2017 IEEE Symposium on Security and Privacy

Is Interaction Necessary for Distributed Private Learning?

Adam Smith*, Abhradeep Thakurta†, Jalaj Upadhyay*
*School of Electrical Engineering and Computer Science,
Pennsylvania State University, Email: {asmith, jalaj}@psu.edu
†Department of Computer Science, University of California Santa Cruz,
Email: aguhatha@ucsc.edu

***Abstract*—Recent large-scale deployments of differentially private algorithms employ the *local* model for privacy (sometimes called *PRAM* or *randomized response*), where data are randomized on each individual's device before being sent to a server that computes approximate, aggregate statistics. The server need not be trusted for privacy, leaving data control in users' hands.**

For an important class of convex optimization problems (including logistic regression, support vector machines, and the Euclidean median), the best known locally differentially-private algorithms are highly interactive, requiring as many rounds of back and forth as there are users in the protocol.

We ask: how much interaction is necessary to optimize convex functions in the local DP model? Existing lower bounds either do not apply to convex optimization, or say nothing about interaction.

We provide new algorithms which are either noninteractive or use relatively few rounds of interaction. We also show lower bounds on the accuracy of an important class of noninteractive algorithms, suggesting a separation between what is possible with and without interaction.

***Keywords*-Differential privacy, local differential privacy, convex optimization, oracle complexity.**

I. Introduction

Each of us generates vast quantities of data as we interact with modern networked devices. Accurate aggregate statistics about those data can generate valuable benefits to society—higher quality healthcare, more efficient systems and lower power consumption, among others. However, those data are highly sensitive, painting detailed pictures of our lives. Private data analysis, broadly, seeks to enable the benefits of learning from these data without exposing individual-level information. *Differential privacy* [17] is a rigorous privacy criterion for data analysis that provides meaningful guarantees regardless of what an adversary knows ahead of time about individuals' data [25]. Differential privacy is now widely studied and algorithms satisfying the criterion are increasingly deployed [1, 2, 19].

There are two well-studied models for implementing differentially-private algorithms. In the *central* model, raw data are collected at a central server where they are processed by a differentially-private algorithm. In the *local* model [20] (dubbed *LDP* and illustrated in Figure 1), each individual applies a differentially-private algorithm locally to their data and shares only the output of the algorithm—called a report or response—with a server that aggregates users' reports. The local model allows individuals to retain control of their data since privacy guarantees are enforced directly by their devices. However, it entails a different set of algorithmic techniques from the central model. In principle, one could also use cryptographic techniques such as secure function evaluation to simulate central model algorithms in a local model, but such algorithms currently impose bandwidth and liveness constraints that make them impractical for large deployments. For example, Google [19] now collects certain usage statistics from users' devices subject to local differential privacy; those algorithms are run by hundreds of millions of users.

A long line of work studies what is achievable by LDP algorithms, and tight upper and lower bounds known on the achievable accuracy for many problems; see Sec. I-C. For a large class of optimization problems, however, the algorithms that achieve the upper bound are highly interactive—the server exhanges messages back and forth in sequence with each user in the system (see Figure 1). Implementing interactive protocols for private data collection is difficult, because network latency introduces delays and because the server must be live throughout the protocol. Consequently, existing large-scale deployments [19] are limited to noninteractive algorithms.

The question naturally arises: how much power is lost by restricting to noninteractive protocols? Kasiviswanathan et al. [26] studied the role of interaction in locally private algorithms, exhibiting a problem that can be solved using a linear (in the dimension) amount of data by a 2-round protocol but for any noninteractive protocol requires an exponential-sized data set. The problem they study is somewhat unatural, based on learning parity functions; their results say little about the computations commonly carried out in machine learning

DOI 10.1109/SP.2017.35

or statistical analysis.

Contributions. This paper initiates the study of interaction in local differential privacy (LDP) for important and natural learning problems. Specifically, we focus on convex optimization, which encompasses the calculation of descriptive statistics, such as the median, as well as more sophisticated computations, such as fitting linear or logistic regression models, training support vector machines and sparse regression. Tight upper and lower bounds are known for the accuracy of LDP convex optimization [14]. However, the upper bounds are highly interactive, requiring as many rounds of back and forth as there are users in the protocol.

We provide new algorithms for noninteractive LDP optimization of convex Lipschitz functions over a bounded parameter space. These algorithms improve considerably over naïve approaches. For one-dimensional problems (e.g., median), our algorithms attain the same optimal error bounds as interactive solutions. For higher-dimensional problems, our algorithms' error guarantees are worse than the bounds for interactive algorithms, since our guarantees decay exponentially as the dimension increases (instead of polynomially).

We provide evidence that this exponential dependence is necessary. We show lower bounds on the error of a natural class of nonadaptive optimization algorithms, which includes the noninteractive LDP variants of first-order methods such as gradient descent. This lower bound applies even to nonprivate algorithms. It demonstrates that the adaptivity of first- and second-order methods is necessary to get a polynomial dependence on the dimension.

We also consider algorithms that use interaction only sparingly. We show that carefully tuned LDP variants of classical first-order methods can return accurate answers with relatively few rounds of interaction. Geometric properties of the loss functions being optimized—smoothness and strong convexity—play a strong role in how quickly the error drops with the number of rounds of interaction.

Finally, we consider a specific problem—linear regression—and show a simple noninteractive protocol that achieves optimal error.

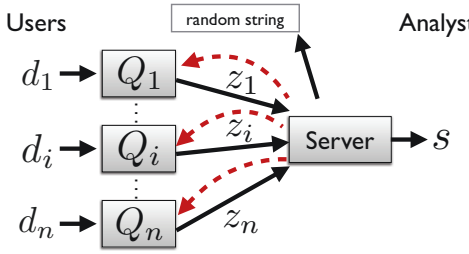

Figure 1. The local model for differential privacy. Each individual i runs a (possibly different) DP algorithm Q_i to randomize her data. In the noninteractive variant (without the dashed arrows), the server sends a single message to all users at the start of the protocol. In the interactive variant (with the dashed arrows), the server sends several messages, each to a subset of users. Each such message, together with responses from users, counts as a *round* of interaction.

A. Background

Differential privacy. In an LDP algorithm, there are n participants, each with a private input d_i from a universe $\mathcal{D}$ of possible values. A protocol proceeds in T rounds. In each round, the server sends a message, called a query, to a subset of the players, requesting that they run a particular algorithm. Based on the queries, each player i in the subset selects an algorithm Q_i, runs it, and sends the output back to the server.

Definition 1 ([20, 17])**.** *An algorithm Q_i is ε-locally differentially private (LDP) if for all pairs $d, d' \in \mathcal{D}$, and for all events E in the output space of Q, we have*

$$\Pr[Q(d) \in E] \leq e^{\varepsilon} \Pr[Q(d') \in E] .$$

A multi-player protocol is ε-LDP if for all possible inputs and runs of the protocol, the transcript of player i's interactions with the server is ε-LDP (for all settings of the remaining data points). [1] *In all the protocols we discuss, each user responds to only a single query over the course of the protocol.*

Convex Optimization. We consider algorithms for convex risk minimization. A particular problem is specified by a convex, closed and bounded constraint set $\mathcal{C}$ in $\mathbb{R}^p$ and a function $\ell : \mathcal{C} \times \mathcal{D} \to \mathbb{R}$ which is convex in its first argument, that is, for all $d \in \mathcal{D}$, $\ell(\cdot; d)$ is convex. We call p the dimension of the problem. A data set $\mathbf{D} = d_1, ..., d_n \in \mathcal{D}^n$ defines a loss (or *empirical risk*) function: $\widehat{L}(\theta; \mathbf{D}) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; d_i)$. For example, finding the median of a data set $\mathbf{D} \in [0, 1]^n$ corresponds to minimizing the loss $\widehat{L}(\theta, \mathbf{D}) = \sum_i |\theta - d_i|$. For standard linear regression, each data point is a pair $d_i = (x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$ and we seek to minimize $\widehat{L}(\theta, \mathbf{D}) = \sum_i (y_i - \langle \theta, x_i \rangle)^2$. Support vector machines correspond to minimizing $\widehat{L}(\theta, \mathbf{D}) = \sum_i (y_i \cdot \langle \theta, x_i \rangle)_+$, where $(x)_+$ denotes $\max(x, 0)$.

When the inputs are drawn i.i.d. from an underlying distribution $\mathcal{P}$ on $\mathcal{D}$, one can also seek to minimize the

[1] In Appendix C, we give an algorithm for linear regression that satisfies an "approximate" variant of this definition, called (ε, δ)-differential privacy [16]. All our lower bounds apply to both variants.

population risk, or *generalization error*, defined as the expected error on a fresh example from the distribution: $L_{\mathcal{P}}(\theta) = \mathbb{E}_{D\sim\mathcal{P}}[\ell(\theta; D)]$. We drop the subscript $\mathcal{P}$ when it is clear from the context.

We state the error of our algorithms in terms of their *excess* (empirical or population) risk. Given an output $\theta_{\mathsf{priv}} \in \mathcal{C}$, we define two variants of excess risk.

$$\text{Empirical: } err_{\mathbf{D}}(\theta_{\mathsf{priv}}) = \widehat{L}(\theta_{\mathsf{priv}}; \mathbf{D}) - \min_{\theta\in\mathcal{C}} \widehat{L}(\theta; \mathbf{D})$$
$$\text{Population: } err_{\mathcal{P}}(\theta_{\mathsf{priv}}) = L_{\mathcal{P}}(\theta_{\mathsf{priv}}) - \min_{\theta\in\mathcal{C}} L_{\mathcal{P}}(\theta)\,.$$

The empirical error measures how well our output does on the data set at hand. The population error assumes the data is drawn from some distribution, and measures how well our algorithm does on unseen examples from the same distribution. The measures are closely related, but not the same (roughly, algorithms that "overfit" may have low empirical error but high population error).

We consider additional restrictions on the loss function ℓ. Ignoring the second argument for a moment, we say a function $\ell : \mathcal{C} \to \mathbb{R}$ is L-Lipschitz if for all $\theta, \theta' \in \mathcal{C}$, $|\ell(\theta) - \ell(\theta')| \leq L\|\theta - \theta'\|_2$. (Unless otherwise specified, we work with the ℓ_2 norm on $\mathbb{R}^p$.) We say ℓ is Δ-strongly convex if, for every $\theta, \theta' \in \mathcal{C}$ and for every subgradient $\nabla\ell(\theta) \in \partial\ell(\theta)$, we have $\ell(\theta') \geq \langle\nabla\ell(\theta), \theta' - \theta\rangle + \frac{1}{2}\Delta^2\|\theta' - \theta\|_2^2$. We say ℓ is β-smooth if it is differentiable and has β-Lipschitz gradients, that is $\|\nabla\ell(\theta) - \nabla\ell(\theta')\|_2 \leq \beta\|\theta - \theta'\|_2$.

Nonprivate methods for optimizing convex functions generally use first- or second-order methods, which gain information about the loss funcion by evaluating the gradient, and possibly the Hessian (matrix of second derivatives) at a sequence of points in $\mathcal{C}$. Examples of such methods include gradient descent, cutting plane algorithms, Frank-Wolfe, and Newton-Raphson.

"Typical" Setting. In what follows, we assume $\mathcal{C} \subset \mathbb{R}^p$ is a convex set ($\|\mathcal{C}\|_2 \leq 1$) and $\ell : \mathcal{C}\times\mathcal{D} \to \mathbb{R}$ is convex and 1-Lipschitz for each setting of its second argument. For this setting, Duchi et al. [14] gave an n-round algorithm with expected population risk $O\left(\sqrt{\frac{p}{\varepsilon^2 n}}\right)$, where n is the number of users. Their algorithm is a LDP version of stochastic gradient descent where, at round i, the server sends the current estimate θ_i to player i, who returns a noisy gradient $Q(\nabla\ell(\theta_i; d_i))$ (where Q adds carefully calibrated noise). They also showed that error bound is tight, using an information theoretic argument. The lower bound applies even to linear loss functions—in particular, the bound shows that assuming smoothness does not change the achievable error.

B. Summary of Results

Noninteractive Algorithms for General Convex Optimization.

Theorem 2 (Theorem 10, informal)**.** *For the typical setting above, there is an ε-LDP algorithm $\mathcal{A}$ such that for all distributions $\mathcal{P}$ on $\mathcal{D}$, with high probability, $err_{\mathcal{P}}(\mathcal{A}(\mathbf{D})) = \widetilde{O}\left(\left(\frac{\sqrt{p}}{\varepsilon^2 n}\right)^{1/(p+1)}\right)$, where $\tilde{O}(\cdot)$ hides $\log(n)$ factors.*

For one-dimensional convex optimization, our protocol nearly matches the lower bound of $\Omega(1/\sqrt{\varepsilon^2 n})$. The dependence on the dimension is exponential, however. To achieve a given level of error α, one requires $n \geq \widetilde{\Omega}(c^p\varepsilon^{-2}\alpha^{-(p+1)})$ data points.

Our one-dimensional algorithm is based on a novel reduction of the general one-dimensional convex optimization to the median problem (a special case), followed by a noninteractive local algorithm for the median that uses a tree-based technique for simultaneously approximating 1D range queries (that arises in [22, 18, 12]). For higher dimensions, we reduce to the one-dimensional case by optimizing, in parallel, over a collection of roughly $1/\alpha^{p-1}$ random lines passing through the center of $\mathcal{C}$.

As a comparison point, the most straightforward approach to noninteractive LDP optimization is to evaluate the loss function at all points in a suitably defined set (a "net") for $\mathcal{C}$, and then output the one with smallest loss. This approach incurs error $\widetilde{O}\left(\left(\frac{p}{\varepsilon^2 n}\right)^{1/(p+2)}\right)$ (or alternatively, requires a sample of size $4^p\varepsilon^{-2}\alpha^{-(p+2)}$ for error α). Our technique saves a factor of $1/\alpha$ in the sample complexity, which is significant for small α.

Bounds on Adaptivity for General Convex Optimization. We show that for a natural class of LDP methods, the exponential dependence on p in our methods is necessary. We start from the observation that methods for general convex optimization, private or not, generally access the loss function by approximating the gradient at a sequence of adaptively chosen points. We model such algorithms by imagining an oracle to which the algorithm makes queries. The oracle is *neighborhood-based* (usually called *local*[2]) if, on query $\theta \in \mathcal{C}$, it returns information about the values of the loss function in an infinitesimal neighborhood of θ (a subgradient or Hessian, for example).

We study, for the first time, the adaptivity of such algorithms: suppose that the algorithm submits queries

[2]This use of "local" is completely different from its use in "local differential privacy". We use "neighborhood-based" for clarity.

in batches to the oracle, with the choice of points in a batch depending only on query answers from previous batches. A nonadaptive algorithm uses only one batch (and corresponds to a noninteractive LDP protocol). We show that for every noninteractive neighborhood-based oracle algorithm requires $(1/\alpha)^{\Omega(p)}$ queries in the worst case to obtain error α for optimizing Lipschitz convex functions:

Theorem 3 (Theorem 13, informal)**.** *There exists $C > 0$ such that for every sufficiently small $\alpha > 0$ and every (not necessarily private) neighborhood-based oracle $\mathcal{O}_.(\cdot)$, every $C\log(1/\alpha)$-round randomized algorithm for optimization of Lipschitz convex functions requires $2^{\Omega(p)}$ queries to succeed with high probability. Furthermore, nonadaptive algorithms require $(1/\alpha)^{\Omega(p)}$ queries.*

This is the first lower bound to demonstrate that the adaptivity of first-order methods such as gradient descent and the cutting-plane method is in fact necessary to get a polynomial dependence on the dimension. It also demonstrates that fundamentally new techniques would be necessary to get noninteractive LDP algorithms with polynomial dependence on the dimension. Previous bounds on oracle optimization [3, 34, 33, 37] used information-theoretic arguments that do not distinguish between adaptive and nonadaptive algorithms (in particular, the instances that arise in those proofs are easy to solve nonadaptively).

