SoK: A Review of Differentially Private Linear Models For High-Dimensional Data

Amol Khanna
Booz Allen Hamilton
Annapolis Junction, MD, USA
Khanna_Amol@bah.com

Edward Raff
Booz Allen Hamilton
University of Maryland, Baltimore County
Annapolis Junction, MD, USA
Raff_Edward@bah.com

Nathan Inkawhich
Air Force Research Laboratory
Rome, NY, USA
Nathan.Inkawhich@us.af.mil

***Abstract*—Linear models are ubiquitous in data science, but are particularly prone to overfitting and data memorization in high dimensions. To guarantee the privacy of training data, differential privacy can be used. Many papers have proposed optimization techniques for high-dimensional differentially private linear models, but a systematic comparison between these methods does not exist. We close this gap by providing a comprehensive review of optimization methods for private high-dimensional linear models. Empirical tests on all methods demonstrate robust and coordinate-optimized algorithms perform best, which can inform future research. Code for implementing all methods is released online.**

***Index Terms*—differential privacy, high-dimensional, linear regression, logistic regression**

I. Introduction

Linear models, like linear and logistic regression, are ubiquitous in current data science and data analytics efforts. Their simple structure enables them to train quickly and generalize well on simple problems. Additionally, they can be interpreted easily to understand model decision making, which is essential in regulated fields like medicine and finance. For example, linear regression has been used to predict future consumer demand, corporate resource requirements, and house prices [1], [2]. Logistic regression has been used for disease prediction and fraud detection [3], [4].

In the low-dimensional regime, where a dataset has many more datapoints than features, linear models typically generalize well without significant tuning. From an information-theoretic perspective, this is because the model has enough data to learn major trends in the dataset, which should generalize to future instances. This is why linear models are often most effective on simple, large datasets [5].

However, many modern datasets are high-dimensional, having more features than datapoints. This is common in genomics or finance, where the expression of many genes or prices of many assets outnumber individual observations. In this case, linear models can overfit to the data, producing poor generalization to future inputs [6].

A common solution to this problem uses regularization, which constrains the weight vector of these models. For example, a common goal is to constrain the number of nonzero coefficients of the model's weight. This both promotes generalization and produces a weight vector which highlights the features which are most strongly related to the model's output [7].

However, linear models' solutions can produce weights which rely on information contained in one or a few datapoints. This can be problematic for models trained on sensitive data, such as those used in medicine or banking, since the parameters of these models can leak information about these datapoints. Specifically, extensive research has demonstrated that membership inference attacks can reconstruct datapoints used in models with high accuracy given access to only the model's outputs or parameters. This is especially true in the high-dimensional regime, where models are easily capable of overfitting to data [8].

A theoretical solution to this problem employs differential privacy. Differential privacy is a statistical guarantee of the privacy of an algorithm's outputs, which ensures that an algorithm does not rely too heavily on information in any one datapoint. Differential privacy prevents membership inference and dataset reconstruction attacks on a trained model [9].

Deferentially private linear and logistic regression have been studied extensively in the past two decades, but primarily from a theoretical perspective. Many optimization strategies and heuristics have been developed, but for most important statistical tasks, systematic comparisons between different algorithms have not been conducted.

This paper reviews methods to develop differentially private high-dimensional linear models. This task is fundamental in private statistics since linear models are typically the first models tested in a data science pipeline. Since differential privacy works best with simple models, using an appropriately trained linear model can avoid false conclusions on linear models' ineffectiveness for certain problems. Finally, differential privacy typically struggles with high-dimensional problems since noise can overwhelm the signals in these problems. This review seeks to identify whether specific optimization methods consistently improve performance on private high-dimensional problems.

Specifically, this review contributes the following:

1) We provide the first centralized review of all methods

Approved for Public Release; Distribution Unlimited. PA Number: AFRL-2023-5408

performing high-dimensional linear modeling with differential privacy, and provide insights about methods' strengths, weaknesses, and assumptions.

2) We implement methods reviewed in code and release all code at https://github.com/afrl-ri/differential-privacy-review. Many reviewed works do not release code, which can make it challenging for their methods to be tested and improved upon by other works. We close this gap.
3) We provide a systematic empirical comparison of each method's performance on a variety of datasets. Current differential privacy literature typically compares methods' theoretical utility. While this is a useful metric, empirical performance is often overlooked. In conducting these empirical tests, we find surprising and previously unidentified trends in performance which can influence future research.

Finally, we remark that in this article, the term "linear models" refers to both linear regression, where the output is an unconstrained numeric prediction (i.e., the mean-squared error loss), and logistic regression, where a binary label is desired. Most proposed methods in the literature support both, but some linear regression and logistic regression specific methods will also be evaluated.

II. Preliminaries

In this section we provide brief overviews of differential privacy and nonprivate optimization methods for high-dimensional linear models. This provides context for our literature review in section III.

A. *Differential Privacy*

Differential privacy (DP) is a statistical guarantee of the privacy of an algorithm's outputs. Specifically, given an algorithm $\mathcal{A}$, it is $(\epsilon, 0)$ DP if for all datasets $\mathcal{D}$ and $\mathcal{D}'$ differing on one datapoint and any event E,

$$\frac{\mathbb{P}[\mathcal{A}(\mathcal{D}) \in E]}{\mathbb{P}[\mathcal{A}(\mathcal{D}') \in E]} \leq e^{\epsilon}.$$

In other words, DP bounds the amount $\mathcal{A}$'s output can depend on a single datapoint. Since it cannot rely too much on any individual datapoint, it cannot reveal significant information about a single individual in a dataset [9].

Implementing the above definition of DP for a nonprivate algorithm $\mathcal{B}$ requires bounding its L_1 sensitivity. This is the maximum amount that $\mathcal{B}$'s output can change between two neighboring datasets as measured by the L_1 norm. Mathematically, this is written as

$$\max_{\mathcal{D},\mathcal{D}':\ \text{dist}(\mathcal{D},\mathcal{D}')=1} \|\mathcal{B}(\mathcal{D}) - \mathcal{B}(\mathcal{D}')\|_1.$$

Once the sensitivity is known, Laplacian noise with appropriate scale can be added to $\mathcal{B}$ to produce a DP version of $\mathcal{B}$ [10]. However, adding Laplacian noise to $\mathcal{B}$ using its L_1 sensitivity can destroy its efficacy. The noise can corrupt its outputs so much that it becomes useless. To address this issue, approximate DP was developed.

Approximate differential privacy, otherwise known as (ϵ, δ) DP, modifies the above definition of DP to

$$\mathbb{P}[\mathcal{A}(\mathcal{D}) \in E] \leq e^{\epsilon}\mathbb{P}[\mathcal{A}(\mathcal{D}') \in E] + \delta,$$

where $0 < \delta \leq 1$. In this case, a nonprivate algorithm $\mathcal{B}$ can be made (ϵ, δ) DP by adding noise from a normal distribution with scale proportional to $\mathcal{B}$'s L_2 sensitivity. This method can add significantly less noise to $\mathcal{B}$'s outputs because the Gaussian distribution's tails are significantly lighter than the Laplace distribution [11].

Finally, we mention some important properties of DP employed in the methods reviewed in section III. We begin with the post-processing property, which states that if $\mathcal{A}(\mathcal{D})$ is DP, then for any function g which does not re-access the dataset $\mathcal{D}$, $g(\mathcal{A}(\mathcal{D}))$ is DP with the same privacy parameters [10].

Next, we discuss composition. Say $\mathcal{A}(\mathcal{D})$ is (ϵ, δ) DP, where $0 \leq \delta \leq 1$. Then under parallel composition, where $\mathcal{D}$ is broken into disjoint sets such that $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_k = \mathcal{D}$, releasing all outputs $\mathcal{A}(\mathcal{D}_1), \ldots \mathcal{A}(\mathcal{D}_k)$ is DP with the same parameters as $\mathcal{A}(\mathcal{D})$ [12]. In contrast, under sequential composition, if $\mathcal{A}_1(\mathcal{D})$ is (ϵ_1, δ_1) DP and $\mathcal{A}_2(\mathcal{D})$ is (ϵ_2, δ_2) DP, then releasing $\mathcal{A}_1(\mathcal{D})$ and $\mathcal{A}_2(\mathcal{D})$ is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ DP [13].

Note that the above bound for sequential composition is called the basic composition theorem. Employing concentration inequalities, using a small value δ', the advanced composition theorem states that we can produce a sublinear increase in ϵ. For example, for k-fold adaptive composition under $(\epsilon, 0)$ DP, the advanced composition theorem states that the output is (ϵ', δ') DP, where $\epsilon' = 2\epsilon\sqrt{2k\log(1/\delta')}$. Similarly, for adaptive composition of (ϵ, δ) DP algorithms, the advanced composition theorem shows a k-fold composition is $(\epsilon', k\delta + \delta')$ DP [13].

