Private Incremental Regression

Shiva Kasiviswanathan
Samsung Research America
kasivisw@gmail.com

Kobbi Nissim*
Ben-Gurion University and Harvard University
kobbi@cs.bgu.ac.il

Hongxia Jin
Samsung Research America
hongxia.jin@samsung.com

ABSTRACT

Data is continuously generated by modern data sources, and a recent challenge in machine learning has been to develop techniques that perform well in an incremental (streaming) setting. A variety of offline machine learning tasks are known to be feasible under differential privacy, where generic construction exist that, given a large enough input sample, perform tasks such as PAC learning, Empirical Risk Minimization (ERM), regression, etc. In this paper, we investigate the problem of private machine learning, where as common in practice, the data is not given at once, but rather arrives incrementally over time.

We introduce the problems of *private incremental ERM* and *private incremental regression* where the general goal is to always maintain a good empirical risk minimizer for the history observed under differential privacy. Our first contribution is a generic transformation of private batch ERM mechanisms into private incremental ERM mechanisms, based on a simple idea of invoking the private batch ERM procedure at some regular time intervals. We take this construction as a baseline for comparison. We then provide two mechanisms for the private incremental regression problem. Our first mechanism is based on privately constructing a noisy incremental gradient function, which is then used in a modified projected gradient procedure at every timestep. This mechanism has an excess empirical risk of $\approx \sqrt{d}$, where d is the dimensionality of the data. While from the results of Bassily *et al.* [2] this bound is tight in the worst-case, we show that certain geometric properties of the input and constraint set can be used to derive significantly better results for certain interesting regression problems. Our second mechanism which achieves this is based on the idea of projecting the data to a lower dimensional space using random projections, and then adding privacy noise in this low dimensional space. The mechanism overcomes the issues of adaptivity inherent with the use of random projections in online streams, and uses recent developments in high-dimensional estimation to achieve an excess empirical risk bound of $\approx T^{1/3}W^{2/3}$, where T is the length of the stream and W is the sum of the Gaussian widths of the input domain and the constraint set that we optimize over.

1. Introduction

Most modern data such as documents, images, social media data, sensor data, and mobile data naturally arrive in a streaming fashion, giving rise to the challenge of incremental machine learning, where the goal is build and publish a model that evolves as data arrives. Learning algorithms are frequently run on sensitive data, such as location information in a mobile setting, and results of such analyses could leak sensitive information. For example, Kasiviswanathan *et al.* [32] show how the results of many convex ERM problems can be combined to carry out reconstruction attacks in the spirit of Dinur and Nissim [11]. Given this, a natural direction to explore, is whether, we can carry out incremental machine learning, without leaking any significant information about individual entries in the data. For example, a data scientist, might want to continuously update the regression parameter of a linear model built on a stream of user profile data gathered from an ongoing survey, but these updates should not reveal whether any one person participated in the survey or not.

Differential privacy [15] is a rigorous notion of privacy that is now widely studied in computer science and statistics. Intuitively, differential privacy requires that datasets differing in only one entry induce similar distributions on the output of a (randomized) algorithm. One of the strengths of differential privacy comes from the large variety of machine learning tasks that it allows. Good generic constructions exist for tasks such as PAC learning [4, 31] and Empirical Risk Minimization [41, 34, 25, 26, 48, 27, 2, 12, 51, 47]. These constructions, however, are typically focused on the batch (offline) setting, where information is first collected and then analyzed. Considering an incremental setting, it is natural to ask whether these tasks can still be performed with high accuracy, under differential privacy.

In this paper, we introduce the problem of private incremental empirical risk minimization (ERM) and provide algorithms for this new setting. Our particular focus will be on the problem of private incremental linear regression. Let us start with a description of the traditional batch convex ERM framework. Given a dataset and a constraint space $\mathcal{C}$, the goal in ERM is to pick a $\theta \in \mathcal{C}$ that minimizes the *empirical error (risk)*. Formally, given n datapoints $\mathbf{z}_1, \ldots, \mathbf{z}_n$ from some domain $\mathcal{Z}$, and a closed, convex set $\mathcal{C} \subseteq \mathbb{R}^d$, consider the optimization problem:

$$\min_{\theta \in \mathcal{C}} \mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_n) \text{ where } \mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_n) = \sum_{i=1}^{n} \jmath(\theta; \mathbf{z}_i). \tag{1}$$

The loss function $\mathcal{J} : \mathcal{C} \times \mathcal{Z}^n \to \mathbb{R}$ measures the fit of $\theta \in \mathcal{C}$ to the given data $\mathbf{z}_1, \ldots, \mathbf{z}_n$, the function $\jmath : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}$ is

*Supported by a grant from the Sloan Foundation, a Simons Investigator grant to Salil Vadhan, and NSF grant CNS-1237235.

the loss associated with a single datapoint and is assumed to be convex in the first parameter θ for every $\mathbf{z} \in \mathcal{Z}$. It is common to assume that the loss function has certain properties, e.g., positive valued. The *M-estimator* (true empirical risk minimizer) $\hat{\theta}$ associated with a given a function $\mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_n) \geq 0$ is defined as:[1]

$$\hat{\theta} \in \operatorname{argmin}_{\theta \in \mathcal{C}} \mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_n) = \operatorname{argmin}_{\theta \in \mathcal{C}} \sum_{i=1}^{n} \jmath(\theta; \mathbf{z}_i).$$

This type of program captures a variety of empirical risk minimization (ERM) problems, e.g., the MLE (Maximum Likelihood Estimators) for linear regression is captured by setting $\jmath(\theta; \mathbf{z}) = (y - \langle \mathbf{x}, \theta \rangle)^2$ in (1), where $\mathbf{z} = (\mathbf{x}, y)$ for $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Similarly, the MLE for logistic regression is captured by setting $\jmath(\theta; \mathbf{z}) = \ln(1 + \exp(-y\langle \mathbf{x}, \theta \rangle))$. Another common example is the support vector machine (SVM), where $\jmath(\theta; \mathbf{z}) = \text{hinge}(y\langle \mathbf{x}, \theta \rangle)$, where $\text{hinge}(a) = 1 - a$ if $a \leq 1$ and 0 otherwise.

The main focus of this paper will be on a particularly important ERM problem of linear regression. Linear regression is a popular statistical technique that is commonly used to model the relationship between the outcome (label) and the explanatory variables (covariates). Informally, in a linear regression, given n covariate-response pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, we wish to find a (regression) parameter vector $\hat{\theta}$ such that $\langle \mathbf{x}_i, \hat{\theta} \rangle \approx y_i$ for most i's. Specifically, let $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$ denote a vector of the responses, and let $X \in \mathbb{R}^{n \times d}$ be the design matrix where $\mathbf{x}_i^\top$ is the ith row. Consider the linear model: $\mathbf{y} = X\theta^\star + \mathbf{w}$, where $\mathbf{w}$ is the noise vector, the goal in linear regression is to estimate the unknown regression vector $\theta^\star$. Assuming that the noise vector $\mathbf{w} = (w_1, \ldots, w_n)$ follows a (sub)Gaussian distribution, estimating the vector $\theta^\star$ amounts to solving the "ordinary least squares" (OLS) problem:

$$\hat{\theta} \in \operatorname{argmin}_{\theta} \sum_{i=1}^{n} (y_i - \langle \mathbf{x}_i, \theta \rangle)^2.$$

Typically, for additional guarantees such as sparsity, stability, etc., θ is constrained to be from a convex set $\mathcal{C} \subset \mathbb{R}^d$. Popular choices of $\mathcal{C}$ include the L_1-ball (referred to as *Lasso* regression) and L_2-ball (referred to as *Ridge* regression). In an incremental setting, the $(\mathbf{x}_i, y_i)$'s arrive over time, and the goal in incremental linear regression is to maintain over time (an estimate of) the regression parameter. We provide a more detailed background on linear regression in Appendix A.1.

Incremental Setting. In an incremental setting the data arrives in a stream at discrete time intervals. The incremental setting is a variant of the traditional batch setting capturing the fact that modern data is rarely collected at one single time and more commonly data gathering and analysis may be interleaved. An incremental algorithm is modeled as follows: at each timestep the algorithm receives an input from the stream, computes, and produces outputs. Typically, constraints are placed on the algorithm in terms of some computational resource (such as memory, computation time) availability. In this paper, the challenge in this setting comes from the differential privacy constraint because frequent releases about the data can lead to privacy loss (see, e.g., [11, 32]).

We focus on the incremental ERM problem. In this problem setting, the data $\mathbf{z}_1, \ldots, \mathbf{z}_T \in \mathcal{Z}$ arrives one point at each timestep $t \in \{1, \ldots, T\}$. The goal of an incremental ERM algorithm is to release at each timestep t, an estimator that minimizes the risk measured on the data $\mathbf{z}_1, \ldots, \mathbf{z}_t$. In more concrete terms, the goal is to output $\hat{\theta}_t$ at every $t \in \{1, \ldots, T\}$, where:

$$\text{Incremental ERM:} \quad \hat{\theta}_t \in \operatorname{argmin}_{\theta \in \mathcal{C}} \sum_{i=1}^{t} \jmath(\theta; \mathbf{z}_i).$$

The goal of *private incremental ERM* algorithm, is to (differential) privately estimate $\hat{\theta}_t$ at every $t \in \{1, \ldots, T\}$.[2] In this paper, we develop the first private incremental ERM algorithms.

There is a long line of work in designing differentially private algorithms for empirical risk minimization problems in the batch setting [41, 34, 26, 48, 27, 2, 12, 51, 47]. A naive approach to transform the existing batch techniques to work in the incremental model is by using them to recompute the outcome after each datapoint's arrival. However, for achieving an overall fixed level of differential privacy, this would result in an unsatisfactory loss in terms of utility. Precise statements can be obtained using composition properties of differential privacy (as in Theorem A.4), but informally if a differentially private algorithm is executed T times on the same input, then the privacy parameter (ϵ in Definition 4) degrades by a factor of $\approx \sqrt{T}$, and this affects the overall utility of this approach as the utility bounds typically depend inversely on the privacy parameter. Therefore, the above naive approach will suffer an additional multiplicative factor of $\approx \sqrt{T}$ over the risk bounds obtained for the batch case. Our goal is to obtain risk bounds in the incremental setting that are comparable to those in the batch setting, i.e., bounds that do not suffer from this additional penalty of $\sqrt{T}$.

Generalization. For the kind of problems we consider in this paper, if the process generating the data satisfies the conditions for *generalization* (e.g., if the data stream contains datapoints where each datapoint is sampled independent and identically from an unknown distribution), the incremental ERM would converge to the true risk on the distribution (via uniform convergence and other ideas, refer to [53, 43, 2] for more details). In this case, the model learned in an incremental fashion will have a good predictive accuracy on unseen arriving data. If however, the data does not satisfy these conditions, then $\hat{\theta}_t$ can be viewed as a "summarizer" for the data seen so far. Generating these summaries could also be useful in many applications, e.g., the regression parameter can be used to explain the associations between the outcome (y_i's) and the covariates ($\mathbf{x}_i$'s).

[1] This formulation also captures *regularized* ERM, in which an additional convex function $R(\theta)$ is added to the loss function to penalize certain type of solutions, e.g., "penalize" for the "complexity" of θ. The loss function $\mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_n)$ then equals $\sum_{i=1}^{n} \jmath(\theta; \mathbf{z}_i) + R(\theta)$, which is same as replacing $\jmath(\theta; \mathbf{z}_i)$ by $\jmath(\theta; \mathbf{z}_i) + R(\theta)/n$ in (1).

[2] A point to note in the above description, while we have the privacy constraint, we have placed no computational constraints on the algorithm. In particular, the above description allows also those algorithms that at time t use the whole input history $\mathbf{z}_1, \ldots, \mathbf{z}_t$. However, as we will discuss in Sections 4 and 5, our proposed approaches for private incremental regression are also efficient in terms of their resource requirements.

These associations are regularly used in domains such as public health, social sciences, biological studies, to understand whether specific variables are important (e.g., statistically significant) or unimportant predictors of the outcome. In practice, these associations would need to be constantly re-evaluated over time as new data arrives.

Comparison to Online Learning. Online learning (or sequential prediction) is another well-studied setting for learning when the data arrives in a sequential order. There are differences between the goals of incremental and online learning. In online ERM learning, the aim is to provide an estimator $\tilde{\theta}_t$ that can be used for future prediction. More concretely, at time t, an online learner, chooses $\tilde{\theta}_t$ and then the adversary picks $\mathbf{z}_t$ and the learner suffers loss of $\jmath(\tilde{\theta}_t; \mathbf{z}_t)$. Online learners try to minimize *regret* defined as the difference between the cumulative loss of the learner and the cumulative loss of the best fixed decision at the end of T rounds [42]. In an incremental setting, the algorithm gets to observe $\mathbf{z}_t$ before committing to the estimator, and the goal is to ensure at each timestep t, the algorithm maintains a *single* estimator that minimizes the risk on the history. There are strong lower bounds on the achievable regret for online ERM learning. In particular, even under the differential privacy constraint, the excess risk bounds on incremental learning that we obtain here have better dependence on T (stream length) than the regret lower bounds for online ERM. Incremental learning model should be viewed as a variant of batch (offline) learning model, where the data arrives over time, and the goal is to output intermediate results.

1.1 Our Results

Before stating our results, let us define how we measure success of our algorithms. As is standard in ERM, we measure the quality of our algorithm by the worst-case (over inputs) excess (empirical) risk (defined as the difference from the minimum possible risk over a function class). In an incremental setting, we want this excess risk to be always small for any sequence of inputs. The following definition captures this requirement. All our algorithms are randomized, and they take a confidence parameter β and produce bounds that hold with probability at least $1 - \beta$.

Definition 1. *A (randomized) incremental algorithm is an* (α, β)-estimator *for loss function* $\mathcal{J}$*, if with probability at least* $1 - \beta$ *over the coin flips of the algorithm, for each* $t \in \{1, \ldots, T\}$*, after processing a prefix of the stream of length* t*, it generates an output* $\theta_t \in \mathcal{C}$ *that satisfies the following bound on excess (empirical) risk:*

$$\mathcal{J}(\theta_t; \mathbf{z}_1, \ldots, \mathbf{z}_t) - \mathcal{J}(\hat{\theta}_t; \mathbf{z}_1, \ldots, \mathbf{z}_t) \leq \alpha,$$

where $\hat{\theta}_t \in argmin_{\theta \in \mathcal{C}} \mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_t)$ *is the true empirical risk minimizer. Here,* α *is referred to as the bound on the excess risk.*

In this paper, we propose incremental ERM algorithms that provide a differential privacy guarantee on data streams. Informally, differential privacy requires that the output of a data analysis mechanism is not overly affected by any single entry in the input dataset. In the case of incremental setting, we insist that this guarantee holds at each timestep for all outputs produced up to that timestep (precise definition in Section 2). This would imply that, an adversary is unable to determine whether a particular datapoint was present or not in the input stream by observing the output of the algorithm over time. Two parameters, ϵ and δ control the level of privacy. Very roughly, ϵ is an upper bound on the amount of influence an individual datapoint has on the outcome and δ is the probability that this bound fails to hold (for a precise interpretation, refer to [30]), so the definition becomes more stringent as $\epsilon, \delta \to 0$. Therefore, while parameters α, β measure the accuracy of an incremental algorithm, the parameters ϵ, δ represent its privacy risk. Our private incremental algorithms take ϵ, δ as parameters and satisfies: a) the differential privacy constraint with parameters ϵ, δ and b) (α, β)-estimator property (Definition 1) for every $\beta > 0$ and some α (parameter that the algorithm tries to minimize).

There is a trivial differentially private incremental ERM algorithm that completely ignores the input and outputs at every $t \in \{1, \ldots, T\}$, any $\theta \in \mathcal{C}$ (this scheme is private as the output is always independent of the input). The excess risk of this algorithm is at most $2TL\|\mathcal{C}\|$;[3] where L is the Lipschitz constant of $\jmath$ (Definition 8) and $\|\mathcal{C}\|$ is the maximum attained norm in the convex set $\mathcal{C}$ (Definition 2). All bounds presented in this paper, as also true for all other existing results in the private ERM literature, are only interesting in the regime where they are less than this trivial bound.

For the purposes of this section, we make some simplifying assumptions and omit[4] dependence on all but key variables (dimension d and stream length T). Slightly more detailed bounds are stated in Table 1. All our algorithms run in time polynomial in d and T (exact bounds depend on the time needed for Euclidean projection onto the constraint set which will differ based on the constraint set). Additionally, our private incremental regression algorithms, which utilize the *Tree Mechanism* of [16, 7] have space requirement whose dependence on the stream length T is only logarithmic.

