. Encouragingly and perhaps somewhat surprisingly, while our method is much more widely applicable (see discussion in Section 4.3), our experiments show that our general algorithm performs as well as objective perturbation. Specifically, Figure 4.5 compares the model accuracy of these three methods, and the maximal gap we observed is only 0.1!

Model Inversion

Improving model utility for a given ε is a theme shared by nearly all previous work on differential privacy. In the final part of our empirical study, we study the impact of the improved DP-utility tradeoff on MI attacks, a new kind of privacy attacks first raised by Fredrikson et al. Fredrikson et al. (2014). This is a sensible goal, because utility has no direct bearing on the privacy guarantee provided by *differential-privacy*—two models can differ significantly on the level of utility the provide, while still conferring the same level of differential privacy. However, we show that MI is orthogonal to differential privacy in this sense, because the improved utility offered by our mechanisms leads to more successful MI attacks.

Better DP mechanisms, more effective MI attacks. Figure 4.6 compares MI accuracy of all the private mechanisms. For all these mechanisms, we see that mechanisms with better DP-utility tradeoff also has higher MI accuracy. Specifically, For Oracle Output Perturbation at $\varepsilon = 0.2$, we see a significant increase in MI accuracy: 45% for Oracle compared to 35% for the functional mechanism. Meanwhile, the utility of our mechanism is much better, with mean squared error 1.82 compared to the functional mechanism's 22.57. This phenomenon holds for larger ε, although the magnitude of the differences gradually shrink.

Figure 4.7 further demonstrates that, similar to their model accuracy,

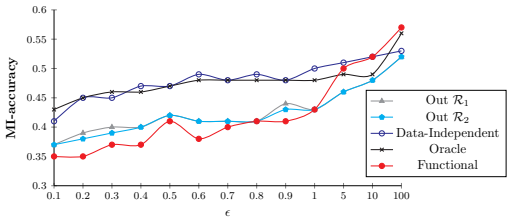


Figure 4.6: MI attack: Output Perturbation vs. Functional Mechanism. The x-axis is ε. The y-axis is the accuracy of MI attack. MI attacks become more effective for all three variants. For Oracle Output Perturbation and Functional Mechanism at $\varepsilon = .2$, the MI attack accuracy for the oracle mechanism is 45%, while is only 35% for the functional mechanism. Data-Independent and Oracle Output Perturbation are similar.

Data-Independent and Oracle Output Perturbation have very similar behavior on model invertibility (note that they both provide similar model accuracy that is better than other methods).

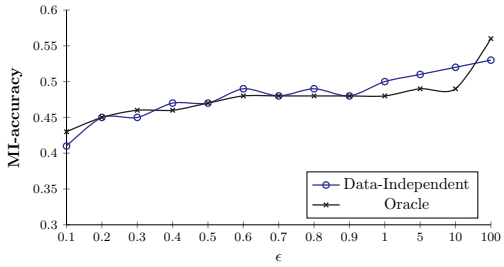
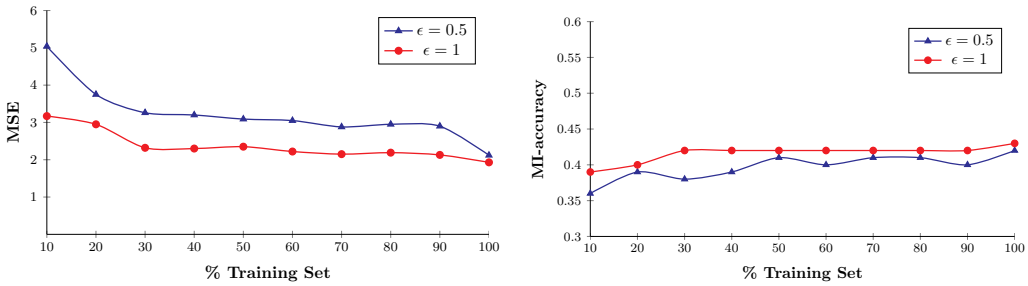


Figure 4.7: Model Invertibility: Data-Independent and Oracle Output Perturbation. The model inversion accuracy of these two algorithms are also very close with each other.

Comparison with Other Private Mechanisms. For MI we also compare our method with projected-histogram algorithm and objective perturbation. Specifically, We notice that in the previous work of Fredrikson et al. Fredrikson et al. (2014), they have demonstrated that projected-histogram algorithm actually leaks more information and produces models with higher MI accuracy than the functional mechanism (see the

discussion under the head "**Private Histograms vs. Linear Regression**" in Section 4 of their paper). For Objective Perturbation we find again that its MI accuracy is almost the same as our Data-Independent and Oracle Output Perturbation. This is not surprising because they have very close model accuracy.

For a fixed mechanism, better utility gives more effective MI attacks. For a fixed mechanism, we demonstrate that MI attacks get more effective as utility increases with the availability of more training data. Figure 4.8



(a) Model Accuracy (b) Model Invertibility

Figure 4.8: Privately-Tuned Output Perturbation with increasingly larger training sets. The x-axis is the size of training chunks, and the y-axis is MSE. We use the following experimental method: The entire data set gets randomly permuted in the beginning. The data is split again into 25 chunks, with the first 24 for training and the last one for validation. For each of the training chunks, a fraction of the training items is sampled at rate r. We then train the model using the data with Tuned Output Perturbation and evaluate their utility and MI attack accuracy. This experiment is repeated 100 time for $r = 10\%, 20\%, \ldots, 100\%$.

shows the results for Tuned Output Perturbation. For $\varepsilon_p = 0.5$, the mean square error drops from 5.05 (with $r = 10\%$ of available training data) to 2.9 ($r = 90\%$), while the MI attack accuracy increases from 36% ($r = 10\%$) to 40% ($r = 90\%$).

A Theoretical Analysis. At first sight the above two phenomena may seem somewhat peculiar. If MI attack is considered as a privacy concern,

then we have empirically observed that some privacy concern becomes "worse", while differential privacy gets "better."

There is no contradiction here. Indeed, it suffices to observe that DP is a property of the learning "process", while MI attack is on the "result" of the process. It is thus valid that the process satisfies a strong privacy guarantee, while the result has some other concerns. In the following, we give a "lower bound" result, which shows that, as long as the optimal solution of the learning problem is susceptible to MI attacks, improving DP/utility tradeoff will give effective MI attacks "eventually."

The intuition is as follows: Suppose that MI attack will be effective at a hypothesis w^* such that $L_{\mathcal{D}}(w^*) = L^*_{\mathcal{D}}$. That is, the MI attack is effective at a hypothesis we want to converge to. Now, suppose that the effectiveness of MI attack grows "monotonically" as we converge to w^*. Then, as long as the result of a learning algorithm converges to w^*, it will gradually give more effective MI attacks. We now give more details of this argument.

Assumptions. We make the following three assumptions on learning and MI attack: (i) The utility of a hypothesis is measured by its generalization error. (ii) Suppose that for w^*, $L_{\mathcal{D}}(w^*) = L^*_{\mathcal{D}}$, and MI attack is effective for w^*. (iii) As $|L_{\mathcal{D}}(w) - L^*_{\mathcal{D}}|$ gets smaller, the MI attack for w becomes more effective.

These assumptions are natural. For (i), almost all previous work measures utility this way. For (ii), since the ultimate goal of learning is to converge to the best possible hypothesis, assuming MI attack will be effective for such w^* is natural. Finally, closeness in $L_{\mathcal{D}}$ indicates that w and w^* are close in terms of their "functioning as a model." Thus (iii) holds intuitively.

Better DP Mechanisms, More Effective MI attack. For any n, let S_n denote a training set of size n. Suppose that A' is an ε_p-differentially private mechanism with better privacy-utility tradeoff than the output perturbation mechanism A. Thus with high probability for $w \sim A'(S_n)$, $L_{\mathcal{D}}(w)$ is

closer to $L^*_{\mathcal{D}}$ than that of w sampled from $A(S_n)$. Combined with (ii) and (iii), we have that MI attack is more effective for $w \sim A'(S_n)$.

One may note that the mechanism A' could be any differentially private mechanism, as long as it has better DP-utility tradeoff than *output perturbation*. For example, one can use the objective perturbation mechanism in Chaudhuri et al. (2011) for generalized linear models, or exponential sampling based mechanisms in Bassily et al. (2014) if minimizing training error is the goal.

For a Fixed Mechanism, Better DP-Utility tradeoff gives More Effective MI attacks.. Theorem 4.5 and our discussion at the end of Section 4.2 have indicated that as n tends to the infinity, the amount of noise injected for ε_p-differential privacy vanishes to zero. Further, Theorem 4.11 and 4.12 imply that for any convex Lipschitz learning problem, the *output perturbation* mechanism converges to the optimal hypothesis w^*. Thus we are in the situation that as n increases, the output model has less noise yet it is closer to w^*. Therefore, by assumptions (ii) and (iii) MI attack is more effective for larger n.

A Bayesian View Point. In a paper by Kasiviswanathan and Smith Kasiviswanathan and Smith (2008), the authors give a Bayesian interpretation of the semantics of differential privacy. Informally speaking, given an arbitrary prior distribution over a collection of databases, what differential privacy guarantees is that *two posteriors* obtained in two worlds of neighboring databases are indistinguishable with each other.

What about the difference between *prior and posterior*? The same paper Kasiviswanathan and Smith (2008), and indeed the original paper by Dwork and NaorDwork and Naor (2008), have pointed out that it is impossible to bound the difference between prior and posterior under *arbitrary background knowledge*. Essentially, as long as the published information is "useful," there exists some background knowledge that allows an adversary to learn and significantly modify his/her prior.

Unfortunately, in MI attacks, some moderate background information allows significant change of one's prior. Therefore, while worsening MI attack is certainly not what one intended, it is also of no surprise that better differentially private mechanisms do not give better resilience against MI attack.

5 A FORMAL STUDY OF MODEL INVERSION ATTACKS

> *There's no sense in being precise when you don't even know what you're talking about.*
>
> — John von Neumann

In the previous chapter we saw that MI attacks are outside of the cocoon of differential privacy. In this chapter we continue the exploration and study MI attacks specifically. MI attacks have received attention shortly after their discovery Fredrikson et al. (2015); Wang et al. (2015). These work give MI-attacks in specific context and use experiments to quantify the advantage of "knowing the model." However, it is unclear theoretically what properties affects the "model invertibility." In fact, there is even no consensus in defining what MI attacks are.

In this chapter we take a first step to fill in this gap. Specifically, we present two methodologies to formalize MI attacks, in both black-box and white-box settings. Then we specialize these methodologies to important special cases, in order to isolate important factors that affect the model invertibility.

Our main results lie in Boolean models in the black-box situation. Using powerful tools from Boolean analysis, we identify *two different notions of stability* which significantly affect the model invertibility. Specifically for black-box MI attacks where the adversary knows the model output and *precisely* all other features, we show that model invertibility is characterized by *influence* from Boolean analysis. In the case where there is noise in the prior knowledge, we show that the invertibility is related to *stable influence* in Boolean analysis. Importantly, these two notions refer to the *stability of the model output when varying a single feature*. Interestingly, our exploration in the noisy situation also unveils a phenomenon where several "unstable features" interfere with each other and render the model

"stable" when little noise present. We study such phenomenon under the name "invertibility interference."