The advanced composition theorem is tight for $(\epsilon, 0)$ DP, meaning that it produces the lowest privacy parameters which can generally be applied to any $(\epsilon, 0)$ DP mechanism. However, advanced composition is not tight for (ϵ, δ) DP where $\delta > 0$ [14]. This has led to the adoption of Rényi and zero concentrated DP, reformulations of DP which are naturally able to produce tight Gaussian mechanism composition for (ϵ, δ) mechanisms [15], [16].

Finally, many DP mechanisms have been developed, but here we mention one specifically: the report-noisy-max mechanism. The report-noisy-max mechanism is one application of the exponential mechanism, and it takes in a set of scores and is used to choose the object with the highest score. It is typically implemented by adding Laplacian noise to each of the scores and then choosing the one which is largest [17]. The report-noisy-max mechanism has found wide use in DP literature, and is used by some of the algorithms we review in section III.

1) Global and Local Differential Privacy: Traditional DP algorithms assume that individuals' raw datapoints are colocated on one machine. Only resulting analyses or models built from the data will be shared publicly with untrusted agents, so only these analyses or models must be noised. This framework is called *global* or *central* DP, in reference to the assumption that all individuals trust one entity to curate their raw data [18].

There are a few common methods to achieving DP under the global assumption. The first is output perturbation, in which the output of an algorithm is noised based on its sensitivity. This method is used often with traditional statistical estimators such as the mean, but it can be difficult to calculate the sensitivity of more complicated outputs like machine learning models [19], [20].

Another method is objective perturbation, which noises the objective function of an optimization problem. This method is flexible to modeling algorithms, but original works required the optimization problem to be solved exactly, which is challenging. More recent frameworks have extended objective perturbation to approximate solutions of optimization problems [20], [21].

Finally, gradient perturbation privatizes algorithms using gradient descent by noising each step of the optimization process. This method has been shown to work better than output and objective perturbation on many problems, and it is easily modified for the stochastic gradient descent case [22]. Its popularity is bolstered by the developments of Réyni and zero-concentrated DP, which allow for tight composition of gradient perturbation methods in the non-stochastic and stochastic settings [15], [16].

However, when individuals do not trust a central actor to curate their data, local DP must be used. In this case, individuals noise data prior to sending it to a central server, thus ensuring that curators of the central server cannot re-identify the data's sensitive attributes. A number of noising algorithms exist, but the main drawback of local DP is reduced accuracy and utility [18].

To combat reduced accuracy and utility while avoiding trust of the central actor, shuffle differential privacy was developed. In shuffle differential privacy, each individual's data is shuffled prior to being provided to the actor so it cannot be readily linked back to a single user through a specific connection. This allows for reducing the noise added to each individual's data, which can improve accuracy and utility. It can be shown that shuffle differential privacy is a stronger than global differential privacy and weaker than local differential privacy [23]. One common specific application of shuffle differential privacy is aggregate differential privacy, which sums many local differentially private metrics prior to providing this result to an untrusted actor. Aggregate differential privacy has been used in real-world systems for tracking user statistics and application performance [24].

In this review, we mention methods for high-dimensional private linear modeling under both the global and local models. However, since methods for high-dimensional linear modeling under local DP are nascent, we only test models built under global DP. Finally, note that we are not aware of any methods which employ shuffle differential privacy for high-dimensional linear models.

2) Sparsity, Stability, and Differential Privacy: Heuristically, an algorithm which is stable produces similar outputs on similar datasets. Algorithmic stability is a well-studied property in statistics and machine learning as it enables a mathematical analysis of algorithm performance [25].

DP algorithms are inherently related to algorithmic stability, as a DP algorithm cannot have a large change in its output when one of its datapoints changes. Indeed, Dwork and Lei formalized this intuition when they studied the relationship between DP and robust statistics [12].

Unfortunately, Xu et al. found that sparse algorithms cannot be stable [25]. Applying this to our case of DP, this means that, in some sense, DP algorithms are at odds with sparsity.

We include this discussion here to highlight the fundamental challenge of producing a sparse DP estimator. While both sparsity and DP are useful tools for data analysis, an effective combination of both is a difficult and open problem. Some of the methods reviewed in this work take steps to resolve this problem.

B. High-Dimensional Optimization for Linear Models

In this section, we provide a brief overview of high-dimensional optimizers for linear and logistic regression. Optimization methods for high-dimensional problems are an important open problem, so we focus on well-established methods relevant to section III here.

First, note that in high-dimensional problems, it is easy for models to overfit the training data. If n is the number of datapoints and d is the number of features, high-dimensional problems have $n < d$, so the number of coefficients in the weight vector exceeds the number of targets [26]. For this reason, we must constrain the weight space to be sparse avoid overfitting [27]. Note that literature on nonprivate optimization often considers sparsity to be a de-facto requirement for high-dimensional problems since it improves utility of solutions greatly; however, as discussed in the prior section, maintaining sparsity with DP can be challenging.

Two common approaches are used to avoid overfitting in high-dimensional linear models. The first is an L_0 contraint on the weight vector, which controls the number of nonzero components in the weight vector. Under this constraint, a linear problem with per-example loss function $\ell(\mathbf{x}_i, y_i; \mathbf{w}_i)$ can be written as

$$\underset{\mathbf{w}:\ \|\mathbf{w}\|_0 \leq k}{\arg\min} \sum_{i=1}^{N} \ell(\mathbf{x}_i, y_i; \mathbf{w}_i).$$

To solve this problem, iterative gradient hard thresholding can be used. This is a greedy method which retains only the top-k coefficients of the weight vector after each gradient step. Since it is greedy, it is not guaranteed to converge to an optimal solution. However, it approaches the L_0 constrained problem in a computationally efficient manner [28].

L_1 constraints are used more often. In this setting, we solve

$$\underset{\mathbf{w}:\ \|\mathbf{w}\|_1 \leq k}{\arg\min} \sum_{i=1}^{N} \ell(\mathbf{x}_i, y_i; \mathbf{w}_i).$$

This is a relaxation of the L_0 constraint which often produces sparse solutions and can be solved efficiently with a number of optimizers; however, the nonzero components of its solutions are biased towards 0, which can reduce the utility of the output [29].

Solving this problem can be done with a number of optimizers. Most commonly, it is solved with the Frank-Wolfe algorithm, which iteratively takes steps towards the vertex of the k-norm L_1-ball which has minimum loss [30]. Compressed learning, in which the dimensionality of the dataset is reduced by multiplication with a random matrix, can also be used [31].

Note that the L_1 constrained problem is often written as an L_1 regularized problem. Specifically, a linear problem can be written as

$$\arg\min_{\mathbf{w}} \sum_{i=1}^{N} \ell(\mathbf{x}_i, y_i; \mathbf{w}_i) + \lambda \|\mathbf{w}\|_1.$$

There is an exact duality between the constrained and regularized setting, where a specific value of k corresponds to a value of λ. However, the closed form conversion from k to λ is not known [32].

To solve the regularized problem, coordinate descent can be used. This algorithm iterates through each coefficient of the weight vector, making gradient updates for only that coefficient. More recent algorithms like the alternating direction method of multipliers can also be used, which split the smooth and nonsmooth parts of the regularized optimization problem for better initial optimization [33], [34].

III. Review

This section provides a literature review of algorithms used for high-dimensional DP linear models. The section is organized by optimization technique, and is in roughly chronological order of the first paper considering each technique. Figure 1 provides an overview of these methods, and employs a color-coding we will use in coming sections. Certain optimization techniques have been considered by multiple works, and as such, their subsections are longer than others. Throughout the sections, we highlight potential strengths and weaknesses of each algorithm. These algorithms are then tested empirically in section V.

In conducting this literature review, we searched for papers considering "sparse," "high-dimensional," "L_1," or "L_0" DP regression. After extensive reading, we identified the works presented in this section as relevant to our review.

Finally, all methods presented in this section employ the unbounded Hamming distance for their adjacency relation. In other words, the definition of differential privacy is employed when $\mathcal{D}$ and $\mathcal{D}'$ differ by one (extra or removed) row. This ensures that the privacy produced by all methods is directly comparable.

A. Model Selection

High-dimensional DP linear models were first considered in 2012 by Kifer et al. [21]. Their work is split into two parts: the first developed a generalized objective perturbation mechanism which employs L_2 regularization to achieve a strongly-convex loss function. They then derive the utility of their algorithm under both $(\epsilon, 0)$ DP and (ϵ, δ) DP, and show that using (ϵ, δ) DP saves a factor of $\sqrt{d}$ in utility.

Their utility analysis shows that this mechanism is not directly applicable to high-dimensional convex optimization. In the second part of their work, they assume that high-dimensional data can be explained by a model with sparsity s, and derive algorithms to select a support set of size s prior to running their generalized objective perturbation mechanism. Note that they make a number of additional assumptions when calculating utility, including those on the norm of the dataset and targets along with the restricted strong convexity of the dataset. However, these assumptions are not necessary to guarantee privacy.