(1) A Generic Transformation. A natural first question is whether *non-trivial* private ERM algorithms exist in general for the incremental setting. Our first contribution is to answer this question in the affirmative – we present a simple generic transformation of private batch ERM algorithms into private incremental ERM algorithms. The construction idea is simple: rather than invoking the batch ERM algorithm every timestep, the batch ERM algorithm is invoked every τ timesteps, where τ is chosen to balance the excess risk factor coming from the stale risk minimizer (because of inaction) with the excess risk factor coming from the increased privacy noise due to reuse of the data. Using this idea along with recent results of Bassily *et al.* [2] for private batch ERM, we obtain an excess risk bound (α in Definition 1) of $\tilde{O}(\min\{(Td)^{1/3}, T\})$.[5] Using this same framework, we also show that when the loss function is strongly convex (Definition 9) the excess risk bound can be improved to $\tilde{O}(\min\{\sqrt{d}, T\})$.

[3] This follows as in each timestep $t \in \{1, \ldots, T\}$, for any $\theta \in \mathcal{C}$, $\mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_t) - \mathcal{J}(\hat{\theta}_t; \mathbf{z}_1, \ldots, \mathbf{z}_t) \leq tL\|\theta - \hat{\theta}_t\| \leq tL(\|\theta\| + \|\hat{\theta}_t\|) \leq 2tL\|\mathcal{C}\| \leq 2TL\|\mathcal{C}\|$.

[4] This includes parameters $\epsilon, \delta, \beta, \|\mathcal{C}\|$, and the Lipschitz constant of the loss function.

[5] For simplicity of exposition, the $\tilde{O}(\cdot)$ notation hides factors polynomial in $\log T$ and $\log d$.

	Incremental ERM Problem Objective	Bound on the Excess (Empirical) Risk under (ϵ, δ)-Differential Privacy (α in Definition 1)
1	Convex Function (using a generic transformation)	$\frac{(Td)^{\frac{1}{3}} \log^{\frac{5}{2}}(1/\delta)}{\epsilon^{2/3}}$
2	Strongly Convex Function (using a generic transformation)	$\frac{\sqrt{d}\log^4(1/\delta)}{\nu^{1/2}\epsilon}$
3	Linear Regression	Mech 1: $\frac{\sqrt{d}\sqrt{\log(1/\delta)}}{\epsilon}$ Mech 2: $\frac{T^{\frac{1}{3}}W^{\frac{2}{3}}\sqrt{\log(1/\delta)}}{\epsilon} + T^{\frac{1}{6}}W^{\frac{1}{3}}\sqrt{\text{OPT}} + T^{\frac{1}{4}}W^{\frac{1}{2}}\sqrt[4]{\text{OPT}}$ (where $W = w(\mathcal{X}) + w(\mathcal{C})$ and OPT is the minimum empirical risk at timestep T)

Table 1: Summary of our results. The stream length is T and d is the dimensionality of the data (number of covariates in the case of regression). The bounds are stated for the setting where both the Lipschitz constant of the function and $\|\mathcal{C}\|$ are scaled to 1. The bounds ignore polylog factors in d and T, and the value in the table gives the bound when it is below T, i.e., the bounds should be read as $\min\{T, \cdot\}$. ν is the strong convexity parameter. For the regression problem, $\mathcal{X}$ is the domain from which the inputs (covariates) are drawn and $\mathcal{C}$ is the constraint space. OPT stands for the minimum empirical risk at time T. The exact results are provided in Theorems 3.1, 4.2, and 5.7, respectively.

A follow-up question is: how much worse is this private incremental ERM risk bound compared to best known private batch ERM risk bound (with a sample size of T datapoints)? In the batch setting, the results of Bassily *et al.* [2] establish that for any convex ERM problem, it is possible to achieve an excess risk bound of $\tilde{O}(\min\{\sqrt{d}, T\})$ (which is also tight in general). Therefore, our transformation from the batch to incremental setting, causes the excess risk to increase by at most a factor of $\approx \max\{T^{1/3}/d^{1/6}, 1\}$. Note that even for a low-dimensional setting (small d case), the factor increase in excess risk of $\approx T^{1/3}$ (as now $\max\{T^{1/3}/d^{1/6}, 1\} \approx T^{1/3}$) is much smaller than the factor increase of $\approx T^{1/2}$ for the earlier described naive approach based on using a private batch ERM algorithm at every timestep. The situation only improves for larger dimensional data.

(2) Private Incremental Linear Regression Using Tree Mechanism. We show that we can improve the generic construction from **(1)** for the important problem of linear regression. We do so by introducing the notion of a *private gradient function* (Definition 5) that allows differentially private evaluation of the gradient at any $\theta \in \mathcal{C}$.[6] More formally, for any $\theta \in \mathcal{C}$, a private gradient function allows differentially private evaluation of the gradient at θ to within a small error (with high probability). Since the data arrives continually, our algorithm utilizes the *Tree Mechanism* of [16, 7] to continually update the private gradient function. Now given access to a private gradient function, we can use any first-order convex optimization technique (such as projected gradient descent) to privately estimate the regression parameter. The idea is, since these optimization techniques operate by iteratively taking steps in the direction opposite to the gradient evaluated at the current point, we can use a private gradient function for evaluating all the required gradients. Using this, we design a private regression algorithm for the incremental setting, that achieves an excess risk bound of $\tilde{O}(\min\{\sqrt{d}, T\})$. It is easy to observe, for private incremental linear regression, this result improves the bounds from the generic construction above for any choice of d, T (as $\min\{\sqrt{d}, T\} \leq \min\{(Td)^{1/3}, T\}$).[7]

Ignoring polylog factors, this worst-case bound matches the lower bounds on the excess risk for squared-loss in the batch case [2], implying that this bound cannot be improved in general (an excess risk upper bound for a problem in the incremental setting trivially holds for the same problem in the batch setting).

(3) Private Incremental Linear Regression: Going Beyond Worst-Case. The noise added in our previous solution **(2)** grows approximately as the square root of the input dimension (for sufficiently large T), which could be prohibitive for a high-dimensional input. While in a worst-case setup this, as we discussed above, seems unavoidable, we investigate whether certain geometric properties of the input/output space could be used to obtain better results in certain interesting scenarios.

A natural strategy for reducing the dependence of the excess risk bounds on d is to use dimensionality reduction techniques such as the Johnson-Lindenstrauss Transform (JLT): The server performing the computation can choose a random projection matrix $\Phi \in \mathbb{R}^{m \times d}$ which is then used for projecting all the $\mathbf{x}_i$'s (covariates) onto a lower-dimensional space. The advantage being that, using the techniques from **(2)**, one could privately minimize the excess risk in the projected subspace, and in doing so, the dependence on the dimension in excess risk is reduced to $\approx \sqrt{m}$ (from $\approx \sqrt{d}$).[8] However, there are two significant challenges in implementing this idea for incremental regression, both of which we overcome using geometric ideas.

The first one being that this only solves the problem in the projected subspace, whereas our goal is to produce an estimate to true empirical risk minimizer. To achieve this we would need to "lift" back the solution from the projected subspace to the original space. We do so by using

[6] Intuitively, this would imply that for any $\theta \in \mathcal{C}$, the output of a private gradient function cannot be used to distinguish two streams that are almost the same (differential privacy guarantee).

[7] A linear regression instance typically does not satisfy strong convexity requirements.

[8] A point to note that this use of random projections for linear regression is different from its typical use in prior work [57], where they are used not for reducing dimensionality but rather for reducing the number of samples used in the regression computation, and hence improving in running time.

recent developments in the problem of high dimensional estimation with constraints [54]. The *Gaussian width* of the constraint space $\mathcal{C}$ plays an important role in this analysis. Gaussian width is a well-studied quantity in convex geometry that captures the geometric complexity of any set of vectors.[9] The rough idea here being that a good estimation (lifting) can be done from few observations (small m) as long as the constraint set $\mathcal{C}$ has small Gaussian width. Many popular sets have low Gaussian width, e.g., the width of L_1-ball (B_1^d is $\Theta(\sqrt{\log d})$, and that of set of all sparse vectors in $\mathbb{R}^d$ with at most k non-zero entries is $\Theta(\sqrt{k \log(d/k)})$.

The second challenge comes from the incremental nature of input generation, because it allows for generation of $\mathbf{x}_i$'s after Φ is *fixed*. This is an issue because the guarantees of random projections, such as JL transform, only hold if the inputs on which it is applied are chosen before choosing the transformation. For example, given a random projection matrix $\Phi \in \mathbb{R}^{m \times d}$ with $m \ll d$, it is simple to generate $\mathbf{x}$ such that the norm on $\mathbf{x}$ is substantially different from the norm of $\Phi\mathbf{x}$.[10] Again to deal with these kind of adaptive choice of inputs, we rely on the geometric properties of problem. In particular, we use Gordon's theorem that states that one can embed a set of points S on a unit sphere into a (much) lower-dimensional space $\mathbb{R}^m$ using a Gaussian random matrix[11] Φ such that, $\sup_{\mathbf{a} \in S} |\|\Phi\mathbf{a}\|^2 - \|\mathbf{a}\|^2|$ is small (with high probability), provided m is at least the square of the Gaussian width of S [21]. In a sense, $w(S)^2$ can be thought as the "effective dimension" of S, so projecting the data onto $m \approx w(S)^2$ dimensional space suffices for guaranteeing the above condition.

Using the above geometric ideas, and the *Tree Mechanism* to incrementally construct the private gradient function (as in **(2)**), we present our second private algorithm for incremental regression, with an

$$\tilde{O}(\min\{T^{\frac{1}{3}} W^{\frac{2}{3}} + T^{\frac{1}{6}} W^{\frac{1}{3}} \sqrt{\text{OPT}} + T^{\frac{1}{4}} W^{\frac{1}{2}} \sqrt[4]{\text{OPT}}, T\})$$

excess risk bound, where $W = w(\mathcal{X}) + w(\mathcal{C})$, $\mathcal{X} \subset \mathbb{R}^d$ is the domain from which the $\mathbf{x}_i$'s (covariates) are drawn, and OPT is the true minimum empirical risk at time T. As we discuss in Section 5, for many practically interesting regression instances, such as when $\mathcal{X}$ is a domain of sparse vectors, and $\mathcal{C}$ is bounded by an L_1-ball (as is the case of popular Lasso regression) or is a polytope defined by polynomial (in dimension) many vertices, $W \approx \text{polylog}(d)$, in which case the risk bound can be simplified to $\tilde{O}(\min\{T^{\frac{1}{3}} + T^{\frac{1}{6}} \sqrt{\text{OPT}} + T^{\frac{1}{4}} \sqrt[4]{\text{OPT}}, T\})$. In most practical scenarios, when the relationship between covariates and response satisfy a linear relationship, one would also expect $\text{OPT} \ll T$. These bounds show that, for certain instances, it is possible to design differentially private risk minimizers in the incremental setting with excess risk that depends only poly-logarithmically on the dimensionality of the data, a desired feature in a high-dimensional setting.

[9] For a set $S \subseteq \mathbb{R}^d$, Gaussian width is defined as $w(S) = \mathbb{E}_{\mathbf{g} \in \mathcal{N}(0,1)^d} [\sup_{\mathbf{a} \in S} \langle \mathbf{a}, \mathbf{g} \rangle]$.

[10] Note that this issue arises independent of the differential privacy requirements, and holds even in a non-private incremental setting.

[11] Recent results [5] have shown other distributions for generating Φ also provide similar guarantees.

Organization. In Section 1.2, we discuss some related work. In Section 2, we present some preliminaries. Our generic transformation from a private batch ERM algorithm to a private incremental ERM algorithm is given in Section 3. We present techniques that improve upon these bounds for the problem of private incremental regression in Sections 4 and 5. The appendices contain some proof details and supplementary material. In Appendix C, we analyze the convergence rate of a noisy projected gradient descent technique, and in Appendix D, we present the *Tree Mechanism* of [16, 7].

1.2 Related Work

Private ERM. Starting from the works of Chaudhuri *et al.* [8, 9], private convex ERM problems have been studied in various settings including the low-dimensional setting [41, 34], high-dimensional sparse regression setting [34, 48], online learning setting [25, 49, 27, 37], local privacy setting [12], and interactive setting [26, 51].

Bassily *et al.* [2] presented algorithms that for a general convex loss function $\jmath(\theta; \mathbf{z})$ which is 1-Lipschitz (Definition 8) for every $\mathbf{z}$, achieves an expected excess risk of $\approx \sqrt{d}$ under (ϵ, δ)-differential privacy and $\approx d$ under ϵ-differential privacy (ignoring the dependence on other parameters for simplicity).[12] We use their batch mechanisms in our generic construction to obtain risk bounds for incremental ERM problems (Theorem 3.1). They also showed that these bounds cannot be improved *in general*, even for the least-squares regression function. However, if the constraint space has low Gaussian width (such as with the L_1-ball), Talwar *et al.* [47, 46] recently showed that, under (ϵ, δ)-differential privacy, the above bound can be improved by exploiting the geometric properties of the constraint space. An analogous result under ϵ-differential privacy for the class of *generalized linear functions* (which includes linear, logistic regression) was recently obtained by Kasiviswanathan and Jin [29]. Our excess risk bounds based on Gaussian width (presented in Section 5) uses a lifting procedure similar to that used by Kasiviswanathan and Jin [29]. All of these above algorithms operate in the batch setting, and ours is the first work dealing with private ERM problems in an incremental setting.

Private Online Convex Optimization. Differentially private algorithms have also been designed for a large class of online (convex optimization) learning problems, in both the full information and bandit settings [25, 49, 27]. Adapting the popular *Follow the Approximate Leader* framework [23], Smith and Thakurta [49] obtained regret bounds for private online learning with nearly optimal dependence on T (though the dependence on the dimensionality d in these results is much worse than the known lower bounds). As discussed earlier, incremental learning is a variant of the batch learning, with goals different from online learning.

Private Incremental Algorithms. Dwork *et al.* [16] introduced the problem of counting under incremental (continual) observations. The goal is to monitor a stream of T bits, and continually release a counter of the number of 1's that have been observed so far, under differential privacy. The elegant *Tree Mechanism* introduced by Dwork *et al.* [16]

[12] Better risk bounds were achieved under strong convexity assumptions [2, 47].

and Chan *et al.* [7] solves this problem, under ϵ-differential privacy, with error roughly $\log^{5/2} T$. The versatility of this mechanism has been utilized in different ways in subsequent works [49, 17, 24]. We use the *Tree Mechanism* as a basic building block for computing the private gradient function incrementally. Dwork *et al.* [16] also achieve *pan-privacy* for their continual release (which means that the mechanism preserves differential privacy even when an adversary can observe snapshots of the mechanism's internal states), a property that we do not investigate in this paper.

Use of JL Transform for Privacy. The use of JL transform for achieving differential privacy with better utility has been well documented for a variety of computational tasks [58, 3, 33, 44, 52]. Blocki *et al.* [3] have shown that if $\Phi \in \mathbb{R}^{m\times n}$ is a Gaussian random matrix of appropriate dimension, then $\Phi X \in \mathbb{R}^{m\times d}$ is differentially private if the least singular value of the matrix $X \in \mathbb{R}^{n\times d}$ is "sufficiently" large. The bound on the least singular value was recently improved by Sheffet [44]. Here the privacy comes as a result of randomization inherent in the transform. However, these results require that the projection matrix is kept private, which is an issue in an incremental setting, where an adversary could learn about Φ over time. Kenthapadi *et al.* [33] use Johnson-Lindenstrauss transform to publish a private sketch that enables estimation of the distance between users. Their main idea is based on projecting a d-dimensional user feature vector into a lower m-dimensional space by first applying a random Johnson-Lindenstrauss transform and then adding Gaussian noise to each entry of the resulting vector. None of these above results deal with an incremental setting, where applying the JL transform is itself a challenge because of the adaptivity issues.

Traditional Streaming Algorithms. The literature on streaming algorithms is replete with various techniques that can solve linear regression and related problems on various streaming models of computation under various computational resource constraints. We refer the reader to the survey by Woodruff [57] for more details. However, incremental regression under differential privacy, poses a different challenge than that faced by traditional streaming algorithms. The issue is that the solution (regression parameter) at each timestep relies on all the datapoints observed in the past, and frequent releases about earlier points can lead to privacy loss.