Algorithms with Limited Adaptivity. Although the design of accurate noninteractive algorithms for high-dimensional optimization remains challenging, we show that LDP algorithms with limited interaction can achieve low error, demonstrating that the n rounds of interaction of previous algorithms (where n is the number of users) are not necessary. These algorithms are noisy, "batch stochastic" versions of two classic first-order methods—gradient descent and the cutting plane method—where in each round many users are queried to get high-accuracy estimates of the gradient at a particular point. We show:

1) In the "typical" setting (optimizing 1-Lipschitz functions over a bounded set) then for every T, there is an ε-LDP algorithm $\mathcal{A}(\cdot)$ such that for every $\mathbf{D} \in \mathcal{D}^n$, $\mathbb{E}[err_{\mathbf{D}}(\mathcal{A}(\mathbf{D}))] = O\left(\min\left(\sqrt{\frac{T}{n\varepsilon^2}} + \left(1-\frac{1}{e}\right)^{T/p}, \frac{1}{\sqrt{T}} + \sqrt{\frac{p}{\varepsilon^2 n}}\right)\right)$. In particular, this algorithm achieves optimal error $O(\sqrt{p/\varepsilon^2 n})$ for $T = n\varepsilon^2/p$ (due to the second term). On the other hand, the first term reaches the optimal error when $T = O(p\log(\varepsilon^2 n/p))$.
2) When the function is 1-Lipschitz, β-smooth and Δ-strongly convex, there is an ε-LDP algorithm $\mathcal{A}(\cdot)$ that for every $\mathbf{D} \in \mathcal{D}^n$, $\mathbb{E}[err_{\mathbf{D}}(\mathcal{A}(\mathbf{D}))] = O\left(\frac{\beta}{2}e^{-\Delta T/\beta} + \frac{pT^2}{n\varepsilon^2}\right)$. Our algorithm requires both smoothness and strong convexity to achieve this bound, and our analysis requires a new analysis of batch stochastic gradient descent under these conditions. The role of smoothness is surprising: without restrictions on interaction, assuming strong convexity helps with accuracy, but adding smoothness does not. However, smoothness is known to accelerate the convergence of gradient descent, and that translates into a much better dependency on T in our context.[3]

Assumptions	Method	Additive Error (big-Oh expression)
1-Lipschitz	GD	$\frac{1}{\sqrt{T}} + \sqrt{\frac{p}{n\epsilon^2}}$
1-Lipschitz	CP	$\sqrt{\frac{T}{n\varepsilon^2}} + 2\left(1-\frac{1}{e}\right)^{T/p}$
Δ-strongly convex	GD	$\frac{1}{\Delta T} + \frac{p}{n\epsilon^2\Delta}$
β-smooth	GD	$\frac{1}{T^2} + \frac{pT^2}{n\epsilon^2\beta}$
β-smooth, Δ-strongly convex	GD	$\frac{\beta}{2}e^{-\Delta T/\beta} + \frac{pT^2}{n\varepsilon^2}$

Figure 2. Upper bounds on achievable error for optimization of 1-Lipschitz convex functions in the local model as a function of the number T of rounds of interaction, the number n of users, and the dimension p of the parameter vector. Methods: GD = gradient descent, CP = cutting plane.

Note. We state all our results in expectation, but they also extend to high probability guarantee using the idea of Bassily *et al.* [7, Appendix D].

Case Study: Linear Regression Our work also raises the question of what can be achieved for specific problems. In Appendix C-A, we study the accuracy of noninteractive LDP protocols for ordinary least squares regression, where each input d is a pair $(\mathbf{x}, y)$ with

[3]One can show that if the loss function is 1-Lipschitz and β-smooth, then the ε-LDP algorithm version of batch stochastic gradient descent can be tuned (using Lan's analysis [29]), so that for every $\mathbf{D} \in \mathcal{D}^n$, $\mathbb{E}[err_{\mathbf{D}}(\mathcal{A}(\mathbf{D}))] = O\left(\frac{1}{T^2} + \frac{pT^2}{n\epsilon^2\beta}\right)$. This achieves optimal error $O(\sqrt{p/\varepsilon^2 n})$ for $T = (n\varepsilon^2/p)^{1/4}$, which is quadratically faster than the case when the function is 1-Lipschitz. Similarly, one can also show (using [40]) that if the loss function is 1-Lipschitz and Δ-strongly convex, then ε-LDP batch SGD satisfies, for every $\mathbf{D} \in \mathcal{D}^n$, $\mathbb{E}[err_{\mathbf{D}}(\mathcal{A}(\mathbf{D}))] = O\left(\frac{1}{\Delta T} + \frac{p}{n\epsilon^2\Delta}\right)$. For constant Δ, this achieves very low error $O(p/\varepsilon^2 n)$ when $T = (n\varepsilon^2/p)$—the same number of rounds, but much lower error than, what one gets from Lipschitz continuity alone.

$\mathbf{x} \in B_p(\mathbf{0}, 1)$ (the p-dimensional ball of radius 1) and $y \in \mathbb{R}$, and $\ell(\theta; (\mathbf{x}, y)) = (y - \langle \mathbf{x}, \theta \rangle)^2$ for $\theta \in B_p(\mathbf{0}, 1)$. We also show that the lower bound of $\sqrt{p/\varepsilon^2 n}$ applies to this special case (regardless of interaction) and that a natural noninteractive algorithm in Appendix C-B, which computes noisy versions of the Hessian and gradient at 0, achieves this error rate. There are nonprivate algorithms with population risk only $1/\sqrt{n}$ (e.g., [39])—thus, our bound shows that privacy imposes a dimension-dependent cost.

The Relation to SQ Learning Kasiviswanathan *et al.* [26] showed a general equivalence between local differential privacy and *statistical query* learning [27]: for distributional problems, the two models are equivalent, in terms of both sample and time complexity, up to polynomial factors in the dimension and the desired accuracy. The main open problem we aim to solve is whether there exist noninteractive LDP algorithms for convex optimization with sample complexity polynomial in p and α. Equivalently, one can ask if there exist SQ algorithms with similar scaling. Our lower bound sheds light on this, showing that such algorithms would have to use the SQ oracle in some other way than to simulate first- or second-order methods (which is the current state of the art, e.g., [21]).

C. Related Work

Local differential privacy was defined by Dwork *et al.* [17] in the paper that introduced differential privacy. Local privacy was implicitly studied in previous works under different names, like γ-amplification [4, 20] and randomized response [46].

Two of the most widely studied problems in local differential privacy are the problem of finding heavy hitters [6, 19, 23, 32] and private local learning [26]. The heavy hitters problem represents a very simple computation (counting elements frequencies), and existing algorithms are all noninteractive. The closely related "heavy hitters over sets" problems is more complex; the state-of-the-art algorithm uses two rounds of interaction [36], though it's unclear whether the extra round is necessary. Algorithms developed for the heavy hitters problem are not directly relevant here, since it is unclear how they can be applied to optimization.

Kasiviswanathan *et al.* [26] initiated the study of private local learning. They showed that every LDP learning algorithm can be simulated in the statistical query model when the data is sampled i.i.d. from a known distribution (see "SQ Learning" above).

In terms of lower bounds, Beimel *et al.* [8] studied the lower bound on the squared error of distributed protocols for the sample mean estimation of the data. This was generalized by Duchi *et al.* [14], who gave a general framework for translating lower bounds on statistical estimation to the local privacy model. While the lower bounds in Duchi *et al.* [14, 6] are optimal, they did not consider the round complexity of the algorithms.

In the optimization literature (independent of work on privacy), several works establish lower bounds on the *oracle complexity*—the number of queries made by a neighborhood-based algorithm. (In contrast, we consider the number of rounds of adaptivity in these queries.) This line of work was started by Nemirovski et al. [33]. Their analysis was simplified by Nesterov [34] under additional restrictions on the aglorithm. Recently, Agarwal et al. [3] and Raginsky and Rakhlin [37] extended the study of lower bounds to the stochastic gradient-based methods, and provided a more structured, information-theoretic approach to these lower bounds. None of these works distinguish between adaptive and nonadaptive methods.

Differentially private convex optimization has also been studied extensively in the central model, starting with the work of Chaudhuri et al. [13] and later in [47, 24, 28, 41, 44, 42, 7, 43]. Noisy variants of first-order methods play an important role in several of those works, starting with Williams and McSherry [47].

II. Noninteractive Private Learning

This section is devoted to our noninteractive LDP algorithms for private learning. In order to better understand our main result (Theorem 10), we start by analyzing a simple mechanism called the *net* mechanism, for which it is easy to establish the following claim.

Claim 4. *For every* 1*-Lipschitz loss function* $\ell : \mathcal{C} \times \mathcal{D}$ *and every distribution* $\mathcal{P}$ *on* $\mathcal{D}$*, when* $\mathbf{D} \sim \mathcal{P}^n$*, the net mechanism* $\mathcal{A}_{net}$ *is* ε*-LDP and satisfies*

$$\mathbb{E}_{\substack{\mathbf{D} \sim \mathcal{P}^n \\ \text{coins of } \mathcal{A}_{net}}} (err_{\mathcal{P}}(\mathcal{A}_{net}(\mathbf{D}))) = O\left(\left(\frac{p}{\varepsilon^2 n}\right)^{1/(p+2)}\right) \tag{1}$$

with high probability. Moreover, this is tight in the worst case. In particular, to achieve a given error α*, the algorithm requires sample complexity* $n = \Omega\left(\frac{C^p}{\varepsilon^2 \alpha^{p+2}}\right)$ *for a constant* $C > 0$ *(roughly 4).*

The net mechanism, $\mathcal{A}_{net}$, to optimizes a loss function L on $\mathcal{C}$ works in three steps: (i) construct an appropriately fine net for $\mathcal{C}$; (ii) approximately evaluate L at all points in the net; and (iii) report the minimizer of the observed values. We defer the proof of Claim 4 to Appendix B.

Having established what one can achieve by the basic net mechanism, we now give our noninteractive

algorithm. For the ease of presentation, we describe our main algorithm through a series of increasing generality. We start by describing and analyzing an algorithm for one-dimensional median problem, 1D-MEDIAN (Figure 3). We then use 1D-MEDIAN to describe an algorithm for one-dimensional convex Lipschitz function, 1D-GENERAL (Figure 4). Finally, we present our main algorithm, HIGHD-MEDIAN, in Figure 5 that uses 1D-GENERAL as a black-box.

For the proof of our main result of this section (Theorem 10), we require our algorithm for 1D-MEDIAN and 1D-GENERAL to satisfy a stronger property known as *uniform approximation*. We first formally define uniform approximation.

Definition 5. (Uniform approximation). *Given real-valued functions f and g on the same domain $\mathcal{C}$, we say f* uniformly approximates *g with error α (denoted $\|f-g\|_\infty \leq \alpha$) if $|f(\theta)-g(\theta)| \leq \alpha$ for all $\theta \in \mathcal{C}$.*

Following our plan mentioned at the start of this section, we first show the following result (Theorem 6) for 1D-MEDIAN. Using Theorem 6, we give an uniform convergence for one-dimensional convex functions (Theorem 9). Finally, we use Theorem 9 to give our main result of this section (Theorem 10) for general convex function in high dimension.

Theorem 6. *For every distribution $\mathcal{P}$ on $[0,1]$, with probability $1-\gamma$ over $\mathbf{D} \sim \mathcal{P}^n$ and $\hat{f} \leftarrow$ 1D-MEDIAN$(\mathbf{D})$, where* 1D-MEDIAN *is the algorithm presented in Figure 3, we have*

$$\left\| \hat{f} - \overline{\mathsf{med}}_{\mathcal{P}} \right\|_\infty \leq O\left(\frac{\log^2(\varepsilon^2 n)\sqrt{\log(\varepsilon^2 n/\gamma)}}{\varepsilon\sqrt{n}} \right), \tag{2}$$

and $\overline{\mathsf{med}}_{\mathcal{P}}(\theta)$ is the expected loss $\mathbb{E}_{d\sim\mathcal{P}}|\theta - d|$ for the median problem. In particular, to achieve error α with probability $1-\gamma$, the algorithm requires sample size $n = \tilde{O}(\log(1/\gamma)/\varepsilon^2\alpha^2)$. Moreover, $\hat{f}$ is ε-LDP.

When $\|\hat{f} - L_{\mathcal{P}}\|_\infty \leq \alpha$, the minimizer θ_{priv} of $\hat{f}$ satisfies $err_{\mathcal{P}}(\theta_{\mathsf{priv}}) \leq \alpha$, so this algorithm suffices for approximate computation of the median.

Our algorithm for one-dimensional median (1D-MEDIAN) presented in Figure 3, is based on the tree-based algorithm for simultaneously approximating the number of points in every subinterval of a range (as in, e.g., [12, 18, 22]).

Proof: The privacy proof follows from the fact that $\mathcal{R}$ is an ε-LDP randomizer [6] and that differential privacy is preserved under arbitrary post-processing [15].

Input. Let the input dataset be $\mathbf{D} = (d_1, \cdots, d_n) \in \mathcal{D}^n$, where every d_i is sampled i.i.d. from the distribution $\mathcal{P}$. Let $\ell : [0,1] \times \mathcal{D} \to \mathbb{R}$ be the loss function defined as $\ell_i(\mathbf{D}) := |\theta - d_i|$. Let $\zeta = 1/\varepsilon\sqrt{n}$, $w = 1/\zeta$, $h = \log w$. Let $\mathcal{B} := \{\theta_1, \cdots, \theta_w\}$ be the set of points in the ζ-net, and $\varepsilon' = \varepsilon/h$.

User side computation. Every user i constructs binary trees $\mathcal{T}_i$ over w leaves as follows:

1) Label the leaves of the binary tree over w leaves as integers from $[1,w]$, and the intermediate nodes n as pairs (a,b) such that $a \in [1,w]$ is the left most leaf and $b \in [1,w]$ is the right most leaf of the subtree rooted at n.
2) For every leaf node $j \in [1,w]$ of $\mathcal{T}_i$, its entry, $\mathsf{u}_j := 1$ for $j := \min\{s : \theta_s > d_j\}$, else $\mathsf{u}_j := 0$. The entry on intermediate node (a,b) is $\sum_{a \leq j \leq b} \mathsf{u}_j$.
3) Let $\mathbf{v}_i$ be the corresponding vector that stores the entries of the nodes of $\mathcal{T}_i$. Send the server a binary tree $\widehat{\mathcal{T}}_i$ with corresponding vectors $\widehat{\mathbf{v}}_i(j) = \mathcal{R}(\mathbf{v}_i(j))$, where $\mathcal{R}(\cdot)$ is the ε'-basic randomizer of Bassily and Smith [6] for all $j \in [1, 2w]$.

Server side computation. On receiving $\widehat{\mathcal{T}}_i$ from all user $i \in [n]$, it performs the following steps:

1) Constructs a binary tree $\mathcal{T}$ with nodes labeled appropriated by vectors $\mathbf{v}$, where $\mathbf{v}(j) := \sum_{i=1}^n \widehat{\mathbf{v}}_i(j)$ for all $j \in [1, 2w]$.
2) For every $j \in [w]$, construct a maximal dyadic partition of the interval $[1, \cdots, j]$ and that of $[j+1, \cdots, w]$. Call the nodes corresponding to the intervals in the binary tree as the set N_j^{left} and N_j^{right}, respectively. Compute $\widehat{\mathbf{x}}(j) = \frac{1}{n}\left| \sum_{\mathsf{n} \in N_j^{\mathsf{left}}} \mathbf{v}(k) - \sum_{\mathsf{n} \in N_j^{\mathsf{right}}} \mathbf{v}(k) \right|$.
3) Define a function $\widehat{g} : [0,1] \to [0,1]$ such that $\widehat{g}(y) = \widehat{\mathbf{x}}(\theta)$ where $\theta = \mathrm{argmin}_{z \in \mathcal{B}} |z - y|$. Define $\widehat{f} : [0,1] \to [0,1]$ as follows: $\widehat{f}(x) = \int_0^x \widehat{g}(t)dt$

Output. The server outputs $\widehat{f}$.

Figure 3. 1-Dimensional Private Non-interactive Median (1D-MEDIAN)

Now we turn our attention to prove equation (2). Before we analyze the algorithm, let us fix some notations. Let $\widetilde{\mathbf{x}}$ be the vector that the server would have computed if there were no $\mathcal{R}(\cdot)$ in Step (3) of the user side computation in Figure 3.

We define a function $\widetilde{g} : [0,1] \to [0,1]$ using the

vector $\widetilde{\mathbf{x}}$ similar to the definition of the function $\widehat{g}(\cdot)$, i.e., $\widetilde{g}(y) = \widetilde{\mathbf{x}}(\theta)$ where $\theta = \operatorname{argmin}_{z \in \mathcal{B}} |z - y|$.

In what follows, we first prove that for all $\theta \in \mathcal{B}$,

$$\|\widehat{g}(\theta) - \widetilde{g}(\theta)\|_\infty \leq O\left(\frac{\log^2(\varepsilon^2 n)\sqrt{\log(\varepsilon^2 n/\gamma)}}{\varepsilon\sqrt{n}}\right) \tag{3}$$

Our analysis uses the observation that, every level of the binary tree $\mathcal{T}$ (i.e., the corresponding entries of the vector $\mathbf{v}$) can be seen as a noisy histogram (over the data universe $\mathcal{B}$) of the user's data and, by the definition of the vector $\widehat{\mathbf{x}}$, the vector $\widehat{\mathbf{x}}$ succinctly stores the estimates of $\nabla\overline{\mathsf{med}}_{\mathcal{P}}$ at different net points.