The first algorithm they analyze is derived from the exponential mechanism. Roughly, this method calculates the loss of each support set with size s, and uses the exponential mechanism to choose the support set with minimum loss. After choosing the support set, these features are passed into the generalized objective perturbation mechanism. They prove utility of their method when $n = \omega(s^3 \log d)$, but is computationally inefficient since the support selection step must calculate the models of all $\binom{d}{s}$ possible support sets. This can be compared to the inefficiency of L_0-constrained minimization in the nonprivate setting.

Their second algorithm uses the sample and aggregate framework to split the training examples into $\sqrt{n}$ blocks and compute the optimal support on each block. After all $\sqrt{n}$ blocks are run, each of the d features has a score associated with it: the number of blocks which chose to include it in their support. A private mechanism can be used to return the top-s features, which are then passed into the generalized objective perturbation mechanism. They prove utility of this method when $n = \omega(s^2 \log^2 d)$, but require an assumption that each of the $\sqrt{n}$ blocks follows the restricted strong convexity assumption, which is strictly stronger than the assumption for the whole dataset.

Since their method for high-dimensional regression involves two stages, we call it TS. Note that because the method is split into two steps, the privacy budget must also be split between these steps. In their paper, Kifer et al. choose to split the budget into $(\frac{\epsilon}{2}, 0)$ for the support selection step and $(\frac{\epsilon}{2}, \delta)$ for the perturbation mechanism. We follow the same convention in our experiments. Additionally, our experiments only test the sample and aggregate support selection mechanism since it is computationally efficient.

Thakurta & Smith (2013) follow up Kifer et al.'s work by developing DP model selection mechanisms based on perturbation and subsampling stability [35]. In short, they call a function f perturbation stable if f outputs the same value on an input dataset $\mathcal{D}$ and all of $\mathcal{D}$'s neighbors. Under this condition, they require an algorithm $\mathcal{A}_{\text{dist}}$ which outputs the distance between $\mathcal{D}$ to its nearest unstable instance. Given the output of $\mathcal{A}_{\text{dist}}$, they use the Propose-Test-Release mechanism to output $f(\mathcal{D})$ in a DP manner.

However, this mechanism is typically not practical since

Model Selection

Summary: Choose a subset of features privately, and then use traditional DP optimization to find a weight.
Assumptions: Feature selection is algorithmically stable.
Privacy: $(\epsilon, 0)$ or (ϵ, δ)
Methods: TS [21]

Frank-Wolfe

Summary: Iteratively choose to move towards a vertex of a polytope constraint in a private manner.
Assumptions: Loss is Lipschitz and smooth. Solutions are found in few iterations. For heavy-tailed methods, truncated gradients must provide effective signal.
Privacy: $(\epsilon, 0)$ or (ϵ, δ)
Methods: FW [37], POLYFW [38], VRFW [39], HTFW [8], HTPL [8]

Compressed Learning

Summary: Compress high-dimensional dataset to low-dimensions by multiplying with a random matrix. Optimize in low-dimensions and find corresponding weight in original space. *Assumptions:* Loss is Lipschitz. Random matrix does not destroy important information in dataset.
Privacy: $(\epsilon, 0)$ or (ϵ, δ)
Methods: PROJERM [40]

ADMM

Summary: Privatize the ADMM algorithm with objective perturbation.
Assumptions: Search through large hyperparameter space possible. ADMM converges.
Privacy: $(\epsilon, 0)$
Methods: ADMM [42], ADMMHALF [42]

Thresholding

Summary: Use IGHT to produce sparse weight. Privatize with gradient perturbation or output perturbation.
Assumptions: IGHT must identify coefficients efficiently. For heavy-tailed methods, truncated gradients must not be overwhelmed by noise.
Privacy: (ϵ, δ)
Methods: DPIGHT [43], DPSLKT [44], HTSL [8], HTSO [8]

Coordinate Descent

Summary: Use greedy coordinate descent to privately a single component of the weight to update.
Assumptions: Greedy coordinate descent can be implemented efficiently. Lipschitz constants for each feature are known.
Privacy: (ϵ, δ)
Methods: GCDGSQ [46], GCDGSR [46], GCDGSS [46]

Mirror Descent

Summary: Use iteratively stronger regularization to solve a constrained optimization problem.
Assumptions: Composing multiple private optimizations is numerically stable.
Privacy: (ϵ, δ)
Methods: NM [39]

Fig. 1. A taxonomy of optimization techniques used for high-dimensional DP linear models.

calculating $\mathcal{A}_{\text{dist}}$ requires searching over all datasets around $\mathcal{D}$, of which there may be infinitely many. Secondly, the mechanism relies on the stability of f at the dataset $\mathcal{D}$, which may change for different functions f.

Thakurta & Smith then analyze subsampling stable functions. For a dataset $\mathcal{D}$, they define $\widehat{\mathcal{D}}$ as a random subset of $\mathcal{D}$ in which each element appears independently with probability q. Then f is called q-subsampling stable if $f(\mathcal{D}) = f(\widehat{\mathcal{D}})$ with probability at least $\frac{3}{4}$. Under this condition, they derive an (ϵ, δ)-DP algorithm which outputs $f(\mathcal{D})$ with high probability when f is q-subsampling stable.

While computationally tractable, this algorithm is still inefficient in that it requires $\mathcal{O}(\frac{1}{q^2})$ runs of f. Given the computational complexity of model selection procedures and the fact that q can be very small for low values of ϵ and δ, this method can be difficult to use in practice. Additionally, there is a nonzero probability that the algorithm outputs $\perp$, a symbol indicating that the model selection procedure cannot be outputted while maintaining DP. Finally, as in the TS approach, using this procedure would require arbitrarily splitting the privacy budget between the model selection and optimization steps. For these reasons, we do not implement this method in our experiments.

Another potential approach within model selection could involve modifying nonprivate screening rules to operate with DP. Recent work has attempted to do this but has shown a negative result [36]. It remains an open question whether an effective screening rule can be derived for model selection with high probability.

B. Frank-Wolfe

The Frank-Wolfe algorithm (FW) is a greedy optimization technique which works well on problems with polytope constraints. At each iteration, FW finds a first-order approximation of the objective function at the current iterate. It then finds a vertex of the polytope constraint which minimizes the first-order approximation, and takes a step in this direction. The algorithm was first developed for quadratic programming, but can be efficiently implemented for many loss functions including mean-squared error and binary cross-entropy.

Talwar et al. (2015) privatize FW for L_1 constrained optimization [37]. The algorithm works well in this setting since the L_1 constraint corresponds to a polytope. In order to produce private FW, at each iteration they treat the values of the first-order approximation at each vertex as scores, and use the exponential mechanism to choose a direction to move in. Using the advanced composition theorem for $(\epsilon, 0)$ DP, they can compute the exact amount of noise to add, achieving (ϵ, δ) DP.

Their utility analysis does not require restricted strong convexity or restricted strong smoothness. Instead, it relies on the curvature constant of the loss function, which is bounded for both linear and logistic regression. This constant and with the Lipschitzness of the loss function with respect to the L_1 norm is the only information necessary to produce the utility bound.

Due to the lack of assumptions in producing a utility bound, private FW was explored by a number of subsequent works. These works identified that private FW requires gradients with respect to the entire dataset at each iteration, which can waste privacy budget. According to the principles of subsampling in DP, if the entire dataset is not accessed at each iteration, less information from the data is used in each update, and less noise is required for privacy. The two following methods consider stochastic private FW methods.

Bassily et al. (2021) consider a stochastic private FW method which reduces variance of updates through a recursive gradient estimator [38]. Specifically, their method computes the gradient of the objective function with respect to $\frac{n}{2}$ of the datapoints. Then it iterates through the remaining $\frac{n}{2}$ points and updates the gradient with a weighted average. The weight is updated after each iteration in a private manner using the exponential mechanism, and advanced composition yields (ϵ, δ)-DP. Using the author's convention, we call this method POLYFW.

POLYFW has error of $\widetilde{\mathcal{O}}(\frac{\log d}{\epsilon\sqrt{n}})$, which is not optimal according to the work below. Their analysis does not explicitly rely on the curvature constant but rather requires L_1 Lipschitzness and a smooth loss function with respect to the L_1 norm.

Asi et al. (2021) develop a variance-reduced FW method (VRFW) for smooth functions using DP binary trees [39]. Nodes in these trees closer to the root are assigned more samples, making their gradient more accurate, while nodes closer to the leaves have less samples. Iterating over the leaves, their algorithm makes FW updates which consider the gradients collected over the entire path from the root to the leaf. By reusing gradients calculated from more datapoints, the algorithm is able to add less noise through the exponential mechanism. Their method can achieve both $(\epsilon, 0)$-DP and (ϵ, δ)-DP.

Asi et al. calculate the error of this method, which is $\widetilde{\mathcal{O}}(1/\sqrt{n}+1/(n\epsilon)^{2/3})$ for (ϵ, δ)-DP. They then calculate lower bounds of error, and show that this error matches the lower bound. For this reason, their algorithm has optimal utility for smooth loss functions.

Finally, Hu et al. (2022) develop two stochastic FW-based algorithms for heavy tailed high-dimensional data [8]. Each of their utility analyses assumes bounded fourth order moments, which is common in heavy tailed statistics.