5.1 Introduction

Privacy concerns surrounding the release of statistical information have received considerable attention in the past decade. The goal of statistical data privacy research is to enable accurate extraction of valuable data patterns, while preserving an individual's privacy in the underlying dataset against *data privacy attacks*. In general, there have been two flavors of data privacy attacks in the literature. The first is against a *specific* privacy notion, such as *differential privacy* Dwork et al. (2006). Investigations of such attacks have led to lower bounds (e.g. Dwork et al. (2015); McGregor et al. (2011)). The second kind of attack is against *attribute privacy*, which is a general concept where one studies how much distortion is needed in order to prevent an adversary from inferring sensitive attributes from non-sensitive ones (for concreteness, see for example *reconstruction attacks* Dinur and Nissim (2003); Kasiviswanathan et al. (2013)). In particular, attribute privacy attacks are widely considered in the applied data privacy literature, where scenarios such as the release of medical information are in focus.

This chapter focuses on a specific class of attacks falling under the second type. The class of attacks we consider relate to inferring sensitive attributes from a released model (e.g. a machine-learning model), or *model inversion* (MI) attacks. Several of these attacks have appeared in the literature. Recently, Fredrikson et al. Fredrikson et al. (2014) explored MI attacks in the context of personalized medicine. Specifically, Fredrikson et al. Fredrikson et al. (2014) "invert" a publicly-released linear regression model in order to infer sensitive genetic markers, based on the model *output* (Warfarin dosage) plus several other non-sensitive attributes (e.g.,

height, age, weight). Interestingly, they demonstrate that knowledge of the model output (Warfarin dosage here), or even a reasonable approximation of it, leads to a statistically-significant increase in leakage of the sensitive attribute. This leads to natural questions of how widely such effective and efficient inversion attacks exist for statistical models, as well as how to quantify the *additional* leakage due to accessing the model.

Recently, more instances of effective MI attacks have been discovered, further stimulating interest in this class of attack. For example, Fredrikson et al. (2015) considered a white-box MI attack on models used to classify images. They demonstrated that by exploiting the additional confidence information provided by such models, one can significantly improve both the effectiveness and efficiency of an MI attack. Interestingly, we note that these attacks are reminiscent of privacy attacks discussed in the context of inverting highly compressed image features, which were explored previously Daneshi and Guo (2011); d'Angelo et al. (2012); Kato and Harada (2014). It is our belief, however, that if we are to develop countermeasures against all these attacks, or even a precise explanation of the dangers they pose, we will need to go beyond example-based definitions and require a methodology to capture this phenomenon. We consider this chapter to be an initial step and much work remains to be done.

In this chapter we take a first step toward providing a formal treatment of MI attacks. Our contributions are summarized as follows:

- We present two methodologies, which are both inspired by the "two world" games common in cryptographic definitions. A methodology for *black-box* attacks, where the adversary has oracle access to the model, and a methodology for *white-box* attacks, where the adversary has information about the model structure. Our methodologies provide a "blueprint" for making these definitions precise for specific cases. Extending this to a precise general definition (such as

the real and the ideal world definition used in the SMC literature) will be an interesting direction to pursue. Our methodology focuses on machine-learning (ML) models because they have been the target of existing MI attacks. One shortcoming of our methodology is that we do not take into account the specific structure of the ML model or the learning task. Again, connecting our methodology to various notions in the ML literature (such as stability) provides an attractive avenue for future work.

- We then specialize our methodology to important special cases, in order to isolate *important factors that affect model invertibility* (i.e. how successfully one can invert the model). Identifying these factors is important for at least two applications. First, as a decision procedure prior to publishing a model, estimating invertibility can help one gauge the leakage of sensitive attributes, and thus help in deciding which part of a model is publishable. The second is to help in preventing MI attacks: if invertibility is low, then little noise may be used to effectively prevent MI attacks without sacrificing too much utility.

- For the case of models that are Boolean functions (e.g., decision trees with attributes having finite domains), we have some concrete results. In this case, we can leverage powerful tools from Boolean analysis. Specifically for black-box MI attacks where the adversary knows the model output and precisely all other features, and there is no noise, we show that model invertibility is characterized by *influence* from Boolean analysis. Unfortunately, it becomes significantly more complicated if there is noise in the prior knowledge of the adversary. Nevertheless, we show that the invertibility is related to *stable influence* in Boolean analysis. Interestingly, our exploration in the noisy situation also unveils a phenomenon where

a highly invertible model quickly becomes highly non-invertible by adding a little noise. We study such phenomenon under the name "invertibility interference."

- For white-box MI attacks, we study a common phenomenon where the computation of a machine learning model is a sequential composition of several layers or models. Exploiting the intermediate information communicated between these layers, even when it is highly compressed, can give a significant advantage to the adversary. In fact, the white-box attack described in Fredrikson et al. (2015) exploits exactly such information where the confidence information is the likelihood probabilities computed at an intermediate layer of the model. We thus study how these restricted communication channels could leak information. Interestingly, our results show, quantitatively, that even with 1 *bit of communication there could be a significant leakage*. Our results also unveil unexpected computational power of these restricted channels, which, to the best of our knowledge, were previously unknown.

The rest of the chapter is organized as follows: Section 5.2 describes our methodologies for black-box and white-box MI attacks. Section 5.3 and Section 5.4, specializes our general formulation to important special cases. Finally, we discuss connections of our formulation with other cryptographic notions in Section 5.5.

5.2 A Methodology for Formalizing MI Attacks

An essential goal of studying MI attacks is to *quantify* the strength of the correlation between sensitive attributes and the output of the model. While this goal is very intuitive, formalizing these attacks poses a challenge due to the diversity of such attacks. Moreover, as we mentioned

earlier, many different attacks can be viewed as "MI attacks." This suggests that it can be difficult to give a "unified" definition of MI attacks without risking over generalization (i.e., even a lot of benign cases with "weak correlation" will be classified as attacks). As a first attempt, our goal is thus to abstract out important factors from existing attacks, and present a methodology. Guided by these methodologies, later in this chapter we identify special cases of MI attacks that lead to theoretical insights.

This section is organized as follows: We start by discussing concepts from machine learning, which provides the background for our methodology. Then we discuss MI attacks in an intuitive manner. In Section 5.2 and 5.2 we present methodologies for black-box MI and white-box MI attacks, respectively. Along the way, we discuss how our methodology captures existing attacks and can be used to model other interesting scenarios that have not been addressed before.

Background. We formalize MI attacks in the *generalized learning setting*. In the generalized learning setting, a machine learning task is represented as a triple (Z, H, ℓ), where Z is a sample space, H is a hypothesis space, and $\ell : H \times Z \mapsto \mathbb{R}$ is a loss function. Given a data generating distribution $\mathscr{D}$, the goal of learning is to solve the following stochastic optimization problem:

$$\min_{h \in H} \mathop{\mathbb{E}}_{x \sim \mathscr{D}} [\ell(h, z)].$$

In machine learning however, $\mathscr{D}$ is unknown and one must find an approximate solution using a dataset S of i.i.d. samples from $\mathscr{D}$.

Recall that in the supervised learning setting, Z is of the form $X \times Y$ where X is called a feature space and Y an output space. Further, a hypothesis $h \in H$ has to take the form $X \mapsto Y$. On the other hand, the generalized learning setting, as formulated above, also incorporates unsupervised learning. For example in clustering, one maps $z \in Z$, a collection of points, to a set of clusters.

MI Attacks: Scenarios and Observations. Intuitively, MI attacks are designed to capture privacy concerns about *participants in a training set*, which arise from the following scenario: An organization trains a model over some dataset collected from a large set of individuals. After restricted access (say under some strict access control) to the model *within the organization*, now they want to release the model to the *public* for general use (e.g. say by a medical-clinic that specializes in providing personalized medicine.) We envision two mechanisms for releasing a model: release the model as a black box so public can use it freely, or release the model as a white box with some information about its architecture and parameters published. The concern is that certain correlation encoded in the model may be too strong such that a potential adversary can leverage the publicly published model, plus additional knowledge about individuals in the training set, to recover *participants'* sensitive information. The essential goal of studying MI attacks is to *quantify the strength of such correlations* so that one can have a better understanding to what degree such concerns matter.

Towards this goal, one thus needs to formulate a reasonable adversary model to capture how an adversary may exploit the model. We have the following simple observations: (1) We are interested in MI attacks in the *test phase* of machine learning, where a model h has already been trained. (2) It is necessary to have some objective for an attack, which can be captured by some function τ that maps a sample $z \in Z$ to some range. (3) The quantification is carried over the training dataset, since the main concern is for participants in the dataset. (4) The quantification is supposed to compare "two worlds", one where the adversary has access to the model, and the other where the adversary does not. This is to capture the fact that we want to quantify the *additional risk* of releasing a model.

Limitations: Next we discuss some limitations of our methodology, and addressing these limitations provides interesting avenue for future work.

Our methodology focusses on one organization, so for example, our model does not cover the following scenario: a different organization can collect data $S^\star$ similar to S and build a model $h^\star$ (may be using the same learning algorithm), which can then be used to infer sensitive information about participants. Moreover, the results need to be interpreted on a case-by-case basis. For example, assume that our definition with parameterization for a specific context yields advantage of $\frac{1}{N}$, where N is the size of the training set S. Should we consider this an attack? This depends on the context. We admit that our methodology does not exploit structure of the ML task and model (e.g., perhaps looking at the loss function l). In general, we believe that what is considered a privacy breach is highly dependent on the context.

Black-Box MI

We now present a methodology for formalizing black-box MI attacks where the adversary has oracle access to a model. Along the way, we introduce notation that will be used later.

Measuring Effectiveness of An Attack. It is attractive to consider the success of an attack on a single sample point. While this might be sensible in some specific scenarios (for example, the adversary wants to get genetic information about a specific individual), it does not seem to be a good formal measure. This is because a machine learning model, in contrast to an encryption, is supposed to communicate some information about the sample, so in the worst case it is always possible to extract some information about a specific individual.

On the other hand, one may attempt to measure an attack over the *data-generating distribution* $\mathscr{D}$, which is in the definition of a machine-learning task. However, this leads to a complication as $\mathscr{D}$ is unknown in general and so one has to impose assumptions on its structure. We choose

to measure an attack over the dataset used to train the model. This thus provides a privacy loss measure for participants in the dataset. Moreover, this allows us to carry out the quantification without an additional parameter $\mathscr{D}$.

Adversaries and Their Power. We first note that a model at the test phase is fixed, so there is no asymptotic behavior since there is no infinite family of models. We thus model an adversary as a probabilistic algorithm without limiting its computational complexity. In other words, the adversary is *all powerful*. We note that other data privacy formulations, such as differential privacy Dwork et al. (2006), also make such an assumption on the adversarial power.

We now present a *methodology* for formulating black-box MI attacks with the goal of measuring the effectiveness of these attacks. To use this methodology as a template to generate precise definitions for specific scenarios, one has to instantiate auxiliary information generators gen and sgen in the methodology for attacks and simulated attacks, respectively. Having two different generators in the two worlds allows us additional flexibility (e.g., in the Warfarin attack the attacker in the MI-Attack world knows some "approximation" of the Warfarin dosage.) Note that this information cannot be computed by using the oracle because the adversary does not know all the feature values. In some cases, gen and sgen will be the same.