Let $\gamma' = \gamma/h$. Since the data universe has size $w = \varepsilon\sqrt{n}$, we can use Theorem 2.3 of Bassily and Smith [6] to estimate the ℓ_∞-error in estimating $\nabla\overline{\mathsf{med}}_{\mathcal{P}}$ at every level $k \in [h]$. Bassily and Smith [6, Theorem 2.3] gives that, with probability $1 - \gamma'$, the ℓ_∞ error in estimating $\nabla\overline{\mathsf{med}}_{\mathcal{P}}$ is at most $O\left(\frac{1}{\varepsilon}\sqrt{\frac{\log(w/\gamma')}{n}}\right)$ for every level $k \in [h]$. Using union bound over all the levels, we have

$$\max_{\theta \in \mathcal{B}} |\widehat{g}(\theta) - \widetilde{g}(\theta)| = O\left(\frac{h}{\varepsilon'}\sqrt{\frac{\log(w/\gamma')}{n}}\right)$$

By substituting $\varepsilon' = \varepsilon/h$, $h = \log w$, and setting $w = \varepsilon\sqrt{n}$, we get equation (3).

We now return to proving Theorem 6. From the definition of $\widehat{f}$ and equation (3), we have the following set of inequalities for all $x \in [0, 1]$:

$$\begin{aligned}
|\widehat{f}(x) - \overline{\mathsf{med}}_{\mathcal{P}}(x)| &= \int_0^x \left|\widehat{g}(\theta) - \nabla\overline{\mathsf{med}}_{\mathcal{P}}(\theta)\right| \mathrm{d}\theta \\
&\leq \sum_{t=1}^{\lceil x/\zeta \rceil - 1} \int_{t\zeta}^{(t+1)\zeta} \left|(\widehat{g}(\theta) - \nabla\overline{\mathsf{med}}_{\mathcal{P}}(\theta))\right| \mathrm{d}\theta \\
&\leq \left(\sum_{t=1}^{\lceil x/\zeta \rceil} \alpha\zeta\right) + \zeta(\nabla\overline{\mathsf{med}}(1) - \nabla\overline{\mathsf{med}}(0)) \\
&\leq \alpha + 2\zeta = O\left(\log^2(\varepsilon^2 n)\sqrt{\log(\varepsilon^2 n/\gamma)}/\left(\varepsilon\sqrt{n}\right)\right),
\end{aligned}$$

where α denotes the left-hand side in Equation (3). The last inequality follows from the fact that the summation is over at most $w = 1/\zeta$ net points and that $\overline{\mathsf{med}}$ is 1-Lipschitz. This completes the proof of Theorem 6. ■

We now proceed to the general one-dimensional convex function. We first state the following key lemma.

Lemma 7. *Let $f : [0, 1] \to [0, 1]$ be a convex 1-Lipschitz function. Then there exists a distribution $\mathcal{Q}$ such that*

$$\forall \theta \in [0, 1],\ f(\theta) = \mathbb{E}_{y \sim \mathcal{Q}}[|\theta - y|] + c$$

Input. A dataset $\mathbf{D} = (d_1, \cdots, d_n) \in \mathcal{D}^n$ with every entry d_i sampled i.i.d. from the distribution $\mathcal{P}$ and a loss function $\ell : [0, 1] \times \mathcal{D} \to \mathbb{R}$.

User side computation. Every user i (for $\ell_i(\theta) := \ell(\theta; d_i)$) performs the following steps:

1) Samples a median function $\overline{\mathsf{med}}_i$ as defined in Corollary 8.
2) Run its end of 1D-MEDIAN algorithm (Figure 3) with $\overline{\mathsf{med}}_i$ and d_i to send $\widehat{\mathcal{T}}_i$ to the server.

Server side computation. On receiving $\widehat{\mathcal{T}}_i$ from all user $i \in [n]$, the server invokes its end of the 1D-MEDIAN algorithm (Figure 3) and gets $\widehat{f}$ as its output.

Output. The server outputs $\widehat{f}$.

Figure 4. 1-Dimensional Private Convex Optimization (1D-GENERAL)

Proof: We first define the randomized algorithm Z whose output is the distribution $\mathcal{Q}$. The algorithm Z does the following.

1) Uniformly sample a $u \in [-1, 1]$.
2) Output $y \in [0, 1]$ such that $u = \nabla f(y)$.

The above distribution is well defined because $f(\cdot)$ is 1-Lipschitz. Now, let $\mathcal{Q}$ be the output distribution of the algorithm Z. Let $g(\theta) = \mathbb{E}_{y \sim \mathcal{Q}}[|y - \theta|]$. Then the following is easy to see

$$\begin{aligned}
\nabla g(\theta) &= \mathbb{E}_{y \sim \mathcal{Q}}[\mathbf{1}_{y \leq \theta}] - \mathbb{E}_{y \sim \mathcal{Q}}[\mathbf{1}_{y > \theta}] \\
&= \frac{(\nabla f(\theta) - (-1)) - (1 - \nabla f(\theta))}{2} = \nabla f(\theta),
\end{aligned}$$

where $\mathbf{1}_{[\cdot]}$ is the indicator random variable. Since the gradient of the function $f(\cdot)$ matches the gradient of the median function at every θ, the two function can differ by at most a constant. This completes the proof of Lemma 7. ■

Since the optimal solution does not change under a translation by a scalar, we have the following as a direct corollary of Lemma 7.

Corollary 8. *For every 1-Lipschitz loss function $\ell : [0, 1] \times \mathcal{D} \to \mathbb{R}$, there is a randomized algorithm $Z : \mathcal{D} \to [0, 1]$ (given by Lemma 7), such that for every distribution $\mathcal{P}$ on $\mathcal{D}$, the distribution $\mathcal{Q}$ on $[0, 1]$ obtained by running Z on a single draw from $\mathcal{P}$ satisfies*

$$L_{\mathcal{P}}(\theta) = \overline{\mathsf{med}}_{\mathcal{Q}}(\theta) \quad \text{for all } \theta \in [0, 1], \tag{4}$$

where $L_{\mathcal{P}}(\theta)$ is the population risk $\mathbb{E}_{d \sim \mathcal{P}}[\ell(\theta, d)]$.

Theorem 6 and Corollary 8 now directly gives us the following result:

Theorem 9. *Let $\ell : [0,1] \times \mathcal{D} \to \mathbb{R}$ be a 1-Lipschitz loss function such that $\ell(\frac{1}{2}, d) = 0$ for all $d \in \mathcal{D}$. For every distribution $\mathcal{P}$ on $\mathcal{P}$, with probability $1-\gamma$ over $\mathbf{D} \sim \mathcal{P}^n$ and $\hat{f} \leftarrow$ 1D-GENERAL$(\mathbf{D}, \ell)$, where* 1D-GENERAL$(\cdot,\cdot)$ *is the algorithm presented in Figure 4, we have*

$$\left\| \hat{f} - L_{\mathcal{P}} \right\|_\infty \leq O\left(\frac{\log(\varepsilon^2 n)\sqrt{\log(\varepsilon^2 n/\gamma)}}{\varepsilon\sqrt{n}} \right),$$

and $L_{\mathcal{P}}(\theta)$ is the population risk $\mathbb{E}_{D\sim\mathcal{P}}(\ell(\theta, D))$. In particular, to achieve error α with probability $1-\gamma$, the algorithm requires sample size $n = \tilde{O}(\log(1/\gamma)/\varepsilon^2\alpha^2)$.

The above theorem basically shows that we can uniformly approximate any 1-Lipschitz convex function defined over $\mathbb{R}$. This observation is crucial for our algorithm in the high dimensional case.

Since an optimization problem is invariant under an affine transformation, without any loss of generality, we can assume that $\ell(\mathbf{0}^p, d) = 0$ in the high dimensional case. Our main result of this section is as follows.

Theorem 10. *Let $\mathcal{C} \subseteq \mathbb{R}^p$ be contained in the unit ball, and let $\ell : \mathcal{C} \times \mathcal{D} \to \mathbb{R}$ be a 1-Lipschitz loss function such that $\ell(\mathbf{0}^p, d) = 0$ for all $d \in \mathcal{D}$. For every distribution $\mathcal{P}$ on $\mathcal{D}$, with probability $1-\gamma$ over $\mathbf{D} \sim \mathcal{P}^n$ and $\theta_{\mathsf{priv}} \leftarrow$ HIGHD-GENERAL$(\mathbf{D}, \ell)$ for* HIGHD-GENERAL$(\cdot,\cdot)$ *presented in Figure 3, we have*

$$err_{\mathcal{P}}(\theta_{\mathsf{priv}}) \leq O\left(\left(\frac{\sqrt{p}\log^3(\varepsilon^2 n)\log^2(1/\gamma)}{\varepsilon^2 n} \right)^{\frac{1}{(p+1)}} \right).$$

In particular, to achieve error α with probability $1-\gamma$, the algorithm requires sample size $n = \widetilde{\Omega}(\frac{c^p}{\varepsilon^2\alpha^{p+1}})$.

Proof: Let us define the function $\widehat{f}$ as follows. On input θ, we perform two steps to compute the value of $\widehat{f}(\theta)$. We first find a point θ' and the direction $\mathbf{u}_j$ such that θ' is parallel to $\mathbf{u}_j$ and θ' is closest to θ in Euclidean distance. We then compute the function $\widehat{f^j}(\theta')$.

Let $\gamma' = \gamma/k$. Fix a $j \in [k]$. From Theorem 6, we have with probability $1-\gamma'$ for all θ on a ζ-net defined on the line parallel to the line $\mathbf{u}_j$,

$$|\widehat{f^j}(\theta) - L_{\mathcal{P}}(\theta)| = O\left(\frac{\log(\varepsilon^2 n/k)}{\varepsilon} \sqrt{\frac{k\log(\varepsilon^2 n/k\gamma')}{n}} \right).$$

This implies that, with probability $1-\gamma$ over the random coins of the algorithm and the local randomizer of 1D-GENERAL,

$$\|\widehat{f} - L_{\mathcal{P}}\|_\infty = O\left(\frac{\log(\varepsilon^2 n/k)}{\varepsilon} \sqrt{\frac{k\log(\varepsilon^2 n/\gamma)}{n}} \right).$$

Input. A dataset $\mathbf{D} = (d_1, \cdots, d_n) \in \mathcal{D}^n$ such that d_i is chosen i.i.d. from the distribution $\mathcal{P}$, and a loss function $\ell : \mathcal{C} \times \mathcal{D} \to \mathbb{R}$, where $\mathcal{C}$ is a convex set and ℓ is 1-Lipschitz function.

Preprocessing step. The algorithm chooses k random directions, $\mathbf{u}_1, \cdots, \mathbf{u}_k$. Then it defines k sets as follows: For $j \in [k]$, define the set $\mathcal{S}_j = \{(j-1)\lfloor \frac{n}{k} \rfloor, \cdots, j\lfloor \frac{n}{k} \rfloor - 1\}$. **If** $j = k$, **then** $\mathcal{S}_j = \mathcal{S}_j \cup \{j\lfloor \frac{n}{k} \rfloor, \cdots, n\}$.

User side computation. Every user i s.t. $\{i \in \mathcal{S}_j\}$ invokes 1D-GENERAL on its side with input $(d_i, \mathbf{u}_i)$ to send $\widehat{\mathcal{T}}_i$ to the server.

Server side computation.: On receiving $\widehat{\mathcal{T}}_i$ from all user $i \in [n]$, the server performs the following steps:

1) For $1 \leq j \leq k$, invokes its end of the 1D-GENERAL algorithm (Figure 4) with $\left\{\widehat{\mathcal{T}}_i\right\}_{i\in S_j}$ to get $\widehat{f^j}$.
2) Compute $\theta_j := \operatorname{argmin}_{\theta \| \mathbf{u}_j} \widehat{f^j}$ and then compute $\theta_{\mathsf{priv}} := \operatorname{argmin}_j \widehat{f^j}(\theta_j)$.

Output. The server outputs θ_{priv}.

Figure 5. Private Non-interactive Optimization (HIGHD-GENERAL)

As before, when $\|\widehat{f} - L_{\mathcal{P}}\|_\infty \leq \alpha$, the minimizer θ_{priv} of $\widehat{f}$ satisfies $err_{\mathcal{P}}(\theta_{\mathsf{priv}}) \leq \alpha$, so bounding the value of α suffices to approximately compute $L_{\mathcal{P}}$.

In order to complete the proof we need to find the value of k. Let us denote by θ^* the true minima of $L_{\mathcal{P}}(\cdot)$. From here on, we drop the subscript $\mathcal{P}$ when clear from the context. Let $\mathcal{B}^p(\theta^*, \alpha)$ be the p-dimensional ball of radius α (to be chosen later) centered at θ^*. Now in order to complete the proof, we need the size of k such that there is at least one line $\mathbf{u}^*$ that intersects $\mathcal{B}^p(\theta^*, \alpha)$. From the Lipschitz property, $L(\theta) - L(\theta^*) \leq \|\theta - \theta^*\|_2 \leq \alpha$ if $\|\theta - \theta^*\|_2 \leq \alpha$. Let $\mathcal{Q}_i$ be the spherical cap[4] formed by a line $\mathbf{u}_i$ picked by the algorithm. Let $\mathbf{S}(\mathcal{X})$ denote the surface area of the convex body $\mathcal{X}$. Let Good be the event that the line $\mathbf{u}_i$ intersects the ball $\mathcal{B}_p(\theta^*, \alpha)$. Using [11, Lemma 2.35], we have $\Pr[\mathsf{Good}] = \mathbf{S}(\mathcal{Q}_i)/\mathbf{S}(\mathcal{C})$.We first prove the following claim.

Claim 11. *Let $\mathsf{Good}, \mathcal{C}, \mathcal{Q}_i$ be as defined above. Let α be the contact angle[5] of the spherical cap. If α is small, then*

$$\Pr[\mathsf{Good}] \geq \sqrt{\frac{2}{\pi p}} \sin^{p-1}(\alpha) \geq \sqrt{\frac{2}{\pi p}} \left(\frac{\alpha}{\sqrt{2}} \right)^{p-1}.$$

[4]A *spherical cap* is the region of a sphere which lies above (or below) a given plane [5]. In this section, the spherical cap $\mathcal{Q}_i$ is defined by the plane whose normal is the line $\mathbf{u}_i$ and is the region of the sphere which does not include the center of the sphere.

[5]The angle between the normal to the sphere at the bottom of the cap and the base plane is called the *contact angle*.

Proof: Using the sine rule, we have $\alpha \cos(\alpha/2) = \sin\alpha$ Using the fact that $\sin^2\alpha + \cos^2\alpha = 1$ and $\cos\alpha = 2\cos^2(\alpha/2) - 1$, rearranging the expression gives us

$$\frac{\alpha}{\sqrt{2}} = \sqrt{\frac{1-\cos^2\alpha}{1+\cos\alpha}} = \sqrt{1-\cos\alpha}. \tag{5}$$

In particular, this implies that $\cos\alpha = 1 - \frac{\alpha^2}{2}$. Therefore,

$$\begin{aligned}\mathsf{Pr}[\mathsf{Good}] &\geq \sqrt{\frac{2}{\pi p}}\sin^{p-1}(\alpha) \\ &= \sqrt{\frac{2}{\pi p}}\left(1-\left(1-\frac{\alpha^2}{2}\right)^2\right)^{1/(p-1)} \\ &\geq \sqrt{\frac{2}{\pi p}}\frac{\alpha^{p-1}}{2^{(p-1)/2}}.\end{aligned}$$

This completes the proof of Claim 11. ∎

If we set $k = O\left(\frac{2^{(p-1)/2}\log(1/\gamma)}{\alpha^{p-1}}\sqrt{\frac{\pi p}{2}}\right)$, then Claim 11 implies that there exists an $j \in [k]$ such that, with probability $1-\gamma$, $\mathbf{u}_j$ intersects $\mathcal{B}_p(\theta^*, \alpha)$. Substituting this value of k in equation (5), we get

$$\|\widehat{f} - L_{\mathcal{P}}\|_\infty = O\left(\frac{\log(1/\gamma)}{\varepsilon}\sqrt{\frac{\log^3(\varepsilon^2 n)\sqrt{p2^{p-2}}}{n\alpha^{p-1}}}\right).$$

Setting $\alpha = O\left(\left(\frac{\sqrt{p}}{\varepsilon^2 n}\log^3(\varepsilon^2 n)\log^2(1/\gamma)\right)^{1/(p+1)}\right)$, we get $\|\widehat{f} - L_{\mathcal{P}}\|_\infty \leq \alpha$. This completes the proof of Theorem 10. ∎

In particular, to achieve a given error α, the theorem states that the algorithm HIGHD-GENERAL requires sample complexity $n = \widetilde{\Omega}(\frac{c^p}{\varepsilon^2\alpha^{p+1}})$ for a constant $c > 0$ (roughly 2).