The first method they use involves a soft truncation and scaling of the gradient. They do this because heavy-tailed data may not have bounded gradients, which can lead to unbounded sensitivity. With these truncated gradients, the exponential mechanism is used to choose a direction for minimization. This algorithm is $(\epsilon, 0)$-DP. We call this method HTFW.

When specifically considering linear regression, they find that truncating each element of the design matrix and target vector to be in $[-K, K]$ can bound sensitivity. After this data processing step, they use the exponential mechanism to choose a direction for minimization. This algorithm is (ϵ, δ)-DP, and Hu et al. abbreviate it as HTPL.

C. Compressed Learning

Another approach to constrained optimization is through compressed learning, which reduces the dimensionality of the input space by multiplying the design matrix by a random matrix $\mathbf{\Phi} \in \mathbb{R}^{m\times d}$. Typically $\mathbf{\Phi}$ is chosen to be a subgaussian random matrix with norm $\mathcal{O}(1)$ and the feasible set $\mathcal{C}$ is the scaled L_1 ball.

Kasiviswanathan & Jin (2016) employ compressed sensing for high-dimensional DP linear modelling [40]. After choosing $\mathbf{\Phi}$ and $\mathcal{C}$, they solve the problem

$$\arg\min_{\vartheta \in \mathbf{\Phi}\mathcal{C}} \frac{1}{n}\sum_{i=1}^{n} \ell(\langle \Phi \mathbf{x}_i, \vartheta\rangle; y_i)$$

with a DP optimizer. To obtain an estimate for the parameter vector in the data space, they solve

$$\theta \in \arg\min_{\theta\in\mathbb{R}^d : \mathbf{\Phi}\theta=\vartheta} \|\theta\|_{\mathcal{C}}$$

where $\|\cdot\|_{\mathcal{C}}$ represents the Minkowski norm induced by the feasible set. When $\mathcal{C}$ is the scaled L_1 ball, this is equivalent to the L_1 norm. Note that the output of this procedure is DP with the same parameters as ϑ since θ is only a function of $\mathbf{\Phi}$, which does not depend on the data, and ϑ, which is DP.

Since their procedure only requires a DP optimizer to guarantee DP, it can be both $(\epsilon, 0)$-DP and (ϵ, δ)-DP. In our experiments, we implement an (ϵ, δ)-DP optimizer to work with their procedure. We refer to their procedure as PROJERM, in reference to its projected empirical risk minimization step.

The utility bounds which Kasiviswanathan & Jin derive follow directly from those known from the Johnson-Lindenstrauss lemma. Their bounds hold under conditions on size of m.

Zheng et al. (2017) used compressed learning to develop an algorithm for high-dimensional linear regression under local DP [41]. Instead of using gradient perturbation like Kasiviswanathan & Jin, Zheng et al. noise the projection of each datapoint $\mathbf{\Phi x}_i$ prior to optimization. Using the noised datapoints, they correct the quadratic loss function of linear regression to ensure it is unbiased, and optimize over this corrected loss function. Their algorithm is (ϵ, δ)-DP.

Zheng et al. use matrix concentration inequalities to derive bounds on utility in the value of the loss. However, they make no claims in directly comparing the output $\mathbf{w}_{\text{priv}}$ from their algorithm to the optimal $\mathbf{w}^*$.

D. ADMM

The alternating direction method of multipliers (ADMM) is a common algorithm used for optimization of nonsmooth or nonconvex regularizers by transforming an optimization problem into two simpler problems. Under assumptions on the strong convexity of the loss function and separability of the regularizer, ADMM can be shown to converge to a stationary point.

Wang & Zhang (2020) use the ADMM algorithm and use it to achieve DP L_1 and $L_{1/2}$ regularized logistic regression (ADMM and ADMMHALF) [42]. First, they show that these methods access the data in only the second of its three steps. Then, to privatize the algorithm, they employ objective perturbation in every second step. Finally, they use the basic composition theorem to show that with appropriate noise, their algorithm is $(\epsilon, 0)$-DP after K iterations.

Although Wang & Zhang do not provide a utility analysis of ADMM and ADMMHALF, they do analyze the three steps of each iteration and find that only the first step is unstable with sparsity. Since the data is only accessed in the second step, privatizing the second step does not require excessive noise since this step is convex and stable. For this reason, they intuitively claim that their algorithm is well-suited for DP, which outputs accurate results on stable functions.

E. Thresholding

A simple approach to sparse optimization is to enforce sparsity throughout training. This can be done with hard thresholding operations, in which only the top s components of a partially learned weight are retained after each optimizer step. Hard thresholding based approaches have been considered in many forms for high-dimensional DP optimization. We cover these methods below.

Wang & Gu (2019) consider DP iterative gradient hard thresholding (DPIGHT), which retains the top s components of parameter vector after each gradient step [43]. Although their method is simple, when they assume that each row of the design matrix is sub-Gaussian, they achieve the same utility as Kifer et al. in the TS procedure without requiring an extra support selection step. However, their utility does assume the sparse eigenvalue condition, which implies both restricted strong convexity and restricted strong smoothness. They analyze the privacy of their algorithm with respect to zero-concentrated DP as this allows for tight composition of gradient descent. Note that zero-concentrated DP can easily be converted to (ϵ, δ)-DP.

Wang & Gu (2020) follow up this work by considering a DP knowledge transfer framework (DPSLKT) to produce sparse linear models [44]. First, they use IGHT to train a teacher model on private data without DP. Next, they generate an auxiliary training set and use output perturbation when passing it through the initially trained model. Now this dataset is DP. Finally, to produce a sparse model, they use IGHT to find a student model which fits the the DP data well. Due to the post-processing property of DP, this model is also DP. Since this method also uses gradient descent, Wang & Gu use zero-concentrated DP, which can be converted to (ϵ, δ)-DP.

The utility guarantees of DPSLKT outperform those of previous methods because they rely on the L_∞ norm of the data, which can be up to $\mathcal{O}(\sqrt{d})$ times less than the L_2 norm. However, their utility analysis requires restricted strong convexity and restricted strong smoothness. For this reason, Wang & Gu give an option to define the loss function with an L_2 regularization term, since this will ensure that the loss is strongly convex. This approach is similar to the TS method of Kifer et al. However, unlike the TS method, this regularization term is not necessary, but without it, an assumption of restricted strong convexity and restricted strong smoothness is necessary, which typically does not hold on real-world datasets.

Hu et al. (2022) employ iterative hard thresholding to develop a private linear regression algorithm for heavy tailed high-dimensional data, which they call HTSL [8]. To do this, they truncate each element of the design matrix and target vector to be in $[-K, K]$, where K is a value set by the user. Then they use gradient descent for optimization, but after each step they privately retain the top s elements of the weight vector. Since their data and targets are truncated, they can calculate the sensitivity of this step, which is passed into the exponential mechanism. Using the advanced composition theorem, they achieve (ϵ, δ)-DP.

Note that truncation is required because the data is heavy tailed. Heavy tailed data is typically not bounded or subgaussian, and its loss function is typically not $\mathcal{O}(1)$ Lipschitz. Some combintation of these assumptions are required to compute the sensitivity for non-heavy tailed algorithms. To avoid this, Hu et al. truncate the "outlier" elements in the design matrix and target vector. However, the utility bound which Hu et al. derive requires that the dataset and targets have bounded fourth order moment. This assumption is similar to those made by other works in heavy tailed optimization.

For general convex optimization, Hu et al. use a soft trunctation and scaling operation on the gradient. Then they use gradient descent, privately retaining the top s elements of the weight vector. In this case, they bound the optimization sensitivity through the truncation and scaling of the gradient. Again, their utility bound requires bounded fourth order moments. This method produces and (ϵ, δ)-DP result. They call this method HTSO, since it performs heavy-tailed sparse optimization.

Finally, we mention Wang & Xu's 2019 work on iterative hard thresholding methods for local DP [45]. In this work, they assume that each datapoint $\mathbf{x}_i$ is sampled from $\{-1, +1\}^d$, which is a very strong condition.

However, this work is valuable since it shows that even under such strong conditions, the utility must have order $\widetilde{\Omega}(d \log d)$. Their iterative hard thresholding algorithm achieves this, but in general, orders greater than $\widetilde{\Omega}(\log d)$ are considered unlearnable in the high-dimensional setting. They then demonstrate that if only the labels y_i are kept private, iterative hard thresholding can achieve a bound of $\widetilde{\mathcal{O}}(\log d)$. However, this setting is dissimilar from the DP setting we consider in this paper, and thus we do not implement this algorithm.

F. Coordinate Descent

Coordinate descent is an optimization algorithm which iteratively updates only one parameter of a model's weight using its gradient. The algorithm typically chooses parameters to update cyclically or at random, but greedy coordinate descent chooses to update the parameter with the highest absolute gradient. In the nonprivate setting, all of these methods perform well on high-dimensional linear problems. Non-greedy methods are typically used since they are computationally efficient, only requiring the gradient with respect to one parameter.