Methodology 1. *The starting point of an MI attack is a machine learning problem, specified as a triple* (Z, H, ℓ)*. We use the following notations:*

1. Γ*: A training algorithm of the learning problem, which outputs a hypothesis* $\Gamma(S) \in H$ *on an input training set* S*.*

2. $\mathscr{D}_S$*: A distribution over the training set* S*.*

3. τ: *The objective function computed by the adversary. For now, one can view it simply as some function that maps* Z *to* $\{0,1\}^*$.

4. $\mathsf{gen}, \mathsf{sgen}$: *Auxiliary information generators. They map a pair* (S, z) *to an advice string in* $\{0,1\}^*$.

As we noted before there are two worlds in our methodology (the MI-attack world) and (the simulated attack world).
The MI-attack world: *This world is described by a tuple* $(A, \mathsf{gen}, \tau, S, \mathscr{D}_S, \Gamma)$, *where the adversary (A) is a probabilistic* oracle machine *(recall that* gen *generates an advice string for the adversary from* (S, z)*). Now the following game is played between the Nature and the adversary A.*

(1) *Nature draws a sample z from* $\mathscr{D}_S$.

(2) *Nature presents* $v = \mathsf{gen}(S, z)$ *to the adversary.*

(3) *Adversary outputs* $A^{\Gamma(S)}(v)$.

The gain of the game is evaluated as

$$\mathsf{gain}(A, \mathsf{gen}, \tau, S, \mathscr{D}_S, \Gamma) = \Pr[A^{\Gamma(S)}(\mathsf{gen}(S, z)) = \tau(z)]$$

where the probability is taken over the randomness of $z \sim \mathscr{D}_S$, *the randomness of* gen, *and the randomness of A. In other words, the gain is the probability that the adversary A with oracle access to the model* $\Gamma(S)$ *and given the advice string generated by* sgen *is able to "guess"* $\tau(z)$.
The simulated world: *is described by a tuple* $(A^*, \mathsf{sgen}, \tau, S, \mathscr{D}_S)$, *where the adversary* $(A^\star)$ *is a non-oracle machine and* sgen *is the second auxiliary information generator. The game between the Nature and* $A^\star$ *is exactly the same as in the MI-attack world, but* $A^\star$ *does not have oracle access to the*

learned model $\Gamma(S)$. *Similarly, the gain is defined as:*

$$\mathsf{sgain}(A^*, \mathsf{sgen}, \tau, S, \mathscr{D}_S) = \Pr[A^*(\mathsf{sgen}(S, z)) = \tau(z)]$$

where the probability is taken over the randomness of $z \sim \mathscr{D}_S$, *the randomness of* sgen, *and the randomness of* A^*.

Advantage: *For* (τ, S, Γ), *the advantage of* (gen, A) *over* (sgen, A^*) *is computed as*

$$\mathsf{adv}^{(\mathsf{gen},A)}_{(\mathsf{sgen},A^*)} = |\mathsf{gain}(A, \mathsf{gen}, \tau, S, \mathscr{D}_S, \Gamma) - \mathsf{sgain}(A^*, \mathsf{sgen}, \tau, S, \mathscr{D}_S)|.$$

Leakage: *We say that* $\Gamma(S)$ *has* ε*-leakage for* $(\tau, \mathscr{D}_S)$ *with respect to* $(\mathsf{gen}, \mathsf{sgen}, A)$ *if there exists an adversary* A^* *such that* $\mathsf{adv}^{(\mathsf{gen},A)}_{(\mathsf{sgen},A^*)} \leq \varepsilon$. *Finally,* $\Gamma(S)$ *has* ε*-leakage for* $(\tau, \mathscr{D}_S)$ *with respect to* $(\mathsf{gen}, \mathsf{sgen})$ *if for any probabilistic adversary* A, *there exists an adversary* A^* *such that* $\mathsf{adv}^{(\mathsf{gen},A)}_{(\mathsf{sgen},A^*)} \leq \varepsilon$.

We remark that an interesting special case is to evaluate the gain against a *uniform distribution* over the training set. This case is interesting because a uniform distribution over the training set gives an approximation of the underlying data generating distribution $\mathscr{D}$, as S is i.i.d. drawn from $\mathscr{D}$. As a result, the gain against the uniform distribution over the training set also approximately measures the strength of the correlation for the data generating distribution.

Modeling Examples. We now discuss several examples of applying our methodology.

Example 5.1 (Warfarin Attack Fredrikson et al. (2014))**.** *Our first example is the Warfarin-dosage attack in the original work of Fredrikson et al. Fredrikson et al. (2014). The Warfarin-dosage attack is a black-box MI attack in the*

The MI-Attack World	The Simulated World
Oracle access to Γ (a linear-regression model).	No access to the oracle.
τ: $\tau(z) = x_i$. (the VKORC1 genetic marker)	τ: $\tau(z) = x_i$.
gen: $\mathsf{gen}(S, z) = (x_{-i}, \mathbf{y}, \text{marginals of } S)$.	sgen: $\mathsf{sgen}(S, z) = (x_{-i}, \text{marginals of } S)$.
A: can access h, x_{-i}, marginals of S and $\mathbf{y}$.	A^*: can access h, x_{-i}, marginals of S.

Table 5.1: Warfarin-dosage Attack of Fredrikson et al. Fredrikson et al. (2014). We describe how to set up various parameters in order to put the attack of Fredrikson et al. (2014) in our methodology. Note that in this formulation z takes the form of (x, y) and we feed y to the adversary (not $h(x)$). This is important because in Fredrikson et al.'s case, for the patients participating in the dataset, y does not come from model output, but rather is determined by medical doctors. Thus $h(x)$ is only an approximation of y.

supervised learning setting. Thus $Z = X \times Y$ and $X = \prod_{i=1}^{n} X_i$ where X_i's are binary encoding of features, such as genotypes, race, etc. The attack, put in our formalization, is summarized in Table 5.1.

Note that in this formulation z takes the form of (x, y) and we feed y to the adversary (not $h(x)$). This is important because in Fredrikson et al.'s case, for the patients participating in the dataset, y does not come from model output, but rather is determined by medical doctors. Thus $h(x)$ is only an approximation of y. □

Example 5.2 (Inferring Participation)**.** *A common privacy attack is to infer whether an individual is in a dataset. For example, differential privacy addresses such attacks, and uses noise to hide the participation of any* individual *(so, with/without a specific individual, the adversary draws the same conclusion with high probability.)*

We note that participation attacks fit naturally into our methodology. In

particular, consider the following goal function τ, for $z \in Z$,

$$\tau(z) = \begin{cases} 1 & z \in S, \\ 0 & \textit{otherwise.} \end{cases}$$

That is, given $z \in Z$, the goal of the adversary is to decide whether z is in the training set or not.

One may think that differential privacy is precisely the countermeasure for this attack. However, in principle it is not, although applying differential privacy may have certainly effect the outcome.

This is because the design of differential privacy allows *learning correlations, subject only to that any* individual participation *will* not *be able to change the correlation significantly. Therefore, once the correlation is found, one may still be able to use this correlation to infer participation of a population with certain accuracy. Nonetheless differential privacy ensures that* localized *to any particular individual, his or her participation will not significantly change the results of such inferences (so the guarantee here is a form of "plausible deniability" for that particular individual). We remark that it would be interesting to carry out this attack empirically in a real-world setting.* □

White-Box MI

We now move on to consider white-box MI attacks. We will now assume that the adversary has some additional knowledge about the model structure. The question is, however, how to model this knowledge about the structure?

We observe that machine learning models typically adopt a sequential composition of computations. For example, in the simplest case of linear models, one first computes a linear representation of the features, and then applies, for example, a logistic function to make a prediction (representing a probability in this case). As another example, in "one-vs-all"

multiclass logistic regression, one trains multiple binary logistic regression models, each encoding the "likelihood" of a particular class, and then makes a final prediction based on these confidence information. As observed in Fredrikson et al. (2015), being able to observe such intermediate information, even though they might be highly compressed compared to the original information, can give the adversary a significant advantage in deducing sensitive values.

We are thus motivated to consider white-box MI attacks in the particular case of sequential composition. We note that this is in sharp contrast with attacks in cryptographic settings where the protocols typically have a significantly more complicated composition structure (compared to sequential composition). We start by defining machine learning models with k layers.

Definition 5.3 (k-Layer Model)**.** *Let X be a feature space and $Y_1, \ldots, Y_k$ be k output spaces. A k-layer model M is a model where its computation can be represented as a composition of k functions $h_1, \ldots, h_k$, where $h_1 : X \mapsto Y_1$ and $h_i : Y_{i-1} \mapsto Y_i$ $(2 \leq i \leq k)$. The output of h_k is the output of the entire model.*

We can now define white-box MI attacks. Compared to the black-box case, the only thing changes now is that: (i) the adversary is aware of the composition structure, and (ii) he might be able to observe intermediate information passing between the layers. We have the following definition.

Methodology 2 (k-layer White-Box MI Attack)**.** *The methodology of white-box MI attack is the same as the black-box one, except for two differences:*

- *Instead of letting the adversary A have oracle access to the model, we feed* the k-layer representation *of the machine-learning model as input to the adversary.*
- *The auxiliary information generators* gen *and* sgen *take an additional parameter Γ (the learning algorithm). This allows the auxiliary infor-*

mation generators to generate information that might depend on the learning algorithm Γ *(see Example 5.5). An important point to note is that in the simulated world the adversary still does not have access to the model* $\Gamma(S)$.

Modeling Examples. As in the black-box case, we now express existing attacks using our methodology.

Example 5.4 (Decision Tree Fredrikson et al. (2015))**.** *In Fredrikson et al. (2015) the authors studied the following attack against decision trees. Not only does the adversary know the model structure, but also he knows, for each path of the tree, how many instances in the training set correspond to that path. In other words, the adversary knows both the exact decision tree, as well as the confidence information of each path (intuitively, one wants to follow the path that more training instances follow). They show that with such confidence information one can significantly improve attack accuracy.*

Such a scenario can be captured by our methodology. The decision tree model is directly fed as input to the adversary. For the confidence information, the adversary can compute on its own by simulating the model on every instance of S*. The adversary can do so because he is all-powerful and has white-box access to the model.*

Example 5.5 (Neural Network Fredrikson et al. (2015))**.** *As we mentioned before, Fredrikson et al. (2015) also studied another attack where for a neural network with a softmax layer (this layer encodes the probabilities corresponding to each class), the adversary can query for probability in that layer. Again, accessing this piece of information significantly improves attack accuracy.*

This attack can also be easily captured by our methodology. One potential subtlety is the softmax probabilities, which cannot be computed by the adversary directly, though he has white-box access to the model. This is because he only knows partially the original input. Nevertheless, this can be generated by

the auxiliary information generator by simulating $\Gamma(S)$ *on* z *and encode the output of the softmax layer in the auxiliary information.*

Interestingly, we observe that several privacy attacks Daneshi and Guo (2011); d'Angelo et al. (2012); Kato and Harada (2014) for recovering image features, which appeared before the work of Fredrikson et al. Fredrikson et al. (2014), can also be captured using white-box MI attacks in a similar way.

5.3 Black-Box MI Attacks

In this section we study black-box MI attacks. Due to lack of space, we will focus on the simplest possible models – binary classification, where all the features are binary as well. Using standard knowledge of Boolean analysis, our results can also be extended to arbitrary generalized but finite domains.