2. Preliminaries

Notation and Data Normalization. We denote $[n] = \{1, \ldots, n\}$. Vectors are in column-wise fashion, denoted by boldface letters. For a vector $\mathbf{v}$, $\mathbf{v}^\top$ denotes its transpose, $\|\mathbf{v}\|$ it's Euclidean (L_2-) norm, and $\|\mathbf{v}\|_1$ it's L_1-norm. For a matrix M, $\|M\|$ denotes its spectral norm which equals its largest singular value, and $\|M\|_F$ its Frobenius norm. We use $\mathbf{0}$ to denote a d-dimensional vector of all zeros. The d-dimensional unit ball in L_p-norm centered at origin is denoted by B_p^d. $\mathbb{I}_d$ represents the $d\times d$ identity matrix. $\mathcal{N}(\mu, \Sigma)$ denotes the Gaussian distribution with mean vector μ and covariance matrix Σ. For a variable n, we use $\mathrm{poly}(n)$ to denote a polynomial function of n and $\mathrm{polylog}(n)$ to denote $\mathrm{poly}(\log(n))$.

We assume all streams are of a fixed length T, which is known to the algorithm. We make this assumption for simplifying the discussion. In fact, in our presented generic transformation for incremental ERM this assumption can be straightforwardly removed. Whereas in the case of algorithms for private incremental regression this assumption can be removed by using a simple trick[13] introduced by Chan *et al.* [7]. For a stream Γ, we use Γ_t to denote the stream prefix of length t.

Throughout this paper, we use ℓ and $\mathcal{L}$ to indicate the least-squared loss on a single datapoint and a collection of datapoints, respectively. Namely,

$$\ell(\theta; (\mathbf{x}, y)) = (y - \langle \mathbf{x}, \theta\rangle)^2 \text{ and}$$
$$\mathcal{L}(\theta; (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)) = \textstyle\sum_{i=1}^n \ell(\theta; (\mathbf{x}_i, y_i)).$$

In Appendix A, we review a few additional definitions related to convex functions and Gaussian concentration. For a set of vectors, we define its diameter as the maximum attained norm in the set.

Definition 2. *(Diameter of Set) The diameter $\|\mathcal{C}\|$ of a closed set $\mathcal{C} \subseteq \mathbb{R}^d$, is defined as $\|\mathcal{C}\| = \sup_{\theta\in\mathcal{C}} \|\theta\|$.*

For improving the worst-case dependence on dimension d, we exploit the geometric properties of the input and constraint space. We use the well-studied quantity of *Gaussian width* that captures the L_2-geometric complexity of a set $S \subseteq \mathbb{R}^d$.

Definition 3 (Gaussian Width)**.** *Given a closed set $S \subseteq \mathbb{R}^d$, its Gaussian width $w(S)$ is defined as:*

$$w(S) = \mathbb{E}_{\mathbf{g}\in\mathcal{N}(0,1)^d}[\sup_{\mathbf{a}\in S}\langle \mathbf{a}, \mathbf{g}\rangle].$$

In particular, $w(S)^2$ can be thought as the "effective dimension" of S. Many popular convex sets have low Gaussian width, e.g., the width of both the unit L_1-ball in $\mathbb{R}^d$ (B_1^d) and the standard d-dimensional probability simplex equals $\Theta(\sqrt{\log d})$, and the width of any ball B_p^d for $1 \le p \le \infty$ is $\approx d^{1-1/p}$. For a set $\mathcal{C}$ contained in the B_2^d, $w(\mathcal{C})$ is always $O(\sqrt{d})$. Another prominent set with low Gaussian width is that made up of sparse vectors. For example, the set of all k-sparse vectors (with at most k non-zero entries) in $\mathbb{R}^d$ has Gaussian width $\Theta(\sqrt{k\log(d/k)})$.

Differential Privacy on Streams. We will consider differential privacy on data streams [16]. A stream is a sequence of points from some domain set $\mathcal{Z}$. We say that two streams $\Gamma, \Gamma' \in \mathcal{Z}^*$ of the same length are neighbors if there exists a datapoint $\mathbf{z} \in \Gamma$ and $\mathbf{z}' \in \mathcal{Z}$ such that if we change $\mathbf{z}$ in Γ to $\mathbf{z}'$ we get the stream Γ'. The result of an algorithm processing a stream is a sequence of outputs.

Definition 4 (Event-level differential privacy [15, 16])**.** *Algorithm* Alg *is (ϵ, δ)-differentially private[14] if for all neighboring streams Γ, Γ' and for all sets of possible output sequences $\mathcal{R} \subseteq \mathbb{R}^{\mathbb{N}}$, we have*

$$\Pr[\mathrm{Alg}(\Gamma) \in \mathcal{R}] \le \exp(\epsilon) \cdot \Pr[\mathrm{Alg}(\Gamma') \in \mathcal{R}] + \delta,$$

[13] Chan *et al.* [7] presented a scheme that provides a generic way for converting the *Tree Mechanism* that requires prior knowledge of T into a mechanism that does not. They also showed that this new mechanism (referred to as the *Hybrid Mechanism*) achieves asymptotically the same error guarantees as the *Tree Mechanism*. The same ideas work in our case too, and the asymptotic excess risk bounds are not affected.

[14] In the practice of differential privacy, we generally think of ϵ as a small non-negligible constant, and δ as a parameter that is cryptographically small.

Mechanism 1: PrivIncERM (ϵ, δ)

Input: A stream $\Gamma = \mathbf{z}_1, \ldots, \mathbf{z}_T$, where each $\mathbf{z}_t$ is from the domain $\mathcal{Z} \subseteq \mathbb{R}^d$, and $\tau \in \mathbb{N}$

Output: A differentially private estimate of $\hat{\theta}_t \in \text{argmin}_{\theta \in \mathcal{C}} \sum_{i=1}^{t} \jmath(\theta; \mathbf{z}_i)$ at every timestep $t \in [T]$

Set $\epsilon' \leftarrow \frac{\epsilon}{\left(2\sqrt{\frac{2T}{\tau}\ln\left(\frac{2}{\delta}\right)}\right)}$ and $\delta' \leftarrow \frac{\delta\tau}{2T}$

$\theta_0^{\text{priv}} \leftarrow \mathbf{0}$

for *all* $t \in [T]$ **do**

  if t *is a multiple of* τ **then**

    $\theta_t^{\text{priv}} \leftarrow$ Output of an (ϵ', δ')-differentially private algorithm minimizing $\mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_t)$

  end

  else

    $\theta_t^{\text{priv}} \leftarrow \theta_{t-1}^{\text{priv}}$

  end

  Return θ_t^{priv}

end

where the probability is taken over the randomness of the algorithm. When $\delta = 0$, the algorithm Alg *is ϵ-differentially private.*

We provide additional background on differential privacy along with some techniques for achieving it in Appendix A.2.

3. Private Incremental ERM: A Generic Mechanism

We present a generic transformation for converting any private batch ERM algorithm into a private incremental ERM algorithm. We take this construction as a baseline for comparison for our private incremental regression algorithms. Missing proofs from this section are presented in Appendix B.

Mechanism PrivIncERM describes this simple transformation. At every timestep, Mechanism PrivIncERM outputs a θ_t^{priv}, a differentially private approximation of

$$\hat{\theta}_t \in \text{argmin}_{\theta \in \mathcal{C}} \mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_t),$$

$$\text{where } \mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_t) = \sum_{i=1}^{t} \jmath(\theta; \mathbf{z}_i).$$

The idea is to perform "relevant" computations only every τ timesteps, thereby ensuring that no $\mathbf{z}_i$ is used in more than T/τ invocations of the private batch ERM algorithm (for simplicity, assume that T is a multiple of τ). This idea is reminiscent of mini-batch processing ideas commonly used in big data processing [6]. In Theorem 3.1, the parameter τ is set to balance the increase in excess risk due to lack of updates on the estimator and the increase in the excess risk due to the change in the privacy risk parameter ϵ (which arises from multiple interactions with the data).

Mechanism PrivIncERM invokes a differentially private (batch) ERM algorithm for timesteps t which are a multiple of τ, and in all other timesteps it just outputs the result from the previous timestep. In Step 5 of Mechanism PrivIncERM any differentially private batch ERM algorithm can be used, and this step dominates the time complexity of this mechanism. Here we present excess risk bounds obtained by invoking Mechanism PrivIncERM with the differentially private ERM algorithms of Bassily *et al.* [2] and Talwar *et al.* [46]. As mentioned earlier, the bounds of Bassily *et al.* [2] are tight in the worst-case. But as shown by Talwar *et al.* [46], if the constraint space $\mathcal{C}$ has a small Gaussian width, then these bounds could be improved. The bounds of Talwar *et al.* depend on the curvature constant of $\jmath$ defined as:

$$C_\jmath = \sup_{\mathbf{z} \in \mathcal{Z}} \sup_{\theta_a, \theta_b \in \mathcal{C}, l \in (0,1], \theta_c = \theta_a + l(\theta_b - \theta_a)} \frac{2}{l^2}(\jmath(\theta_c; \mathbf{z}) - \jmath(\theta_a; \mathbf{z}) - \langle \theta_c - \theta_a, \nabla \jmath(\theta_a; \mathbf{z}) \rangle).$$

For linear regression where $\jmath(\theta; \mathbf{z}) = \ell(\theta; (\mathbf{x}, y)) = (y - \langle \mathbf{x}, \theta \rangle)^2$, then with $\|\mathbf{x}\| \leq 1$ and $|y| \leq 1$, it follows that $C_\ell \leq \|\mathcal{C}\|^2$ [10].

We now show that Mechanism PrivIncERM is event-level differentially private (Definition 4), and analyze its utility under various invocations in Step 5.

Theorem 3.1. *Mechanism* PrivIncERM *is (ϵ, δ)-differentially private with respect to a single datapoint change in the stream Γ. Also,*

1. (Using Theorem 2.4, Bassily et al. *[2]). If the function $\jmath(\theta; \mathbf{z}) : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}$ is a positive-valued function that is convex with respect to θ over the domain $\mathcal{C} \subseteq \mathbb{R}^d$. Then for any $\beta > 0$, with probability at least $1 - \beta$, for each $t \in [T]$, θ_t^{priv} generated by Mechanism* PrivIncERM *with $\tau = \lceil \frac{(Td)^{1/3}}{\epsilon^{2/3}} \rceil$ satisfies:*

$$\mathcal{J}(\theta_t^{\text{priv}}; \Gamma_t) - \min_{\theta \in \mathcal{C}} \mathcal{J}(\theta; \Gamma_t) = O(\min\{\frac{(Td)^{1/3} L \|\mathcal{C}\| \log^{5/2}(1/\delta) \,\text{polylog}(T/\beta)}{\epsilon^{2/3}}, TL\|\mathcal{C}\|\}),$$

where L is the Lipschitz parameter of the function $\jmath$.

2. (Using Theorem 2.4, Bassily et al. *[2]). If the function $\jmath(\theta; \mathbf{z}) : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}$ is a positive-valued function which is ν-strongly convex with respect to θ over the domain $\mathcal{C} \subseteq \mathbb{R}^d$. Then for any $\beta > 0$, with probability at least $1 - \beta$, for each $t \in [T]$, θ_t^{priv} generated by Mechanism* PrivIncERM *with $\tau = \lceil \frac{\sqrt{dL}}{(\nu^{1/2}\epsilon\|\mathcal{C}\|^{1/2})} \rceil$ satisfies:*

$$\mathcal{J}(\theta_t^{\text{priv}}; \Gamma_t) - \min_{\theta \in \mathcal{C}} \mathcal{J}(\theta; \Gamma_t) = O(\min\{\frac{\sqrt{d} L^{3/2} \|\mathcal{C}\|^{1/2} \log^4(1/\delta) \text{polylog}(T/\beta)}{\nu^{1/2}\epsilon}, TL\|\mathcal{C}\|\}),$$

where L is the Lipschitz parameter of the function $\jmath$.

3. (Using Theorem 2.6 of Talwar et al. *[46]). If the function $\jmath(\theta; \mathbf{z}) : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}$ is a positive-valued function that is convex with respect to θ over the domain $\mathcal{C} \subseteq \mathbb{R}^d$. Then for any $\beta > 0$, with probability at least $1 - \beta$, for each $t \in [T]$, θ_t^{priv} generated by Mechanism* PrivIncERM *with $\tau = \lceil \frac{\sqrt{Tw(\mathcal{C})} C_\jmath^{1/4}}{((L\|\mathcal{C}\|)^{1/4}\epsilon^{1/2})} \rceil$ satisfies:*

$$\mathcal{J}(\theta_t^{\text{priv}}; \Gamma_t) - \min_{\theta \in \mathcal{C}} \mathcal{J}(\theta; \Gamma_t) = O(\min\{\frac{\sqrt{Tw(\mathcal{C})} C_\jmath^{1/4} (L\|\mathcal{C}\|)^{3/4} \log^{7/3}(1/\delta) \,\text{polylog}(T/\beta)}{\epsilon^{1/2}}, TL\|\mathcal{C}\|\}),$$

where L is the Lipschitz parameter and $C_\jmath$ is the curvature constant of the function $\jmath$.

4. Private Incremental Linear Regression using Tree Mechanism

We now focus on the problem of private incremental linear regression. Our first approach for this problem is based on a private incremental computation of the gradient. The algorithm is particularly effective in the regime of large T and small d. A central idea of our approach is the construction of a *private gradient function* defined as follows.

Definition 5. *Let $\mathcal{C} \subseteq \mathbb{R}^d$. Algorithm* Alg *computes an (α, β)-accurate gradient of the loss function $\mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_t)$ with respect to $\theta \in \mathcal{C}$, if given $\mathbf{z}_1, \ldots, \mathbf{z}_t \in \mathcal{Z}$ it outputs a function $g_t : \mathcal{C} \to \mathbb{R}^d$ such that:*

(i) Privacy*: Alg is (ϵ, δ)-differentially private (as in Definition 4), i.e., for all neighboring streams $\Gamma, \Gamma' \in \mathcal{Z}^*$ and subsets $\mathcal{R} \subseteq \mathcal{C} \to \mathbb{R}^d$,*

$$\Pr[\text{Alg}(\Gamma) \in \mathcal{R}] \leq \exp(\epsilon) \cdot \Pr[\text{Alg}(\Gamma') \in \mathcal{R}] + \delta.$$

(ii) Utility*: The function g_t is an (α, β)-approximation to the true gradient, in that,*

$$\Pr_{\text{Alg}}\left[\max_{\mathbf{z}_1, \ldots, \mathbf{z}_t \in \mathcal{Z}, \theta \in \mathcal{C}} \|g_t(\theta) - \nabla \mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_t)\| \geq \alpha\right] \leq \beta.$$

Note that the output of Alg in the above definition is a function g_t. The first requirement on Alg specifies that it satisfies the differential privacy condition (Definition 4). The second requirement on Alg specifies that for any $\theta \in \mathcal{C}$, it gives a "sufficiently" accurate estimate of the true gradient $\nabla \mathcal{J}(\theta; \mathbf{z}_1, \ldots, \mathbf{z}_t)$.

Let $\Gamma = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ represent the stream of covariate-response pairs, we use Γ_t to denote $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_t, y_t)$. Consider the gradient of the loss function $\mathcal{L}(\theta; \Gamma_t)$ where $X_t \in \mathbb{R}^{t \times d}$ is a matrix with rows $\mathbf{x}_1^\top, \ldots, \mathbf{x}_t^\top$ and $\mathbf{y}_t = (y_1, \ldots, y_t)$:

$$\nabla \mathcal{L}(\theta; \Gamma_t) = 2(X_t^\top X_t \theta - X_t^\top \mathbf{y}_t) = 2\left(\sum_{i=1}^{t} \mathbf{x}_i \mathbf{x}_i^\top \theta - \sum_{i=1}^{t} \mathbf{x}_i y_i\right). \tag{2}$$

A simple observation from the gradient form of (2) is that if we can maintain the streaming sum of $\mathbf{x}_1 y_1, \ldots, \mathbf{x}_T y_T$ and the streaming sum of $\mathbf{x}_1 \mathbf{x}_1^\top, \ldots, \mathbf{x}_T \mathbf{x}_T^\top$, then we can maintain the necessary ingredients needed to compute $\nabla \mathcal{L}(\theta; \Gamma_t)$ for any $t \in [T]$. We use this observation to construct a private gradient function $g_t : \mathcal{C} \to \mathbb{R}^d$ at every timestep t. The idea is to privately maintain $\sum_{i=1}^{T} \mathbf{x}_i \mathbf{x}_i^\top$ and $\sum_{i=1}^{T} \mathbf{x}_i y_i$ over the steam using the *Tree Mechanism* of [16, 7]. We present the entire construction of the *Tree Mechanism* in Appendix D. The rough idea behind this mechanism is to build a binary tree where the leaves are the actual inputs from the stream, and the internal nodes store the partial sums of all the leaves in its sub-tree. For a stream of vectors, $\upsilon_1, \ldots, \upsilon_T$ in the unit ball, the *Tree Mechanism* allows private estimation of $\sum_{i=1}^{t} \upsilon_i$, for every $t \in [T]$, with error roughly $\sqrt{d} \log^2 T$ (under (ϵ, δ)-differential privacy, ignoring other parameters).