Mangold et al. (2022) develop a DP coordinate descent algorithm for high-dimensional ERM [46]. They choose to use greedy coordinate descent (GCD), which updates the coordinate with the largest absolute value gradient, because under DP fewer computations enable less added noise. They acknowledge that this is an inherent tradeoff to computational efficiency, since their procedure must compute the gradient with respect to every parameter at every iteration.

Since they update only one parameter per iteration, Mangold et al. are able to employ per-feature smoothness and Lipschitz constants. In some datasets, some features can have significantly lower smoothness and Lipschitz constants as compared to the entire dataset. For these features, less noise is added during coefficient selection and optimization. Finally, using the advanced composition theorem, their method achieves (ϵ, δ)-DP.

Their utility bound demonstrates that their algorithm converges to a ball around the optimal weight, where the radius of the ball is determined by the level of privacy. However, it is worth noting that their utility bounds consider only smooth loss functions, and are thus not applicable to L_1 regularizers. To use L_1 regularizers, Mangold et al. develop DP proximal coordinate descent algorithms with a variety of proximal operators. These algorithms are named GCDGSQ, GCDGSR, and GCDGSS, to represent three different proximal operators, and empirical results indicate that these modifications greatly improve performance on high-dimensional datasets. However, they do not consider the utility of the algorithm with proximal operators.

G. Mirror Descent

Asi et al. (2021) privatize an iterative localization framework with private optimization. They call their method private mirror descent with noise multipliers (NM) [39]. This method finds the optimal parameter vector for a sequence of optimization problems with increasing regularization. By sequentially increasing the regularization, the method finds iteratively smaller regions for optimization, making the optimization problem easier. In addition, the regularization makes the optimization problem strongly convex, making it easy to use well-known optimal optimization techniques for private strongly convex problems. They prove that NM achieves optimal utility for non-smooth loss functions by demonstrating that using private optimization with iterative localization produces a loss which matches a known lower bound up to logarithmic factors. This algorithm is (ϵ, δ)-DP.

IV. Implementation Details

In the following section, we perform experiments on the central DP methods previously discussed. We release all code for these experiments online. This is one of the major contributions of this work, as previous works do not release code or test algorithms empirically. While theoretical utility bounds can provide an even ground for algorithm comparison, these bounds often make significant (and different) assumptions; empirical performance on a variety of datasets better demonstrates how these algorithms would perform if used in application. In this section, we list challenges we faced when implementing some of these methods.

- Without an optimized linear algebra implementation, the FW algorithm is very inefficient. An optimized implementation can be found in [47]. This holds for the other variants of FW as well.
- VRFW required smoothness constants to be greater than a threshold. Since a k-smooth function is also l-smooth when $l \geq k$, when implementing this method we set the smoothness constant to $\max(k, l)$ where k is the smoothness constant of our loss function and l is the required lower bound on smoothness.
- For heavy-tailed methods requiring the robust gradient, the expression of the correction factor can be found in Lemma 3.2 in [48]. The authors considering heavy-tailed private optimizers claimed to provide this expression in their appendix, but we could not find it.
- The private optimization method recommended by the authors of PROJERM requires n^2 steps of stochastic gradient descent, where n is the number of datapoints. Additionally, the method requires solving a optimization problem in d variables. We did not modify these steps, but for larger datasets PROJERM did not converge within 48 hours.
- ADMM and ADMMHALF have a significant number hyperparamters and can be unstable for many hyperparameters and datasets. We had to refactor portions of their algorithm into numerically stable mathematical equivalents. Refactoring improved performance but certain datasets were still unstable.
- We did not implement GCD methods but instead used code released with the GCD paper to run these methods.
- When implementing NM, we tried to use CVXPY to solve each optimization problem [49], [50]. However, the constraints in this problem are unstable, and the free ECOS solver included in CVXPY would produce errors prior to convergence [51]. The free SCS solver would take up to 10 seconds for a single iteration, and the cumulative sequence of problems would require over 10,000 iterations, making it infeasible [52]. For this reason, we approximated the solution to each optimization problem using gradient descent.
- We employed differentially private stochastic gradient descent (DPSGD) as a baseline for each dataset [53]. In addition to being a common DP optimization method, works have shown that the noise added during DPSGD can be an effective regularization strategy which prevents overfitting. This may cause it to perform well in high-dimensional settings [54].

Finally, it is important to note that all experiments were conducted with Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz. For each dataset-algorithm combination, 20 trials were performed on different CPUs to speed up training. A 48 hour wall-clock time limit on computation time was set for each dataset-algorithm combination.

V. Experiments

This section runs experiments on six datasets to test the methods described in Section III. These experiments provide a systematic comparison between algorithms' performance on varying privacy budgets. Our study focuses on two of the most prevalent linear models: linear regression and logistic regression.

All datasets were chosen from the libsvm and OpenML libraries [55], [56]. For linear regression, we used the Bodyfat, PAH, and E2006 datasets. The raw datasets for Bodyfat and PAH were used, but for computational efficiency, 500 datapoints from the E2006 dataset were chosen and their dimensionality was reduced to 500 with PCA. Similarly, the Heart, DBworld-subjects-stemmed, and RCV1 datasets were used for logistic regression. We used the raw Heart and DBworld datasets, but chose 500 datapoints of the RCV1 dataset and reduced their dimensionality to 500 with PCA. These datasets were chosen to identify algorithms' performance on different scales of dimensionality while retaining computational efficiency. Table I provides a summary of the datasets' dimensionalities.

All features in datasets were demeaned, and then samples in each dataset were rescaled so the maximum L_1-norm of a sample in any given dataset was 1. This was done to bound the Lipschitz and smoothness constants of the datasets under linear and logistic regression, a common requirement in DP optimization literature and detailed well in [40]. Finally, linear or logistic regression models were trained on the datasets under a variety of hyperparameters. Each set of hyperparameters was tested for 20 trials, with a unique 60%/20%/20% split between training, validation, and testing datasets used for each trial. A detailed list of the hyperparameters used for each algorithm can be found in the appendix.

Note that algorithms were tested with privacy parameters $\epsilon \in \{0.1, 0.5, 1, 2, 5\}$ with $\delta = \frac{1}{n_{\text{train}}^2}$, where n_{train} is the number of training samples. These privacy parameters were chosen according to common recommendations in DP literature. Finally, each algorithm was run with $\epsilon = 100$ and $\delta = 0.999$ to approximate a nonprivate solution. We did this for each algorithm so that we find the best nonprivate solution using the same choices of hyperparameters used for private training.

A. Linear Regression

Table II, Table III, and Table IV demonstrate the mean of 20 trials of mean-squared errors for the test portions of the Bodyfat, PAH, and E2006 datasets, respectively. Additionally, subscripts designating two times the standard error of the mean are provided. For each value of ϵ, the best performing algorithm's results are bolded. Note that ProjERM is not

TABLE I
Summary of Datasets

Dataset	N	D	Regression Type
Bodyfat	252	14	Linear
PAH	80	112	Linear
E2006	500	500	Linear
Heart	270	13	Logistic
DBworld	64	229	Logistic
RCV1	500	500	Logistic

included in Table IV since it did not converge within 48 hours. We now discuss notable performance trends which are seen in these tables. Note that in the appendix, Figure 2, Figure 3, and Figure 4 provide boxplots of the mean-squared errors for the trials on each dataset.

Perhaps the most significant trend among all three tables is that HTSO, or heavy-tailed sparse optimization, performs well on all values of ϵ. HTSO performs iterative gradient hard thresholding using the robust gradient of the data. Intuitively, this makes sense; while most other methods tested require the loss to be Lipschitz, this constant is often a worst-case bound. For example, although bounding the Lipschitz constant of the loss requires rescaling the datapoints, datapoints with larger rescaled norms might be outliers, and most datapoints have norms near 0. In contrast, heavy tailed optimizers do not assume Lipschtizness of the loss, and instead use parameters to calculate a robust gradient which is resistant to outliers. Thus, even in settings where the Lipschitz constant of the loss can be set, heavy-tailed private gradient optimizers can perform better.

Next, we note that GCD, or greedy coordinate descent, can perform also well. Similar to HTSO, these methods do not rely on the Lipschitz constant for optimization of the entire weight vector but rather use different Lipschitz constants for each coefficient. For this reason, private optimization can add less noise to coefficients which are smaller than others, thus improving the utility of the method. Finally, as was reported in [46], different greedy selection rules can perform very differently depending on the dataset.

Note that GCD methods can be made to better approximate the behavior of HTSO through a clipping parameter. Specifically, by aggressively clipping gradients before applying coordinate descent, GCD can resist outlier influence on gradients. However, our goal was to observe the performance of algorithms without additional modifications, and we did not try to use significant clipping here.

From these results, we identify that future research on private statistical models which incorporate measures of scale and robustness could be particularly promising. Specifically, robust calculations are statistically stable, which reduces the noise required for DP. Considering the scale of features in a linear model allows the information from features "squashed" after normalization to not be lost.