Recall that our main technical goal is to isolate important factors that can affect model invertibility. Unfortunately, our formulation in Section 5.2 is quite complex so many factors may play a role. For example, intuitively the training process may have an impact – if we know that a model is trained using linear regression, would this give us an advantage? On the other hand, if one thinks about MI attacks at the application phase, where a model is fixed anyway, then it suggests that invertibility should be *independent* of the training.

To gain more understanding, we choose to start with simple scenarios where we can characterize model invertibility exactly. Interestingly, even these very abstract and seemingly oversimplified scenarios provide insights to our main question. Perhaps more importantly, they also give rise to intriguing and natural questions that provide ample scope for future investigations.

Specifically, in this section we specialize Methodology 1 in the following ways:

- We consider a Boolean model $h : \{\text{-}1, 1\}^n \mapsto \{\text{-}1, 1\}$ in the test phase.
- We assume that the model invertibility is evaluated over the uniform distribution. That is, we assume that a feature vector is drawn uniformly from $U_{\{\text{-}1,1\}^n}$.
- We consider two simple auxiliary information generators. In the first, *noiseless generator* gen_1,

$$\mathsf{gen}_1(S, (x, y)) = (x_{-i}, y).$$

  In the second *independent perturbation generator*,

$$\mathsf{gen}_\rho(S, (x, y)) = (z_{-i}, y)$$

  where each bit of z_{-i} equals that of x_{-i} with probability $\frac{1}{2} + \frac{\rho}{2}$, and is flipped otherwise, or $z_{-i} \sim N_\rho(x_{-i})$. Note that for $\rho = 1$ it degenerates to our noiseless generator.

Under these specializations our main results are summarized as follows[1].

1. In the noiseless case, we characterize model invertibility using the influence of a Boolean function. Interestingly, it turns out in this case, model invertibility is *independent* of the training. These results are presented in Section 5.3.
2. In the noisy case, we show that model invertibility is related to the stable influence of a Boolean function, though stable influence does

[1] For sake of exposition, the proofs are put in the appendix.

not exactly capture the invertibility. These results are presented in Section 5.3.

3. Interestingly, we find that under noise, there is an interesting phenomenon where a *highly invertible model quickly becomes highly non-invertible with only a little noise*. We study this phenomenon under the name "invertibility interference." The results are presented in Section 5.3.

Model Invertibility with No Noise

We now specialize our methodology for black-box MI attacks to the noiseless scenario.

Definition 5.6 (Noiseless Uniform Black-Box attack)**.** *Let* (Z, H, ℓ) *be a learning problem where* H *consists of hypotheses of the form* $\{-1,1\}^n \mapsto \{-1,1\}$. *Let* Γ *be a learning algorithm and* S *be a training set. For simplicity we denote* $\Gamma(S)$ *as* h. Noiseless Uniform Black-Box MI attack for coordinate i *is the following game.*
The MI-attack world: Let A *be a probabilistic algorithm with binary output, then*

i. *Nature draws* (x, y) *from* $\mathscr{D}_U$. *That is, nature draws* $x \sim \{-1,1\}^n$, *and set* $y = h(x)$.

ii. *Nature presents*

$$\mathsf{gen}_1(S, (x, y)) = (x_{-i}, y)$$

to the adversary.

iii. *Adversary outputs* $A^{\Gamma(S)}(x_{-i}, h(x))$.

The gain *of this game is* $\Pr[A^{\Gamma(S)}(x_{-i}, h(x)) = x_i]$, *where the probability is over samples* $x_1, \ldots, x_n$, *and the randomness (if any) of the adversary.*

The Simulated World: In this case sgen_1 *is defined as* $\mathsf{sgen}_1(S,(x,y)) = x_{-i}$. *Because* x_i *is independently and uniformly drawn from* {-1, 1}, *so for any simulated attack* A^*, $\Pr[A^*(x_{-i}) = x_i] = 1/2$.

Therefore, the advantage *of the game is defined to be* $\Pr[A^{\Gamma(S)}(x_{-i}, h(x)) = x_i] - 1/2$.

For this type of MI attack, we consider the following *deterministic* algorithm for the adversary A.

Algorithm 8 A Deterministic Algorithm for Noiseless Uniform Flat MI Attack.

Input: $x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n$. $y \in \{-1,1\}$. Oracle access to f.

1: **function** NoiselessBlackboxMI($x_1,\ldots,x_n,y$)
2: Compute $y_1 = f(x_1,\ldots,x_{i-1},-1,x_{i+1},\ldots,x_n)$, and $y_2 = f(x_1,\ldots,x_{i-1},+1,x_{i+1},\ldots,x_n)$.
3: If $y_1 \neq y_2$, then if $y_1 = y$, output -1, otherwise output $+1$. Otherwise, output the constant 1.

We have the following simple lemma.

Lemma 5.7. *Let* A_1 *denote Algorithm 8. Then* $\mathsf{gain}(A_1, \mathsf{gen}_1, x_i, \mathscr{D}_U, S, \Gamma) = \frac{1}{2} + \frac{\mathbf{Inf}_i[\Gamma(S)]}{2}$, *and so the advantage is* $\frac{\mathbf{Inf}_i[\Gamma(S)]}{2}$. *Further this gain is optimal.*

Proof. For any fixed $x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n$, if $f(x) \neq f(x^{\oplus i})$, then we guess x_i right with probability 1. Otherwise, conditioned on $f(x) = y$, x_i is uniformly and independently distributed over {-1, 1}. In this case, by constantly guessing 1 the correct probability is 1/2. This gives the desired gain of Algorithm 8, as well as its optimality. □

While this lemma is trivial to prove, an interesting observation regarding it is that the invertibility is *independent of the training*. That is, no matter what Γ and S are (and what distribution S is drawn from), the invertibility is *characterized* by influence, which is an intrinsic property of the model itself.

For noiseless MI attacks, it is also easy to characterize the most and least invertible functions. In the following recall that we assume the nontrivial feature assumption.

Most Invertible Functions. With the no trivial feature assumption, the most invertible function is $\chi_{[n]}(x) = \prod_{i=1}^{n} x_i$, where every coordinate has influence 1, and so the advantage is 1/2.

Least Invertible Functions. What functions are least invertible if we measure the invertibility by the maximum influence $\mathbf{MaxInf}_i[h]$? In this direction, a natural candidate is the majority function. Indeed, using Stirling's formula (see Exercise 2.22 O'Donnell (2014).), one can estimate that $\mathbf{Inf}_i[\mathrm{MAJ}_n] \approx O(1/\sqrt{n})$ for every $i \in [n]$. There are functions with much smaller influence. For example,

$$\mathrm{OR}_n(x) = \begin{cases} 1 & x_1 = x_2 = \cdots = x_n = 1. \\ -1 & \text{otherwise.} \end{cases}$$

Then it is easy to check that $\mathbf{Inf}_i[\mathrm{OR}_n] = 2^{1-n}$ for every $i \in [n]$. We can also characterize the structure of the least invertible functions. Under the no trivial feature assumption, these functions are those that are "constant except at one point."

Lemma 5.8. *Consider any* $h : \{-1,1\}^n \mapsto \{-1,1\}$. *If* $\mathbf{Inf}_i[h] > 0$, *then* $\mathbf{Inf}_i[h] \geq 2^{1-n}$.

Proof. If for any input x, $f(x) \neq f(x^{\oplus i})$, then there are at least two inputs, namely $y = x, x^{\oplus i}$ such that $f(y) \neq f(y^{\oplus i})$, this shows the probability $\Pr_{x\sim\{-1,1\}^n}[f(x) \neq f(x^{\oplus i})] \geq 2^{1-n}$. □

Lemma 5.9. *Let* $h : \{-1,1\}^n \mapsto \{-1,1\}$ *be a Boolean function. If* $\mathbf{Inf}_i[h] = 2^{1-n}$ *for some* $i \in [n]$, *then* $\mathbf{Inf}_i[h] > 0$ *for every* $i \in [n]$.

Proof. Without loss of generality, assume for contradiction that x_1 has influence 0. Then by Lemma 2.28, there is a function $g(x_2,\dots,x_n)$ such that $\mathbf{Inf}_j[g] = \mathbf{Inf}_j[f]$ for every $j \geq 2$. Now by Lemma 5.8, $\mathbf{Inf}_j[f] = \mathbf{Inf}_j[g] \geq 2^{2-n} > 2^{1-n}$ for every $j \geq 2$, contradiction. □

Theorem 5.10. *Let $h : \{-1,1\}^n \mapsto \{-1,1\}$ be a Boolean function. Then h satisfies the property that for every $i \in [n]$, $\mathbf{Inf}_i[h] = 2^{1-n}$ if and only if h is constant except at a unique point x_0. In other words, there exist $x_0 \in \{-1,1\}^n$ and $b \in \{-1,1\}$ such that*

$$h(x) = \begin{cases} b & \text{if } x = x_0, \\ -b & \text{otherwise.} \end{cases}$$

Proof. **(⇐)** If f is constant except at a unique point x_0, then for any i, the only inputs x on which $f(x) \neq f(x^{\oplus i})$ are $x = x_0$ and $x = x_0^{\oplus i}$. This proves that $\mathbf{Inf}_i[f] = 2^{1-n}$.

(⇒) We induct on n. If $n = 1$, then $\mathbf{Inf}_1[f] = 1$, so $f(x_1) = x_1$ or $-x_1$ and the result is true. Fix any $n \geq 2$. The induction hypothesis is the following: Let g be a Boolean function on $k \leq (n-1)$ variables. If every coordinate of g has influence 2^{1-k}, then g is constant except at one point.

Now let f be a function on n variables that $\mathbf{Inf}_i[f] = 2^{1-n}$ for every $i \in [n]$. Consider two Boolean functions on $n-1$ variables, $g(x_2,\dots,x_n) = f(1,x_2,\dots,x_n)$ and $h(x_2,\dots,x_n) = f(-1,x_2,\dots,x_n)$. We claim that the influence of x_2 is 2^{2-n} in one of g and h, and 0 in the other. Indeed, by the assumption that $\mathbf{Inf}_2[f] = 2^{1-n}$, there must be a unique setting $z_{-2} = (z_1,z_3,\dots,z_n)$ such that $f(z^{2\to 1}) \neq f(z^{2\to -1})$. Note that z_1 is fixed to be 1 or -1, thus the influence of x_2 is 2^{2-n} in g or h, and 0 in the other.

Without loss of generality, suppose $\mathbf{Inf}_2[g] = 2^{2-n}$. Note that g is defined over $n-1$ variables, so by Lemma 5.9, $\mathbf{Inf}_j[g] > 0$ for every $2 \leq j \leq n$. On the other hand, clearly $\mathbf{Inf}_j[g] \leq 2^{2-n}$ for every $j = 2,\dots,n$. Thus we can apply the induction hypothesis to conclude that g is constant

except one point. Moreover, h is constant. Finally, suppose $g(y) = b$ except $g(y_0) = -b$ where $y_0 \in \{\text{-}1,1\}^{n-1}$ and $b \in \{\text{-}1,1\}$ is some fixed point. Because $\mathbf{Inf}_1[f] = 2^{1-n}$, so it must be that $h \equiv b$. This shows that f is constant except at one point. □

Recall that a Boolean-valued function is *unanimous* if $h(1,\ldots,1) = 1$ and $h(-1,\ldots,-1) = -1$. Therefore we have the following two corollaries,

Corollary 5.11. OR_n *and* AND_n *are the only unanimous Boolean functions where maximum influence is* 2^{1-n}.