Somewhat similar to our approach, Smith and Thakurta also use the *Tree Mechanism* to maintain the sum of gradients in their private online learning algorithm [49], however, unlike our approach, they do not construct a private gradient function.

Algorithm 2: PRIVINCREG$_1$ (ϵ, δ)

Input: A stream $\Gamma = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$, where each $(\mathbf{x}_t, y_t)$ in Γ is from the domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X} \subset \mathbb{R}^d$ with $\|\mathcal{X}\| \leq 1$ and $\mathcal{Y} \subset \mathbb{R}$ with $\|\mathcal{Y}\| \leq 1$

Output: A differentially private estimate of $\hat{\theta}_t \in \text{argmin}_{\theta \in \mathcal{C}} \sum_{i=1}^{t} (y_i - \langle \mathbf{x}_i, \theta \rangle)^2$ at every timestep $t \in [T]$

1 Set $\epsilon' \leftarrow \frac{\epsilon}{2}$, $\delta' \leftarrow \frac{\delta}{2}$, $\kappa \leftarrow \frac{\log^{3/2}(T)\sqrt{\log\left(\frac{1}{\delta'}\right)}}{\epsilon'}$, $\alpha' \leftarrow O(\kappa \|\mathcal{C}\| \sqrt{d})$, and $r \leftarrow \Theta\left(\left(1 + \frac{T\|\mathcal{C}\|}{\alpha'}\right)^2\right)$

2 **for** *all* $t \in [T]$ **do**

3 $\quad \mathbf{q}_t \leftarrow$ output of TREEMECH $(\epsilon', \delta', 2)$ at t when invoked on the stream $\mathbf{x}_1 y_1, \ldots, \mathbf{x}_T y_T$

4 $\quad Q_t \leftarrow$ output of TREEMECH $(\epsilon', \delta', 2)$ at t when invoked on the stream $\mathbf{x}_1 \mathbf{x}_1^\top, \ldots, \mathbf{x}_T \mathbf{x}_T^\top$ which can be viewed as d^2-dimensional vectors (the outputs are converted back to form $d \times d$ matrices)

5 $\quad$ Define a private gradient function, $g_t : \mathcal{C} \to \mathbb{R}^d$ as:

$$g_t(\theta) = 2(Q_t \theta - \mathbf{q}_t)$$

6 $\quad \theta_t^{\text{priv}} \leftarrow$ NOISYPROJGRAD$(\mathcal{C}, g_t, r)$ (described in Appendix C)

7 $\quad$ Return θ_t^{priv}

8 **end**

Once the private gradient function g_t is released, it could be used to obtain an approximation to the estimator $\hat{\theta}$ by using any traditional gradient-based optimization technique. In this paper, we use a variant of the classical *projected gradient descent* approach, described in Algorithm NOISYPROJGRAD (Appendix C). Algorithm NOISYPROJGRAD is a iterative algorithm that takes in the constraint set $\mathcal{C}$, private gradient function g_t, and a parameter r denoting the number of iterations, and returns back (θ_t^{priv}) a private estimate of regression parameter.

An important point to note is that evaluating the gradient function at different θ's, as needed for any gradient descent technique, does not affect the privacy parameters ϵ and δ. We analyze the convergence of Algorithm NOISYPROJGRAD in Appendix C. This convergence result (Corollary C.2) is used to set the parameter r in Algorithm PRIVINCREG$_1$.

Algorithm PRIVINCREG$_1$ is (ϵ, δ)-differentially private with respect to a single datapoint change in the stream Γ. This follows as the L_2-sensitivity of the stream in both the invocations (Steps 3 and 4) of Algorithm TREEMECH is less than 2 (because of the normalization on the $\mathbf{x}_i$'s and y_i's). Using the standard composition properties (Theorem A.3) of differential privacy gives that the Algorithm PRIVINCREG$_1$ is (ϵ, δ)-differentially private. The algorithm only requires $O(d^2 \log T)$ space (see Appendix D), therefore having only a logarithmic dependence on the length of the stream. The running time of the algorithm is dominated by Steps 4 and 6;

at every timestep t, Step 4 has time complexity of $O(d^2 \log T)$, whereas the time complexity of Step 6 is r times the time complexity of projecting a datapoint onto $\mathcal{C}$ (the $P_{\mathcal{C}}$ operation defined in Appendix C). We now analyze the utility of this algorithm, using the error bound on *Tree Mechanism* from Proposition D.1.

Lemma 4.1. *For any $\beta > 0$, $\theta \in \mathcal{C}$, and $t \in [T]$, with probability at least $1 - \beta$, the function g_t defined in Algorithm* PrivIncReg$_1$ *satisfies:*

$$\|g_t(\theta) - \nabla \mathcal{L}(\theta; \Gamma_t)\| = O\left(\kappa \|\mathcal{C}\| (\sqrt{d} + \sqrt{\log(1/\beta)})\right).$$

Proof. Applying Proposition D.1, we know with probability at least $1 - \beta$,

$$\left\|\mathbf{q}_t - \textstyle\sum_{i=1}^t \mathbf{x}_i y_i\right\| = O\left(\kappa(\sqrt{d} + \sqrt{\log(1/\beta)})\right) \text{ and}$$
$$\left\|Q_t - \textstyle\sum_{i=1}^t \mathbf{x}_i \mathbf{x}_i^\top\right\| = O\left(\kappa(\sqrt{d} + \sqrt{\log(1/\beta)})\right).$$

Therefore, we get with probability at least $1 - \beta$,

$$\begin{aligned}
&\|g_t(\theta) - \nabla \mathcal{L}(\theta; \Gamma_t)\| \\
&= \left\| 2\left(Q_t - \sum_{i=1}^t \mathbf{x}_i \mathbf{x}_i^\top\right)\theta - 2\left(\mathbf{q}_t - \sum_{i=1}^t \mathbf{x}_i y_i\right)\right\| \\
&= O\left(\kappa \|\mathcal{C}\| (\sqrt{d} + \sqrt{\log(1/\beta)})\right),
\end{aligned}$$

where we used the fact that $\|\theta\| \leq \|\mathcal{C}\|$ for any $\theta \in \mathcal{C}$. □

Theorem 4.2. *Algorithm* PrivIncReg$_1$ *is (ϵ, δ)-differentially private with respect to a single datapoint change in the stream Γ. For any $\beta > 0$, with probability at least $1-\beta$, for each $t \in [T]$, θ_t^{priv} generated by Algorithm* PrivIncReg$_1$ *satisfies:*

$$\begin{aligned}
&\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t) - \min_{\theta \in \mathcal{C}} \mathcal{L}(\theta; \Gamma_t) \\
&= O\left(\frac{\log^{3/2} T \sqrt{\log(1/\delta)} \|\mathcal{C}\|^2 (\sqrt{d} + \sqrt{\log(T/\beta)})}{\epsilon}\right).
\end{aligned}$$

Proof. The (ϵ, δ)-differential privacy follows from the above discussed global sensitivity bound.

Fix any $t \in [T]$. Let $\hat{\theta}_t \in \text{argmin}_{\theta \in \mathcal{C}} \sum_{i=1}^t (y_i - \langle \mathbf{x}_i, \theta \rangle)^2$. Combining Lemma 4.1 and Corollary C.2, with probability at least $1 - r\beta'$

$$\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t) - \mathcal{L}(\hat{\theta}_t; \Gamma_t) = O\left(\kappa \|\mathcal{C}\|^2 (\sqrt{d} + \sqrt{\log(1/\beta')})\right).$$

Replacing β' by β/r, substituting for κ, and taking a union bound over all $t \in [T]$ gives the claimed result. □

Remark 4.3. *For linear regression instances, which typically do not satisfy the strong convexity property, the $\approx \min\{(Td)^{1/3}, T\}$ risk bound obtained from Mechanism* PrivIncERM *is substantially worse than the $\approx \min\{\sqrt{d}, T\}$ risk bound provided by Algorithm* PrivIncReg$_1$.

The dependence on dimension d is tight in the worst-case; due to $\approx \sqrt{d}$ excess risk lower bounds established by Bassily et al. *[2].*

Remark 4.4. *The techniques developed in this section (based on Tree Mechanism) can be applied to any convex ERM problem whose gradient has a* linear *form, in which case we obtain an excess risk bound as in Theorem 4.2. It is an interesting open question to obtain similar bounds for general convex ERM problems in an incremental setting.*

5. Private Incremental Linear Regression: Going Beyond Worst-Case

The noise added in Theorem 4.2 for privacy grows proportionately as $\sqrt{d}$. While this seems unavoidable in the worst-case, we ask whether it is possible to go beyond this worst-case bound under some realistic assumptions. An intuitive idea to overcome this curse of dimensionality is to project (compress) the data to a lower dimension, before the addition of noise. In this section, we use this and other geometric ideas to cope with the high-dependency on dimensionality in Theorem 4.2. The resulting bound will depend on the Gaussian width of the input and constraint space which for many interesting problem instances will be smaller than $\sqrt{d}$. We mention some specific instantiations and extensions of our result in Section 5.2 including a scenario when not all inputs are drawn from a domain with a "small" Gaussian width. Missing proofs from this section are presented in Appendix E.

Our general approach in this section will be based on this simple principle: reduce the dimensionality of the problem, solve it privately in the lower dimensional space, and then "lift" the solution back to the original space. Our lifting procedure is similar to that used by Kasiviswanathan and Jin [29] in their recent work on bounding excess risk for private ERM in a batch setting.

Fix t and consider solving the following projected least-squares problem, defined as:[15]

$$\mathcal{L}_{\text{proj}}(\theta; \Gamma_t; \Phi) = \sum_{i=1}^t \left(y_i - \frac{\|\mathbf{x}_i\|}{\|\Phi \mathbf{x}_i\|} \langle \Phi \mathbf{x}_i, \Phi \theta \rangle\right)^2, \quad (3)$$

where $\Phi \in \mathbb{R}^{m \times d}$ is a random projection matrix (to be defined later). The loss function, $\mathcal{L}_{\text{proj}}$ is also referred to as *compressed least-squares* in the literature [36, 20, 28].

As a first step, we investigate the relationship between $\mathcal{L}(\theta; \Gamma_t)$ and $\mathcal{L}_{\text{proj}}(\theta; \Gamma_t; \Phi)$. A fundamental tool in dimensionality reduction, the Johnson-Lindenstrauss (JL) lemma, states that for any set $S \subseteq \mathbb{R}^d$, given $\gamma > 0$ and $m = \Omega(\log |S|/\gamma^2)$, there exists a map that embeds the set into $\mathbb{R}^m$, distorting all pairwise distances within at most $1 \pm \gamma$ factor. A simple consequence of the JL Lemma is that, for any set of n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, $\theta \in \mathbb{R}^d$, $\gamma > 0$, and $\beta > 0$, if Φ is an $m \times d$ matrix with entries drawn i.i.d. from $\mathcal{N}(0, 1/m)$ with $m = \Theta(\log(n/\beta)/\gamma^2)$, we get,

$$\Pr\left[|\langle \Phi \mathbf{x}_i, \Phi \theta \rangle - \langle \mathbf{x}_i, \theta \rangle| \geq \gamma \|\mathbf{x}_i\| \|\theta\| \text{ for all } i \in [n]\right] \leq \beta. \quad (4)$$

Using standard inequalities such as (4), establishing the relationship between $\mathcal{L}$ and $\mathcal{L}_{\text{proj}}$ is relatively straightforward in a batch setting. However, the incremental setting raises the challenge of dealing with adaptive inputs in JL transformation. The issue being that JL transformation is inherently non-adaptive, in that "success" of the JL transformation (properties such as (4)) depends on the fact that the inputs on which it gets applied are chosen before (independent of) fixing the transformation Φ.

To deal with this kind of adaptive generation of inputs, we use a generalization of JL lemma which yields similar

[15] The scaling factor of $\frac{\|\mathbf{x}_i\|}{\|\Phi \mathbf{x}_i\|}$ is for simplicity of analysis only. One could omit it and still obtain the same results using a slightly different analysis. Also without loss of generality, we assume $\mathbf{x}_i \neq 0$ for all i.

guarantees but with a much smaller (than $\log|S|$) requirement on m, if the set S has certain geometric characteristics. For ease of exposition, in the rest of this section, we are going to assume that Φ is a matrix in $\mathbb{R}^{m\times d}$ with i.i.d. entries from $\mathcal{N}(0,1/m)$.[16] Gordon [21] showed that using a Gaussian random matrix, one can embed the set S into a lower-dimensional space $\mathbb{R}^m$, where m roughly scales as the square of Gaussian width of S. This result has found several interesting applications in high-dimensional convex geometry, statistics, and compressed sensing [39].

Theorem 5.1 (Gordon [21]). *Let $\tilde{\Phi}$ be an $m\times d$ random matrix, whose rows $\phi_1^\top,\ldots,\phi_m^\top$ are i.i.d. Gaussian random vectors in $\mathbb{R}^d$ chosen according to the standard normal distribution $\mathcal{N}(\mathbf{0},\mathbb{I}_d)$. Let $\Phi=\tilde{\Phi}/\sqrt{m}$. Let S be a set of points in $\mathbb{R}^d$. There is a constant $C>0$ such that for any $0<\gamma,\beta<1$,*

$$\Pr\left[\sup_{\mathbf{a}\in S}\left|\|\Phi\mathbf{a}\|^2-\|\mathbf{a}\|^2\right|\geq\gamma\|\mathbf{a}\|^2\right]\leq\beta,$$

provided that $m\geq\frac{C}{\gamma^2}\max\left\{w(S)^2,\log\left(\frac{1}{\beta}\right\}\right)$.

Note that the $w(S)$ is defined for all sets, not just convex sets, a fact that we use below as the input domain $\mathcal{X}$ may not be convex. As a simple corollary to the above theorem it also follows that,

Corollary 5.2. *Under the setting of Theorem 5.1, there exists a constant $C'>0$ such that for any $0<\gamma,\beta<1$,*

$$\Pr\left[\sup_{\mathbf{a},\mathbf{b}\in S}|\langle\Phi\mathbf{a},\Phi\mathbf{b}\rangle-\langle\mathbf{a},\mathbf{b}\rangle|\geq\gamma\|\mathbf{a}\|\|\mathbf{b}\|\right]\leq\beta,$$

provided that $m\geq\frac{C'}{\gamma^2}\max\left\{w(S)^2,\log\left(\frac{1}{\beta}\right\}\right)$.

Applying the above corollary to the set of vectors in $\mathcal{X}\cup\mathcal{C}$, and by noting that $w(\mathcal{X}\cup\mathcal{C})\leq w(\mathcal{X})+w(\mathcal{C})$, gives that if $m=\Theta((1/\gamma^2)\max\{(w(\mathcal{X})+w(\mathcal{C}))^2,\log(1/\beta)\})$, then with probability at least $1-\beta$,

$$\Pr\left[\sup_{\mathbf{x}\in\mathcal{X},\theta\in\mathcal{C}}|\langle\Phi\mathbf{x},\Phi\theta\rangle-\langle\mathbf{x},\theta\rangle|\geq\gamma\|\mathbf{x}\|\|\theta\|\right]\leq\beta. \tag{5}$$

5.1 Algorithm for the Streaming Setting

We now present a mechanism (Algorithm PrivIncReg$_2$) for private incremental linear regression based on minimizing the projected least-squares objective (3), under differential privacy. The idea is to again construct a private gradient function g_t, but of function $\mathcal{L}_{\text{proj}}$ (instead of $\mathcal{L}$ as done in Algorithm PrivIncReg$_1$).