III. ROUND COMPLEXITY OF ADAPTIVE ALGORITHMS USING NEIGHBORHOOD ORACLES

In this section, we show that, for a natural class of algorithms, the exponential dependence on the dimension is essential. We first define the basic model and then state our result. We assume that an algorithm has an access to a special kind of oracle which we call a *neighborhood-based oracle* and can make queries in batches, where queries in a particular batch may depend on the queries and response from previous batches.

Neighborhood-based oracles. We assume that the algorithm has access to an oracle $\mathcal{O}_F(\cdot)$ for a convex function $F : \mathcal{C} \to [0,1]$, where $\mathcal{C} \subseteq \mathbb{R}^p$ is a convex set over which the function is defined. We say the oracle is *neighborhood-based* (also called *local*, see the footnote on page 3) if, given $\theta \in \mathcal{C}$, the oracle outputs a value that depends on the function $F(\cdot)$ and the infinitesimal neighborhood of the query point θ. For example, one such query can be to compute the h-th order gradient, $\nabla^h F(\theta)$, at the point θ.

Query model. We assume that the algorithm is randomized and can make at most q queries in T batches, where batch-i consists of q_i queries such that $\sum_{i=1}^T q_i = q$. We call any such algorithms that makes at most q queries in T-batches a (q,T)-*adaptive algorithm* if, for all $1 \leq i \leq T-1$, the queries in the $(i+1)$-th batch depends only on the queries made and the responses received during the first i batches. When $T = 1$, we call such algorithm a q-*nonadaptive algorithm*. Note that our query model does not assume the queries made in a particular batch are independent, i.e., queries in a single batch may have non-trivial correlation.

Our first claim shows that there exists a distribution over convex function defined on a convex set $\mathcal{C} \subseteq \mathbb{R}^p$, such that for a function F chosen from this distribution, any q-nonadaptive algorithm can outputs a θ^* such that $F(\theta^*) - \min_{\theta\in\mathcal{C}} F(\theta) \leq \alpha/21$ with probability at most $q\alpha^p$. That is, to get a constant success probability, any non-adaptive algorithm has to make number of queries exponential in the dimension of the underlying convex set. More precisely, we show the following.

Theorem 12. *Let $\mathcal{B}^p(\mathbf{0}^p, 1)$ denote a unit ball centered around the origin. There exists a distribution $\mathcal{F}$ of convex functions from $\mathcal{B}^p(\mathbf{0}^p,1)$ to $[0,1]$ such that, for $F \sim \mathcal{F}$, the following holds:*

1) *Let $\mathcal{O}_F(\cdot)$ be a neighborhood based oracle for F as defined above. Then the output θ^* of any q-nonadaptive and randomized algorithm, with oracle access to $\mathcal{O}_F(\cdot)$, satisfies*

$$\mathsf{Pr}\left[\left|F(\theta^*) - \min_{\theta\in\mathcal{B}^p(\mathbf{0}^p,1)} F(\theta)\right| \leq \alpha/21\right] \leq q\alpha^p.$$

2) *There exists a $(\mathrm{poly}(\log(1/\alpha)), 2)$-adaptive algorithm that can compute θ^* with probability at least $2/3$, such that $\left|F(\theta^*) - \min_{\theta\in\mathcal{B}^p(\mathbf{0}^p,1)} F(\theta)\right| \leq \alpha$.*

Remark 1. *We first note the implication of this result in the model of local-differential privacy. Theorem 12 rules out any nonadaptive algorithms, including non-private algorithms, that just uses gradient information. This in particular implies that there exists a convex function for which no local-differentially private algorithm, with access to neighborhood oracle, can output a θ_{priv} such that $err_{\mathbf{D}}(\theta_{\mathsf{priv}}) \leq \alpha$ with high probability. In other words, it shows that adaptivity is necessary for gradient*

based methods to achieve polynomial dependence on the dimension.

The idea behind of our proof is as follows. Recall that we need to construct a distribution of functions $\mathcal{F}$ such that a function sampled from $\mathcal{F}$ is hard to optimized using only non-adaptive queries. We define a distribution $\mathcal{F}$ that satisfies the following properties: (i) the minimum of any function sampled from $\mathcal{F}$ lies in a uniformly at random chosen p-dimensional ball $\mathcal{B}$ of radius α, (ii) any algorithm that makes neighborhood queries does not learn anything about the optimum point unless it queries a point inside the ball $\mathcal{B}$, and (iii) if an algorithm queries a point inside the ball, it learns the optimum value. Once we have such a distribution of function, we are basically done because the probability with which any query point is a point inside a small p-dimensional ball depends on the volume of $\mathcal{B}$ and the volume of $\mathcal{B}$ decays exponentially with p.

We now return to proving Theorem 12.

Proof of Theorem 12: Let $\mathsf{C} := \{\mathbf{c}_1, \cdots, \mathbf{c}_N\}$ be the centers of N balls that forms an α-packing of $\mathcal{B}^p(\mathbf{0}^p, 1)$. That is,

$$\forall i \in [N] \text{ and } \theta \text{ such that } \|\theta - \mathbf{c}_i\|_2 \leq \alpha,\ \theta \in \mathcal{B}^p(\mathbf{0}^p, 1)$$
$$\forall i \neq j \in [N],\ \|\mathbf{c}_i - \mathbf{c}_j\|_2 \geq 2\alpha.$$

Pick a point $c \in \mathsf{C}$ uniformly at random. Pick two points $\hat{\mathbf{c}} \in \mathcal{B}_p(\mathbf{0}^p, 1)$ and $\tilde{\mathbf{c}} \in \mathcal{B}_p(\mathbf{0}^p, 1)$ such that $\alpha/6 \leq \|\mathbf{c} - \hat{\mathbf{c}}\|_2 \leq \alpha/3$, $\alpha/6 \leq \|\mathbf{c} - \tilde{\mathbf{c}}\|_2 \leq \alpha/3$, and $\|\tilde{\mathbf{c}} - \hat{\mathbf{c}}\|_2 \geq 2\alpha/3$. Define the following functions:

$$F_0(\theta) := \max\left\{\|\theta - \mathbf{c}\|_2, \frac{1}{3}\|\theta - \hat{\mathbf{c}}\|_2 + \frac{1}{3}\right\}$$
$$F_1(\theta) := \max\left\{\|\theta - \mathbf{c}\|_2, \frac{1}{3}\|\theta - \tilde{\mathbf{c}}\|_2 + \frac{1}{3}\right\}$$

The function F is chosen by flipping a uniform coin b and setting $F(\theta) := F_b(\theta)$. Since all three functions f, f_0, and f_1 are convex function and point-wise maximum of two convex functions is a convex function, both F_0 and F_1 are convex functions. Similarly, it is easy to verify that the function $F(\theta)$ is 1-Lipschitz. Moreover, $\nabla F(\theta) = 1$ for all $\theta \in \mathcal{B}^p(\mathbf{0}^p, 1) \backslash \mathcal{B}^p(\mathbf{c}, \alpha)$ and $\nabla F(\theta) = 1/3$ for all $\theta \in \mathcal{B}^p(\mathbf{c}, \alpha)$.

We first enumerate some key properties of $F_0(\cdot)$. The same properties holds for $F_1(\cdot)$ as well. Let $\theta^* := \operatorname{argmin}_{\theta \in \mathcal{B}^p(\mathbf{c}, \alpha)} F(\theta)$. First note that $F(\hat{\mathbf{c}}) = F_0(\hat{\mathbf{c}})$ if $b = 0$. That is the minimum of the function F is $\theta^* = \hat{\mathbf{c}}$ when $b = 0$. Similarly, $\theta^* = \tilde{\mathbf{c}}$ when $b = 1$.

We fix a notation that we use very often in this proof. For a function G, let

$$\mathsf{disc}(G; \theta) := \left|G(\theta) - \min_{\theta \in \mathcal{B}^p(\mathbf{0}^p, 1)} G(\theta)\right|.$$

When $b = 0$, for all $\theta \in \mathcal{B}^p(\mathbf{0}^p, 1)$ such that $\mathsf{disc}(F_0; \theta) \leq \alpha/21$, we have $\|\theta - \hat{\mathbf{c}}\|_2 \leq \alpha/7$. This is because of the construction of f_0 and that the minimum of F_0 occurs at $\theta = \hat{\mathbf{c}}$. That is, for all such θ,

$$\|\mathbf{c} - \theta\|_2 \geq \|\mathbf{c} - \hat{\mathbf{c}}\|_2 - \|\theta - \hat{\mathbf{c}}\|_2 \geq \frac{\alpha}{6} - \frac{\alpha}{7} > 0$$
$$\|\tilde{\mathbf{c}} - \theta\|_2 \geq \|\tilde{\mathbf{c}} - \hat{\mathbf{c}}\|_2 - \|\theta - \hat{\mathbf{c}}\|_2 \geq \frac{2\alpha}{3} - \frac{\alpha}{7} > \frac{\alpha}{2}.$$

Moreover, since $\|\theta - \hat{\mathbf{c}}\|_2 \leq \alpha/7$, we have $\|\theta - \mathbf{c}\|_2 \leq \alpha/3 + \alpha/7 < \alpha$. Therefore, all θ such that $\mathsf{disc}(F_0; \theta)$, we have $\theta \in \mathcal{B}^p(\mathbf{c}, \alpha)$. Similarly, when $b = 1$, for all $\zeta \in \mathcal{B}^p(\mathbf{0}^p, 1)$ such that $\mathsf{disc}(F_1; \theta) \leq \alpha/21$, we have $\zeta \in \mathcal{B}^p(\mathbf{c}, \alpha)$, $\|\mathbf{c} - \zeta\|_2 > 0$, and $\|\hat{\mathbf{c}} - \theta\|_2 > \alpha/2$. The second and higher order derivative of the function F is identically zero and the $\nabla F(\theta) = 1$ for all $\theta \in \mathcal{B}^p(\mathbf{0}^p, 1) \backslash \mathcal{B}^p(\mathbf{c}, \alpha)$ and $\nabla F = 1/3$ for all $\theta \in \mathcal{B}^p(\mathbf{c}, \alpha)$.

This implies that in order to output whether $\theta^* = \hat{\mathbf{c}}$ or $\theta^* = \tilde{\mathbf{c}}$, at least one of the queries of any q-nonadaptive algorithm has to be a point inside the ball $\mathcal{B}^p(\mathbf{c}, \alpha)$. Now fix a query i. Since the ratio of the volume of ball $\mathcal{B}^p(\mathbf{c}, \alpha)$ to the volume of $\mathcal{B}^p(\mathbf{0}^p, 1)$ is α^p, the probability that the query i is a point inside the ball is α^p. By a union bound, the probability that any query lands in the ball is at most $q\alpha^p$, as desired.

To prove the second part of the claim, we show that there is an efficient adaptive algorithm Adaptive that outputs θ^* such that $\mathsf{disc}(F; \theta^*) \leq \alpha$. Adaptive makes two batches of queries:

1) In the first batch, Adaptive randomly picks a point $\theta \in \mathcal{B}^p(\mathbf{0}^p, 1)$ and queries for the gradient $\nabla F(\theta)$.
2) Let $\mathbf{g}$ be the direction of the gradient returned. In the second batch, Adaptive randomly picks $\mathrm{poly}(\log(1/\alpha))$ points in $\mathcal{B}^p(\mathbf{0}^p, 1)$ along the direction of $\mathbf{g}$ and makes gradient queries at all these points in the second round.

By Chernoff bound, with probability at least $2/3$, a query in the second batch is a point that lies in $\mathcal{B}^p(\mathbf{c}, \alpha)$. This allows the algorithm to output θ^* such that $\mathsf{disc}(F; \theta^*) \leq \alpha$. This completes the proof. ■

In the view of Theorem 12, a natural question arises is whether, is it always possible to output a θ^* using constant round of adaptivity and neighborhood oracles for a function F sampled from any arbitrary distribution $\mathcal{F}$ of functions defined over a convex set $\mathcal{C}$, such that $\mathsf{disc}(F, \theta^*) \leq \alpha$? Our next result shows that this is not true in general. That is, there exists a distribution of functions from $\mathcal{B}^p(\mathbf{0}^p, 1)$ to $[0, 1]$, such that, for $F(\cdot)$ sampled from that distribution, no $(\mathrm{poly}(p), O(1))$-adaptive (and randomized) algorithm with access to $\mathcal{O}_F$ can output a θ^* such that $\mathsf{disc}(F, \theta^*) \leq \alpha$ with

a constant probability. More formally, we have the following theorem.

Theorem 13. *Let $\mathcal{B}^p(\mathbf{0}^p, 1)$ denotes a unit ball centered around the origin. There exists a distribution $\mathcal{F}$ of convex functions from a unit ball $\mathcal{B}^p(\mathbf{0}^p, 1)$ to $[0, 1]$ such that, for $F \sim \mathcal{F}$, the following holds. Let $\mathcal{O}_F(\cdot)$ be a neighborhood based oracle for F as defined above. For every $\alpha > 0$, there exists a $T := T(\alpha)$, with $T(\alpha) = \Theta(\log(1/\alpha))$, such that the output θ^* of any (q, T)-adaptive and randomized algorithm with oracle access to $\mathcal{O}_F(\cdot)$, satisfies*

$$\mathsf{Pr}\left[\left|F(\theta^*) - \min_{\theta \in \mathcal{B}^p(\mathbf{0}^p,1)} F(\theta)\right| \leq \alpha\right] \leq \frac{(T+1)q2^{-p}}{1-3^{-p}}.$$

Proof: Let $\mathcal{B}_0 = \mathcal{B}^p(\mathbf{0}^p, 1)$ and $r_0 = 1$. Recursively for all level of recursion, $i = \{1, \cdots, T\}$, define the following:

1) Define an packing $\mathcal{V}_i$ of the ball $\mathcal{B}_{i-1}$ with balls of radius $r_i := \frac{1}{2}\left(\frac{7}{18}\right)^{i-1}$.
2) Pick a random ball from the packing $\mathcal{V}_i$ with center $\mathbf{c}^{(i)}$. Set this ball as $\mathcal{B}_i$ for the next level of recursion.
3) Randomly pick $\mathbf{c}_0^{(i)}, \mathbf{c}_1^{(i)} \in \mathcal{B}_i$ such that (i) balls of radius $r_i/6$ with centers $\mathbf{c}_0^{(i)}$ and $\mathbf{c}_1^{(i)}$ lies in $\mathcal{B}_i$ are disjoint, (ii) $r_i/12 \leq \|\mathbf{c}^{(i)} - \mathbf{c}_0^{(i)}\|_2, \|\mathbf{c}^{(i)} - \mathbf{c}_1^{(i)}\|_2 \leq r_i/6$ and (iii) $\|\mathbf{c}_0^{(i)} - \mathbf{c}_1^{(i)}\|_2 \geq r_i/3$.
4) If all the conditions in step 3 are not satisfied, go back to step 2 and repeat.

We define two functions at every level of the recursion as follows:

$$f_0^{(i)} := \left(\frac{4}{7}\right)^i \|\theta - \mathbf{c}_0^{(i)}\|_2 + \left(\frac{1}{3} + \frac{r_i}{3}\sum_{j=1}^{i-1}\left(\frac{4}{7}\right)^j\right)$$

$$f_1^{(i)} := \left(\frac{4}{7}\right)^i \|\theta - \mathbf{c}_1^{(i)}\|_2 + \left(\frac{1}{3} + \frac{r_i}{3}\sum_{j=1}^{i-1}\left(\frac{4}{7}\right)^j\right).$$

Set $f(\theta) = \|\theta - \mathbf{c}^{(1)}\|_2$. Now pick T random bits $b_1, \cdots, b_T$ and set the function as follows:

$$F(\theta) := \max\left\{f(\theta), \left\{f_{b_i}^{(i)}(\theta)\right\}_{i=1}^{T}\right\}. \tag{6}$$

Claim 14. *The function $F(\cdot)$ is convex.*

Proof: First note that the two functions defined in every levels of recursion are convex. Further, $f(\cdot)$ is convex. Since point-wise maximum of convex functions is convex, the claim follows. ∎

We now return to the proof of Theorem 13. Without loss of generality, let us assume that all the random bits $b_1, \cdots, b_T$ are 0. We can make this assumption because the functions at the same level sets are defined analogously. We first prove the following structural property of our function. The claim basically says that in a small ball around the points chosen in step 3 above in any level of recursion, $F(\cdot)$ is defined by only the functions at the lower level of recursion.