Corollary 5.12. OR_n *and* AND_n *are the only monotone Boolean functions where maximum influence is* 2^{1-n}.

Model Invertibility with Independent Noise

We now move on to the independent perturbation case.

Definition 5.13 (ρ-Independent Perturbation Uniform Black-Box MI Attack)**.** *Let* (Z, H, ℓ) *be a learning problem where* H *consists of hypotheses of the form* $\{\text{-}1,1\}^n \mapsto \{\text{-}1,1\}$. *Let* Γ *be a learning algorithm and* S *be a training set. For simplicity we denote* $\Gamma(S)$ *as* h. ρ-Independent Perturbation Uniform Black-Box MI attack *for coordinate* i *is the following game.*

The MI-Attack world: Let A be a probabilistic algorithm with binary output, then

i. *Nature draws* (x, y) *from* $\mathscr{D}_U$. *That is, nature draws* $x \sim \{\text{-}1,1\}^n$, *and set* $y = h(x)$.

ii. *Nature presents*

$$\mathsf{gen}_\rho(S,(x,y)) = (z_{-i}, y)$$

to the adversary.

iii. *Adversary outputs* $A^{\Gamma(S)}(z_{-i}, y)$.

The gain *of this game is* $\Pr[A^{\Gamma(S)}(z_{-i}, y) = x_i]$*, where the probability is over samples* $x_1, \ldots, x_n$*, the randomness of* gen_ρ*, and the randomness (if any) of the adversary.*

The simulated world: For simulation, sgen_ρ *is defined as* $\mathsf{sgen}_\rho(S, x_{-i}, y) = z_{-i}$*. Because* x_i *is independently and uniformly drawn from* $\{-1, 1\}$*, so for any simulated attack* A^**,* $\Pr[A^*(z_{-i}) = x_i] = 1/2$*.*

Therefore, the advantage *is defined as* $\Pr[A^{\Gamma(S)}(x_{-i}, y) = x_i] - 1/2$*.*

We now consider the following algorithm corresponding to the adversary A. The algorithm is the same as Algorithm 8, we repeat it here and note that now the input to the algorithm is z_{-i}, instead of x_{-i}.

Algorithm 9 A Deterministic Algorithm for ρ-Perturbation Uniform Singleton Flat MI Attack.

Input: $z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n$. $y \in \{-1, 1\}$. Oracle access to f.
function IndNoiseBlackboxMI($z_1, \ldots, z_n, y$)
Compute $y_1 = f(z_1, \ldots, z_{i-1}, -1, z_{i+1}, \ldots, z_n)$, and $y_2 = f(z_1, \ldots, z_{i-1}, +1, z_{i+1}, \ldots, z_n)$.
If $y_1 \neq y_2$, then if $y_1 = y$, output -1, otherwise output $+1$. Otherwise, output the constant 1.

The gain of this algorithm is exactly the so called *stable influence*. Intuitively this is clear: Recall from Definition 2.24 that stable influence is defined as $\mathbb{E}_{x \sim \{-1,1\}^n, z \sim N_\rho(x)}[\mathrm{D}_i f(x) \mathrm{D}_i f(z)]$. Thus if $\mathrm{D}_i f(x)$ and $\mathrm{D}_i f(z)$ are of the same sign then Algorithm 9 guessed correctly. If the signs are different, then the guess is incorrect. Otherwise, one can show that the gain is 1/2. Formally, we have the following theorem.

Theorem 5.14. *Let* $\rho \in [0, 1]$*. Let* A_ρ *denote Algorithm 9. Then*

$$\mathsf{gain}(A_\rho, \mathsf{gen}_\rho, x_i, \mathscr{D}_U, S, \Gamma) = \frac{1}{2} + \frac{\mathbf{Inf}_i^{(\rho)}[h]}{2}$$

where $\mathbf{Inf}_i^{(\rho)}[h]$ *is the* ρ*-stable influence of* h *(See Definition 2.24) at* i*-th coordinate.*

Proof. Let A denote Algorithm 9. Let $\alpha = f(x^{i\to 1}) - f(x^{i\to -1})$ and $\beta = f(z^{i\to 1}) - f(z^{i\to -1})$. Consider the following four disjoint events:

(E1) $\alpha\beta > 0$. Conditioned on E1 happens, A always outputs the correct bit. Thus in this case $\Pr[b = x_i \mid E_1] = 1$.

(E2) $\alpha\beta < 0$. Conditioned on E2 happens, A always output $-x_i$ upon input z_{-i}. Thus in this case $\Pr[b = x_i \mid E_2] = 0$.

(E3) $\beta = 0$. In this case A outputs a uniform random bit. Thus $\Pr[b = x_i \mid E_3] = \frac{1}{2}$.

(E4) $\alpha = 0$ but $\beta \neq 0$. We claim that $\Pr[b = x_i \mid E_4] = 1/2$. Indeed, noting that A is deterministic, writing out all the randomness we have

$$\begin{aligned}
&\Pr[b = x_i \mid E_4] \\
=\ &\Pr\Big[A(z_{-i}, f(x)) = x_i \;\Big|\; \\
&f(x^{i\to 1}) = f(x^{i\to -1}), \\
&f(z^{i\to 1}) \neq f(z^{i\to -1})\Big]
\end{aligned}$$

Note that the event $E_4 = \left\{f(x^{i\to 1}) = f(x^{i\to -1}), f(z^{i\to 1}) \neq f(z^{i\to -1})\right\}$ only depends on x_{-i}, and is independent of x_i, so the above is

$$\begin{aligned}
&\Pr[b = x_i \mid E_4] \\
=\ &\frac{1}{2}\Big(\Pr[A(z_{-i}, f(x^{i\to 1})) = 1 \mid E_4] \\
&+ \Pr[A(z_{-i}, f(x^{i\to -1})) = -1 \mid E_4]\Big)
\end{aligned}$$

Note that conditioned on E_4,

$$\begin{aligned}&\Pr_{\substack{x_{-i}\sim\{\text{-}1,1\}^{n-1}\\ z_{-i}\sim N_\rho(x_{-i})}}[A(z_{-i}, f(x^{i\to -1})) = -1 \mid E_4]\\ =&\Pr_{\substack{x_{-i}\sim\{\text{-}1,1\}^{n-1}\\ z_{-i}\sim N_\rho(x_{-i})}}[A(z_{-i}, f(x^{i\to 1})) = -1 \mid E_4]\end{aligned}$$

This gives that

$$\begin{aligned}&\Pr_{\substack{x_{-i}\sim\{\text{-}1,1\}^{n-1}\\ z_{-i}\sim N_\rho(x_{-i})}}[A(z_{-i}, f(x^{i\to 1})) = 1 \mid E_4]\\ +&\Pr_{\substack{x_{-i}\sim\{\text{-}1,1\}^{n-1}\\ z_{-i}\sim N_\rho(x_{-i})}}[A(z_{-i}, f(x^{i\to -1})) = -1 \mid E_4]\\ =&1\end{aligned}$$

Therefore, the probability of guessing correctly is $\Pr[E_1] + \frac{\Pr[E_3]+\Pr[E_4]}{2}$. Observe that $\Pr[E_1] - \Pr[E_2]$ is exactly the ρ-stable influence ($\rho \in [0,1]$),

$$\begin{aligned}\Pr[E_1] - \Pr[E_2] &= \mathbb{E}[\mathrm{D}_i f(x)\, \mathrm{D}_i f(z)]\\ &= \mathbf{Stab}_\rho[\mathrm{D}_i f]\\ &= \mathbf{Inf}_i^{(\rho)}[f].\end{aligned}$$

Combining with $\Pr[E_1] + \Pr[E_2] + \Pr[E_3] + \Pr[E_4] = 1$, we have that

$$2\Pr[E_1] + \Pr[E_3] + \Pr[E_4] = 1 + \mathbf{Inf}_i^{(\rho)}[f]$$

Dividing by 2 on both sides gives the desired gain. □

For $\rho = 1$, $\mathbf{Inf}_i^1[h] = \mathbb{E}[\mathrm{D}_i f(x)\, \mathrm{D}_i f(z)] = \mathbb{E}[\mathrm{D}_i f(x)^2] = \mathbf{Inf}_i[h]$. We thus get back the influence in the noiseless model of Lemma 5.7.

Remark 5.15 (On Optimality)**.** *Unfortunately, for the ρ-independent perturbation model, Algorithm 9 no longer achieves the maximum possible gain. That is, there exists some model $h : \{-1,1\}^n \mapsto \{-1,1\}$ and some inversion algorithm A', such that the gain of A' is larger than $(1 + \mathbf{Inf}_i^{(\rho)}[h])/2$. The intuition is that, since the adversary knows $h(x)$ exactly, so it can leverage the function table of h to "de-noise", For example, consider OR_n. As long as we see that the model output is 1, we know that all input bits are 1. Therefore, the advantage we can achieve, even in the independent perturbation model, is $\mathbf{Inf}_i[\mathrm{OR}_n]/2 = 2^{-n}$ for any $0 \le \rho \le 1$. On the other hand, the advantage of Algorithm 9 is $\rho^{n-1}2^{-n}$. We pose the following question that we would like to investigate in the future.*

Question 1. *Consider ρ-independent perturbation model. Let A_ρ denote Algorithm 9. Is it the case that for any $h : \{-1,1\}^n \mapsto \{-1,1\}$, and any probabilistic algorithm A',*

$$\begin{aligned} &\mathsf{gain}(A', \mathsf{gen}_\rho, x_i, \mathscr{D}_U, S, \Gamma) \\ \le\ &\mathsf{gain}(A_\rho, \mathsf{gen}_\rho, x_i, \mathscr{D}_U, S, \Gamma) + o_n(1)\ ? \end{aligned}$$

Invertibility Interference

Intuitively it is clear that noise will negatively the gain of the adversary. Theorem 5.14 quantifies this intuition using stable influence. For example, as we saw in the above, the gain of a natural algorithm (Algorithm 9) on OR_n goes from $\mathbf{Inf}_i[\mathrm{OR}_n]$ to $\rho^{n-1}\mathbf{Inf}_i[\mathrm{OR}_n]$, which is exponentially small in the influence. However, this example is not very interesting in the sense that the influence of OR_n is already very small (2^{1-n}) in the noiseless case.

A more interesting phenomenon regarding noise is that highly invertible models in the noiseless case quickly becomes highly non-invertible due to a little noise. The reason behind is that multiple influential coordinates interfere with each other under noise. Let us see an exam-

ple. In the noiseless model the most invertible function is the parity function $\chi_{[n]} = \prod_{i\in[n]} x_i$. In this case, $\mathbf{Inf}_i[f] = 1$ for every i. On the other hand, under independent noise, the invertibility of χ_n becomes $\mathbf{Inf}_n^{(\rho)}\left[\chi_{[n]}\right] = \mathbf{Stab}_\rho\left[\mathrm{D}_n \chi_{[n]}\right] = \left\langle \chi_{[n-1]}, \mathrm{T}_\rho \chi_{[n-1]}\right\rangle = \rho^{n-1}$. Therefore the invertibility decays exponentially fast in n.