Let X_t be a matrix with rows $\mathbf{x}_1^\top,\ldots,\mathbf{x}_t^\top$, and $\tilde{X}_t\in\mathbb{R}^{n\times d}$ be a matrix with rows $\tilde{\mathbf{x}}_1^\top,\ldots,\tilde{\mathbf{x}}_t^\top$. As before, let $\mathbf{y}_t$ be the vector $(y_1,\ldots,y_t)$. Under these notation, $\mathcal{L}_{\text{proj}}(\theta;\Gamma_t;\Phi)$ from (3) can be re-expressed as:

$$\mathcal{L}_{\text{proj}}(\theta;\Gamma_t;\Phi)=\|\mathbf{y}_t-\tilde{X}_t\Phi^\top\Phi\theta\|^2.$$

The gradient of $\mathcal{L}_{\text{proj}}$ with respect to $\Phi\theta$ equals:

$$\nabla_{(\Phi\theta)}\mathcal{L}_{\text{proj}}(\theta;\Gamma_t;\Phi)=\frac{\partial\,\|\mathbf{y}_t-(\tilde{X}_t\Phi^\top)(\Phi\theta)\|^2}{\partial\,(\Phi\theta)}$$
$$=2((\tilde{X}_t\Phi^\top)^\top(\tilde{X}_t\Phi^\top))(\Phi\theta)-2(\tilde{X}_t\Phi^\top)^\top\mathbf{y}_t.$$

[16] One could also use other (better) constructions of Φ, such as those that create sparse Φ matrix, using recent results by Bourgain *et al.* [5] extending Theorem 5.1 to other distributions.

Algorithm 3: PrivIncReg$_2$ (ϵ,δ)

Input: A stream $\Gamma=(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_T,y_T)$, where each $(\mathbf{x}_t,y_t)$ in Γ is from the domain $\mathcal{X}\times\mathcal{Y}$ where $\mathcal{X}\subset\mathbb{R}^d$ with $\|\mathcal{X}\|\leq 1$ and $\mathcal{Y}\subset\mathbb{R}$ with $\|\mathcal{Y}\|\leq 1$

Output: θ_t^{priv} a differentially private estimate of $\hat{\theta}_t\in\text{argmin}_{\theta\in\mathcal{C}}\sum_{i=1}^t(y_i-\langle\mathbf{x}_i,\theta\rangle)^2$ at every timestep $t\in[T]$

Set $\epsilon'\leftarrow\frac{\epsilon}{2}$, $\delta'\leftarrow\frac{\delta}{2}$, $\kappa\leftarrow\frac{\log^{3/2}(T)\sqrt{\log\left(\frac{1}{\delta'}\right)}}{\epsilon'}$, $\alpha'\leftarrow O(\kappa\|\mathcal{C}\|\sqrt{m})$, $r\leftarrow\Theta\left(\left(1+\frac{T\|\mathcal{C}\|}{\alpha'}\right)^2\right)$, $\gamma\leftarrow\frac{(w(\mathcal{X})+w(\mathcal{C}))^{1/3}}{T^{1/3}}$, and $m\leftarrow\Theta\left(\frac{1}{\gamma^2}\max\{(w(\mathcal{X})+w(\mathcal{C}))^2,\log(\frac{T}{\beta})\}\right)$

Let $\Phi\leftarrow m\times d$ random matrix with entries drawn i.i.d. from $\mathcal{N}(0,1/m)$

for *all* $t\in[T]$ **do**

  Let $\tilde{\mathbf{x}}_t\leftarrow\frac{\|\mathbf{x}_t\|}{\|\Phi\mathbf{x}_t\|}\mathbf{x}_t$

  $\mathbf{q}_t\leftarrow$ output of TreeMech $(\epsilon',\delta',2)$ at t when invoked on the stream $\Phi\tilde{\mathbf{x}}_1y_1,\ldots,\Phi\tilde{\mathbf{x}}_Ty_T$

  $Q_t\leftarrow$ output of TreeMech $(\epsilon',\delta',2)$ at t when invoked on the stream $(\Phi\tilde{\mathbf{x}}_1)(\Phi\tilde{\mathbf{x}}_1)^\top,\ldots,(\Phi\tilde{\mathbf{x}}_T)(\Phi\tilde{\mathbf{x}}_T)^\top$ which can be viewed as m^2-dimensional vectors (the outputs are converted back to form $m\times m$ matrices)

  Define a private gradient function, $g_t:\Phi\mathcal{C}\to\mathbb{R}^m$ as:
  $$g_t(\vartheta)=2(Q_t\vartheta-\mathbf{q}_t)$$

  $\vartheta_t^{\text{priv}}\leftarrow$ NoisyProjGrad$(\Phi\mathcal{C},g_t,r)$ (described in Appendix C)

  $\theta_t^{\text{priv}}\leftarrow\text{argmin}_{\theta\in\mathbb{R}^d}\|\theta\|_{\mathcal{C}}$ subject to $\Phi\theta=\vartheta_t^{\text{priv}}$ (can be solved using any convex optimization technique)

  Return θ_t^{priv}

end

Note that $\nabla_{(\Phi\theta)}\mathcal{L}_{\text{proj}}\in\mathbb{R}^m$.

Let $\Phi\mathcal{C}=\{\vartheta\in\Phi\theta:\theta\in\mathcal{C}\}$. Note for a convex $\mathcal{C}$, $\Phi\mathcal{C}\subset\mathbb{R}^m$ is also convex. In Algorithm PrivIncReg$_2$, $\vartheta_t^{\text{priv}}$ is a private estimate of $\hat{\vartheta}_t$, where

$$\hat{\vartheta}_t\in\text{argmin}_{\vartheta\in\Phi\mathcal{C}}\sum_{i=1}^t\left(y_i-\frac{\|\mathbf{x}_i\|}{\|\Phi\mathbf{x}_i\|}\langle\Phi\mathbf{x}_i,\vartheta\rangle\right)^2.$$

Algorithm PrivIncReg$_2$ only requires $O(m^2\log T+\log d)$ space, therefore is slightly more memory efficient than Algorithm PrivIncReg$_1$ (as $m\leq d$). The time complexity can be analyzed as for Algorithm PrivIncReg$_1$.

Algorithm PrivIncReg$_2$ is (ϵ,δ)-differentially private with respect to a single datapoint change in the stream Γ. The L_2-sensitivity for both invocations of Algorithm TreeMech

is 2. In Step 6, this holds because

$$\begin{aligned}&\max_{\mathbf{x}_a,\mathbf{x}_b\in\mathcal{X}} \|(\Phi\tilde{\mathbf{x}}_a)(\Phi\tilde{\mathbf{x}}_a)^\top - (\Phi\tilde{\mathbf{x}}_b)(\Phi\tilde{\mathbf{x}}_b)^\top\|_F\\ &\le \|(\Phi\tilde{\mathbf{x}}_a)(\Phi\tilde{\mathbf{x}}_a)^\top\|_F + \|(\Phi\tilde{\mathbf{x}}_b)(\Phi\tilde{\mathbf{x}}_b)^\top\|_F\\ &= \|\Phi\tilde{\mathbf{x}}_a\|^2 + \|\Phi\tilde{\mathbf{x}}_b\|^2 = \|\mathbf{x}_a\|^2 + \|\mathbf{x}_b\|^2 = 2,\end{aligned}$$

the second to last equality follows as, for every $\mathbf{x} \in \mathcal{X}$, $\|\Phi\tilde{\mathbf{x}}\| = \|\mathbf{x}\|$ (by construction). Since $\Phi\mathbf{x}_i$'s are in the projected subspace ($\mathbb{R}^m$), the noise needed for differential privacy (in Steps 5 and 6) of Algorithm PrivIncReg$_2$ roughly scales as $\sqrt{m}$.

In Step 9 of Algorithm PrivIncReg$_2$, we lift $\vartheta_t^{\text{priv}}$ into the original d-dimensional constraint space $\mathcal{C}$. Since $\vartheta_t^{\text{priv}} \in \Phi\mathcal{C}$, we know that there exists a $\theta_t^{\text{true}} \in \mathcal{C}$, such that $\Phi\theta_t^{\text{true}} = \vartheta_t^{\text{priv}}$. Then the goal is to estimate θ_t^{true} from $\Phi\theta_t^{\text{true}}$. Again geometry of $\mathcal{C}$ (Gaussian width) plays an important role, as it controls the diameter of high-dimensional random sections of $\mathcal{C}$ (referred to as $M^\star$ bound [35, 54]). We refer the reader to the excellent tutorial by Vershynin [54] for more details.

We define Minkowski functional, as commonly used in geometric functional analysis and convex analysis.

Definition 6 (Minkowski functional)**.** *For any vector $\theta \in \mathbb{R}^d$, the Minkowski functional of $\mathcal{C}$ is the non-negative number $\|\theta\|_{\mathcal{C}}$ defined by the rule: $\|\theta\|_{\mathcal{C}} = \inf\{\rho \in \mathbb{R} : \theta \in \rho\mathcal{C}\}$.*

For the typical situation in ERM problems, where $\mathcal{C}$ is a symmetric convex body, $\|\cdot\|_{\mathcal{C}}$ defines a norm. The optimization problem solved in Step 9 of Algorithm PrivIncReg$_2$ is convex if $\mathcal{C}$ is convex, and hence can be efficiently solved. The existence of θ_t^{priv} follows from the following theorem.

Theorem 5.3. *[54] Let Φ be an $m \times d$ matrix, whose rows $\phi_1^\top, \ldots, \phi_m^\top$ are i.i.d. Gaussian random vectors in $\mathbb{R}^d$ chosen according to the standard normal distribution $\mathcal{N}(0, \mathbb{I}_d)$. Let $\mathcal{C}$ be a convex set. Given $\mathbf{v} = \Phi\mathbf{u}$ and Φ, let $\hat{\mathbf{u}}$ be the solution to the following convex program: $\min_{\mathbf{u}'\in\mathbb{R}^d} \|\mathbf{u}'\|_{\mathcal{C}}$ subject to $\Phi\mathbf{u}' = \mathbf{v}$. Then for any $\beta > 0$, with probability at least $1-\beta$,*

$$\sup_{\mathbf{u}:\mathbf{v}=\Phi\mathbf{u}} \|\mathbf{u} - \hat{\mathbf{u}}\| = O\left(\frac{w(\mathcal{C})}{\sqrt{m}} + \frac{\|\mathcal{C}\|\sqrt{\log(1/\beta)}}{\sqrt{m}}\right).$$

The next thing to be verified is that θ_t^{priv} generated by Algorithm PrivIncReg$_2$ is in $\mathcal{C}$. This is simple as by definition of Minkowski functional, as any closed set $\mathcal{C} = \{\theta \in \mathbb{R}^d : \|\theta\|_{\mathcal{C}} \le 1\}$. Hence, $\|\theta_t^{\text{true}}\|_{\mathcal{C}} \le 1$. By choice of θ_t^{priv} in Step 9, ensures that $\|\theta_t^{\text{priv}}\|_{\mathcal{C}} \le \|\theta_t^{\text{true}}\|_{\mathcal{C}} \le 1$, which guarantees that $\theta_t^{\text{priv}} \in \mathcal{C}$. Finally, note that the lifting is a post-processing operation on a differentially private output $\vartheta_t^{\text{priv}}$, and hence does not affect the differential privacy guarantee.

Utility Analysis of Algorithm PrivIncReg$_2$. In Lemma 5.4, by using the fact that the Gaussian noise for privacy (in the *Tree Mechanism*) is added on a lower dimensional (m) instance, we show that the difference between $\mathcal{L}_{\text{proj}}(\theta_t^{\text{priv}}; \Gamma_t; \Phi)$ and $\mathcal{L}_{\text{proj}}(\hat{\theta}_t; \Gamma_t; \Phi)$ (the minimum empirical risk) roughly scales as $\sqrt{m}$, for sufficiently large m. The Lipschitz constant of the function $\mathcal{L}_{\text{proj}}(\theta; \Gamma_t; \Phi)$ is $O(\|\Phi\mathcal{C}\|)$, which by Theorem 5.1, with probability at least $1-\beta$ is $O(\|\mathcal{C}\|)$, when

$$m = \Theta((1/\gamma^2)\max\{(w(\mathcal{X}) + w(\mathcal{C}))^2, \log(T/\beta)\}).$$

Let $\mathcal{E}_0$ be the event that the above Lipschitz bound holds.

Lemma 5.4. *For any $\beta > 0$, with probability at least $1-\beta$, for each $t \in [T]$, θ_t^{priv} generated by Algorithm PrivIncReg$_2$ satisfies:*

$$\begin{aligned}&\mathcal{L}_{\text{proj}}(\theta_t^{\text{priv}}; \Gamma_t; \Phi) - \mathcal{L}_{\text{proj}}(\hat{\theta}_t; \Gamma_t; \Phi)\\ &\qquad = O\left(\frac{\sqrt{m}\log^{3/2} T\sqrt{\log(1/\delta)}\|\mathcal{C}\|^2}{\epsilon}\right)\end{aligned}$$

where $\hat{\theta}_t \in \text{argmin}_{\theta\in\mathcal{C}} \sum_{i=1}^t (y_i - \langle\mathbf{x}_i, \theta\rangle)^2$.

Using properties of random projections, we now bound $\mathcal{L}_{\text{proj}}(\hat{\theta}_t; \Gamma_t; \Phi)$ in terms of $\mathcal{L}(\hat{\theta}_t; \Gamma_t)$.

Lemma 5.5. *Let Φ be a random matrix as defined in Theorem 5.1 with $m = \Theta((1/\gamma^2)\max\{(w(\mathcal{X})+w(\mathcal{C}))^2, \log(T/\beta)\})$, and let $\beta > 0$. Then with probability at least $1-\beta$, for each $t \in [T]$,*

$$\begin{aligned}\mathcal{L}_{\text{proj}}(\hat{\theta}_t; \Gamma_t; \Phi) \le \mathcal{L}(\hat{\theta}_t; \Gamma_t) &+ 4\gamma^2\|\mathcal{C}\|^2 t + 2\gamma\|\mathcal{C}\|\sqrt{t\mathcal{L}(\hat{\theta}_t; \Gamma_t)}\\ &+ 2\sqrt{2}\gamma^{3/2}\|\mathcal{C}\|^{3/2} t^{3/4}\mathcal{L}(\hat{\theta}_t; \Gamma_t)^{1/4}.\end{aligned}$$

The next step is to lower bound $\mathcal{L}_{\text{proj}}(\theta_t^{\text{priv}}; \Gamma_t; \Phi)$ in terms of $\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t)$.

Lemma 5.6. *For any $\beta > 0$, with probability at least $1-\beta$, for each $t \in [T]$, θ_t^{priv} generated by Algorithm PrivIncReg$_2$ satisfies:*

$$\begin{aligned}\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t) \le \mathcal{L}_{\text{proj}}(\theta_t^{\text{priv}}; \Gamma_t; \Phi) &+ 2\gamma\|\mathcal{C}\|\sqrt{T\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t)}\\ &+ 2\sqrt{2}\gamma^{3/2}\|\mathcal{C}\|^{3/2}T^{3/4}\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t)^{1/4} + 4\gamma^2\|\mathcal{C}\|^2 T.\end{aligned}$$

Putting together Lemmas 5.4, 5.5, and 5.6 and simple arithmetic manipulation gives: that with probability at least $1-\beta$, for each $t \in [T]$,

$$\begin{aligned}\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t) - \mathcal{L}(\hat{\theta}_t; \Gamma_t) &= O\left(\frac{\sqrt{m}\log^{3/2} T\sqrt{\log(1/\delta)}\|\mathcal{C}\|^2}{\epsilon}\right)\\ &+ 8\gamma^2\|\mathcal{C}\|^2 T + 2\gamma\|\mathcal{C}\|\sqrt{T\mathcal{L}(\hat{\theta}_t; \Gamma_t)}\\ &+ 2\gamma\|\mathcal{C}\|\sqrt{T\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t)} + 2\sqrt{2}\gamma^{3/2}\|\mathcal{C}\|^{3/2}T^{3/4}\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t)^{1/4}\\ &+ 2\sqrt{2}\gamma^{3/2}\|\mathcal{C}\|^{3/2}T^{3/4}\mathcal{L}(\hat{\theta}_t; \Gamma_t)^{1/4}. \quad (6)\end{aligned}$$

To simplify (6) in terms of its dependence on $\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t)$, start by noting that it is of the form $p - a\sqrt{p} - a^{3/2}\sqrt[4]{p} - b - 2a^2 \ge 0$, where

$$\begin{aligned}p &= \mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t),\\ a &= 2\gamma\|\mathcal{C}\|\sqrt{T}, \text{and}\\ b &= \mathcal{L}(\hat{\theta}_t; \Gamma_t) + O\left(\frac{\sqrt{m}\log^{3/2} T\sqrt{\log(1/\delta)}\|\mathcal{C}\|^2}{\epsilon}\right)\\ &\quad + 2\gamma\|\mathcal{C}\|\sqrt{T\mathcal{L}(\hat{\theta}_t; \Gamma_t)} + 2\sqrt{2}\gamma^{\frac{3}{2}}\|\mathcal{C}\|^{\frac{3}{2}}T^{\frac{3}{4}}\mathcal{L}(\hat{\theta}_t; \Gamma_t)^{\frac{1}{4}}.\end{aligned}$$