B. Logistic Regression

Table V, Table VI, and Table VII demonstrate the mean of 20 trials of accuracies for the test portions of the Heart, DBworld, and RCV1 datasets, respectively. Subscripts are provided denoting two times the standard error of the mean. Note that ADMM, ADMMHalf, and ProjERM are not included in Table VII since they did not converge within 48 hours. Additionally, the ECOS solver failed to converge in the TS procedure in Table VI. These three tables are presented in the appendix due to the page limit of the pain body of the paper. Note that in the appendix, Figure 5, Figure 6, and Figure 7 provide boxplots of the accuracies for the trials on each dataset.

From these three tables, it is clear that GCD, or greedy coordinate descent, performs well on all values of ϵ. As discussed in the previous subsection, GCD methods do not rely on the Lipschitz constant for optimization of the entire weight vector but instead use different Lipschitz constants for each coefficient. This allows GCD to add less noise than other optimizers.

Thus we conclude this section similar to that of linear regression: future research on private logistic regression models should incorporate the scale of each feature separately when adding private noise. This allows for improved empirical utility, in which features with different scales can be noised effectively.

VI. Trends in Results

Having completed the implementation and empirical evaluation of each surveyed method, we note three high-level themes in the research that may form valuable directions for future work. Each of these is significant in that they inhibit practitioner adoption of DP methods by imposing unexpected costs of behaviors compared to non-private methods.

First is the extreme computational cost of using DP linear models compared to their original non-private counterparts, taking orders of magnitude more time to train. This limits real-world utility where datasets can not be subsampled for convenience. A related issue is that reguarlizaition penalties do not have the same (mostly) intuitive behavior in a DP model, and there appears to be no clear guidance on how and when to apply regularizers to a DP linear model. Finally, we find that accuracy does not necessarily decrease with stricter privacy parameters. We discuss these issues in the following subsections.

A. Computational Inefficiency

We have evaluated six values of ϵ for ≈ 16 algorithms for each table. Filling each cell required 12-36 hours of computing time for datasets that are considered small by modern machine learning standards. Indeed, training a linear model via scikit-learn or other APIs takes minutes. This extreme cost disparity is perhaps the biggest hindrance to the adoption of DP methods in practice, as a practitioner has no reasonable expectation that they could train a model on a real-world larger dataset.

TABLE II

BODYFAT: MEAN SQUARED ERROR

ϵ	0.1	0.5	1.0	2.0	5.0	NONPRIVATE
TS	$0.1142_{(0.0067)}$	$0.1133_{(0.0066)}$	$0.1122_{(0.0065)}$	$0.1103_{(0.0064)}$	$0.1051_{(0.0060)}$	$\mathbf{0.0112}_{(0.0070)}$
FW	$0.0947_{(0.0053)}$	$0.0947_{(0.0054)}$	$0.0947_{(0.0054)}$	$0.0947_{(0.0054)}$	$0.0947_{(0.0054)}$	$0.0844_{(0.0047)}$
POLYFW	$0.0955_{(0.0053)}$	$0.0955_{(0.0053)}$	$0.0955_{(0.0053)}$	$0.0955_{(0.0053)}$	$0.0955_{(0.0053)}$	$0.0955_{(0.0053)}$
VRFW	$0.0965_{(0.0057)}$	$0.0965_{(0.0057)}$	$0.0965_{(0.0057)}$	$0.0965_{(0.0057)}$	$0.0965_{(0.0057)}$	$0.0892_{(0.0058)}$
HTFW	$0.0946_{(0.0053)}$	$0.0946_{(0.0053)}$	$0.0946_{(0.0053)}$	$0.0937_{(0.0058)}$	$0.0896_{(0.0049)}$	$0.0886_{(0.0049)}$
HTPL	$0.0946_{(0.0053)}$	$0.0946_{(0.0053)}$	$0.0946_{(0.0053)}$	$0.0946_{(0.0053)}$	$0.0946_{(0.0053)}$	$0.0847_{(0.0047)}$
PROJERM	$0.0945_{(0.0053)}$	$0.0945_{(0.0053)}$	$0.0945_{(0.0053)}$	$0.0945_{(0.0053)}$	$0.0945_{(0.0053)}$	$0.0890_{(0.0051)}$
DPIGHT	$0.0891_{(0.0052)}$	$0.0926_{(0.0054)}$	$0.0896_{(0.0055)}$	$0.0892_{(0.0056)}$	$0.0916_{(0.0052)}$	$0.0932_{(0.0053)}$
DPSLKT	$0.0902_{(0.0060)}$	$0.0902_{(0.0060)}$	$0.0902_{(0.0060)}$	$0.0902_{(0.0060)}$	$0.0908_{(0.0052)}$	$0.0744_{(0.0075)}$
HTSL	$0.0896_{(0.0065)}$	$0.0885_{(0.0059)}$	$0.0894_{(0.0059)}$	$0.0886_{(0.0054)}$	$0.0886_{(0.0056)}$	$0.0942_{(0.0055)}$
HTSO	$\mathbf{0.0685}_{(0.0091)}$	$0.0727_{(0.0153)}$	$0.0699_{(0.0458)}$	$\mathbf{0.0521}_{(0.0131)}$	$0.0607_{(0.0104)}$	$0.0729_{(0.0068)}$
GCDGSQ	$1.0307_{(0.1095)}$	$0.0768_{(0.0082)}$	$0.0661_{(0.0046)}$	$0.0701_{(0.0046)}$	$0.0755_{(0.0057)}$	$0.0162_{(0.0022)}$
GCDGSR	$0.0956_{(0.0053)}$	$\mathbf{0.0041}_{(0.0018)}$	$\mathbf{0.0312}_{(0.0024)}$	$0.0582_{(0.0033)}$	$0.0697_{(0.0053)}$	$0.0615_{(0.0064)}$
GCDGSS	$0.0956_{(0.0053)}$	$0.0779_{(0.0048)}$	$0.0894_{(0.0051)}$	$0.0956_{(0.0069)}$	$\mathbf{0.0462}_{(0.0090)}$	$0.0435_{(0.0042)}$
NM	$0.0956_{(0.0053)}$	$0.0956_{(0.0053)}$	$0.0953_{(0.0053)}$	$0.0973_{(0.0053)}$	$0.0948_{(0.0054)}$	$0.0980_{(0.0052)}$
SGD	$0.0916_{(0.0051)}$	$0.0916_{(0.0051)}$	$0.0916_{(0.0051)}$	$0.0916_{(0.0051)}$	$0.0916_{(0.0051)}$	$0.0983_{(0.0055)}$

TABLE III

PAH: MEAN SQUARED ERROR

ϵ	0.1	0.5	1.0	2.0	5.0	NONPRIVATE
TS	$0.0607_{(0.0252)}$	$0.0572_{(0.0245)}$	$0.0607_{(0.0257)}$	$0.0608_{(0.0262)}$	$0.0611_{(0.0275)}$	$0.0837_{(0.0224)}$
FW	$0.0910_{(0.0168)}$	$0.0910_{(0.0168)}$	$0.0910_{(0.0168)}$	$0.0910_{(0.0168)}$	$0.0910_{(0.0168)}$	$0.0902_{(0.0166)}$
POLYFW	$0.0909_{(0.0168)}$	$0.0909_{(0.0168)}$	$0.0909_{(0.0168)}$	$0.0909_{(0.0168)}$	$0.0909_{(0.0168)}$	$0.0911_{(0.0168)}$
VRFW	$0.0918_{(0.0170)}$	$0.0918_{(0.0170)}$	$0.0918_{(0.0170)}$	$0.0918_{(0.0170)}$	$0.0918_{(0.0170)}$	$0.0911_{(0.0169)}$
HTFW	$0.0907_{(0.0167)}$	$0.0907_{(0.0167)}$	$0.0907_{(0.0167)}$	$0.0907_{(0.0167)}$	$0.0905_{(0.0167)}$	$0.0910_{(0.0168)}$
HTPL	$0.0907_{(0.0167)}$	$0.0908_{(0.0168)}$	$0.0907_{(0.0167)}$	$0.0907_{(0.0167)}$	$0.0907_{(0.0167)}$	$0.0905_{(0.0167)}$
PROJERM	$0.0910_{(0.0168)}$	$0.0910_{(0.0168)}$	$0.0910_{(0.0168)}$	$0.0910_{(0.0168)}$	$0.0910_{(0.0168)}$	$0.0911_{(0.0168)}$
DPIGHT	$0.0892_{(0.0163)}$	$0.0897_{(0.0164)}$	$0.0892_{(0.0163)}$	$0.0904_{(0.0167)}$	$0.0904_{(0.0166)}$	$0.0910_{(0.0168)}$
DPSLKT	$\mathbf{0.0538}_{(0.0068)}$	$0.0850_{(0.0146)}$	$0.0538_{(0.0068)}$	$0.0852_{(0.0138)}$	$0.0850_{(0.0146)}$	$0.0828_{(0.0170)}$
HTSL	$0.0905_{(0.0168)}$	$0.0901_{(0.0166)}$	$0.0903_{(0.0166)}$	$0.0905_{(0.0167)}$	$0.0908_{(0.0168)}$	$0.0911_{(0.0168)}$
HTSO	$0.0581_{(0.0157)}$	$\mathbf{0.0352}_{(0.0063)}$	$\mathbf{0.0432}_{(0.0062)}$	$\mathbf{0.0530}_{(0.0077)}$	$0.0647_{(0.0102)}$	$0.0909_{(0.0168)}$
GCDGSQ	$429.1658_{(195.0152)}$	$15.6425_{(4.2077)}$	$3.4062_{(0.9211)}$	$1.1643_{(1.2433)}$	$\mathbf{0.0591}_{(0.0231)}$	$0.0474_{(0.0076)}$
GCDGSR	$1366.9646_{(1497.3976)}$	$56.6219_{(61.293)}$	$38.2438_{(49.2854)}$	$1.5510_{(0.8699)}$	$0.1607_{(0.0751)}$	$0.0855_{(0.0169)}$
GCDGSS	$1.4834_{(0.9205)}$	$0.2219_{(0.0879)}$	$0.1485_{(0.0459)}$	$0.1188_{(0.0301)}$	$0.1179_{(0.0522)}$	$\mathbf{0.0395}_{(0.0065)}$
NM	$0.0911_{(0.0168)}$	$0.0911_{(0.0168)}$	$0.0911_{(0.0168)}$	$0.0911_{(0.0168)}$	$0.0911_{(0.0168)}$	$0.0911_{(0.0168)}$
SGD	$0.0914_{(0.0170)}$	$0.0914_{(0.0170)}$	$0.0914_{(0.0170)}$	$0.0914_{(0.0170)}$	$0.0914_{(0.0170)}$	$0.0910_{(0.0168)}$