We term this phenomenon "invertibility interference:" When noise presents, if one does not know one of these influential coordinates exactly, then he cannot effectively invert the model to deduce the target feature. What is the stable influence if we have t influential coordinates? In this direction, we have the following simple result:

Theorem 5.16. *Suppose that $h : \{-1,1\}^n \mapsto \{-1,1\}$ has t coordinates with influence 1. Let $0 < \rho \le 1$, then for any $i \in [n]$, $\mathbf{Inf}_i^{(\rho)}[h] \le \rho^{t-1}\,\mathbf{Inf}_i[h]$.*

Proof. We know that $\mathbf{Inf}_i^{(\rho)}[f] = \sum_{S\ni i} \rho^{|S|-1}\widehat{f}(S)^2$. Consider any $S \subseteq [n]$ such that $|S| < t$. We show that $\widehat{f}(S) = 0$. For this let $A = \{x : f(x) = \chi_S(x)\}$. We show that $|A| = |A^c|$ where A^c is the complement of A. Indeed, consider any position $i^* \notin S$ such that $\mathbf{Inf}_{i^*}[f] = 1$. Such i^* must exist by our assumption. Thus the mapping $x \mapsto x^{\oplus i^*}$ is a mapping from A to A^c that is one-to-one and onto. Therefore,

$$\begin{aligned}\mathbf{Inf}_i^{(\rho)}[f] =& \sum_{S\ni i:|S|\ge t} \rho^{|S|-1}\widehat{f}(S)^2\\ \le& \rho^{t-1}\sum_{S\ni i}\widehat{f}(S)^2\\ \le& \rho^{t-1}\,\mathbf{Inf}_i[f].\end{aligned}$$

The proof is complete. □

Question 2. *If, instead of having coordinates of influence 1, we are only guaranteed that individual influence is lower bounded by $1-\delta$ for some $\delta > 0$, how fast will the stable influence decay with respect to δ?*

5.4 White-Box MI Attacks

We now move on to study white-box MI attacks. As discussed before, we assume that the computation of the models follows a sequential composition. This thus gives a natural view of MI attacks as *communication games*: One can think of *each layer of the model* as a player who sends a message to the next player, and the adversary as *another player* who observes the model output and has some additional information. Together, the goal of this game is to compute some function τ. This view gives a natural question:

"How would knowing the communication structure and (possibly) observing some intermediate information in the communication help MI attacks?"

Empirically, the answer is that it helps a lot. As mentioned earlier, it is essential for white-box attacks as studied in Fredrikson et al. (2015) to have access to the auxiliary confidence information, which makes the inversion algorithm much more effective.

The main purpose of this section is to give *theoretical justifications* for these empirical observations. At a high level, our results are summarized as follows:

1. Similar to our study of black-box MI attacks, we choose to specialize our methodology so as to obtain theoretical insights. To do so, we focus on white-box MI attacks on *decision trees*. An advantage of studying attacks on decision trees is its simplicity: The communication channel is very restricted, not only it is a sequential composition, but also in each iteration a player only reads a single bit of the input, and decides a binary output.

2. We show how to interpret white-box MI attacks on decision trees as alternating (communication) games. Specifically, these are communication games where the communication channel is one-way,

unicast (following the sequential composition), and players *alternatively* hold two inputs. We give examples showing that these communication games are very restricted.

3. We show that, however, even when restricting these communication games to have 1 bit of communication between two neighboring players, it is still the case that for *any* goal function τ, there exists a game with enough players (corresponding to a machine learning model with enough many layers), such that there is an adversary who can compute τ correctly *everywhere*. This result illustrates the unexpected computational power of a restricted communication game, and in particular, that the leakage can be significant even in a very restricted white-box case.

We now give more details in the rest of this section.

Decision Trees, MI Attacks, and Alternating Games

From now on we consider *oblivious decision trees*, which are decision trees in which the same feature is examined at each level (of the tree). This restricts the machine learning models we consider (it is even a subclass of decision trees). Note that, however, the more we restrict the model (and its communication), the stronger our conclusion (regarding leakage in the white-box case) is if we can show significant information leakage.

We have mentioned that for a model with sequential composition, the communication channel is restricted: it is *one-way* and *unicast*. That is, each player only sends one message to the next player, in a fixed order. This is in sharp contrast with communication games studied in typical communication complexity literature Damm et al. (1998); Chakrabarti (2007); Braverman et al. (2013); Fischer et al. (1985); Karp et al. (2000); Kempe et al. (2003); Ganor and Raz (2014), where the channel is either bidirectional or the messages are broadcasted.

We note that for oblivious decision trees, such communication games are further restricted. Consider an adversary who knows part of the input to the decision tree, then the communication game *alternates* between input he knows and input he does not know. Specifically, suppose that the input to the decision tree is $z \in \{0,1\}^n$, and assume that the adversary can see the bits at positions $K \subseteq [n]$ (K stands for "known" positions), then, without loss of generality, the communication game can be viewed as: the first player examines several variables at positions in K, then sends a bit to the next player, who then examines several variables in $[n] \setminus K$, and so on. The final player, which is the adversary (who knows bits in K), determines an output.

The following definition captures our discussion so far mathematically:

Definition 5.17 (Alternating MI Attacks (AMI Attacks))**.** *Let n, ℓ be natural numbers. Let $k \geq 3$ be also a natural number. In Alternating MI Attack there are $(k-1) \geq 2$ functions: $h_1, \ldots, h_{k-1}$ in the form of $h_1 : \{0,1\}^n \mapsto \{0,1\}^\ell$ and $h_i : \{0,1\}^n \times \{0,1\}^\ell \mapsto \{0,1\}^\ell$ $(i = 1, \ldots, k-1)$. let $h^{(1)}, \ldots, h^{(k-1)} : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^\ell$ be the following sequence:*

$$
\begin{aligned}
h^{(1)}(x,y) &= h_1(x) \\
h^{(i)}(x,y) &= h_i(y, h^{(i-1)}(x,y)) \quad i = 2, 4, \ldots \\
h^{(i)}(x,y) &= h_i(x, h^{(i-1)}(x,y)) \quad i = 3, 5, \ldots
\end{aligned}
$$

Let A be a probabilistic algorithm that is an "adversary." Let $\tau : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$ be a Boolean function on $2n$ bits. The alternating MI attack proceeds as follows:

i. *Nature samples x, y uniformly random from $\{0,1\}^n$.*

ii. *If k is odd, then nature presents x to A, but not y. Otherwise, nature presents y to A, but not x.*

iii. *Nature also presents the output of $h^{(k-1)}$, that is the output of the "outermost model", to A.*

For odd k, the gain of the alternating MI attack is measured by

$$\Pr[A(x, h^{(k-1)}(x, y)) = \tau(x, y)].$$

Similarly for even k, the gain is defined as $\Pr[A\left(y, h^{(k-1)}(x, y)\right) = \tau(x, y)\Big]$. *Both probabilities are taken over all the randomness: the randomness of sampling x, y (uniformly), the private randomness of $h_1, \ldots, h_k$, and the randomness of A.*

Note that the adversary in this formulation can only see the output of the outer model ($h^{(k-1)}$), beyond knowing part of the input. However, we also want to capture the intuition that the adversary may "inspect" some messages passed between layers in a machine learning model. We thus consider the following modification of the definition:

Definition 5.18 (Alternating MI Attacks with Early Inspection)**.** *In alternating MI attacks with early inspection, the only difference is that instead of feeding $h^{(k-1)}$ to the adversary, the adversary can* choose once *to inspect the output of $h^{(i)}$ ($1 \le i \le k-1$), and based on that to compute the output.*

Note that we restrict the adversary to be only able to inspect once — this is, again, to pose restriction on the communication, which gives stronger implication on the risk of leakage.

In the above definition of alternating MI attacks, we still need to distinguish between "layers" in a machine model, and the adversary. Towards our main result, which states that for any goal function τ there exists a model (with enough layers) that can allow an adversary to compute τ everywhere, we find that it is more convenient to work with a definition where we do not distinguish between functions inside a model

and the function computed by the adversary. This leads to the following definition.

Definition 5.19 (Alternating One-Way Unicast Communication Games (AOWU))**.** *Let n, k, ℓ be natural numbers. In Alternating One-Way Unicast Communication Games we have:*

(1) *The goal of the communication is to compute some function $\tau : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$.*

(2) *There are k players, $P_1, \ldots, P_k$. These players are allowed to use private randomness.*

(3) *The players communicate in the way of* one-way unicast. *Namely, they play in the fixed order of $P_1, \ldots, P_k$, and player P_i is only allowed to send one message, a bit string of length ℓ, to player P_{i+1}, for $i = 1, \ldots, k-1$.*

(4) *P_k is required to output a single bit, which is viewed as the output of the protocol.*

Similar to alternating MI attack, one can define the sequence of composed function $P^{(1)}, \ldots, P^{(k)}$, where $P^{(i)}$ is the function computed by the first i players on $\{0,1\}^n \times \{0,1\}^n$:

$$
\begin{aligned}
P^{(1)}(x,y) &= P_1(x) \\
P^{(i)}(x,y) &= P_i(y, P^{(i-1)}(x,y)) \quad i = 2, 4, \ldots \\
P^{(i)}(x,y) &= P_i(x, P^{(i-1)}(x,y)) \quad i = 3, 5, \ldots
\end{aligned}
$$

For AMI Attacks with Early Inspection, we have the following definition,

Definition 5.20 (AOWU*)**.** *An AOWU game with early stopping, or called an AOWU* game, is an AOWU game where any player P_i, $i \in [k]$, can stop the protocol, and claim his or her output as the output of the protocol.*

From our discussion so far it follows that

Lemma 5.21. *(n,k,ℓ)-alternating MI attack and (n,k,ℓ)-AOWU games are equivalent. Further, $(n,k,\ell$-alternating MI attack with early inspection and (n,k,ℓ)-AOWU* games are equivalent.*

Proof. By viewing the first k players $P_1,\ldots,P_k$ as models, the last player P_{k+1} as the adversary $\mathscr{A}$, and message strings in $\{0,1\}^\ell$ as the model output in the model composition, the proof is complete. □

On the Power of AMI Attacks with Early Inspection

We now study the power of alternating MI attacks with early inspection. Clearly, we can equivalently study AOWU games with early stop. We show that, even when restricting to 1 bit of communication, it is still surprisingly powerful.

To motivate this result, let us fist give an example, which illustrates "how restricted" these games are.

Example 5.22. *Let* $\mathrm{IP}(x,y)$ *be the inner product of x and y, that is for $x,y \in \{0,1\}^n$, $\mathrm{IP}(x,y) = \bigoplus_{i=1}^n (x_i \wedge y_i)$. Consider one-way unicast alternating games that try to compute* $\mathrm{IP}(x,y)$*. The communicated messages are restricted to be of* 1*-bit long (that is $\ell = 1$).*

*Let us consider the simple case that $n = 2$. That is, we want to compute the inner product of two length-*2 *bit strings. Note that in the traditional two-player communication model where Alice holds x and Bob holds y, then there is a trivial protocol where Alice sends to Bob* 2 *bit messages, one x_1 and one x_2, so that Bob then has complete knowledge of x and can compute* any function *on x,y.*

However, with AOWU games, there is a now a difficulty. Suppose that P_1 sends x_1 to P_2, and P_2 computes $x_1 y_1$. Then what will P_2 send to P_3? If P_2 sends y_2 to P_3, then the progress that P_2 has made is essentially lost. However,

if he sends x_1y_1, P_3 still does not know any information about y_2. Therefore, at least with 2 bits communication they cannot solve the problem. What is the "right" lower bound on the number of players that are needed in order to compute inner product with 1 bit of communication? □

We now construct a "universal" protocol that can compute any $\tau : \{0,1\}^n \times \{0,1\}^n$ using an AOWU* protocol with 1 *bit of communication*.