Solving for p from $p - a\sqrt{p} - a^{3/2}p^{1/4} - b - 2a^2 = 0$ (we use WolframAlpha solver [56]), and using that to simplify (6) yields, that with probability at least $1-\beta$, for each $t \in [T]$,

$$\begin{aligned}\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t) - \mathcal{L}(\hat{\theta}_t; \Gamma_t) = O(&\frac{\sqrt{m}\log^{3/2} T\sqrt{\log(1/\delta)}\|\mathcal{C}\|^2}{\epsilon}\\ &+\gamma^2\|\mathcal{C}\|^2 T + \gamma\|\mathcal{C}\|\sqrt{T\mathcal{L}(\hat{\theta}_t; \Gamma_t)} + \gamma^{3/2}\|\mathcal{C}\|^{3/2}T^{3/4}\mathcal{L}(\hat{\theta}_t; \Gamma_t)^{1/4}).\end{aligned} \quad (7)$$

Theorem 5.7. *Algorithm* PrivIncReg$_2$ *is* (ϵ, δ)*-differentially private with respect to a single datapoint change in the stream* Γ*. For any* $\beta > 0$*, with probability at least* $1-\beta$*, for each* $t \in [T]$*,* θ_t^{priv} *generated by Algorithm* PrivIncReg$_2$ *satisfies:*

$$\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t) - \mathcal{L}(\hat{\theta}_t; \Gamma_t) = O(\frac{T^{\frac{1}{3}} W^{\frac{2}{3}} \log^2 T \|\mathcal{C}\|^2 \sqrt{\log(1/\delta)\log(1/\beta)}}{\epsilon} + T^{\frac{1}{6}} W^{\frac{1}{3}} \|\mathcal{C}\| \sqrt{\mathcal{L}(\hat{\theta}_t; \Gamma_t)} + T^{\frac{1}{4}} W^{\frac{1}{2}} \|\mathcal{C}\|^{\frac{3}{2}} \sqrt[4]{\mathcal{L}(\hat{\theta}_t; \Gamma_t)}),$$

where $\hat{\theta}_t \in \min_{\theta \in \mathcal{C}} \mathcal{L}(\theta; \Gamma_t)$ *and* $W = w(\mathcal{X}) + w(\mathcal{C})$*.*

Remark 5.8. *Since* $\mathcal{L}$ *is an non-decreasing function in* t*, in the above theorem* $\mathcal{L}(\hat{\theta}_t; \Gamma_t)$ *in the right-hand side could be replaced by* $\mathcal{L}(\hat{\theta}_T; \Gamma_T)$ *(defined as* OPT *in the introduction).*

5.2 Discussion about Theorem 5.7

We start this discussion by mentioning a few instantiations of Theorem 5.7. Let $\text{OPT} = \mathcal{L}(\hat{\theta}_T; \Gamma_T)$ (remember, that $\text{OPT} \leq T$). For simplicity, below we ignore dependence on the privacy and confidence parameters.

For arbitrary $\mathcal{X}$ and $\mathcal{C}$, with just an L_2-diameter assumption as in Theorem 4.2, $W = w(\mathcal{X}) + w(\mathcal{C}) = O(\sqrt{d})$, and therefore the excess risk bound provided by Theorem 5.7 (accounting for the trivial excess risk bound of T) is $\tilde{O}(\min\{T^{1/3}d^{1/3} + T^{1/6}d^{1/6}\sqrt{\text{OPT}} + T^{1/4}d^{1/4}\sqrt[4]{\text{OPT}}, T\})$, which is worse than the $\tilde{O}(\min\{\sqrt{d}, T\})$ excess risk bound provided by Theorem 4.2. However, as we discuss below, for many interesting high-dimensional problems, one could get substantially better bounds using Theorem 5.7. This happens when W is "small".

In many practical regression instances, the input data is high-dimensional and sparse [38, 22, 59] which leads to a small $w(\mathcal{X})$. For example, if the $\mathbf{x}_i$'s are k-sparse, then $w(\mathcal{X}) = O(\sqrt{k \log(d/k)})$. Another common scenario is to have the $\mathbf{x}_i$'s come from a bounded L_1-diameter ball, in which case $w(\mathcal{X}) = O(\sqrt{\log d})$. Now under any of these assumptions, let us look at different choices of constraint spaces ($\mathcal{C}$) that have a small Gaussian width.

- If $\mathcal{C} = \text{conv}\{\mathbf{a}_1, \ldots, \mathbf{a}_l\}$ be convex hull of vectors $\mathbf{a}_i \in \mathbb{R}^d$, such that for all $i \in [d], \|\mathbf{a}_i\| \leq c$ with $c \in \mathbb{R}^+$, then $w(\mathcal{C}) = O(c\sqrt{\log l})$. A popular subcase of the above is the *cross-polytope* B_1^d (unit L_1-ball). For example, the popular Lasso formulation [50] used for high-dimensional linear regression is:

  $$\hat{\theta}_t \in \text{argmin}_{\theta \in cB_1^d} \sum_{i=1}^{t} (y_i - \langle \mathbf{x}_i, \theta \rangle)^2.$$

  Another popular subcase is the probability simplex where, $\mathcal{C} = \{\theta \in \mathbb{R}^d : \sum_i \theta_i = 1, \forall i \in [d], \theta_i \geq 0\}$.
- Group/Block L_1-norm is another prominent sparsity inducing norm used in many applications [1]. For a vector $\theta \in \mathbb{R}^d$, and a parameter k, this norm is defined as:[17]

  $$\|\theta\|_{k, L_{1,2}} = \sum_{i=1}^{\lceil d/k \rceil} \sqrt{\sum_{j=(i-1)k+1}^{\min\{ik, d\}} |\theta_j|^2}.$$

  If $\mathcal{C}$ denotes the convex set centered with radius one with respect to $\|\cdot\|_{k, L_{1,2}}$-norm then the Gaussian width of $\mathcal{C}$ is $O(\sqrt{k \log(d/k)})$ [47].
- L_p-balls ($1 < p < 2$) are another popular choice of constraint space [40]. The regression problem in this case is defined as:

  $$\hat{\theta}_t \in \text{argmin}_{\theta \in cB_p^d} \sum_{i=1}^{t} (y_i - \langle \mathbf{x}_i, \theta \rangle)^2,$$

  and $w(cB_p^d) = O(cd^{1-1/p})$.

As the reader may notice, in all of the above problem settings, W is much smaller than $\sqrt{d}$. As a comparison to the bound in Theorem 4.2, if $W = O(\text{polylog}(d))$, then Theorem 5.7 yields an excess risk bound of $\tilde{O}(T^{1/3} + T^{1/6}\sqrt{\text{OPT}} + T^{1/4}\sqrt[4]{\text{OPT}})$. This is significantly better than the $\tilde{O}(\sqrt{d})$ risk bound from Theorem 4.2 for many setting of T, d, and OPT, e.g., if $d \gg T^{4/3}$.

One could also compare the result from Theorem 5.7 to the bound obtained by applying the differentially private ERM algorithm of Talwar *et al.* [46] in the generic mechanism (Theorem 3.1, Part 3). It is hard to do a precise comparison because of the dependence on different parameters in these bounds. In general, when OPT is not very big (say $\ll T^{2/3}$) and $W = O(\text{polylog}(d))$ then the excess risk bound from Theorem 5.7 is significantly better than $\tilde{O}(\sqrt{T})$ risk bound obtained in Theorem 3.1, Part 3.

Extension to a case where not all inputs are drawn from a domain with small Gaussian Width. The previous analysis assumes that $w(\mathcal{X})$ is small (all inputs are drawn from a domain with small Gaussian Width). We now show that the techniques and results in the previous section extend to a more robust setting, where not all inputs are assumed to come from a domain with small Gaussian width.

In particular, we assume that there exists a set $\mathcal{G} \subseteq \mathcal{X}$, such that $w(\mathcal{G})$ is small, and only some inputs in the stream come from $\mathcal{G}$ (e.g., only a fraction of the covariates could be sparse). We also assume that the algorithm has access to an oracle which given a point $\mathbf{x} \in \mathcal{X}$, returns whether $\mathbf{x} \in \mathcal{G}$ or not. The goal of the algorithm is to perform private incremental linear regression on inputs from $\mathcal{G}$. In a non-private world, this is trivial as the algorithm can simply ignore when $\mathbf{x}_t$ is not in $\mathcal{G}$, but this operation is not private. However, a simple change to Algorithm PrivIncReg$_2$ can handle this scenario without a breach in the privacy. The idea is to check whether $\mathbf{x}_t \in \mathcal{G}$, if so $(\mathbf{x}_t, y_t)$ is used as currently in Algorithm PrivIncReg$_2$. Otherwise, $(\mathbf{x}_t, y_t)$ is replaced by $(\mathbf{0}, 0)$ before invoking Algorithm TreeMech (in Steps 5 and 6 of Algorithm PrivIncReg$_2$). With this change, the resulting algorithm is (ϵ, δ)-differentially private, and with probability at least $1-\beta$, for each $t \in [T]$, its output θ_t^{priv} will satisfy:

$$\sum_{\mathbf{x}_i \in \mathcal{G}, i \in [t]} (y_i - \langle \mathbf{x}_i, \theta_t^{\text{priv}} \rangle)^2 - \sum_{\mathbf{x}_i \in \mathcal{G}, i \in [t]} (y_i - \langle \mathbf{x}_i, \hat{\theta}_t \rangle)^2 = O(\frac{T^{\frac{1}{3}} W^{\frac{2}{3}} \log^2 T \|\mathcal{C}\|^2 \sqrt{\log(1/\delta)\log(1/\beta)}}{\epsilon} + T^{\frac{1}{6}} W^{\frac{1}{3}} \|\mathcal{C}\| \sqrt{\mathcal{L}(\hat{\theta}_t; \Gamma_t)} + T^{\frac{1}{4}} W^{\frac{1}{2}} \|\mathcal{C}\|^{\frac{3}{2}} \sqrt[4]{\mathcal{L}(\hat{\theta}_t; \Gamma_t)}),$$

where $\hat{\theta}_t \in \min_{\theta \in \mathcal{C}} \sum_{\mathbf{x}_i \in \mathcal{G}, i \in [t]} (y_i - \langle \mathbf{x}_i, \theta \rangle)^2$ and $W = w(\mathcal{G}) + w(\mathcal{C})$.

[17] There are generalizations of this norm that can handle different group (block) sizes.

References

[1] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[2] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Differentially private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*. IEEE, 2014.

[3] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 410–419. IEEE, 2012.

[4] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The SuLQ framework. In *PODS*, pages 128–138. ACM, 2005.

[5] Jean Bourgain, Dirksen Sjoerd, and Jelani Nelson. Toward a unified theory of sparse dimensionality reduction in euclidean space. In *Proceedings of the 47th ACM Symposium on Theory of Computing*. Association for Computing Machinery, 2015.

[6] John Canny and Huasha Zhao. Bidmach: Large-scale learning with zero memory allocation. In *BigLearning, NIPS Workshop*, 2013.

[7] T-H Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):26, 2011.

[8] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, pages 289–296, 2009.

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

[10] Kenneth L Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):63, 2010.

[11] Irit Dinur and Kobbi Nissim. Revealing Information while Preserving Privacy. In *PODS*, pages 202–210. ACM, 2003.

[12] John C Duchi, Michael Jordan, Martin J Wainwright, et al. Local privacy and statistical minimax rates. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 429–438. IEEE, 2013.

[13] John C Duchi, Michael I Jordan, and Martin J Wainwright. Privacy aware learning. *Journal of the ACM (JACM)*, 61(6):38, 2014.

[14] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, LNCS, pages 486–503. Springer, 2006.

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.

[16] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

[17] Cynthia Dwork, Moni Naor, Omer Reingold, and Guy Rothblum. Pure differential privacy for rectangle queries via private partitions. In *ASIACRYPT*, 2015.

[18] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Theoretical Computer Science*, 9(3-4):211–407, 2013.

[19] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 51–60. IEEE, 2010.

[20] Mahdi Milani Fard, Yuri Grinberg, Joelle Pineau, and Doina Precup. Compressed least-squares regression on sparse spaces. In *AAAI*, 2012.

[21] Yehoram Gordon. *On Milman's inequality and random subspaces which escape through a mesh in* $\mathbb{R}^n$. Springer, 1988.

[22] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[23] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *Learning theory*, pages 499–513. Springer, 2006.

[24] Justin Hsu, Zhiyi Huang, Aaron Roth, Tim Roughgarden, and Zhiwei Steven Wu. Private matchings and allocations. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 21–30. ACM, 2014.

[25] Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially private online learning. In *COLT 2012*, pages 24.1–24.34, 2012.

[26] Prateek Jain and Abhradeep Thakurta. Differentially private learning with kernels. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 118–126, 2013.

[27] Prateek Jain and Abhradeep Guha Thakurta. (near) dimension independent risk bounds for differentially private learning. In *Proceedings of The 31st International Conference on Machine Learning*, pages 476–484, 2014.

[28] Ata Kabán. New bounds on compressive linear least squares regression. In *The 17-th International Conference on Artificial Intelligence and Statistics (AISTATS 2014)*, volume 33, pages 448–456, 2014.

[29] Shiva Kasiviswanathan and Hongxia Jin. Efficient private empirical risk minimization for high-dimensional learning. In *ICML*, 2016.

[30] Shiva P Kasiviswanathan and Adam Smith. On the'semantics' of differential privacy: A bayesian

formulation. *Journal of Privacy and Confidentiality*, 6(1):1, 2014.

[31] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *FOCS*, pages 531–540. IEEE Computer Society, 2008.

[32] Shiva Prasad Kasiviswanathan, Mark Rudelson, and Adam Smith. The power of linear reconstruction attacks. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1415–1433. SIAM, 2013.

[33] Krishnaram Kenthapadi, Aleksandra Korolova, Ilya Mironov, and Nina Mishra. Privacy via the johnson-lindenstrauss transform. *Journal of Privacy and Confidentiality*, 5(1):39–71, 2013.

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

[35] Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: isoperimetry and processes*, volume 23. Springer Science & Business Media, 2013.

[36] Odalric Maillard and Rémi Munos. Compressed least-squares regression. In *Advances in Neural Information Processing Systems*, pages 1213–1221, 2009.

[37] Nikita Mishra and Abhradeep Thakurta. (nearly) optimal differentially private stochastic multi-arm bandits. In *UAI*, pages 592–601, 2015.

[38] John Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 117–126. IEEE, 2013.

[39] Götz E Pfander. *Sampling Theory, a Renaissance*. Springer, 2015.

[40] Azar Rahimi, Jingjia Xu, and Linwei Wang. -norm regularization in volumetric imaging of cardiac current sources. *Computational and mathematical methods in medicine*, 2013, 2013.

[41] Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *arXiv preprint arXiv:0911.5708*, 2009.

[42] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

[43] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *The Journal of Machine Learning Research*, 11:2635–2670, 2010.

[44] Or Sheffet. Private approximations of the 2nd-moment matrix using existing techniques in linear regression. *arXiv preprint arXiv:1507.00056*, 2015.

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

[46] Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Nearly optimal private lasso. In *Advances in Neural Information Processing Systems*, pages 3007–3015, 2015.

[47] Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Private empirical risk minimization beyond the worst case: The effect of the constraint set geometry. *arXiv preprint arXiv:1411.5417, To appear in NIPS*, 2015.

[48] Abhradeep Guha Thakurta and Adam Smith. Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Conference on Learning Theory*, pages 819–850, 2013.

[49] Abhradeep Guha Thakurta and Adam Smith. (nearly) optimal algorithms for private online learning in full-information and bandit settings. In *Advances in Neural Information Processing Systems*, pages 2733–2741, 2013.

[50] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[51] Jonathan Ullman. Private multiplicative weights beyond linear queries. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems*, pages 303–312. ACM, 2015.

[52] Jalaj Upadhyay. Randomness efficient fast-johnson-lindenstrauss transform with applications in differential privacy and compressed sensing. *arXiv preprint arXiv:1410.2470*, 2014.

[53] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[54] Roman Vershynin. Estimation in high dimensions: a geometric perspective. *arXiv preprint arXiv:1405.5103*, 2014.

[55] Oliver Williams and Frank McSherry. Probabilistic inference and differential privacy. In *Advances in Neural Information Processing Systems*, pages 2451–2459, 2010.

[56] WolframAlpha. `https://www.wolframalpha.com/examples/EquationSolving.html`, 2016.

[57] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and TrendsÂő in Theoretical Computer Science*, 10(1âĂŞ2):1–157, 2014.

[58] Shuheng Zhou, Katrina Ligett, and Larry Wasserman. Differential privacy with compression. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 2718–2722. IEEE, 2009.