Claim 15. *Let i be any index in $[T]$. Then for every θ in the ball of radius $r_{i+1} := \frac{1}{2}\left(\frac{7}{18}\right)^i$ around $\mathbf{c}_0^{(i)}$, $F(\theta) = \max_{i \leq j \leq T}\left\{f_0^{(j)}(\theta)\right\}$.*

Proof: We prove the claim by induction.

Base case. When $i = 1$, then we have for all $\theta \in \mathcal{B}_2$, $f(\theta) = \|\theta - \mathbf{c}^{(1)}\|_2 \in [1/12, 5/18]$. However, by construction, $f_0^{(1)}(\theta) := \frac{4}{7}\|\theta - \mathbf{c}_0^{(1)}\|_2 + \frac{1}{3} \in [1/3, 4/9]$. Therefore, for all $\theta \in \mathcal{B}_2$, $F(\theta) = \max_{1 \leq j \leq T}\left\{f_0^j\right\}_{j=1}^{T}$. This completes the base case.

Inductive case. Let us assume that for every θ in the ball of radius r_{i+1} around $\mathbf{c}_0^{(i)}$, $F(\theta) = \max_{i \leq j \leq T}\left\{f_0^{(j)}(\theta)\right\}$. To prove the claim, we need to prove that, for $\theta \in \mathcal{B}_{i+2}$ around $\mathbf{c}_0^{(i+1)}$, $F(\theta) = \max_{i < j \leq T}\left\{f_0^{(j)}(\theta)\right\}$. From the induction hypothesis we know that $F(\theta) = \max_{i \leq j \leq T}\left\{f_0^j(\theta)\right\}$ in the ball $\mathcal{B}_i$. Since $\mathcal{B}_{i+2} \subseteq \mathcal{B}_{i+1}$ by construction, we need to show that $f^{(i)}(\theta) \leq \max\left\{f_0^j(\theta)\right\}_{j=i+1}^{T}$. Note that $f_0^{(i)}(\theta)$ and $f_0^{(i+1)}(\theta)$ grows linearly as the distance of the point θ from $\mathbf{c}_0^{(i)}$ with slope $\left(\frac{4}{7}\right)^i$ and $\left(\frac{4}{7}\right)^{i+1}$, respectively. Therefore, if we prove that $f_0^{(i+1)}(\theta) \geq f_0^{(i)}(\theta)$ at the boundary of the ball $\mathcal{B}_{i+2}$ and at $\theta = \mathbf{c}_0^{(i+1)}$, we are done. At $\theta = \mathbf{c}_0^{(i+1)}$, we have

$$\begin{aligned} f_0^{(i)}(\theta) &\leq \left(\frac{4}{7}\right)^i \frac{r_i}{6} + \frac{1}{3} + \frac{r_i}{3}\sum_{j=1}^{i-1}\left(\frac{4}{7}\right)^j \\ &\leq \frac{1}{3} + \frac{r_i}{3}\sum_{j=1}^{i}\left(\frac{4}{7}\right)^j \leq f_0^{(i+1)}(\theta). \end{aligned}$$

For θ at the boundary of the ball $\mathcal{B}_{i+2}$, we have

$$\begin{aligned} f_0^{(i)}(\theta) &= \left(\frac{4}{7}\right)^i \|\theta - \mathbf{c}_0^{(i)}\| + \frac{1}{3} + \frac{r_i}{3}\sum_{j=1}^{i-1}\left(\frac{4}{7}\right)^j \\ &\leq \left(\frac{4}{7}\right)^i \|\mathbf{c}_0^{(i+1)} - \mathbf{c}_0^{(i)}\|_2 + \left(\frac{4}{7}\right)^i \|\mathbf{c}_0^{(i+1)} - \theta\|_2 \\ &\quad + \frac{1}{3} + \frac{r_i}{3}\sum_{j=1}^{i-1}\left(\frac{4}{7}\right)^j \\ &\leq \left(\frac{4}{7}\right)^i \frac{r_i}{6} + \left(\frac{4}{7}\right)^i \|\mathbf{c}_0^{(i+1)} - \theta\|_2 = f_0^{(i+1)}(\theta). \end{aligned}$$

This completes the proof that for every θ in the ball of radius r_{i+1} around $\mathbf{c}_0^{(i+1)}$, $F(\theta) = \max\left\{f_0^{(j)}(\theta)\right\}_{j=i}^{T}$ and Claim 15 follows. ∎

One of the main corollaries of Claim 15 is that $\mathbf{c}_0^{(T)} = \operatorname{argmin}_{\theta\in\mathcal{B}^p(\mathbf{0}^p,1)} F(\theta)$. Another direct consequence of this structural theorem is that $F(\theta)$ is 1-Lipschitz.

Now suppose $\theta \in \mathcal{B}^p(\mathbf{0}^p, 1)$ is such that $\mathsf{disc}(F;\theta) \le \frac{2}{7}\left(\frac{7}{18}\right)^T$ .We know that $F(\theta) = f_0^{(T)}(\theta)$ in the ball of radius $\frac{1}{2}\left(\frac{7}{18}\right)^T$. Since $\mathbf{c}_0^{(T)} = \operatorname{argmin}_{\theta\in\mathcal{B}^p(\mathbf{0}^p,1)} F(\theta)$, we know that $\|\theta - \mathbf{c}_0^{(T)}\|_2 \le \frac{1}{14}\left(\frac{7}{18}\right)^T \le \|\mathbf{c}_0^{(r-1)} - \mathbf{c}_0^{(T)}\|_2$. Since balls of radius $\frac{1}{2}\left(\frac{7}{18}\right)^T$ around $\mathbf{c}_0^{(T)}$ and $\mathbf{c}_1^{(T)}$ are disjoint, this implies that any such θ is closer to $\mathbf{c}_0^{(T)}$. In other words, for any algorithm to output a θ such that $\mathsf{disc}(F;\theta) \le \frac{2}{7}\left(\frac{7}{18}\right)^T$, it has to distinguish where the minimizer is $\mathbf{c}_0^{(T)}$ or $\mathbf{c}_1^{(T)}$. That is, any such algorithm has to query for a point inside the ball $\mathcal{B}_T$ in at least one of the T batches.

Now the adversary can compute the minimizer in less than T rounds if, in the batch-j query, one of its queries is a point inside $\mathcal{B}_i$ for $i \ge j$. Let $E_{i,j}$ be such an event. Then for $i \ge 1$, $\mathsf{Pr}[E_{i,j}] = q_j \mathsf{vol}(\mathcal{B}_i)/\mathsf{vol}(\mathcal{B}_j) \le q\left(\frac{7}{18}\right)^{(i-j)p}$. Therefore, the probability with which the adversary succeeds in computing a minimizer in less than T rounds is

$$\mathsf{Pr}\left[\bigvee_{i=1}^{T} E_i\right] \le q\left(\left(\frac{1}{2}\right)^{Tp} + T\left(\frac{7}{18}\right)^{Tp}\right).$$

Setting $T := \Theta(\log(1/\alpha))$ completes the proof. ∎

IV. Noisy-Gradient Based Methods

In this section we give noisy versions of the gradient computation based algorithms to solve convex optimization. Before we present our noisy gradient based method, we first provide an exposition of an algorithm (in Section IV-A), which we will use heavily in this and later sections, for estimating the gradient (and also the loss) of the population risk $L(\theta) = \mathbb{E}_{d\sim\mathcal{P}}[\ell(\theta;d)]$ at any θ.

A. Locally Differentially Private (LDP) Function and Gradient Oracles

Gradient oracle: For a given data set $\mathbf{D} = \{d_1, \cdots, d_n\}$ drawn i.i.d. from the population distribution $\mathcal{P}$, the objective is to estimate the gradient of $L(\theta) = \mathbb{E}_{d\sim\mathcal{P}}[\ell(\theta;d)]$ at a given θ, using local reports from n/T samples from $\mathbf{D}$. Algorithm Noisy-Gradient-Oracle is expected to be called to estimate the gradient at T different θ's $(\theta_1, \cdots, \theta_T)$ using T sets of n/T disjoint samples.

Input. Current partition number: t, number of partitions: T, current model estimate: $\theta_{\mathsf{current}}$, privacy parameter: ε, value flag: $\mathsf{flag} \in \{\mathsf{T}, \mathsf{F}\}$, fixed model: θ_0. Let $\mathbf{D} = \{d_1, \cdots, d_n\} \sim \mathcal{P}^n$ be the data set, $L(\theta) = \mathbb{E}_{d\sim\mathcal{P}}[\ell(\theta;d)]$ is a 1-Lipschitz function.

Algorithm. Perform the following steps:

1) Define $\mathcal{S}_t = \left\{(t-1)\lfloor\frac{n}{T}\rfloor, \cdots, t\lfloor\frac{n}{T}\rfloor - 1\right\}$. **If** $t = T$, **then** $\mathcal{S}_t = \mathcal{S}_t \cup \left\{t\lfloor\frac{n}{T}\rfloor, \cdots, n\right\}$.
2) The server sends $\theta_{\mathsf{current}}$ to all the users. Every user i s.t. $\{i \in \mathcal{S}_t\}$ does the following: Compute $\mathbf{z}_i := \mathcal{R}_\varepsilon(\nabla\ell(\theta_{\mathsf{current}}; d_i))$ and $z_i := \mathcal{R}_\varepsilon(\ell(\theta_{\mathsf{current}}; d_i) - \ell(\theta_0; d_i))$, where $\mathcal{R}_\varepsilon(\cdot)$ is the randomizer defined in Appendix A.
3) Compute $\mathsf{Loss}_{\mathsf{noisy}}(t) = \frac{1}{\mathcal{S}_t}\sum_{i\in\mathcal{S}_t} z_i$ and $\mathsf{Grad}_{\mathsf{noisy}}(t) = \frac{1}{\mathcal{S}_t}\sum_{i\in\mathcal{S}_t} \mathbf{z}_i$.
4) **If** $\mathsf{flag}=\mathsf{F}$, **return** $\mathsf{Grad}_{\mathsf{noisy}}(t)$ **else return** $(\mathsf{Grad}_{\mathsf{noisy}}(t), \mathsf{Loss}_{\mathsf{noisy}}(t))$.

Figure 6. Locally Noisy Gradient Oracle (Noisy-Gradient-Oracle)

Privacy guarantee. The following theorem is immediate from the privacy result of Duchi *et al.* [14].

Theorem 16. Noisy-Gradient-Oracle *is ε-LDP.*

In particular, since differential privacy is preserved under post-processing, all our algorithms in this section are ε-LDP. In the following, we state the following utility guarantees for the algorithm Noisy-Gradient-Oracle. We will use both the unbiasedness and the concentration properties below heavily in the later algorithms (see Appendix A for a proof).

Theorem 17. *Let $\mathbf{D} = \{d_1, \cdots, d_n\}$ be drawn i.i.d. from the distribution $\mathcal{P}$, and let $L(\theta) = \mathbb{E}_{d\sim\mathcal{P}}[\ell(\theta;d)]$. Then for a fixed partition number $t \in [T]$ and a $\theta_{\mathsf{current}}$ (independent of the data samples in t), the following are true for Algorithm* Noisy-Gradient-Oracle.

1) **Unbiasedness:** $\mathbb{E}[\mathsf{Grad}_{\mathsf{noisy}}(t)] = \nabla L(\theta_{\mathsf{current}})$.
2) **Bounded variance:**

$$\mathbb{E}\left[\|\mathsf{Grad}_{\mathsf{noisy}}(t) - \nabla L(\theta_{\mathsf{current}})\|_2^2\right] = O\left(Tp/n\varepsilon^2\right).$$

3) **Tight concentration:** *Let $\boldsymbol{x}$ be any vector (with $\|\boldsymbol{x}\|_2 \le 1$) independent of the sample set S_t and the gaussian noise added in Figure 6. Then with probability at least $1-\gamma$, $|\langle\mathsf{Grad}_{\mathsf{noisy}}(t), \boldsymbol{x}\rangle - \langle\nabla L(\theta_{\mathsf{current}}), \boldsymbol{x}\rangle|$ is bounded by $O\left(\sqrt{T\log(1/\gamma)/n\varepsilon^2}\right)$.*

Function oracle: We have the following bound on the estimation of the loss (population risk) at a given $\theta_{\mathsf{current}}$. The proof of this theorem is analogous to Theorem 17.

Theorem 18. *Let θ_0 be a fixed model in the convex set $\mathcal{C}$. Let $\boldsymbol{x}$ be any vector (with $\|\boldsymbol{x}\|_2 \leq 1$) independent of the Gaussian noise added in Figure 6. Using the notation from Theorem 17, for Algorithm* NOISY-GRADIENT-ORACLE *with probability at least $1-\gamma$, $|\mathsf{Loss}_{\mathsf{noisy}}(t) - (L(\theta_{\mathsf{current}}) - L(\theta_0))| = O\left(\sqrt{T\log(1/\gamma)/n\varepsilon^2}\right)$.*

B. Noisy Cutting Plane Method

In this section we provide the details of the main cutting plane method (Figure 7) The algorithm for noisy cutting plane method is a simple variation of the cutting plane method of Levin [30] and Newman [35]. However, we need to be careful due to the effect of the noise at every iteration. To better understand the algorithm, we first review the basic cutting plane method. In a cutting plane method, we assume there is an oracle which, in every iteration $t \in [T]$ of the first stage, either returns a point θ_t in the convex set $\mathcal{C}$, with sufficiently small error, or a hyperplane that divides the solution space roughly by half. The idea is that after T iterations, there will be at least one iteration $t^* \in [T]$ such that $err_{\mathcal{P}}(\theta_{t^*}) \leq \alpha$, where α is the desired accuracy. Finally, the output is an $\theta \in \{\theta_1, \cdots, \theta_T\}$ such that it minimizes the loss $L(\theta)$.

In the differentially private variant of the cutting-plane method (Algorithm 7 (PRIVATE-COG)), instead of using the exact gradient (or the function oracle for the loss $L(\theta)$), we use their differentially private variant from Section IV-A. The ε-local differential privacy of PRIVATE-COG follows immediately from Theorem 16 and that differential privacy is preserved under post-processing. In Theorem 19 we provide the utility guarantee. (See Appendix B-B for the proof.)

Theorem 19. *Let $\mathcal{C}$ be a bounded convex set such that $\|\mathcal{C}\|_2 \leq 1$. Let $\mathbf{D} = \{d_1, \cdots, d_n\}$ be drawn i.i.d. from the distribution $\mathcal{P}$. Let $L(\theta) = \mathbb{E}_{d\sim\mathcal{P}}[\ell(\theta; d)]$. Then with probability $1-\gamma$ over the coin tosses of* NOISY-COG*,*

$$err_{\mathcal{P}}(\theta_{\mathsf{priv}}) = O\left(T\ln(T/\gamma)/\left(n\varepsilon^2\right) + (1-1/e)^{T/p}\right).$$

Remark 2. *The algorithm can be made efficient by using the randomized algorithm of Bertsimas and Vempala [9] to compute the center of gravity.*

C. Noisy Gradient Descent

In this section we focus on understanding the role of interaction in the convergence rate of gradient descent

Input. Given a convex set $\mathcal{C}$ such that $\|\mathcal{C}\|_2 \leq 1$. Let $\mathbf{D} = \{d_1, \cdots, d_n\} \sim \mathcal{P}^n$ be the data set, $L(\theta) = \mathbb{E}_{d\sim\mathcal{P}}[\ell(\theta; d)]$ is a 1-Lipschitz function.

Algorithm. The server does the following.

1) Set $\mathcal{C}_0 := \mathcal{C}$. Pick a point $\theta_0 \in \mathcal{C}$.
2) **for** $t = 1, \cdots, T$,
   a) Compute $\theta_t \leftarrow$ COG$(\mathcal{C}_{t-1})$ and estimate $(\nabla L_{\mathsf{priv}}(\theta_t), L_{\mathsf{priv}}(\theta_t)) \leftarrow$NOISY-GRADIENT-ORACLE $(t, T, \theta_t, \varepsilon, \mathsf{T}, \theta_0)$.
   b) The server updates the convex set as follows: $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1} \cap \{\theta \in \mathbb{R}^p : \langle \nabla L_{\mathsf{priv}}(\theta_t), \theta - \theta_t \rangle \leq 0\}$.