Construction 1 (Universal-1 Protocol)**.** *Let $\tau : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$ be any function. Consider the following protocol:*

- *There are $2 \cdot 2^n = 2^{n+1}$ players, split into pairs, $(1,2)$, $(3,4)$, $(2i-1, 2i), \ldots, (2^{n+1}-1, 2^{n+1})$. Note that odd-numbered players hold x, and even-numbered players hold y.*
- *Player $2i-1$ sends the following bit to player $2i$: He sends 1 if x is the lexicographically the i-th smallest string of all binary strings of length n. For example, player 1 sends 1 to player 2 if $x = 0$, and sends a bit 0 otherwise.*
- *For player $2i$, if she receives a bit 1, then she can be certain about the value of x. Because she also knows y, she can compute $\tau(x,y)$, stops the protocol early by asserting the special stopping bit, and claims the output. Otherwise, she keeps the special stop bit as 0 to indicate player $2i+1$ to continue the protocol.*

Note that early stopping is essential here, otherwise player $2i+1$ cannot distinguish between a "value" that is for computing τ and a "signal" which indicates that the computation is already done. Following the construction we immediately have the following theorem,

Theorem 5.23. *Universal-1 protocols compute any τ with 1-bit message and 2^{n+1} players.*

Thus we have obtained the claimed main result regarding alternating MI attacks with early inspection.

Theorem 5.24. *For any goal function $\tau : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$, there exists a machine learning model with $O(2^n)$ layers, and an alternating MI attack with 1-bit communication, that computes τ correctly everywhere.*

We close this section with two open questions.

Question 3. *For 1-bit communication, is universal-1 protocol essentially optimal in the sense that there is a function $\tau : \{0,1\}^n \times \{0,1\}^n$ where any protocol computing τ requires $\Omega(2^n)$ rounds of communications?*

Question 4. *For 1-bit communication, is there a universal AOWU protocol (instead of AOWU*) that computes every function $\tau : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$?*

5.5 Connections with Other Cryptographic Notions

In this section we compare MI attack with two classic cryptographic primitives: Hard-Core Predicate and Secure Multiparty Computation. We assume that the readers have some basic familiarity with cryptographic terminologies.

Connection with Hard-Core Predicate. Let us first recall the definition of hard-core predicate

Definition 5.25 (Hard-Core Predicate Goldreich (2001))**.** *Let U_n be a uniform distribution over $\{0,1\}^n$, and $f : \{0,1\}^* \mapsto \{0,1\}^*$. A polynomial time computable predicate $b : \{0,1\}^* \mapsto \{0,1\}$ is called a* hard-core *of f if for every probabilistic polynomial time algorithm A', there exists a negligible function*

$\mu(\cdot)$ such that for all sufficiently large n's

$$\Pr_{x\sim U_n}[A'(f(x)) = b(x)] \leq \frac{1}{2} + \mu(n).$$

By viewing f as a "model", one can then simulate this definition by a black-box MI attack. Specifically, let $\mathsf{out}(x) = b(x)$, which is to compute a single bit. The joint distribution is $J_U = (U_n, f(U_n))$. In the real world, given $x \sim U_n$, the auxiliary information generator gen gives the advice string $\mathsf{gen}(d_f, x, f(x)) = f(x)$ to the adversary A'. One can then observe that the gain of A' in the real world, $\mathsf{gain}_{J_U,f,b}(\mathsf{gen}, A')$, is exactly $\Pr_{(x,y)\sim J_U}[A'(y) = b(x)] = \Pr[A'(f(U_n)) = b(U_n)]$. Therefore the goal of hard-core predicate is to find a "hard" predicate b so that for any adversary A', any negligible function $\mu(\cdot)$, and all sufficiently large n's, the gain $\mathsf{gain}_{J_U,f,b}(\mathsf{gen}, A') \leq 1/2 + \mu(n)$.

Following this simulation one can also observe two notable differences: First, in an MI attack an adversary typically has more auxiliary information than the case of hard-core predicates (which only observes the function output). For example, in a black-box MI attack as defined in Definition 5.6, an adversary has information about all except one feature. Second, we note that the goal of MI attacks is not to construct hard predicates. Rather, its goal is somewhat its "dual": The purpose is to study the leakage of certain model with respect to computing some output function. For statistical output, understanding this leakage may help us decide what information can be safely published.

Connection with Secure Multiparty Computation (SMC). A more interesting notion to compare with is the Secure Multiparty Computation (SMC). To this end, we recall first the definition of m-party secure protocols

Definition 5.26 (m-party secure protocols – sketch Goldreich (2004))**.** *Let f be an m-ary functionality and Π be an m-party protocol operating in the*

real model.

- *For a real-model adversary A, controlling some minority of the parties (and tapping all communication channels), and an m-sequence $\overline{x}$, we denote by* $\mathrm{REAL}_{\Pi,A}(\overline{x})$ *the sequence of m outputs resulting from the execution of* Π *on input $\overline{x}$ under attack of the adversary A.*

- *For an ideal-model adversary A', controlling some minority of the parties, and an m sequence $\overline{x}$, we denote by* $\mathrm{IDEAL}_{\Pi,A}(\overline{x})$ *the sequence of m outputs resulting from the ideal process (that they together send input to a trusted third party, which then gives back the output) on input $\overline{x}$ under attack of the adversary A'.*

We say that Π *securely implements f with honest majority if for every feasible real model adversary A, controlling some minority of the parties, there exists a feasible ideal model, controlling the same parties, so that the probability ensembles* $\{\mathrm{REAL}_{\Pi,A}(\overline{x})\}_{\overline{x}}$ *and* $\{\mathrm{IDEAL}_{\Pi,A}(\overline{x})\}_{\overline{x}}$ *are computationally indistinguishable.*

We observe that in this definition the privacy concerns of the *output* is *not* considered. Specifically, it can happen that in the ideal case, upon receiving the output from the trusted third party, one can infer partial information about the input of the other party. Yet such concerns will not factor in the distinguishability as they are contained in the ideal world. Put in another way, the privacy concerns considered in a secure protocol is whether the *communication* among parties in the real world leaks *additional* information compared to the ideal case.

Black-box MI Attacks and SMC. By contrast, for *black-box MI attack*, one considers precisely the privacy concerns of the *output*. That is how much sensitive information an adversary can recover from the output. To this end, one may argue that it is questionable why should one be

bothered with the concern of the output, since this is the purpose of the computation.

On one hand, we feel that a fundamental difference here is what information constitutes the output. In the setting of SMC, the output is precisely defined (for example, whether two inputs are equal). However, in the setting of MI attack, the output is statistical and "noisy." For example, a model may carry too much information of some individuals if the learning procedure over-fits. Publishing such models may thus induce effective MI attacks and unwanted disclosure. Studying MI attacks can help us identify and quantify such leakage.

On the other hand, we note that, frequently, additional information is intentionally revealed by an SMC protocol due to performance considerations (for example, revealing the centroid in privacy-preserving clustering or revealing a bit of a honest party's input in the dual-execution SMC protocol Huang et al. (2012)). However, the ramifications of leaking this additional information are unclear. Perhaps our framework can be used to address such problems.

More concretely, let us consider a simple example where our current results for MI attacks (though studied in the setting of machine learning) can be applied to SMC. Consider two parties Alice and Bob who jointly compute a Boolean function $f(x, y)$, where x is Alice's input and y is Bob's input. Suppose that the inputs are drawn uniformly from two sets X, Y. Now if Bob is malicious, and can see insensitive information x_{-i} of Alice for $x \sim X$, then with access to f how much better can he guess x_i over X, compared to random guessing?

Let $f_y(x) = f(x, y)$ (i.e. f_y is the specialization of f where the second input is y). The answer to this question becomes exactly an MI problem against uniform distribution over X, where the adversary Bob has auxiliary information x_{-i}. Our results in Section 5.3 tells that the advantage is the influence of coordinate i of the function $f_y(\cdot)$

Therefore, what remains is to estimate the influence of coordinate i of f_y. In this direction, we note that recent years there has been interesting progress on the algorithmic side of the influence theory. For example, a recent work by Ron, Rubinfeld, Safra, Samorodnistsky and Weinstein Ron et al. (2012) proves lower bounds in approximating influence and gives a better upper bound for monotone Boolean functions. By invoking their influence estimation algorithms (for example, their Algorithm 1), one can thus obtain a quantitative understanding of the risk of outputting the function f.

White-box MI Attacks and SMC. This situation changes when we consider *white-box MI attack*. Intuitively, in composed MI attack, if one view sub-models as "parties", then we can ask how much information do these compositions (or communication) leak. Therefore, even if one modulo the concerns of the output of the *outer-most* model, one could still investigate the *additional* leakage caused by the composition (or communication). This is closer to the goal of SMC.

Unlike SMC, however, is that the communication pattern of the white-box MI attacks is usually much more restricted. For example, composition of models in the usual sense gives "one-way unicast communication", rather than broadcasts, or arbitrary point-to-point communication (so it is not one-way). Indeed, as we saw in the chapter, this way of communication induces intriguing and somewhat unexpected connections with communication complexity that does not seem to have been studied before.

6 CONCLUSION AND FUTURE DIRECTIONS

In this thesis we study the interplay of stability with data privacy. From differential privacy to model inversion attacks, our study shows that stability, though may be in different forms, plays a key role. Specifically, in Chapter 3 we achieve better differentially private stochastic gradient descent by giving a novel analysis of the global stability of stochastic gradient descent. In Chapter 4 we clarify and quantify the relationship between differential privacy and model inversion by analyzing the stability of empirical risk minimization. Finally, in Chapter 5 we study a different form of stability, the influence, that is closely related to model inversion attacks.

There are many intriguing future directions to pursue. For differentially private optimization, an important direction is to have a better understanding of the convergence behavior of private SGD when we can only afford to do a constant number of passes over the data. BST14 Bassily et al. (2014) provides a convergence bound for private SGD when $O(m)$ passes are made over the data. SCS13 Song et al. (2013) does not provide a convergence proof; however, the work of Duchi, Jordan and Wainwright Duchi et al. (2013), which considers *local differential privacy*, a privacy model where data providers do not even trust the data collector, can be used to conclude convergence for SCS13, though at a very slow rate. Finally, while our method converges very well in practice with multiple passes, we can only prove convergence with *one pass*, and the bound is weaker than BST14, which uses m passes. Can we prove convergence bounds of our algorithms for multiple-passes and match the bounds of BST14?

For model inversion attacks our study barely scratches the surface. We cannot answer even basic questions such as the optimality of stable influence. Further, our understanding of "invertibility interference" is

also shallow. There are also many questions regarding white-box model inversion attacks and their connection to restricted communication protocols. To this end, it seems that the natural first step is to further study AOWU and AOWU* protocols.