TABLE IV

E2006: MEAN SQUARED ERROR

ϵ	0.1	0.5	1.0	2.0	5.0	NONPRIVATE
TS	$0.0392_{(0.0033)}$	$0.0393_{(0.0033)}$	$0.0393_{(0.0033)}$	$0.0393_{(0.0033)}$	$0.0380_{(0.0035)}$	$0.0444_{(0.0052)}$
FW	$0.0390_{(0.0033)}$	$0.0390_{(0.0033)}$	$0.0390_{(0.0033)}$	$0.0390_{(0.0033)}$	$0.0390_{(0.0033)}$	$\mathbf{0.0382}_{(0.0033)}$
POLYFW	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$
VRFW	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$
HTFW	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$
HTPL	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0385_{(0.0033)}$
DPIGHT	$0.0390_{(0.0033)}$	$0.0390_{(0.0033)}$	$0.0390_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$
DPSLKT	$0.0391_{(0.0033)}$	$0.0400_{(0.0035)}$	$0.0394_{(0.0034)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$
HTSL	$0.0384_{(0.0033)}$	$0.0388_{(0.0033)}$	$0.0388_{(0.0033)}$	$0.0389_{(0.0033)}$	$0.0390_{(0.0033)}$	$0.0391_{(0.0033)}$
HTSO	$\mathbf{0.0308}_{(0.0031)}$	$\mathbf{0.0247}_{(0.0043)}$	$\mathbf{0.0229}_{(0.0044)}$	$\mathbf{0.0246}_{(0.0040)}$	$\mathbf{0.0282}_{(0.0034)}$	$0.0392_{(0.0034)}$
GCDGSQ	$0.0909_{(0.0132)}$	$0.0658_{(0.0064)}$	$0.0657_{(0.0064)}$	$0.0656_{(0.0064)}$	$0.0656_{(0.0064)}$	$0.0566_{(0.0058)}$
GCDGSR	$0.0656_{(0.0064)}$	$0.0656_{(0.0064)}$	$0.0656_{(0.0064)}$	$0.0656_{(0.0064)}$	$0.0656_{(0.0064)}$	$0.0656_{(0.0064)}$
GCDGSS	$0.6121_{(0.1639)}$	$0.0834_{(0.0085)}$	$0.0690_{(0.0061)}$	$0.0660_{(0.0064)}$	$0.0656_{(0.0064)}$	$0.0656_{(0.0064)}$
NM	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$
SGD	$4.0000_{(0.0000)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$	$0.0391_{(0.0033)}$

As it exists today, only one work has attempted to make the FW algorithm scalable to sparse datasets [57]. While they were able to obtain significant speedups of up to four orders of magnitude, the FW algorithm never performed best in any of the $\epsilon \leq 5$ experiments. It remains to be determined how many DP algorithms for linear models can be made computationally tractable while balancing their efficacy. A key issue in this task is the nature of DP itself; addining noise at each step of the process is a computationally demanding task in-and-of itself that tends to remove all sparsity that one would exploit in high dimensional problems.

B. Unclear Impact of Regularization

In the linear model case, the L_1 penalty is particularly desirable because it is robust to spurious uninformative features and produces sparse solutions [58]. For both L_1 and L_2 models, a regularization path is a common desiderata where the penalty λ is varied from small-to-large, and a smooth relationship between coefficients (and accuracy) with the value of λ is used to inform the model selection and interrogate the data [59]. This is often achieved via a "warm start" where the solution of one value of λ is used as the starting solution for an adjacent value of $\lambda' \neq \lambda$ [60].

When training differentially private linear models, we do not see these common benefits occur. In all cases, even when using a L_1 penalty on the weight vector, the solution is completely dense. This simultaneously removes the benefit of sparse solutions (larger models) and mitigates the benefit of robustness in the face of irrelevant features (because all features are always used). Warm starting the solution via a previous value of λ is also non-viable as it would require expending additional privacy budget.

The nature of how to regularize DP linear models and their ultimate benefit is an open problem. One possible hypothesis is that the noise added by DP itself acts as a regularize. This is supported by prior results in deep learning on the utility of noise [61]. Other works in DP learning with a backbone network trained on non-private data also support an implicit regularization effect in obtaining good results without any additional regularization [54], though the use of a backbone confounds drawing a conclusion.

C. Accuracy Asperity with Privacy

It is natural that as ϵ decreases, the accuracy should also decrease, as the limit of $\epsilon \to 0$ implies a constant prediction. However, this is only a weakly observable trend in our experiments despite running each trial 20 times and averaging the result. This has also been observed by [62], [63], and appears to be a broader issue with DP learning. To some degree the issue is unavoidable because noise is added to the model, and so noise necessarily increases in the output. However, it remains to be determined if the amount of asperity in results for a given change in ϵ is "optimal" and how much this could be reduced.

One possible issue is the intersection of the optimization algorithm used, and that of the noise added. Running a classical optimizer until convergence to a global minima (in the case of strongly convex losses) is possible regardless of the convergence rate of the algorithm being tested. This is not the case for DP, as each access to the data must be accounted for in the total privacy budget. This creates an interplay between target ϵ and convergence where it may be best to perform fewer optimization iterations to achieve a lower total ϵ, but thus result in an irregular effective convergence rate. Another possibility is implied by the results of [64] who observe that there is a finite practical range of ϵ values that have a meaningful impact on privacy. Values of ϵ below or above this range add/reduce the noise without changing the identifiability of the data and thus could lead to lower ϵ values obtaining better accuracy by chance because chance was the only factor in relative ranking.

VII. Conclusion

In this paper, we provide the first unified review of optimization methods for high dimensional DP linear models. In doing so, we give an overview of the strengths and weaknesses of many methods, highlighting the different approaches taken in prior literature. Next, we conduct empirical experiments for a systematic comparison of the optimization methods, which has not been done in works prior to this. Finally, we release our code for easier future use and better analysis of future algorithms.

Our empirical experiments highlight surprising and previously unobserved trends of optimization for high-dimensional DP linear models. Specifically, we find that methods which are able to take into account the scale of each feature perform better than those which rely on the Lipschitz constant of the loss function. Indeed, even when the Lipschitz constant of the loss function is bounded, heavy-tailed or coordinate-based optimization techniques can perform better since they are more robust and add less noise.

We believe that this paper and the surprising result highlighted above can influence future research on differentially private optimization, which is an active research field. Further study on heavy-tailed or coordinate-wise optimization can improve performance on the tasks listed in this review, and may even translate to more complicated models such as neural networks.

References

[1] P. Lyssiotou, P. Pashardes, and T. Stengos, "Age effects on consumer demand: an additive partially linear regression model," *Canadian Journal of Economics/Revue canadienne d'économique*, vol. 35, no. 1, pp. 153–165, 2002.

[2] C. R. Madhuri, G. Anuradha, and M. V. Pujitha, "House price prediction using regression techniques: A comparative study," in *2019 International conference on smart structures and systems (ICSSS)*. IEEE, 2019, pp. 1–5.

[3] I. Kurt, M. Ture, and A. T. Kurum, "Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease," *Expert systems with applications*, vol. 34, no. 1, pp. 366–374, 2008.

[4] Y. Sahin and E. Duman, "Detecting credit card fraud by ann and logistic regression," in *2011 international symposium on innovations in intelligent systems and applications*. IEEE, 2011, pp. 315–319.