Output. The server outputs the following.

$$\theta_{\mathsf{priv}} := \operatorname*{argmin}_{\theta_t \in \{\theta_1, \cdots, \theta_T\}} \{L_{\mathsf{priv}}(\theta_t)\} \tag{7}$$

Figure 7. A Noisy Cutting Plane Method (NOISY-COG)

Input. Given a convex set $\mathcal{C}$ such that $\|\mathcal{C}\|_2 \leq 1$. Let $\mathbf{D} = \{d_1, \cdots, d_n\} \sim \mathcal{P}^n$ be the data set, $L(\theta) = \mathbb{E}_{d\sim\mathcal{P}}[\ell(\theta; d)]$ is a 1-Lipschitz function.

Algorithm. The server defines a step size η_t and follows the following steps.

1) Pick a point $\theta_0 \in \mathcal{C}$.
2) **for** $t = 1, \cdots, T$,
   a) Estimate $\nabla L_{\mathsf{priv}}(\theta_t)$=NOISY-GRADIENT-ORACLE$(t, T, \theta_t, \varepsilon, \mathsf{F})$.
   b) Computes $\theta_{t+1} = \Pi_{\mathcal{C}}\left(\theta_t - \eta_t \nabla L_{\mathsf{priv}}(\theta_t)\right)$

Output. $\theta_{priv} := \theta_T$

Figure 8. Local and Noisy Gradient Descent Algorithm (NOISY-GRADIENT-DESCENT)

style algorithms in the local model with access to noisy gradient oracle (NOISY-GRADIENT-DESCENT (Figure 8)). We analyze the gradient descent algorithm when the convex function has certain properties, such as Lipschitz property, smoothness and strong convexity. While noisy gradient descent has been extensively studied in [7, 14, 29] for optimality of excess empirical risk (both in the local and the central model), we focus on the interaction complexity of noisy gradient descent to achieve those errors. The rounds of interaction differ significantly under various assumptions on the loss function $\ell(\cdot, d)$. As a result, we state them categorically below. Moreover, in each of the setting, we get ϵ-LDP algorithm by using the noisy gradient oracle from Section IV-A.

The fact that NOISY-GRADIENT-DESCENT is ϵ-LDP

follows immediately from Theorem 16. In the following we state the interaction round complexity of NOISY-GRADIENT-DESCENT. While the first three results in Theorem 20 follow immediately from prior works ([40] and [29] respectively) in combination with Theorem 16, the bound for smooth and strongly convex functions require a new analysis, which we state in Appendix B-C.

Theorem 20 (Utility guarantees for noisy gradient descent)**.** *Let $\mathcal{C}$ be a bounded convex set such that $\|\mathcal{C}\|_2 \leq 1$. Let $\mathbf{D} = \{d_1, \cdots, d_n\}$ be drawn i.i.d. from the distribution $\mathcal{P}$, and let $L(\theta) = \mathbb{E}_{d\sim\mathcal{P}}[\ell(\theta; d)]$. Let $\ell(\cdot;\cdot)$ be 1-Lipschitz in the first parameter. The following are true for the* NOISY-GRADIENT-DESCENT *algorithm.*

1) *If learning rate $\eta = \frac{1}{\sqrt{t}}$, then $\mathbb{E}[err_{\mathcal{P}}(\theta_{\mathsf{priv}})] = O\left(\log T\left(1/\sqrt{T} + \sqrt{p/n\varepsilon^2}\right)\right)$.*
2) *If $\ell(\cdot;\cdot)$ is Δ-strongly convex and the learning rate $\eta = \frac{1}{\Delta t}$, then $\mathbb{E}\left[err_{\mathcal{P}}(\theta_{\mathsf{priv}})\right] = O\left(\frac{\log T}{\Delta T} + \frac{p\log(T)}{n\varepsilon^2\Delta}\right)$.*
3) *If $\ell(\cdot;\cdot)$ is β-smooth and the learning rate $\eta = \frac{1}{\Delta t}$, then Nestrov's accelerated variant of* NOISY-GRADIENT-DESCENT *[29] satisfies $\mathbb{E}\left[err_{\mathcal{P}}(\theta_{\mathsf{priv}})\right] = O\left(\frac{1}{T^2} + \frac{pT^2\log^2 T}{n\epsilon^2\beta}\right)$.*
4) *If $\ell(\cdot;\cdot)$ is β-smooth and Δ-strongly convex, then with learning rate $\eta = \frac{1}{\beta}$ we have $\mathbb{E}\left[err_{\mathcal{P}}(\theta_{\mathsf{priv}})\right] = O\left(\frac{\beta}{2}e^{-\Delta T/\beta} + \frac{pT}{n\varepsilon^2}\right)$.*

Remark 3. *For 1-Lipschitz convex functions, one can achieve an error of $\widetilde{O}(\sqrt{p/n})$ in $T = \sqrt{n/p}$ rounds of interaction. Variants of this result has appeared in [7, 14, 43]. Under Δ-strong convexity, one can achieve an error of $\widetilde{O}(p/n)$ in $T = n/p$ rounds of interaction. Variants of this result has appeared in [7, 43]. Under β-smoothness, one can achieve an error of $\widetilde{O}(\sqrt{p/n})$ in $T = (n/p)^{1/4}$ rounds of interaction. The algorithm is a variant of the classic Nestrov's* accelerated *gradient descent (see [29] for a complete exposition.) Further, if we assume Δ-strong convexity and β-smoothness along with the default 1-Lipschitzness property, then one can show that with $T = \frac{\beta\log(n)}{\Delta}$,* NOISY-GRADIENT-DESCENT *algorithm achieves an error of $\widetilde{O}(p/n)$.*

All the number of rounds of interaction above are tight [33, 34], and equals that of the non-private counterparts. Notice that Theorem 20 in essence implies that NOISY-GRADIENT-DESCENT algorithm reaches the "error level" in computing the gradient of the loss function at the same rate as the non-private counterpart. The main takeaway from this section is that differentially private (noisy) gradient descent can be shown to converge to the optimal error level, with the same asymptotic number of interactions as the non-private variant.

V. CONCLUSIONS

This paper looks at how the structure of an optimization problem affects the amount of interaction necessary for a locally-private algorithm to arrive at an accurate answer. The starting observation for our work is that the parallelizability of local protocol queries matters—protocols that require multiple rounds of interaction are slower and harder to engineer. The ideal is a noninteractive protocol, in which all queries are asked in parallel. However, protocols with just a few rounds of interaction (that is, batches of queries) can be useful.

There are accurate, noninteractive protocols for some tasks, such as computing sums or finding heavy hitters. But current protocols for convex optimization use as many rounds of interaction as there are participants in the protocol. We focus on convex optimization since it is a fundamental tool in statistics and machine learning.

We find that the structure of an optimization problem affects not only the accuracy that can be achieved by differentially private protocols, but also the amount of interaction necessary to get that accuracy.

Our paper provides both good and bad news for the prospective implementer of locally private optimization. **On the positive side**, we draw three main conclusions:

(*i*) For very low-dimensional convex optimization, there are reasonable noninteractive protocols. These protocols have nearly the same theoretical guarantees as the best interactive protocols when the dimension is constant. Evaluating their effectiveness in practice—and especially understanding the range of dimensions at which they offer an attractive tradeoff—requires further study.

(*ii*) Sometimes problem structure can be exploited to find good noninteractive protocols. We illustrate this with a noninteractive algorithm for linear regression that has the same asymptotic error as the best interactive one, even in high dimensions (see, Appendix C). This exploits the fact that the quadratic loss function itself has a compact description.

(*iii*) A few rounds of interaction help enormously. The same ideas that go into designing protocols with many rounds of interaction can be adapted to get protocols with optimal error and just a few rounds of interaction. Specifically, an optimization algorithm is "first-order" if it gets information only by approximating the gradient or the loss function at a small number of "query" points. First-order optimization methods that make few queries and handle noise gracefully can be turned into protocols

that use few rounds of interaction. We demonstrate this for gradient descent and the cutting plane method.

The best choice of algorithm depends on the tradeoff between various aspects of the problem structure (dimension, smoothness, strong convexity); see Figure 2. Interestingly, the smoothness of a problem does not affect the error achievable by the best differentially private algorithm, but it can drastically reduce interaction.

Unfortunately, for general convex optimization (that is, without assuming both smoothness and strong convexity), our protocols still require a number of rounds of interaction that grows polynomially with the problem dimension.

On the negative side, we show:

(*iv*) Known algorithmic design techniques for local private learning cannot provide noninteractive protocols—or even ones with few rounds of interaction—for general convex optimization in high dimensions. Specifically, any first-order optimization algorithm must either have "many" batches of queries, or make exponentially-many queries (in the dimension p) in order to have nontrivial accuracy. Here "many" depends on the desired accuracy—the lower bound is about $\log(1/\alpha)$ for desired error α. That means that every *nonadaptive* first-order algorithm (one that asks all its queries in one batch) must make exponentially many queries to be useful. Consequently, very different techniques are needed for noninteractive, general purpose local optimization.

Both our positive and negative results highlight the role of several key structural properties: (*a*) The dimension p, that is the number of real parameters in the vectors θ over which we aim to minimize a loss function. (*b*) The variability of the loss function we aim to minimize; measures of variability include the "Lipschitz constant"—an upper bound on the amount that any one individual can change the gradient of the loss function—and the *smoothness*, which is an upper bound on the rate at which the gradient changes as θ varies. For example, the loss function for support vector machines has low Lipschitiz constant but is not smooth (the gradient changes abruptly). (*c*) The *strong convexity* of the loss function. A strongly convex function is bounded below by a quadratic function at every point and, in particular, has a well-defined minimum. (*d*) The shape and size of the constraint set $\mathcal{C}$ in which θ resides. We focus here on the role of the diameter of $\mathcal{C}$, though other work on (central) differential privacy suggests that properties (the Gaussian width, and the number of exptreme points) [43] also play a role.

Perhaps the simplest conclusion is that these structural properties are important, and a prospective implementer's first task is to extract as much structure as possible from the specific problem and include these extra properties explicitly in the problem formulation.

The importance and difficulty of evaluation. This paper focuses on the analytical evaluation of several algorithms and advances some basic design principles. One of the lessons of our work is that there is currently no single all-purpose algorithm for local private learning, and so the choice of the best algorithm is likely to depend on the problem and the data at hand. That highlights the need for further algorithmic research—what new techniques can we bring to bear? can we bypass the lower bounds in this paper by using algorithms that access the loss function differently?—as well as careful, application-specific empirical evaluation. We hope this study informs such efforts, both by suggesting specific algorithms and highlighting structural properties that play an important role.

ACKNOWLEDGMENTS

We are grateful to Vitaly Feldman, Ilya Mironov, and Kunal Talwar for helpful conversations about this problem, as well as to David Evans and several anonymous referees for their detailed and constructive feedback on the submitted version. We would like to thank Prateek Jain especially for some of the ideas in the convergence proof of gradient descent with smooth and strongly convex functions. A.S. and J.U. are supported by NSF award IIS-1447700, a Google Faculty Award and a Sloan Foundation research award.

REFERENCES

[1] Privacy by the numbers: A new approach to safeguarding data. *Scientific American*, 2012.

[2] Apple tries to peek at user habits without violating privacy. *The Wall Street Journal*, 2016.

[3] A. Agarwal, M. J. Wainwright, P. L. Bartlett, and P. K. Ravikumar. Information-theoretic lower bounds on the oracle complexity of convex optimization. In *NIPS*, pages 1–9, 2009.

[4] S. Agrawal, J. R. Haritsa, and B. A. Prakash. Frapp: a framework for high-accuracy privacy-preserving mining. *Data Mining and Knowledge Discovery*, 18(1):101–139, 2009.

[5] K. Ball. An elementary introduction to modern convex geometry. *Flavors of geometry*, 31:1–58, 1997.

[6] R. Bassily and A. Smith. Local, private, efficient protocols for succinct histograms. In *STOC*, pages 127–135. ACM, 2015.

[7] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*, pages 464–473. IEEE, 2014.

[8] A. Beimel, K. Nissim, and E. Omri. Distributed private data analysis: Simultaneously solving how and what. In *CRYPTO*, pages 451–468. Springer, 2008.

[9] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *JACM*, 51(4):540–556, 2004.
[10] S. Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.
[11] P. Bürgisser and F. Cucker. *Condition: The geometry of numerical algorithms*, volume 349. Springer Science & Business Media, 2013.
[12] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26, 2011.
[13] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *JMLR*, 12(Mar):1069–1109, 2011.
[14] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438. IEEE, 2013.
[15] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.
[16] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, 2006.
[17] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284. Springer, 2006.
[18] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *STOC*, pages 715-724, ACM, 2010.
[19] Ú. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067. ACM, 2014.
[20] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, pages 211–222. ACM, 2003.
[21] V. Feldman, C. Guzman, and S. Vempala. Statistical query algorithms for stochastic convex optimization. *arXiv:1512.09170*, 2015.
[22] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the Accuracy of Differentially Private Histograms Through Consistency. *PVLDB*, 3(1):1021–1032, 2010.
[23] J. Hsu, S. Khanna, and A. Roth. Distributed private heavy hitters. In *ICALP*, pages 461–472. Springer, 2012.
[24] P. Jain, P. Kothari, and A. Thakurta. Differentially private online learning. In *COLT*, pages 24–1, 2012.
[25] S. P. Kasiviswanathan and A. Smith. On the'semantics' of differential privacy: A bayesian formulation. *Journal of Privacy and Confidentiality*, 6(1):1, 2014.
[26] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.
[27] M. Kearns. Efficient noise-tolerant learning from statistical queries. *JACM*, 45(6):983–1006, 1998.
[28] D. Kifer, A. Smith, and A. Thakurta. Private convex empirical risk minimization and high-dimensional regression. *JMLR*, 1:41, 2012.
[29] G. Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1-2): 365–397, 2012.
[30] A. Y. Levin. On an algorithm for the minimization of convex functions. In *Soviet Mathematics Doklady*, volume 160, pages 1244–1247, 1965.
[31] F. J. MacWilliams and N. J. A. Sloane. *The theory of error correcting codes*. Elsevier, 1977.
[32] N. Mishra and M. Sandler. Privacy via pseudorandom sketches. In *SIGMOD*, pages 143–152. ACM, 2006.
[33] A. Nemirovski, D.-B. Yudin, and E.-R. Dawson. Problem complexity and method efficiency in optimization. 1982.
[34] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.
[35] D. J. Newman. Location of the maximum on unimodal surfaces. *JACM*, 12(3):395–398, 1965.
[36] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *CCS*, pages 192–203. ACM, 2016.
[37] M. Raginsky and A. Rakhlin. Information-based complexity, feedback and dynamics in convex programming. *IEEE Transactions on Information Theory*, 57 (10):7036–7056, 2011.
[38] M. Rudelson and R. Vershynin. The smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics*, pages 1707 – 1739, 2009.
[39] S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan. Stochastic Convex Optimization. In *COLT*, 2009.
[40] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML (1)*, pages 71–79, 2013.
[41] A. Smith and A. Thakurta. Differentially private model selection via stability arguments and the robustness of the lasso. *J Mach Learn Res Proc Track*, 30:819–850, 2013.
[42] S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *GlobalSIP*, pages 245–248. IEEE, 2013.
[43] K. Talwar, A. Thakurta, and L. Zhang. Nearly optimal private lasso. In *NIPS*, 2015.
[44] A. G. Thakurta and A. Smith. (nearly) optimal algorithms for private online learning in full-information and bandit settings. In *NIPS*, pages 2733–2741, 2013.
[45] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv:1011.3027*, 2010.
[46] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.
[47] O. Williams and F. McSherry. Probabilistic inference and differential privacy. In *NIPS*, pages 2451–2459, 2010.
[48] Y. Yang and A. Barron. Information-theoretic determination of minimax rates of convergence. *Annals of Statistics*, pages 1564–1599, 1999.