REFERENCES

Apache Mahout. mahout.apache.org.

Aono, Yoshinori, Takuya Hayashi Le Trieu Phong, and Lihua Wang. 2015. Fast and secure linear regression and biometric authentication with security update.

Applegate, David, and Ravi Kannan. 1991. Sampling and integration of near log-concave functions. In *Proceedings of the 23rd annual ACM symposium on theory of computing, may 5-8, 1991, new orleans, louisiana, USA*, 156–163.

Barthe, Gilles, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2016. Proving differential privacy via probabilistic couplings. *CoRR* abs/1601.05047.

Bassily, Raef, Adam Smith, and Abhradeep Thakurta. 2014. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*.

Braverman, Mark, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. 2013. Tight bounds for set disjointness in the message passing model. *CoRR* abs/1305.4696.

Bubeck, Sébastien. 2015. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning* 8(3-4):231–357.

Chakrabarti, Amit. 2007. Lower bounds for multi-player pointer jumping. In *22nd annual IEEE conference on computational complexity (CCC 2007), 13-16 june 2007, san diego, california, USA*, 33–45.

Chaudhuri, Kamalika. personal communication.

Chaudhuri, Kamalika, Claire Monteleoni, and Anand D. Sarwate. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12:1069–1109.

Chaudhuri, Kamalika, and Staal A. Vinterbo. 2013. A stability-based validation procedure for differentially private machine learning. In *NIPS*.

Damm, Carsten, Stasys Jukna, and Jiri Sgall. 1998. Some bounds on multiparty communication complexity of pointer jumping. *Computational Complexity* 7(2):109–127.

Daneshi, Maryam, and JQ Guo. 2011. Image reconstruction based on local feature descriptors. *Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, Tech. Rep*.

d'Angelo, Emmanuel, Laurent Jacques, Alexandre Alahi, and Pierre Vandergheynst. 2012. From bits to images: Inversion of local binary descriptors. *CoRR* abs/1211.1265.

Diaz, Francisco J., and Jose Yeh, Hung-Wen de Leon. 2012. Role of statistical random-effects linear models in personalized medicine. *Current Pharmacogenomics and Personalized Medicine* 10(1):22–32.

Dinur, Irit, and Kobbi Nissim. 2003. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, june 9-12, 2003, san diego, ca, USA*, 202–210.

Duchi, John C., Michael I. Jordan, and Martin J. Wainwright. 2013. Local privacy and statistical minimax rates. In *FOCS*.

Dwork, Cynthia, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*.

Dwork, Cynthia, and Moni Naor. 2008. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality* 2(1):8.

Dwork, Cynthia, and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9(3-4):211–407.

Dwork, Cynthia, Adam D. Smith, Thomas Steinke, Jonathan Ullman, and Salil P. Vadhan. 2015. Robust traceability from trace amounts. In *IEEE 56th annual symposium on foundations of computer science, FOCS 2015, berkeley, ca, usa, 17-20 october, 2015*, 650–669.

Erlingsson, Úlfar, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In *ACM ccs*.

Feng, Xixuan, Arun Kumar, Benjamin Recht, and Christopher Ré. 2012. Towards a unified architecture for in-rdbms analytics. In *SIGMOD*.

Fischer, Michael J., Nancy A. Lynch, and Mike Paterson. 1985. Impossibility of distributed consensus with one faulty process. *J. ACM* 32(2): 374–382.

Fredrikson, Matthew, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd acm conference on computer and communications security*.

Fredrikson, Matthew, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *Proceedings of the 23rd USENIX security symposium, san diego, ca, usa, august 20-22, 2014.*, 17–32.

Ganor, Anat, and Ran Raz. 2014. Space pseudorandom generators by communication complexity lower bounds. In *Approximation, randomization, and combinatorial optimization. algorithms and techniques, APPROX-/RANDOM 2014, september 4-6, 2014, barcelona, spain*, 692–703.

Goldreich, Oded. 2001. *The foundations of cryptography - volume 1, basic techniques*. Cambridge University Press.

———. 2004. *The foundations of cryptography - volume 2, basic applications*. Cambridge University Press.

Gray, Jim, et al. 1997. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Min. Knowl. Discov.* 1(1):29–53.

Hardt, M., B. Recht, and Y. Singer. 2015. Train faster, generalize better: Stability of stochastic gradient descent. *ArXiv e-prints*. 1509.01240.

Hardt, Moritz. 2015. Towards practicing differential privacy. http://blog.mrtz.org/2015/03/13/practicing-differential-privacy.html.

Haussler, David. 1992. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inf. Comput.* 100(1): 78–150.

Hellerstein, Joe, et al. 2012. The MADlib Analytics Library or MAD Skills, the SQL. In *VLDB*.

Huang, Yan, Jonathan Katz, and David Evans. 2012. Quid-pro-quo-tocols: Strengthening semi-honest protocols with dual execution. In *IEEE symposium on security and privacy, SP 2012, 21-23 may 2012, san francisco, california, USA*, 272–284.

International Warfarin Pharmacogenetic Consortium. 2009. Estimation of the warfarin dose with clinical and pharmacogenetic data. *New England Journal of Medicine* 360(8):753–764.

Jain, Prateek, Pravesh Kothari, and Abhradeep Thakurta. 2012. Differentially private online learning. In *COLT*.

Jain, Prateek, and Abhradeep Thakurta. 2013. Differentially private learning with kernels. In *ICML*.

Johnson, Rie, and Tong Zhang. 2013. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*.

Karp, Richard M., Christian Schindelhauer, Scott Shenker, and Berthold Vöcking. 2000. Randomized rumor spreading. In *41st annual symposium on foundations of computer science, FOCS 2000, 12-14 november 2000, redondo beach, california, USA*, 565–574.

Kasiviswanathan, Shiva Prasad, Mark Rudelson, and Adam Smith. 2013. The power of linear reconstruction attacks. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on discrete algorithms, SODA 2013, new orleans, louisiana, usa, january 6-8, 2013*, 1415–1433.

Kasiviswanathan, Shiva Prasad, and Adam Smith. 2008. A note on differential privacy: Defining resistance to arbitrary side information. *CoRR* abs/0803.3946.

Kato, Hiroharu, and Tatsuya Harada. 2014. Image reconstruction from bag-of-visual-words. In *2014 IEEE conference on computer vision and pattern recognition, CVPR 2014, columbus, oh, usa, june 23-28, 2014*, 955–962.

Kempe, David, Alin Dobra, and Johannes Gehrke. 2003. Gossip-based computation of aggregate information. In *44th symposium on foundations*

of computer science (FOCS 2003), 11-14 october 2003, cambridge, ma, usa, proceedings, 482–491.

Kifer, Daniel, and Ashwin Machanavajjhala. 2011. No free lunch in data privacy. In *Sigmod conference*, 193–204.

———. 2014. Pufferfish: A framework for mathematical privacy definitions. *ACM Trans. Database Syst.* 39(1):3.

Kifer, Daniel, Adam D. Smith, and Abhradeep Thakurta. 2012. Private convex optimization for empirical risk minimization with applications to high-dimensional regression. In *COLT*.

Li, Ninghui, Wahbeh H. Qardaji, Dong Su, Yi Wu, and Weining Yang. 2013. Membership privacy: a unifying framework for privacy definitions. In *ACM ccs*, 889–900.

Lindell, Yehuda, and Eran Omri. 2011. A practical application of differential privacy to personalized online advertising. *IACR Cryptology ePrint Archive* 2011:152.

McGregor, Andrew, Ilya Mironov, Toniann Pitassi, Omer Reingold, Kunal Talwar, and Salil P. Vadhan. 2011. The limits of two-party differential privacy. *Electronic Colloquium on Computational Complexity (ECCC)* 18: 106.

McKeague, IanW., and Min Qian. 2011. Sparse functional linear regression with applications to personalized medicine. In *Recent advances in functional data analysis and related topics*. Contributions to Statistics, Physica-Verlag HD.

McSherry, Frank, and Kunal Talwar. 2007. Mechanism design via differential privacy. In *48th annual IEEE symposium on foundations of computer science (FOCS 2007), october 20-23, 2007, providence, ri, usa, proceedings*, 94–103.

Nemirovsky, AS, and DB Yudin. 1983. Problem complexity and method efficiency in optimization.

Nesterov, Yurii. 2004. *Introductory lectures on convex optimization : a basic course*. Applied optimization, Kluwer Academic Publ.

O'Donnell, Ryan. 2014. *Analysis of boolean functions*. Cambridge University Press.

Parikh, Neal, and Stephen P. Boyd. 2014. Proximal algorithms. *Foundations and Trends in Optimization* 1(3):127–239.

Polyak, Boris T. 1987. *Introduction to optimization*. Optimization Software.

Reed, Jason, Adam J. Aviv, Daniel Wagner, Andreas Haeberlen, Benjamin C. Pierce, and Jonathan M. Smith. 2010. Differential privacy for collaborative security. In *Proceedings of the third european workshop on system security, EUROSEC 2010, paris, france, april 13, 2010*, 1–7.

Ron, Dana, Ronitt Rubinfeld, Muli Safra, Alex Samorodnitsky, and Omri Weinstein. 2012. Approximating the influence of monotone boolean functions in $o(\sqrt{n})$ query complexity. *TOCT* 4(4):11.

Roux, Nicolas Le, Mark W. Schmidt, and Francis R. Bach. 2012. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*.

Rubinstein, Benjamin I. P., Peter L. Bartlett, Ling Huang, and Nina Taft. 2009. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *CoRR* abs/0911.5708.

Shalev-Shwartz, S., and S. Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

Shalev-Shwartz, Shai, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. 2009. Stochastic convex optimization. In *COLT*.

Shalev-Shwartz, Shai, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. 2010. Learnability, stability and uniform convergence. *The Journal of Machine Learning Research* 11:2635–2670.

Shamir, O. 2016. Without-Replacement Sampling for Stochastic Gradient Methods: Convergence Results and Application to Distributed Optimization. *ArXiv e-prints*. 1603.00570.

Song, Shuang, Kamalika Chaudhuri, and Anand D. Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *GlobalSIP*.

Valiant, Leslie G. 1984. A theory of the learnable. *Commun. ACM* 27(11): 1134–1142.

Vapnik, Vladimir. 1998. *Statistical learning theory*. Wiley.

Wang, Yue, Cheng Si, and Xintao Wu. 2015. Regression model fitting under differential privacy and model inversion attack. In *Proceedings of the twenty-fourth international joint conference on artificial intelligence, IJCAI 2015, buenos aires, argentina, july 25-31, 2015*, 1003–1009.

Winslett, Marianne, Yin Yang, and Zhenjie Zhang. 2012. Demonstration of damson: Differential privacy for analysis of large data. In *18th IEEE international conference on parallel and distributed systems, ICPADS 2012, singapore, december 17-19, 2012*, 840–844.

Zhang, Jun, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. 2013. Privgene: differentially private model fitting using genetic algorithms. In *SIGMOD*.

Zhang, Jun, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. 2012. Functional mechanism: Regression analysis under differential privacy. *PVLDB*.

Zinkevich, Martin. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*.

Zinkevich, Martin, Markus Weimer, Alexander J. Smola, and Lihong Li. 2010. Parallelized stochastic gradient descent. In *NIPS*.