[59] Shuheng Zhou, Larry Wasserman, and John D Lafferty. Compressed regression. In *Advances in Neural Information Processing Systems*, pages 1713–1720, 2008.

APPENDIX

A. Additional Preliminaries

We start by reviewing some standard definitions in convex optimization. In our setting, for a loss function $\jmath(\theta; \mathbf{z})$, all

the following properties (such as convexity, Lipschitzness, strong convexity) are defined with respect to the first argument θ.

In the following, we use the notation $\nabla\jmath(\theta;\mathbf{z})$ to denote the gradient (if it exists) or any subgradient of function $\jmath(\cdot;\mathbf{z})$ at θ.

Definition 7. *(Convex Functions) A loss function $\jmath : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}$ is convex with respect to θ over the domain $\mathcal{C}$ if for all $\mathbf{z} \in \mathcal{Z}$ the following inequality holds: $\jmath(\lambda\theta_a + (1-\lambda)\theta_b;\mathbf{z}) \leq \lambda\cdot\jmath(\theta_a;\mathbf{z}) + (1-\lambda)\cdot\jmath(\theta_b;\mathbf{z})$ for all $\theta_a, \theta_b \in \mathcal{C}$ and $\lambda \in [0,1]$. For a continuously differentiable $\jmath$, this inequality can be equivalently replaced with: $\jmath(\theta_b;\mathbf{z}) \geq \jmath(\theta_a;\mathbf{z}) + \langle\nabla\jmath(\theta_a;\mathbf{z}), \theta_b - \theta_a\rangle$ for all $\theta_a, \theta_b \in \mathcal{C}$, where $\nabla\jmath(\theta_a;\mathbf{z})$ is the gradient of $\jmath(\cdot;\mathbf{z})$ at θ_a.*

Definition 8. *(Lipschitz Functions) A loss function $\jmath : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}$ is L-Lipschitz with respect to θ over the domain $\mathcal{C}$, if for all $\mathbf{z} \in \mathcal{Z}$ and $\theta_a, \theta_b \in \mathcal{C}$, we have $|\jmath(\theta_a;\mathbf{z}) - \jmath(\theta_b;\mathbf{z})| \leq L\|\theta_a - \theta_b\|$. If $\jmath$ is a convex function, then $\jmath$ is L-Lipschitz iff for all $\theta \in \mathcal{C}$ and subgradients $\mathbf{g}$ of $\jmath$ at θ we have $\|\mathbf{g}\| \leq L$.*

Definition 9. *(Strongly Convex Functions) A loss function $\jmath : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}$ is ν-strongly convex if for all $\mathbf{z} \in \mathcal{Z}$ and $\theta_a, \theta_b \in \mathcal{C}$, all subgradients $\mathbf{g}$ of $\jmath(\theta_a;\mathbf{z})$, we have $\jmath(\theta_b;\mathbf{z}) \geq \jmath(\theta_a;\mathbf{z}) + \langle\mathbf{g}, \theta_b - \theta_a\rangle + (\nu/2)\|\theta_b - \theta_a\|^2$ (i.e., $\jmath$ is bounded below by a quadratic function tangent at θ_a).*

Gaussian Norm Bounds. Let $\mathcal{N}(0,\sigma^2)$ denote the Gaussian (normal) distribution with mean 0 and variance σ^2. We use the following standard result on the spectral norm (largest singular value) of an i.i.d. Gaussian random matrix throughout this paper.

Proposition A.1. *Let A be an $N \times n$ matrix whose entries are independent standard normal random variables. Then for every $t > 0$, with probability at least $1 - 2\exp(-t^2/2)$ one has, $\|A\| = O(\sqrt{N} + \sqrt{n} + t)$. In particular, with probability at least $1-\beta$, $\|A\| = O(\sqrt{N} + \sqrt{n} + \sqrt{\log(1/\beta)})$.*

A.1 Background on Linear Regression

Linear regression is a statistical method used to create a linear model. It attempts to model the relationship between two variables (known as covariate-response pairs) by fitting a linear function to observed data. More formally, given $\mathbf{y} = X\theta^\star + \mathbf{w}$, where $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$ is a vector of observed responses, $X \in \mathbb{R}^{n\times d}$ is the covariate matrix in which ith row $\mathbf{x}_i^\top$ represents the covariates (features) for the ith observation, and $\mathbf{w} = (w_1, \ldots, w_n)$ is a noise vector, the goal of linear regression is to estimate the unknown regression vector $\theta^\star$.

Assuming that the noise vector $\mathbf{w}$ follows a (sub)Gaussian distribution, estimating $\theta^\star$ amounts to solving the ordinary least-squares problem:

$$\hat{\theta} \in \operatorname{argmin}_{\theta\in\mathbb{R}^d} \|\mathbf{y} - X\theta\|^2 = \operatorname{argmin}_{\theta\in\mathbb{R}^d} \sum_{i=1}^{n} (y_i - \langle\mathbf{x}_i, \theta\rangle)^2.$$

Typically, for additional guarantees such as sparsity and stability, constraints are added to the least-squares estimator. This leads to the constrained linear regression formulation:

$$\text{Linear Regression:} \quad \hat{\theta} \in \operatorname{argmin}_{\theta\in\mathcal{C}} \|\mathbf{y} - X\theta\|^2 = \operatorname{argmin}_{\theta\in\mathcal{C}} \sum_{i=1}^{n} (y_i - \langle\mathbf{x}_i, \theta\rangle)^2, \quad (8)$$

for some convex set $\mathcal{C} \subseteq \mathbb{R}^d$. In this paper, we work with this formulation. Two well-known regression problems are obtained by choosing $\mathcal{C}$ as the L_2/L_1-ball:

$$\text{Ridge Regression:} \quad \hat{\theta} \in \operatorname{argmin}_{\theta\in cB_2^d} \sum_{i=1}^{n} (y_i - \langle\mathbf{x}_i, \theta\rangle)^2,$$

$$\text{Lasso Regression:} \quad \hat{\theta} \in \operatorname{argmin}_{\theta\in cB_1^d} \sum_{i=1}^{n} (y_i - \langle\mathbf{x}_i, \theta\rangle)^2,$$

where $c \in \mathbb{R}^+$. Another popular example is the Elastic-net regression which combines the Lasso and Ridge regression. Note that, while in this paper we focus on the constrained formulation of regression, from duality and the KKT conditions, the constrained formulation is equivalent to a penalized (regularized) formulation.

In the Streaming Setting. Let $\Gamma = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ denote a data stream. Let Γ_t denote the stream prefix of Γ of length t. Informally, the goal of *incremental linear regression* is to release at each timestep $t \in [T]$, $\theta_t \in \mathcal{C}$, that minimizes $\mathcal{L}(\theta;\Gamma_t) = \sum_{i=1}^{t} (y_i - \langle\mathbf{x}_i, \theta\rangle)^2$.

Definition 10 (Adaptation of Definition 1 for Incremental Linear Regression)**.** *A randomized streaming algorithm is* (α,β)-estimator *for incremental linear regression, if with probability at least $1-\beta$ over the coin flips of the algorithm, for each $t \in [T]$, after processing a prefix of the stream of length t, it generates an output $\theta_t \in \mathcal{C}$ that satisfies the following bound on excess (empirical) risk:*

$$\sum_{i=1}^{t} (y_i - \langle\mathbf{x}_i, \theta_t\rangle)^2 - \left(\min_{\theta\in\mathcal{C}} \sum_{i=1}^{t} (y_i - \langle\mathbf{x}_i, \theta\rangle)^2\right) \leq \alpha.$$

A.2 Background on Differential Privacy

In this section, we review some basic constructions in differential privacy. The literature on differential privacy is now rich with tools for constructing differentially private analyses, and we refer the reader to a survey by Dwork and Roth [18] for a comprehensive review of developments there.

One of the most basic technique, for achieving differential privacy is by adding noise to the outcome of a computed function where the noise magnitude is scaled to the *(global) sensitivity* of the function defined as:

Definition 11 (Sensitivity)**.** *Let f be a function mapping streams $\Gamma \in \mathcal{Z}^*$ to $\mathbb{R}^d$. The L_2-sensitivity Δ_2 of f is the maximum of $\|f(\Gamma) - f(\Gamma')\|$ over neighboring streams Γ, Γ'.*

Theorem A.2 (Framework of Global Sensitivity [15])**.** *Let $f : \mathcal{Z}^* \to \mathbb{R}^d$ be a function with L_2-sensitivity Δ_2. The algorithm that on an input Γ outputs $f(\Gamma) + Y$ where $Y \sim \mathcal{N}(0, (2\Delta_2^2 \ln(2/\delta))/\epsilon^2)^d$ is (ϵ,δ)-differentially private.*

Composition theorems for differential privacy allow a modular design of privacy preserving algorithms based on algorithms for simpler sub tasks:

Theorem A.3 ([14])**.** *A mechanism that permits k adaptive interactions with mechanisms that preserves (ϵ,δ)-differential privacy (and does not access the database otherwise) ensures $(k\epsilon, k\delta)$-differential privacy.*

A stronger composition is also possible as shown by Dwork *et al.* [19].

Theorem A.4 ([19])**.** *Let $\epsilon, \delta, \delta^* > 0$ and $\epsilon \leq 1$. A mechanism that permits k adaptive interactions with mechanisms that preserves (ϵ,δ)-differential privacy ensures $(\epsilon\sqrt{2k\ln(1/\delta^*)} + 2k\epsilon^2, k\delta + \delta^*)$-differential privacy.*

B. Proof of Theorem 3.1

Proof of Theorem 3.1. **Privacy Analysis.** Each $\mathbf{z}_i$ is accessed at most T/τ times by the algorithm invoked in Step 5. Let $l = T/\tau$. By using the composition Theorem A.4 with $\delta^* = \delta/2$ it follows that the entire algorithm is $(\epsilon'\sqrt{2l\ln(2/\delta)} + 2l\epsilon'^2, l\cdot(\delta/2l) + \delta/2)$-differentially private. We set ϵ' as $\epsilon/2 = \epsilon'\sqrt{2l\ln(2/\delta)}$. Note that with this setting of ϵ', $2l\epsilon'^2 \leq \epsilon/2$, therefore, $\epsilon'\sqrt{2l\ln(2/\delta)} + 2l\epsilon'^2 \leq \epsilon$. Hence, Mechanism PrivIncERM is (ϵ, δ)-differentially private.

Utility Analysis. If $T < \tau$, algorithm does not access the data, and in that case, the excess risk can be bounded by $TL\|\mathcal{C}\|$. Now assume $T \geq \tau$. Note that the algorithm performs no computation when t is not a multiple of τ. Let Γ_t denote the prefix of stream Γ till time t. Let t lie in the interval of $[j\tau, (j+1)\tau]$ for some $j \in \mathbb{N}$. The total loss accumulated by the algorithm at time t can be split as:

$$\sum_{i=1}^{t} \jmath(\theta_t^{\text{priv}}; \mathbf{z}_i) = \sum_{i=1}^{j\tau} \jmath(\theta_{j\tau}^{\text{priv}}; \mathbf{z}_i) + \sum_{i=j\tau+1}^{t} \jmath(\theta_{j\tau}^{\text{priv}}; \mathbf{z}_i)$$
$$\leq \sum_{i=1}^{j\tau} \jmath(\theta_{j\tau}^{\text{priv}}; \mathbf{z}_i) + \tau L\|\mathcal{C}\|,$$

as $\theta_t^{\text{priv}} = \theta_{j\tau}^{\text{priv}}$.

Let $\hat{\theta}_t \in \text{argmin}_{\theta\in\mathcal{C}} \sum_{i=1}^{t} \jmath(\theta; \mathbf{z}_i)$. As $\jmath$ is positive-valued,

$$\sum_{i=1}^{t} \jmath(\hat{\theta}_t; \mathbf{z}_i) \geq \sum_{i=1}^{j\tau} \jmath(\hat{\theta}_t; \mathbf{z}_i) \geq \sum_{i=1}^{j\tau} \jmath(\hat{\theta}_{j\tau}; \mathbf{z}_i).$$

Hence, we get,

$$\sum_{i=1}^{t} \jmath(\theta_t^{\text{priv}}; \mathbf{z}_i) - \sum_{i=1}^{t} \jmath(\hat{\theta}_t; \mathbf{z}_i) \leq \sum_{i=1}^{j\tau} \jmath(\theta_{j\tau}^{\text{priv}}; \mathbf{z}_i)$$
$$- \sum_{i=1}^{j\tau} \jmath(\hat{\theta}_{j\tau}; \mathbf{z}_i) + \tau L\|\mathcal{C}\|.$$

Using the results from Bassily *et al.* [2] or Talwar *et al.* [46] to bound $\sum_{i=1}^{j\tau} \jmath(\theta_{j\tau}^{\text{priv}}; \mathbf{z}_i) - \sum_{i=1}^{j\tau} \jmath(\hat{\theta}_{j\tau}; \mathbf{z}_i)$, setting τ to balance the various opposing terms, and a final union bound provide the claimed bounds. □

C. Noisy Projected Gradient Descent

In this section, we investigate the convergence rate of noisy projected gradient descent. Williams and McSherry first investigated gradient descent with noisy updates for probabilistic inference [55]; noisy stochastic variants of gradient descent have also been studied by [25, 12, 45, 2] in various private convex optimization settings. Other convex optimization techniques such as mirror descent [13, 47] and Frank-Wolfe scheme [47] have also been considered for designing private ERM algorithms.

For completeness, in this section, we present an analysis of the projected gradient descent procedure that operates with only access to a private gradient function (Definition 5). The analysis relies on standard ideas in convex optimization literature.

Consider the following constrained optimization problem,

$$\min_{\theta\in\mathcal{C}} f(\theta) \text{ where } \mathcal{C} \subseteq \mathbb{R}^d, \tag{9}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a convex function and $\mathcal{C}$ is some non-empty closed convex set. Define projection of $\theta \in \mathbb{R}^d$ onto a convex set $\mathcal{C}$ as:

$$P_{\mathcal{C}}(\theta) = \text{argmin}_{\mathbf{z}\in\mathcal{C}} \|\theta - \mathbf{z}\|^2.$$

Projected gradient descent algorithm uses the following update to solve (9)

ProjGrad$(\mathcal{C}, r)$: Initialize $\theta_1 \in \mathcal{C}$, Repeat r times:

$$\theta_{k+1} = P_{\mathcal{C}}(\theta_k - \eta_k \nabla f(\theta_k)), \text{ Output } \bar{\theta} = \frac{1}{r}\sum_{i=1}^{r} \theta_i, \tag{10}$$

for some stepsize η_k. Here $P_{\mathcal{C}}(\theta)$ defines the projection of θ onto $\mathcal{C}$.

Let $g : \mathbb{R}^d \to \mathbb{R}^d$ be an (α, β)-approximation of the true gradient of f (as in Definition 5),

$$\Pr\left[\max_{\theta\in\mathcal{C}} \|g(\theta) - \nabla f(\theta)\| > \alpha\right] \leq \beta.$$

The noisy projected gradient descent is a simple modification of the projected gradient descent algorithm (10), where instead of the true gradient, a noisy gradient is used. In other words, noisy projected gradient descent takes the form,

NoisyProjGrad$(\mathcal{C}, g, r)$: Initialize $\theta_1 \in \mathcal{C}$, Repeat r times:

$$\theta_{k+1} = P_{\mathcal{C}}(\theta_k - \eta_k g(\theta_k)), \text{ Output } \bar{\theta} = \frac{1}{r}\sum_{i=1}^{r} \theta_i. \tag{11}$$

The following proposition analyzes the convergence of the above NoisyProjGrad procedure assuming g is an (α, β)-approximation of the true gradient of f. Note that the proposition holds, even if f is not differentiable, in which case, $\nabla f(\theta)$ represents any subgradient of f at θ.