APPENDIX A.
LOCAL RANDOMIZER FOR ϵ-LDP

Randomizer of Duchi *et al.* [14]. On input $\mathbf{x} \in \mathbb{R}^p$, the randomizer $\mathcal{R}_\varepsilon(\mathbf{x})$ does the following. It first sets $\widetilde{\mathbf{x}} =$

$b\mathbf{x}/\|\mathbf{x}\|_2$, where $b \in \{-1, 1\}$ Bernoulli random variable $\mathsf{Ber}(1/2 + \|\mathbf{x}\|/2)$. We then sample $T \sim \mathsf{Ber}(e^\varepsilon/(e^\varepsilon + 1))$ and outputs $\mathcal{R}_\varepsilon(\mathbf{x})$, where

$$\mathcal{R}_\varepsilon(\mathbf{x}) := \begin{cases} \mathsf{Uni}(\mathbf{u} \in \mathbb{R}^p : \langle \mathbf{u}, \widetilde{\mathbf{x}} \rangle > 0) & \text{if } T = 1 \\ \mathsf{Uni}(\mathbf{u} \in \mathbb{R}^p : \langle \mathbf{u}, \widetilde{\mathbf{x}} \rangle \leq 0) & \text{if } T = 0 \end{cases}.$$

Let $\mathbf{y}$ be any unit vector independent of the input to the randomizer $\mathbf{x}$. An alternate way to see the randomizer $\mathcal{R}_\varepsilon(\mathbf{x})$ is that the output of $\mathcal{R}_\varepsilon(\cdot)$ is such that $\langle \mathbf{y}, \mathcal{R}_\varepsilon(\mathbf{x}) \rangle = \langle \mathbf{y}, c_{\varepsilon,d} B_{\mathbf{u},\mathbf{x}} \mathbf{u} \rangle$ for $c_{\varepsilon,p} = O(\sqrt{p}/\varepsilon)$ and $B_{\mathbf{u},\mathbf{x}}$ is ± 1 random variable chosen such that $\mathbb{E}[\mathcal{R}_\varepsilon(\mathbf{x})] = \mathbf{x}$. In the following we prove Theorem 17, the utility guarantee we desire from our NOISY-GRADIENT-ORACLE.

Proof of Theorem 17: Duchi *et al.* [14, Section V.C] proved that $\mathcal{R}_\varepsilon(\mathbf{x})$ is an unbiased estimator, i.e., $\mathbb{E}\left[\mathsf{Grad}_{\mathsf{noisy}}(t)\right] = \nabla L(\theta_{\mathsf{current}})$; therefore, the first part follows. In order to show the second and the third part of Theorem 17 , we show that the randomizer has subgaussian tail (see Definition 21 and Theorem 22 below).

Definition 21. *A real-valued random variable X is* subgaussian *if it has the property that there is some $c > 0$ such that for every $t \in \mathbb{R}$ one has $\mathbb{E}[e^{tX}] \leq e^{c^2 t^2/2}$. Moreover, $\mathsf{Var}(X) = c^2$.*

Theorem 22. *Given a vector $\boldsymbol{x} \in \mathbb{R}^p$, the randomizer of Duchi et al. [14] defined above is a subgaussian random vector with variance $O(p/\varepsilon^2)$.*

Proof: In the following, we use the notation $M_X(t) = \mathbb{E}[e^{tX}]$ to denote the moment generating function of the random variable X. Let $A_{\mathbf{v},\mathbf{x}} = \langle \mathbf{v}, c_{\varepsilon,d} B_{\mathbf{u},\mathbf{x}} \mathbf{u} \rangle$. In order to prove that the randomizer of Duchi *et al.* [14] outputs a subgaussian random vector, it suffices to prove that $A_{\mathbf{v},\mathbf{x}}$ is subgaussian.

$$\begin{aligned} M_{A_{\mathbf{v},\mathbf{x}}}(t) &= \mathbb{E}[e^{tA_{\mathbf{v},\mathbf{x}}}] = \mathbb{E}[e^{tc_{\varepsilon,p}\langle \mathbf{v},\mathbf{u}\rangle} + e^{-tc_{\varepsilon,p}\langle \mathbf{v},\mathbf{u}\rangle}] \\ &= M_{\langle \mathbf{v},\mathbf{u}\rangle}(c_{\varepsilon,p}t) + M_{\langle \mathbf{v},\mathbf{u}\rangle}(-c_{\varepsilon,p}t) \\ &= M_{\langle \mathbf{v},\mathbf{u}\rangle}(c_{\varepsilon,p}t) \leq 2e^{c_{\varepsilon,p}^2 t^2/2} = 2e^{t^2 p/\varepsilon^2}. \end{aligned}$$

Using Definition 21 completes the proof. ∎

For the second part, notice that, by definition, the set $\mathcal{S}_t$ (the current batch in Algorithm 6) has $\Theta(n/k)$ entries and every user in the set $\mathcal{S}_t$ randomizes its output independently of the others. This observation, along with with Theorem 22, allows us to bound the variance of the estimator $\mathsf{Grad}_{\mathsf{noisy}}(t)$ in (8) below. Here we use the fact that $\|\nabla \ell(\theta; d)\|_2 \leq 1$ for for all $\theta \in \mathcal{C}$ and $d \in \mathcal{D}$ (where $\mathcal{C}$ is the convex set over which the optimization is performed, and $\mathcal{D}$ is the domain of the data entries).

$$\begin{aligned} \mathbb{E}\left[\left\|\mathsf{Grad}_{\mathsf{noisy}}(t) - \nabla L(\theta_{\mathsf{current}})\right\|_2^2\right] &= (T/n)\mathsf{Var}(\mathcal{R}_\varepsilon(\mathbf{x})) \\ &= O\left(Tp/n\varepsilon^2\right). \end{aligned} \tag{8}$$

To prove the third part of Theorem 17, we provide a tail bound on the the inner product $\langle \mathsf{Grad}_{\mathsf{noisy}}(t), \mathbf{y} \rangle$ for any vector $\mathbf{y}$ s.t. $\|\mathbf{y}\|_2 \leq 1$ and is independent of $\mathcal{R}_\varepsilon(\mathbf{x})$. Now, by standard tail bound for sub-gaussian distribution [45], we have, $\Pr\left[|\langle \mathcal{R}_\varepsilon(\mathbf{x}), \mathbf{y} \rangle| = O\left(\sqrt{T \log(1/\gamma)/\varepsilon^2 n}\right)\right] \geq 1 - \gamma$.This completes the proof of Theorem 17. ∎

APPENDIX B. MISSING PROOFS

A. Proof of Claim 4

Proof: In the net-based argument, one would first construct an $\alpha/2$-net over the convex set of size m and then find the minimum over each of these net points. To make it local-differentially private, we need to use the randomizer of Duchi *et al.* [14]. This would incur a total population error, $O\left(\sqrt{\frac{m \log m \log(1/\gamma)}{\varepsilon^2 n}} + \frac{\alpha}{2}\right)$ with probability $1 - \gamma$. In a p-dimensional space $[0, 1]^p$, we need to have $m = O((2/\alpha)^p)$ net points. Setting this value of m, we get the population error

$$O\left(\sqrt{\left(\frac{2}{\alpha}\right)^p \frac{p \log(2/\alpha) \log(1/\gamma)}{\varepsilon^2 n}} + \frac{\alpha}{2}\right). \tag{9}$$

To minimize equation (9), we need $\alpha^{p+2}\varepsilon^2 n = 2^{p+1} p \log(2/\alpha) \log(1/\gamma)$. In other words, the population error is at most $O\left(\left(\frac{p \log(1/\gamma)}{\varepsilon^2 n}\right)^{1/(p+2)}\right)$ In other words, equation (1) implies that one need $n = \widetilde{\Omega}\left(\frac{p4^p}{\varepsilon^2 \alpha^{p+2}}\right)$ to get $err_{\mathbf{D}}(\theta_{\mathsf{priv}}) \leq \alpha$. ∎

B. Proof of Theorem 19

Proof: We first state the following claim. The claims allows us to work with an easy to handle function $M(\cdot)$.

Claim 23. *Let $\theta^* = \operatorname{argmin}_{\theta \in \mathcal{C}} L(\theta)$, and let $M(\theta) := L(\theta) - L(\theta_0)$, where θ_0 is as in the first step of Figure 7. Then the following holds : (i) $\theta^* = \operatorname{argmin}_{\theta \in \mathcal{C}} M(\theta)$, (ii) $\forall \theta \in \mathcal{C}$, $L(\theta) - L(\theta^*) = M(\theta) - M(\theta^*)$.*

Proof: The first part of the claim follows from the fact that $L(\theta_0)$ is a constant, and subtracting a constant from the objective function does not change the argument of the optimum value. The second part follows from the following straightforward computation: $M(\theta) - M(\theta^*) = L(\theta) - L(\theta_0) - L(\theta^*) + L(\theta_0)$.This completes the proof of Claim 23. ∎

We will use the following result about non-private cutting plane method.

Lemma 24. [10, Theorem 2.1] *Let $\mathcal{C}$ and p be as defined in Theorem 19. Let $\theta^* = \operatorname{argmin}_{\theta\in\mathcal{C}} L(\theta)$ and $\mathcal{C}_t$ be as defined in Figure 7 for iteration $t \in [T]$. Let E be the event that $\theta^* \in \mathcal{C}_t$ for all iterations $t \le T$. Conditioned on the event E, there exists an iteration $\tau \in [T]$, such that $M(\theta_\tau)$ satisfies the following bound.*

$$M(\theta_\tau) - M(\theta^*) \le 2\left(1 - 1/e\right)^{T/p}.$$

We now analyze the algorithm presented in Figure 7. Our first lemma states the condition when the optimal point is not excluded by NOISY-COG.

Lemma 25. *Let $\theta^* = \operatorname{argmin}_{\theta\in\mathcal{C}} L(\theta)$. Let $\mathbf{g}_t = \nabla L_{\mathsf{priv}}(\theta_t) - \nabla L(\theta_t)$ for $0 < t \le T$. If for all $\tau \le t \in [T]$, $\langle \mathbf{g}_t, \theta_t - \theta^*\rangle \ge -\alpha$ and $M(\theta_t) - M(\theta^*) \ge \alpha$, then $\theta^* \in \mathcal{C}_\tau$, where $\mathcal{C}_\tau$ is the convex set included after τ rounds.*

Proof: Since the function $M(\cdot)$ is convex, we have $M(\theta) \ge M(\theta_t) + \langle \nabla M(\theta_t), \theta - \theta_t\rangle$ for all $\theta \in \mathcal{C}$. This implies that $\langle \nabla M(\theta_t), \theta_t - \theta\rangle \ge M(\theta_t) - M(\theta)$

Let α be a parameter that we will fix later. Now, if $M(\theta_t) - M(\theta^*) \ge \alpha$, then we have $\langle \nabla M(\theta_t), \theta_t - \theta^*\rangle \ge M(\theta_t) - M(\theta^*) \ge \alpha$. This further implies that $\langle \nabla M(\theta_t), \theta^* - \theta_t\rangle \le -\alpha$. From the description of NOISY-COG, we exclude all $\theta \in \mathcal{C}$ such that $\langle \nabla M(\theta_t), \theta - \theta_t\rangle \ge \langle \mathbf{g}_t, \theta - \theta_t\rangle$. Combining these observations completes the proof of Lemma 25. ■

Lemma 26. *Let $\theta^* = \operatorname{argmin}_{\theta\in\mathcal{C}} M(\theta)$. Then with probability $1-\gamma$ over the coin tosses of* PRIVATE-COG*, there exists a $t \in [T]$ such that*

$$M(\theta_t) - M(\theta^*) = O\left(\sqrt{T\log(T/\gamma)/n\epsilon^2} + (1-1/e)^{\frac{T}{p}}\right).$$

Proof: From Theorem 17, for all $t \in [T]$ we have with probability at least $1-\gamma$, $|\langle \mathbf{g}_t, \theta_t - \theta^*\rangle| = O\left(\sqrt{T\log(1/\gamma)/n\epsilon^2}\right)$, where $\mathbf{g}_t = \nabla L_{\mathsf{priv}}(\theta_t) - \nabla L(\theta_t)$ for $0 < t \le T$. If we pick $\alpha = O\left(\sqrt{T\log(T/\gamma)/n\epsilon^2}\right)$, then taking union bound over all the T rounds, $\Pr\left[\forall t \in [T], \langle \mathbf{g}_t, \theta^* - \theta_t\rangle \le -\alpha\right] \le \gamma$. Lemma 26 now follows from Lemma 24 and 25. ■

Lemma 26 and Theorem 18 gives Theorem 19. ■

C. Analysis for 1-Lipschitz, β-smooth and Δ-strongly Convex Functions

In the following we provide the analysis of part (4) from Theorem 20.

Proof of Theorem 20 (4): Let $\theta^* = \min_{\theta\in\mathcal{C}} L(\theta)$, $D_t = \theta_t - \theta^*$ and $\mathbf{z}_{t+1} = \theta_t - \eta(\mathcal{R}_\varepsilon(\nabla L(\theta_t)))$. Let $\mathbf{g}_t = (\nabla L_{\mathsf{priv}}(\theta_t) - \nabla L(\theta_t))$ be the error in the estimation in Algorithm PRIVATE-GRADIENT-ORACLE (Figure 6) for $t \in [T]$. First, by the Pythogorean theorem, we have,

$$\|D_{t+1} - D_t\|_2^2 = \|D_t\|_2^2 + \|D_{t+1}\|_2^2 - 2\langle D_t, D_{t+1}\rangle. \tag{10}$$

Using the β-smoothness and Δ-strong convexity, we have the following:

$$\begin{aligned}
L(\theta_{t+1}) - L(\theta^*) &= L(\theta_{t+1}) - L(\theta_t) + L(\theta_t) - L(\theta^*)\\
&\le \langle \nabla L(\theta_t), D_{t+1}\rangle + \frac{\beta}{2}\|\theta_{t+1} - \theta_t\|_2^2 - \frac{\Delta}{2}\|D_t\|_2^2\\
&= \langle \frac{1}{\eta}(\theta_t - \mathbf{z}_{t+1}) - \mathbf{g}_t, \theta_{t+1} - \theta^*\rangle\\
&\quad + \frac{\beta}{2}\|\theta_{t+1} - \theta_t\|_2^2 - \frac{\Delta}{2}\|\theta_t - \theta^*\|_2^2\\
&= \langle \frac{1}{\eta}(\theta_t + \theta_{t+1} - \theta_{t+1} - \mathbf{z}_{t+1}) - \mathbf{g}_t, \theta_{t+1} - \theta^*\rangle\\
&\quad + \frac{\beta}{2}\|\theta_{t+1} - \theta_t\|_2^2 - \frac{\Delta}{2}\|\theta_t - \theta^*\|_2^2\\
&\le \frac{1}{\eta}\langle \theta_t - \theta^*, \theta_{t+1} - \theta^*\rangle - \langle \mathbf{g}_t, \theta_{t+1} - \theta^*\rangle\\
&\quad - \frac{1}{\eta}\|D_{t+1}\|_2^2 + \frac{\beta}{2}\|\theta_{t+1} - \theta_t\|_2^2 - \frac{\Delta}{2}\|D_t\|_2^2.
\end{aligned} \tag{11}$$

Let $w_{t+1} = \theta_t - \eta\nabla L(\theta_t)$. Now,

$$\begin{aligned}
\mathbb{E}[\langle \mathbf{g}_t, \theta_{t+1} - \theta^*\rangle] &= \mathbb{E}[\langle \mathbf{g}_t, \theta_{t+1} - \Pi_{\mathcal{C}}(w_{t+1})\rangle]\\
&\quad + \mathbb{E}[\langle \mathbf{g}_t, \Pi_{\mathcal{C}}(w_{t+1}) - \theta^*\rangle]\\
&= \mathbb{E}[\langle \mathbf{g}_t, \theta_{t+1} - \Pi_{\mathcal{C}}(w_{t+1})\rangle]\\
&\le \mathbb{E}[\|\mathbf{g}_t\|_2\|\theta_{t+1} - \Pi_{\mathcal{C}}(w_{t+1})\|_2]\\
&\le \mathbb{E}[\|\mathbf{g}_t\|_2\|\mathbf{z}_{t+1} - w_{t+1})\|_2].
\end{aligned} \tag{12}$$

Substituting equation (10) and equation (12) in equation (11), and then using the bound on $\mathbf{g}_t$ from Theorem 17, for all $1 \le t \le T$, we get

$$\begin{aligned}
&2\mathbb{E}[L(\theta_{t+1}, d) - L(\theta^*, d)]\\
&\le \left(\frac{1}{\eta} - \Delta\right)\mathbb{E}[\|D_t\|_2^2] - \frac{1}{\eta}\mathbb{E}[\|D_{t+1}\|_2^2]\\
&\quad + \left(\beta - \frac{1}{\eta}\right)\mathbb{E}[\|D_t - D_{t+1}\|_2^2] + O\left(Tp/\left(n\varepsilon^2\right)\right).\\
&\le \left(\frac{1}{\eta} - \Delta\right)\mathbb{E}[\|D_t\|_2^2] - \frac{1}{\eta}\mathbb{E}[\|D_{t+1}\|_2^2] + O\left(Tp/\left(n\varepsilon^2\right)\right).
\end{aligned} \tag{13}$$

Since $\mathbb{E}[L(\theta_{t+1}) - L(\theta^*)] \ge 0$, equation (13) implies that

$$\mathbb{E}[\|D_{t+1}\|_2^2] \le (1 - \eta\Delta)\mathbb{E}[\|D_t\|_2^2] + \eta \cdot O\left(Tp/\left(n\varepsilon^2\right)\right). \tag{14}$$

Let $\|\theta_0 - \theta^*\|_2 \le 1$, then using the above equations, we get

$$\mathbb{E}[\|D_T\|_2^2] \le e^{-\eta\Delta T} + \eta\left(\frac{1-e^{-\eta\Delta T}}{1-e^{-\eta\Delta}}\right)\cdot O\left(\frac{Tp\sigma^2}{n\varepsilon^2}\right)$$
$$\le e^{-\eta\Delta T} + \eta\cdot O\left(Tp/\left(n\varepsilon^2\right)\right). \quad (15)$$

Using the fact that the function is β-smooth, i.e., $\mathbb{E}[err_{\mathbf{D}}(\theta_{\mathsf{priv}})] \le \frac{\beta}{2}\mathbb{E}[\|D_T\|_2^2]$ and setting $\eta = \frac{1}{\beta}$, and we have Theorem 20. ∎

Appendix C.
Case Study: Linear Regression

In this section we concentrate on a specific problem of linear regression in the LDP model. We give a simple non-interactive algorithm that outputs an estimate (Theorem 27). We show that the error guarantee achieved by this algorithm is the best one can hope for by showing a matching lower bound (Theorem 31). Our algorithm satisfies a slightly weaker variant of the privacy definition in Definition 1, called (ϵ,δ)-approxmiate differential privacy [16], where the definition of closeness in Definition 1 is modified to $\Pr[Q(d) \in E] \le e^{\varepsilon}\Pr[Q(d') \in E] + \delta$. Consider $\delta \approx 1/n^{\omega(1)}$, where n is the number of data samples.