Proposition C.1. *Suppose f is convex and is L-Lipschitz. Let $\|\mathcal{C}\|$ be the diameter of $\mathcal{C}$. Let $g : \mathbb{R}^d \to \mathbb{R}^d$ be an (α, β)-approximation of the true gradient of f. Then after r steps of Algorithm* NoisyProjGrad*, starting with any $\theta_0 \in \mathcal{C}$ and constant stepsize of $\eta_k = \frac{\|\mathcal{C}\|}{\sqrt{r}(\alpha+L)}$, with probability at least $1 - r\beta$,*

$$f(\bar{\theta}) - f(\theta^*) \leq \frac{(\alpha+L)\|\mathcal{C}\|}{\sqrt{r}} + \alpha\|\mathcal{C}\|.$$

Proof. Let $\theta^* \in \mathcal{C}$ be the optimal solution of (9). Let $g(\theta_k) = \nabla f(\theta_k) + e(\theta_k)$. By Definition 8, $L \geq \max_{\theta\in\mathcal{C}} \|\nabla f(\theta)\|$.

$$\begin{aligned}
&\|x_{k+1} - \theta^*\|^2 = \|P_{\mathcal{C}}(\theta_k - \eta_k g(\theta_k)) - P_{\mathcal{C}}(\theta^*)\|^2 \\
&\leq \|\theta_k - \eta_k g(\theta_k) - \theta^*\|^2 \\
&\leq \|\theta_k - \theta^*\|^2 + 2\eta_k\langle g(\theta_k), \theta^* - \theta_k\rangle + \eta_k^2\|g(\theta_k)\|^2 \\
&= \|\theta_k - \theta^*\|^2 + 2\eta_k\langle \nabla f(\theta_k) + e(\theta_k), \theta^* - \theta_k\rangle \\
&\qquad + \eta_k^2\|\nabla f(\theta_k) + e(\theta_k)\|^2 \\
&\leq \|\theta_k - \theta^*\|^2 + 2\eta_k\langle \nabla f(\theta_k), \theta^* - \theta_k\rangle + 2\eta_k\|e(\theta_k)\|\|\mathcal{C}\| \\
&\qquad + \eta_k^2\|\nabla f(\theta_k) + e(\theta_k)\|^2 \\
&\leq \|\theta_k - \theta^*\|^2 + 2\eta_k(f(\theta^*) - f(\theta_k)) + 2\eta_k\|e(\theta_k)\|\|\mathcal{C}\| \\
&\qquad + \eta_k^2(\|\nabla f(\theta_k)\|^2 + \|e(\theta_k)\|^2 + 2\|\nabla f(\theta_k)\|\|e(\theta_k)\|) \\
&\leq \|\theta_k - \theta^*\|^2 + 2\eta_k(f(\theta^*) - f(\theta_k)) + 2\eta_k\|e(\theta_k)\|\|\mathcal{C}\| \\
&\qquad + \eta_k^2(L^2 + \|e(\theta_k)\|^2 + 2L\|e(\theta_k)\|).
\end{aligned}$$

The first inequality follows because projection cannot increase distances (projection operator is *contractive*).

By the assumption on $g(\theta_k)$, with probability at least $1-\beta$, $\|e(\theta_k)\| \leq \alpha$. Hence, with probability at least $1-\beta$,

$$\|\theta_{k+1} - \theta^*\|^2 \leq \|\theta_k - \theta^*\|^2 + 2\eta_k(f(\theta^*) - f(\theta_k)) + 2\eta_k\alpha\|\mathcal{C}\| + \eta_k^2(L^2 + \alpha^2 + 2\alpha L).$$

Summing the above expression and taking a union bound, yields that with probability at least $1 - r\beta$,

$$0 \leq \|\theta_{r+1} - \theta^*\|^2 \leq \|\theta_1 - \theta^*\|^2 + 2\sum_{k=1}^{r}\eta_k(f(\theta^*) - f(\theta_k)) + 2r\eta_k\alpha\|\mathcal{C}\| + r\eta_k^2(L^2 + \alpha^2 + 2\alpha L).$$

Rearranging the above, and setting constant step size $\eta_k = \frac{\|\mathcal{C}\|}{\sqrt{r}(\alpha+L)}$, with probability at least $1 - r\beta$,

$$\sum_{k=1}^{r}(f(\theta_k) - f(\theta^*)) \leq \|\mathcal{C}\|\sqrt{r}(\alpha + L) + r\alpha\|\mathcal{C}\|.$$

Now by Jensen's inequality,

$$f(\bar{\theta}) = f\left(\frac{1}{r}\sum_{k=1}^{r}\theta_k\right) \leq \frac{1}{r}\sum_{k=1}^{r}f(\theta_k).$$

Therefore,

$$f(\bar{\theta}) - f(\theta^*) \leq \frac{1}{r}\sum_{k=1}^{r}(f(\theta_k) - f(\theta^*)) \leq \frac{(\alpha+L)\|\mathcal{C}\|}{\sqrt{r}} + \alpha\|\mathcal{C}\|.$$

□

Corollary C.2. *By setting $r = \frac{(\alpha+L)^2\|\mathcal{C}\|^2}{\zeta^2}$, in Proposition C.1 gives that with probability at least $1 - r\beta$,*

$$f(\bar{\theta}) - f(\theta^*) \leq \zeta + \alpha\|\mathcal{C}\|.$$

If $\alpha > 0$, then we can set $r = \left(1 + \frac{L}{\alpha}\right)^2$ gives $f(\bar{\theta}) - f(\theta^*) \leq 2\alpha\|\mathcal{C}\|$.

After constructing the private gradient function, evaluating the gradient function at any θ can be done without affecting the privacy budget, as this is just a post-processing of private outputs.

D. Tree Mechanism for Continually Releasing Private Sums

Given a bit stream $b_1, \ldots, b_T \in \{0,1\}$, the *private streaming counter* problem is to release at every timestep t, (an approximation to) $\sum_{i=1}^{t} b_i$ while satisfying differential privacy (Definition 4). Chan *et al.* [7] and Dwork *et al.* [16] proposed an elegant differentially private mechanism (referred to as the *Tree Mechanism*) for this problem. We use this mechanism as a basic building block in our private incremental regression algorithms.[18]

For completeness, in Algorithm TreeMech, we present the entire *Tree Mechanism* as applied to a set of vectors. Given a data stream $\Upsilon = \upsilon_1, \ldots, \upsilon_T \in \mathcal{Z}$, the algorithm releases at each timestep t, (an approximation to) the sum function $\sum_{i=1}^{t}\upsilon_i$, while satisfying differential privacy.

Algorithm 4: TreeMech $(\epsilon, \delta, \Delta_2)$

Input: A stream $\Upsilon = \upsilon_1, \ldots, \upsilon_T$, where each υ_t is from the domain $\mathcal{Z} \subseteq \mathbb{R}^d$, and $\Delta_2 = \max_{\upsilon,\upsilon' \in \mathcal{Z}} \|\upsilon - \upsilon'\|$

Output: A differentially private estimate of $\sum_{i=1}^{t}\upsilon_i$ at every timestep $t \in [T]$

for *all* $t \in [T]$ **do**
  Express $t = \sum_{j=0}^{\log t} 2^j \mathrm{Bin}_j(t)$ (where $\mathrm{Bin}_j(t)$ is the bit at the jth index in the binary representation of t)
  $i \leftarrow \min_{0 \leq j \leq \log T}\{\mathrm{Bin}_j(t) \neq 0\}$
  $\mathbf{a}_i \leftarrow \sum_{j<i}\mathbf{a}_j + \upsilon_t$
  for $0 \leq j \leq i-1$ **do**
    $\mathbf{a}_j \leftarrow \mathbf{0}$ and $\mathbf{b}_j \leftarrow \mathbf{0}$
  end
  $\mathbf{b}_i \leftarrow \mathbf{a}_i + \mathcal{N}\left(\mathbf{0}, \frac{2\log^2(T)\Delta_2^2\ln(2/\delta)\mathbb{I}_d}{\epsilon^2}\right)$
  $\mathbf{s}_t \leftarrow \sum_{j:\mathrm{Bin}_j(t)\neq 0}\mathbf{b}_j$
  Return $\mathbf{s}_t$
end

Algorithm TreeMech can be viewed as releasing partial sums of different ranges at each timestep t and computing the final sum is simply a post-processing of the partial sums. At most $\log T$ partial sums are used for constructing each private sum. The following theorem follows by using the standard upper deviation inequality for Gaussian random variables (Proposition A.1) in the analysis of *Tree Mechanism* from [16, 7]. Another advantage of this mechanism is that it can be implemented with small memory, as only $O(\log t)$ partial sums are needed at any time t.

Proposition D.1. *Algorithm* TreeMech *is (ϵ, δ)-differentially private with respect to a single datapoint change in the stream Υ. For any $\beta > 0$ and $t \in [T]$, with probability at least $1-\beta$, $\mathbf{s}_t$ computed by Algorithm* TreeMech *satisfies:*

$$\left\|\mathbf{s}_t - \sum_{i=1}^{t}\upsilon_i\right\| = O\left(\frac{\Delta_2(\sqrt{d} + \sqrt{\log(1/\beta)})\log^{3/2}T\sqrt{\log(1/\delta)}}{\epsilon}\right),$$

where Δ_2 is the L_2-sensitivity of the sum function from Definition 11.

E. Missing Proofs from Section 5

Proof of Lemma 5.4. Let us condition on event $\mathcal{E}_0$ to hold. By definition,

$$\min_{\vartheta \in \Phi\mathcal{C}} \sum_{i=1}^{t}\left(y_i - \frac{\|\mathbf{x}_i\|}{\|\Phi\mathbf{x}_i\|}\langle\Phi\mathbf{x}_i, \vartheta\rangle\right)^2 \equiv \min_{\theta \in \mathcal{C}} \sum_{i=1}^{t}\left(y_i - \frac{\|\mathbf{x}_i\|}{\|\Phi\mathbf{x}_i\|}\langle\Phi\mathbf{x}_i, \Phi\theta\rangle\right)^2.$$

[18] Dwork *et al.* [17] have recently improved the bounds for the private streaming counter problem in the case where the bit stream is *sparse*, i.e., have many fewer 1's than 0's.

Since the inputs are m-dimensional and $\vartheta_t^{\text{priv}} = \Phi\theta_t^{\text{priv}}$, using an analysis similar to Theorem 4.2 gives that, with probability at least $1-\beta$, for each $t \in [T]$,

$$\sum_{i=1}^{t}\left(y_i - \frac{\|\mathbf{x}_i\|}{\|\Phi\mathbf{x}_i\|}\langle\Phi\mathbf{x}_i, \Phi\theta_t^{\text{priv}}\rangle\right)^2 - \min_{\theta\in\mathcal{C}}\sum_{i=1}^{t}\left(y_i - \frac{\|\mathbf{x}_i\|}{\|\Phi\mathbf{x}_i\|}\langle\Phi\mathbf{x}_i, \Phi\theta\rangle\right)^2 = O\left(\frac{\log^{3/2} T\sqrt{\log(1/\delta)}\|\mathcal{C}\|^2(\sqrt{m}+\sqrt{\log(T/\beta)})}{\epsilon}\right).$$

In other words, with probability at least $1-\beta$, for each $t \in [T]$,

$$\mathcal{L}_{\text{proj}}(\theta_t^{\text{priv}}; \Gamma_t; \Phi) - \min_{\theta\in\mathcal{C}}\mathcal{L}_{\text{proj}}(\theta; \Gamma_t; \Phi) = O\left(\frac{\log^{3/2} T\sqrt{\log(1/\delta)}\|\mathcal{C}\|^2(\sqrt{m}+\sqrt{\log(T/\beta)})}{\epsilon}\right).$$

By noting that $\min_{\theta\in\mathcal{C}}\mathcal{L}_{\text{proj}}(\theta; \Gamma_t; \Phi) \le \mathcal{L}_{\text{proj}}(\hat{\theta}_t; \Gamma_t; \Phi)$ and removing the conditioning on $\mathcal{E}_0$ (by adjusting β), completes the proof. □

Proof of Lemma 5.5. Fix a $t \in [T]$. From Theorem 5.1, for the chosen value of m, with probability at least $1-\beta$,

$$\begin{aligned}
&\mathcal{L}(\hat{\theta}_t; (\tilde{\mathbf{x}}_1, y_1), \ldots, (\tilde{\mathbf{x}}_t, y_t)) = \sum_{i=1}^{t}(y_i - \langle\tilde{\mathbf{x}}_i, \hat{\theta}_t\rangle)^2 \\
&= \sum_{i=1}^{t}\left(y_i - \frac{\|\mathbf{x}_i\|}{\|\Phi\mathbf{x}_i\|}\langle\mathbf{x}_i, \hat{\theta}_t\rangle\right)^2 \\
&\le \sum_{i=1}^{t}(|y_i - \langle\mathbf{x}_i, \hat{\theta}_t\rangle| + \gamma\|\hat{\theta}_t\|)^2 \le \sum_{i=1}^{t}(|y_i - \langle\mathbf{x}_i, \hat{\theta}_t\rangle| + \gamma\|\mathcal{C}\|)^2 \\
&\le \mathcal{L}(\hat{\theta}_t; \Gamma_t) + \gamma^2\|\mathcal{C}\|^2 t + 2\gamma\|\mathcal{C}\|\sum_{i=1}^{t}|y_i - \langle\mathbf{x}_i, \hat{\theta}_t\rangle| \\
&\le \mathcal{L}(\hat{\theta}_t; \Gamma_t) + \gamma^2\|\mathcal{C}\|^2 t + 2\gamma\|\mathcal{C}\|\sqrt{t\mathcal{L}(\hat{\theta}_t; \Gamma_t)}. \qquad (12)
\end{aligned}$$

Consider $\mathcal{L}_{\text{proj}}(\hat{\theta}_t; \Gamma_t; \Phi)$. From Corollary 5.2, for the chosen value of m, with probability at least $1-\beta$,

$$\begin{aligned}
&\mathcal{L}_{\text{proj}}(\hat{\theta}_t; \Gamma_t; \Phi) = \sum_{i=1}^{t}(y_i - \langle\Phi\tilde{\mathbf{x}}_i, \Phi\hat{\theta}_t\rangle)^2 \\
&\le \sum_{i=1}^{t}(|y_i - \langle\tilde{\mathbf{x}}_i, \hat{\theta}_t\rangle| + \gamma\|\hat{\theta}_t\|)^2 \\
&\le \sum_{i=1}^{t}(|y_i - \langle\tilde{\mathbf{x}}_i, \hat{\theta}_t\rangle| + \gamma\|\mathcal{C}\|)^2 \\
&= \mathcal{L}(\hat{\theta}_t; (\tilde{\mathbf{x}}_1, y_1), \ldots, (\tilde{\mathbf{x}}_t, y_t)) + \gamma^2\|\mathcal{C}\|^2 t + 2\gamma\|\mathcal{C}\|\sum_{i=1}^{t}|y_i - \langle\tilde{\mathbf{x}}_i, \hat{\theta}_t\rangle| \\
&\le \mathcal{L}(\hat{\theta}_t; (\tilde{\mathbf{x}}_1, y_1), \ldots, (\tilde{\mathbf{x}}_t, y_t)) + \gamma^2\|\mathcal{C}\|^2 t \\
&\qquad + 2\gamma\|\mathcal{C}\|\sqrt{t\mathcal{L}(\hat{\theta}_t; (\tilde{\mathbf{x}}_1, y_1), \ldots, (\tilde{\mathbf{x}}_t, y_t))},
\end{aligned}$$

where the first inequality is by application of Corollary 5.2. Substituting the result from (12) and taking a union bound over all $t \in [T]$ (i.e., replacing β by β/T), completes the proof. □

Proof of Lemma 5.6. Fix a $t \in [T]$. For the chosen value of m,

$$\Pr\left[|\|\Phi\mathbf{x}_i\| - \|\mathbf{x}_i\|| \ge \gamma\|\mathbf{x}_i\| \text{ for all } i \in [t]\right] \le \beta.$$

Similarly for the chosen value of m by using (5),

$$\Pr\left[\left|\langle\Phi\mathbf{x}_i, \Phi\theta_t^{\text{priv}}\rangle - \langle\mathbf{x}_i, \theta_t^{\text{priv}}\rangle\right| \ge \gamma\|\mathbf{x}_i\|\|\theta_t^{\text{priv}}\| \text{ for all } i \in [t]\right] \le \beta.$$

Using arguments as in Lemma 5.5, but by focusing on the lower bounds, we get that with probability at least $1-\beta$,

$$\begin{aligned}
\mathcal{L}_{\text{proj}}(\theta_t^{\text{priv}}; \Gamma_t; \Phi) \ge \mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t) - 2\gamma\|\mathcal{C}\|\sqrt{T\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t)} \\
- 2\sqrt{2}\gamma^{3/2}\|\mathcal{C}\|^{3/2}T^{3/4}\mathcal{L}(\theta_t^{\text{priv}}; \Gamma_t)^{1/4} - 4\gamma^2\|\mathcal{C}\|^2 T.
\end{aligned}$$

Taking a union bound over all $t \in [T]$ completes the proof. □

Proof of Theorem 5.7. The (ϵ, δ)-differential privacy follows from the established global sensitivity bound.

For the utility analysis, we start from (7), and substitute $\gamma = (w(\mathcal{X}) + w(\mathcal{C}))^{1/3}/T^{1/3}$ to get the claimed bound. The value of γ is picked to balance the various opposing factors. □