A. Local-Differentially Private Linear Regression

Let $\mathbf{X}_1,\cdots,\mathbf{X}_n \in \mathbb{R}^p$ be row vectors and $\mathbf{y} \in \mathbb{R}^n$ be a column vector, where $d_i := (\mathbf{X}_i, \mathbf{y}_i) \in \mathbb{R}^{p+1}$ is the data of the user i and $\mathbf{D} := (d_1,\cdots,d_n)$ be the dataset. We present this section in the terms of empirical risk. Using standard results [39], this bound can be converted to population risk with no difference in the asymptotic bound. For linear regression, we know that the loss function has the following closed form expression.

$$\ell(\theta;(\mathbf{X}_i,\mathbf{y}_i)) := (\mathbf{y}_i - \langle\mathbf{X}_i,\theta\rangle)^2, \quad (16)$$

$$\widehat{L}(\theta;\mathbf{D}) := \frac{1}{2n}\sum_{i=1}^{n}\ell(\theta;d_i) = \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\theta\|_2^2. \quad (17)$$

and the minimizer has the form $\theta^* := \operatorname{argmin}_{\theta\in\mathcal{C}}\frac{1}{2n}\|\mathbf{y}-\mathbf{X}\theta\|_2^2$. We show that it is possible to non-interactively compute a θ_{priv} in (ε,δ)-local-differentially private manner. More precisely, we show the following.

Theorem 27. *Let $\mathcal{C} \subseteq \mathbb{R}^p$ be a bounded convex set ($\|\mathcal{C}\|_2 \le 1$) over which the linear regression problem is defined. There is an efficient non-interactive algorithm that solves linear regression problem, stated in equation (17), while preserving (ε,δ)-local differential privacy such that with probability at least $99/100$ over the coin tosses of the algorithm,*

$$err_{\mathbf{D}}(\theta_{\mathsf{priv}}) = O\left(\sqrt{p\log(1/\delta)/\left(n\varepsilon^2\right)}\right).$$

Proof: Note that $\|\mathbf{y}-\mathbf{X}\theta\|_2^2 = \|\mathbf{y}\|_2^2 - 2\langle\mathbf{X}\theta,\mathbf{y}\rangle + \langle\mathbf{X}\theta,\mathbf{X}\theta\rangle$. Since the first term does not depend on θ, this implies that

$$\operatorname*{argmin}_{\theta\in\mathcal{C}}\frac{1}{2n}\|\mathbf{y}-\mathbf{X}\theta\|_2^2 = \operatorname*{argmin}_{\theta\in\mathcal{C}}\frac{1}{2n}\langle\mathbf{X}\theta,\mathbf{X}\theta\rangle - 2\langle\mathbf{X}\theta,\mathbf{y}\rangle.$$

We maintain the following two quantities:

1) Noisy covariance: $\mathbf{Z} := \mathbf{X}^{\mathsf{T}}\mathbf{X} + \mathbf{B}$, where $\mathbf{B} \sim \mathcal{N}(0, 2n\log(1/\delta)/\varepsilon^2)^{p\times p}$; and
2) Noisy terms, $\mathbf{z} := (\sum \mathbf{y}_i\mathbf{X}_i) + \mathbf{b}$, where $\mathbf{b} \sim \mathcal{N}(0, 2n\log(1/\delta)/\varepsilon^2)^{1\times p}$.

This can be achieved in the local model as follows: every user computes $\mathbf{Z}_i := \mathbf{X}_i^{\mathsf{T}}\mathbf{X}_i + \mathbf{B}_i$, where $\mathbf{B}_i \sim \mathcal{N}(0, 2\log(1/\delta)/\varepsilon^2)^{p\times p}$ and $\mathbf{z}_i := \mathbf{y}_i\mathbf{X}_i + \mathbf{b}_i$, where $\mathbf{b}_i \sim \mathcal{N}(0, 2\log(1/\delta)/\varepsilon^2)^{1\times p}$. The user then sends $(\mathbf{Z}_i, \mathbf{z}_i)$ to the server. The server can now compute $\mathbf{Z} = \sum_i \mathbf{Z}_i$ and $\mathbf{z} = \sum \mathbf{z}_i$. This results in the same noisy covariance and noisy term as above because $\mathbf{X}^{\mathsf{T}}\mathbf{X} = \sum_{i=1}^n \mathbf{X}_i^{\mathsf{T}}\mathbf{X}_i$ and the noise matrix $\mathbf{B}_1,\cdots,\mathbf{B}_n$ and the noise vector $\mathbf{b}_1,\cdots,\mathbf{b}_n$ are independent.

Now let us define

$$\widehat{L}_{\mathsf{priv}}(\theta;\mathbf{D}) := \frac{1}{2n}\left(\|\mathbf{y}\|_2^2 - 2\sum\langle\mathbf{y}_i\mathbf{X}_i^{\mathsf{T}} + \mathbf{b},\theta\rangle\right) + \frac{1}{2n}\left(\theta^{\mathsf{T}}(\mathbf{X}^{\mathsf{T}}\mathbf{X}+\mathbf{B})\theta\right)$$

and let the minimizer of $\widehat{L}_{\mathsf{priv}}(\theta;\mathbf{D})$ be θ_{priv}.

From now on, we write $\widehat{L}(\cdot)$ instead of $\widehat{L}(\cdot;\mathbf{D})$ and $\widehat{L}_{\mathsf{priv}}(\cdot)$ instead of $\widehat{L}_{\mathsf{priv}}(\cdot;\mathbf{D})$. We can decompose $\widehat{L}(\theta_{\mathsf{priv}}) - \widehat{L}(\theta^*)$ as following:

$$\begin{aligned}\widehat{L}(\theta_{\mathsf{priv}}) - \widehat{L}(\theta^*) &= \widehat{L}(\theta_{\mathsf{priv}}) - \widehat{L}_{\mathsf{priv}}(\theta_{\mathsf{priv}}) + \widehat{L}_{\mathsf{priv}}(\theta_{\mathsf{priv}}) \\ &\quad - \widehat{L}_{\mathsf{priv}}(\theta^*) + \widehat{L}_{\mathsf{priv}}(\theta^*) - \widehat{L}(\theta^*) \\ &\le \widehat{L}(\theta_{\mathsf{priv}}) - \widehat{L}_{\mathsf{priv}}(\theta_{\mathsf{priv}}) + \widehat{L}_{\mathsf{priv}}(\theta^*) - \widehat{L}(\theta^*) \\ &\le \frac{2\|\mathbf{b}\|_2\|\theta_{\mathsf{priv}}-\theta^*\|_2 + \theta_{\mathsf{priv}}^{\mathsf{T}}\mathbf{B}\theta_{\mathsf{priv}} - (\theta^*)^{\mathsf{T}}\mathbf{B}\theta^*}{2n}.\end{aligned} \quad (18)$$

The first inequality follows from the fact that $\widehat{L}_{\mathsf{priv}}(\theta_{\mathsf{priv}}) - \widehat{L}_{\mathsf{priv}}(\theta^*) \le 0$ (since θ^* is the minimizer) and the second inequality follows from the fact that $\langle\cdot,\cdot\rangle \le \|\cdot\|_2\|\cdot\|_2$. Let $\beta = \theta_{\mathsf{priv}} - \theta^*$. We can further decompose $\theta_{\mathsf{priv}}^{\mathsf{T}}\mathbf{B}\theta_{\mathsf{priv}}$ as follows.

$$\theta_{\mathsf{priv}}^{\mathsf{T}}\mathbf{B}\theta_{\mathsf{priv}} = (\theta^*)^{\mathsf{T}}\mathbf{B}\theta^* + \beta^{\mathsf{T}}\mathbf{B}\theta^* + (\theta^*)^{\mathsf{T}}\mathbf{B}\beta + \beta^{\mathsf{T}}\mathbf{B}\beta \quad (19)$$

Now using equation (19) in equation (18), the bound on the singular values of Gaussian matrices [38] and that $\|\mathcal{C}\|_2 \leq 1$, we get

$$\begin{aligned}\widehat{L}(\theta_{\mathsf{priv}}) - \widehat{L}(\theta^*) &\leq \frac{1}{2n}\left(2\|\mathbf{b}\|_2\|\theta_{\mathsf{priv}} - \theta^*\|_2\right) \\ &\quad + \frac{1}{2n}\left(\lambda_{max}(\mathbf{B})\|\theta_{\mathsf{priv}} - \theta^*\|_2^2\right) \\ &\quad + \frac{1}{2n}\left(2\lambda_{max}(\mathbf{B})\|\theta_{\mathsf{priv}} - \theta^*\|_2\|\mathcal{C}\|_2\right) \\ &= O\left(\sqrt{p\log(1/\delta)/\left(n\varepsilon^2\right)}\right).\end{aligned}$$

This completes the proof ot Theorem 27. ■

B. Lower Bound on Linear Regression

In this section we prove that the error bound achieved in Theorem 27 is optimal.

Minimax rates. We use the framework of minimax rates introduced by Duchi *et al.* [14] to prove the lower bound. Let $\mathbb{S} \subset [0,1]^{|\mathcal{D}|}$ denote the simplex with $|\mathcal{D}|$ corners, each representing one of the possible datasets. Let $\mathbf{P} \in \mathbb{S}$ be a probability distribution over the datasets $\mathcal{D}$. User i data d_i is assumed to be drawn i.i.d. using the distribution $\mathbf{P}$. For every user i, let $\mathcal{R}_i(\cdot)$ be the local randomizer of the user i which is used to generate the report $Z_i = \mathcal{R}_i(d_i) \in \mathcal{Z}$. Each of these randomizers are assumed to use independent random bits. Let Alg be an algorithm that is used to estimate the distribution $\mathbf{P}$ based on $Z_1, \cdots, Z_n$. The minimax rate is defined as follows:

$$\mathsf{MinMax} := \min_{\mathsf{Alg}} \max_{\mathbf{P}} \mathbb{E}[\|\mathsf{Alg}(Z_1, \cdots, Z_n) - \mathbf{P}\|_\infty].$$

Now that we have described our framework to prove the lower bound, we sketch our proof. Our proof uses idea in Bassily and Smith [6]. In order to use their idea, we need to first introduce the notion of an γ-degrading channel. For any $\gamma \in [0,1]$, an γ-degrading channel $W^\gamma : \mathcal{D} \to \mathcal{D}$ is a randomized mapping that is defined as follows: for every $d \in \mathcal{D}$,

$$W^\gamma(d) := \begin{cases} d & \text{with probability } \gamma \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

Let MinMax_γ be the minimax error resulting from the scenario where each user $i \in [n]$ with data $d_i \in \mathcal{D}$ first applies d_i to an independent copy W^γ of an γ-degrading channel, and then apply the output to its (ε, δ) local-differentially private algorithm $\mathcal{R}_i$ that outputs Z_i. That is, MinMax_γ is the minimax error when $\mathcal{R}_i(\cdot)$ is replaced by $\mathcal{R}_i(W_i^\gamma(\cdot))$ for all $i \in [n]$. We first prove the following claim.

Claim 28. *If* $\mathsf{MinMax} \geq 0.01$*, then* $\mathsf{MinMax}_\gamma \geq 0.01\gamma$

Proof: Let $W^\gamma = \gamma\mathcal{D}$. Then $\mathbb{E}_{d\sim W^\gamma}[\ell(\theta; d)] = \gamma\mathbb{E}_{d\sim\mathcal{D}}[\ell(\theta; d)] \geq \frac{\gamma}{100}$. This completes the proof of Claim 28. ■

We now construct a hard instance of input to any LDP algorithm. Let $\mathcal{H}$ be the p-dimensional hypercube that is just contained inside $\mathcal{B}^p$. Consider a codeword which is the subset of the coordinates of $\mathcal{H}$ and has distance $p/8$. Using Gilbert-Varshamov bound [31], we know that the size of the codeword is $N \geq 2^{p/8}$. We pick N distributions $\mathcal{D}_1, \cdots, \mathcal{D}_N$. Each of these distributions are a point distribution such that $y - \langle \mathbf{X}, \theta\rangle = 1 \Leftrightarrow \theta_i^* = \mathbf{X}$. Our result follows from the following claims:

Claim 29. *Let* θ_{priv} *be the output of an algorithm that solves the linear regression problem. Then the mutual information between* θ_{priv} *and the distribution of the data points is* $I(\theta_{\mathsf{priv}}; \mathcal{D}) \leq n\varepsilon^2\gamma^2$*, where* γ *is the parameter for the degrading channel.*

Proof: Using the chain rule of mutual information and Duchi *et al.* [14], we have the following:

$$I(\theta_{\mathsf{priv}}; d) \leq I(Z_1, \cdots, Z_n; d) \leq 4(e^\varepsilon - 1)^2 n \leq 16\varepsilon^2 n$$

The claim now follows from Bassily and Smith [6]. ■

Claim 30. *For any* $\mathcal{D}_i$*, let* $S^i_{1/100} := \left\{\theta \in \mathcal{B}^p : \widehat{L}(\theta; D) \leq 1/100\right\}$*, where* $D \sim \mathcal{D}_i$*, then* $\forall i, j, i \neq j,\ S^i_{1/100} \cap S^j_{1/100} = 0$.

Proof: Let i and j be adjacent node on the hypercube $\mathcal{H}$. Let Q be the intersection (of the extension) of the sets $S^i_{1/100}$ and $S^i_{1/100}$. Let C be the center of the ball $\mathcal{B}^p$. First note that the boundary of $S^i_{1/100}$ and $S^j_{1/100}$ are hyperplane whose normal intersects at C. Let ζ be the angle made by the normal of the hyperplane corresponding to $S^i_{1/100}$ and the line passing through C and Q. First of all, $\sin\zeta = \sqrt{1/32}$, and $\cos\zeta = \sqrt{31/32}$.

By symmetry, the line CQ cuts the line between node-i and node-j into half. Therefore, the length of CQ is $\frac{9}{10\sqrt{31/32}} \geq 1$. This implies that Q lies outside $\mathcal{B}^p$. ■

We now complete the proof of the main result.

Theorem 31. *Any* ε*-local differential private algorithm for linear regression incurs an error* $\Omega(\sqrt{p/n\varepsilon^2})$.

Proof: By Fano's inequality [48] and Claim 29,

$$\mathsf{Pr}\left[\theta^i_{\mathsf{priv}} \notin S^i_{1/100}\right] \geq 1 - \frac{I(\theta_{\mathsf{priv}}; D)}{\log N} \geq 1 - \frac{8cn\varepsilon^2\gamma^2}{p}.$$

Then Claim 30 gives us that $\mathsf{MinMax} \geq 10^{-2}\left(1 - 8cn\varepsilon^2\gamma^2/p\right)$. For $\gamma = O(\sqrt{\frac{p}{n\varepsilon^2}})$, we get $\mathsf{MinMax} \geq 1/100$. By Claim 28, we have the overall error $\Omega\left(\sqrt{\frac{p}{n\varepsilon^2}}\right)$. ■