[5] K. P. Murphy, *Probabilistic machine learning: an introduction*. MIT press, 2022.

[6] P. Bühlmann, "Statistical significance in high-dimensional linear models," 2013.

[7] P. Rigollet, "18. s997: High dimensional statistics," *Lecture Notes, Cambridge, MA, USA: MIT Open-CourseWare*, 2015.

[8] L. Hu, S. Ni, H. Xiao, and D. Wang, "High dimensional differentially private stochastic optimization with heavy-tailed data," in *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2022, pp. 227–236.

[9] C. Dwork, "Differential privacy," in *International colloquium on automata, languages, and programming*. Springer, 2006, pp. 1–12.

[10] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[11] A. Beimel, K. Nissim, and U. Stemmer, "Private learning and sanitization: Pure vs. approximate differential privacy," in *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, 2013, pp. 363–378.

[12] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 371–380.

[13] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," in *International conference on machine learning*. PMLR, 2015, pp. 1376–1385.

[14] J. P. Near and C. Abuah, "Programming differential privacy," 2021.

[15] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Theory of Cryptography Conference*. Springer, 2016, pp. 635–658.

[16] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 2017, pp. 263–275.

[17] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 2007, pp. 94–103.

[18] X. Xiong, S. Liu, D. Li, Z. Cai, and X. Niu, "A comprehensive survey on local differential privacy," *Security and Communication Networks*, vol. 2020, pp. 1–29, 2020.

[19] B. I. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft, "Learning in a large function space: Privacy-preserving mechanisms for svm learning," *arXiv preprint arXiv:0911.5708*, 2009.

[20] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimiza tion." *Journal of Machine Learning Research*, vol. 12, no. 3, 2011.

[21] D. Kifer, A. Smith, and A. Thakurta, "Private convex empirical risk minimization and high-dimensional regression," in *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, 2012, pp. 25–1.

[22] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *2014 IEEE 55th annual symposium on foundations of computer science*. IEEE, 2014, pp. 464–473.

[23] A. Cheu, "Differential privacy in the shuffle model: A survey of separations," *arXiv preprint arXiv:2107.11839*, 2021.

[24] A. McMillan, O. Javidbakht, K. Talwar, E. Briggs, M. Chatzidakis, J. Chen, J. Duchi, V. Feldman, Y. Goren, M. Hesse *et al.*, "Private federated statistics in an interactive setting," *arXiv preprint arXiv:2211.10082*, 2022.

[25] H. Xu, C. Caramanis, and S. Mannor, "Sparse algorithms are not stable: A no-free-lunch theorem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 1, pp. 187–193, 2011.

[26] J. Subramanian and R. Simon, "Overfitting in prediction models–is it a problem only in high dimensions?" *Contemporary clinical trials*, vol. 36, no. 2, pp. 636–641, 2013.

[27] M. J. Wainwright, *High-dimensional statistics: A non-asymptotic viewpoint*. Cambridge university press, 2019, vol. 48.

[28] P. Jain, A. Tewari, and P. Kar, "On iterative hard thresholding methods for high-dimensional m-estimation," *Advances in neural information processing systems*, vol. 27, 2014.

[29] D. Bertsimas, A. King, and R. Mazumder, "Best subset selection via a modern optimization lens," 2016.

[30] M. Frank, P. Wolfe *et al.*, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[31] O. Maillard and R. Munos, "Compressed least-squares regression," *Advances in neural information processing systems*, vol. 22, 2009.

[32] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[33] S. J. Wright, "Coordinate descent algorithms," *Mathematical programming*, vol. 151, no. 1, pp. 3–34, 2015.

[34] R. Glowinski, "On alternating direction methods of multipliers: a historical perspective," *Modeling, simulation and optimization for science and technology*, pp. 59–82, 2014.

[35] A. G. Thakurta and A. Smith, "Differentially private feature selection via stability arguments, and the robustness of the lasso," in *Conference on Learning Theory*. PMLR, 2013, pp. 819–850.

[36] A. Khanna, F. Lu, and E. Raff, "The challenge of differentially private screening rules," *arXiv preprint arXiv:2303.10303*, 2023.

[37] K. Talwar, A. Guha Thakurta, and L. Zhang, "Nearly optimal private lasso," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[38] R. Bassily, C. Guzmán, and A. Nandi, "Non-euclidean differentially private stochastic convex optimization," in *Conference on Learning Theory*. PMLR, 2021, pp. 474–499.

[39] H. Asi, V. Feldman, T. Koren, and K. Talwar, "Private stochastic convex optimization: Optimal rates in l1 geometry," in *International Conference on Machine Learning*. PMLR, 2021, pp. 393–403.

[40] S. P. Kasiviswanathan and H. Jin, "Efficient private empirical risk minimization for high-dimensional learning," in *International Conference on Machine Learning*. PMLR, 2016, pp. 488–497.

[41] K. Zheng, W. Mou, and L. Wang, "Collect at once, use effectively: Making non-interactive locally private learning possible," in *International Conference on Machine Learning*. PMLR, 2017, pp. 4130–4139.

[42] P. Wang and H. Zhang, "Differential privacy for sparse classification learning," *Neurocomputing*, vol. 375, pp. 91–101, 2020.

[43] L. Wang and Q. Gu, "Differentially private iterative gradient hard thresholding for sparse learning," in *28th International Joint Conference on Artificial Intelligence*, 2019.

[44] ——, "A knowledge transfer framework for differentially private sparse learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6235–6242.

[45] D. Wang and J. Xu, "On sparse linear regression in the local differential privacy model," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6628–6637.

[46] P. Mangold, A. Bellet, J. Salmon, and M. Tommasi, "High-dimensional private empirical risk minimization by greedy coordinate descent," *arXiv preprint arXiv:2207.01560*, 2022.

[47] A. Khanna, F. Lu, E. Raff, and B. Testa, "Differentially private logistic regression with sparse solutions," in *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1–9. [Online]. Available: https://doi.org/10.1145/3605764.3623910

[48] O. Catoni and I. Giulini, "Dimension-free pac-bayesian bounds for matrices, vectors, and linear least squares regression," *arXiv preprint arXiv:1712.02747*, 2017.

[49] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[50] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[51] A. Domahidi, E. Chu, and S. Boyd, "Ecos: An socp solver for embedded systems," in *2013 European control conference (ECC)*. IEEE, 2013, pp. 3071–3076.

[52] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *Journal of Optimization Theory and Applications*, vol. 169, no. 3, pp. 1042–1068, June 2016.

[53] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[54] M. Tobaben, A. Shysheya, J. Bronskill, A. Paverd, S. Tople, S. Zanella-Beguelin, R. E. Turner, and A. Honkela, "On the efficacy of differentially private few-shot image classification," *arXiv preprint arXiv:2302.01190*, 2023.

[55] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[56] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: Networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.

[57] E. Raff, A. Khanna, and F. Lu, “Scaling up differentially private lasso regularized logistic regression via faster frank-wolfe iterations,” in *Advances in Neural Information Processing Systems*, vol. 36. Curran Associates, Inc., 2023.

[58] A. Y. Ng, “Feature selection, l1 vs. l2 regularization, and rotational invariance,” *Twenty-first international conference on Machine learning - ICML '04*, p. 78, 2004.

[59] J. Friedman, T. Hastie, and R. Tibshirani, “Regularization paths for generalized linear models via coordinate descent,” *Journal of Statistical Software*, vol. 33, no. 1, p. 1–22, 2010.

[60] C.-H. Tsai, C.-Y. Lin, and C.-J. Lin, “Incremental and decremental training for linear classification,” in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 343–352. [Online]. Available: https://doi.org/10.1145/2623330.2623661

[61] H. Noh, T. You, J. Mun, and B. Han, “Regularizing deep neural networks by noise: Its interpretation and optimization,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 5115–5124.

[62] M. S. Alvim, N. Fernandes, A. McIver, C. Morgan, and G. H. Nunes, “A novel analysis of utility in privacy pipelines, using kronecker products and quantitative information flow,” in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1718–1731. [Online]. Available: https://doi.org/10.1145/3576915.3623081

[63] A. Khanna, V. Schaffer, G. Gürsoy, and M. Gerstein, “Privacy-preserving model training for disease prediction using federated learning with differential privacy,” in *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2022, pp. 1358–1361.

[64] T. LeBlond, J. Munoz, F. Lu, M. Fuchs, E. Zaresky-Williams, E. Raff, and B. Testa, “Probing the transition to dataset-level privacy in ml models using an output-specific and data-resolved privacy profile,” in *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 23–33. [Online]. Available: https://doi.org/10.1145/3605764.3623904

APPENDIX

First, we provide tabular results for the logistic regression datasets which did not fit into the main body of the paper:

TABLE V

HEART: ACCURACY

ϵ	0.1	0.5	1.0	2.0	5.0	NONPRIVATE
TS	0.4639$_{(0.0823)}$	0.4713$_{(0.0829)}$	0.6509$_{(0.0475)}$	0.6722$_{(0.0347)}$	0.7333$_{(0.0258)}$	0.8009$_{(0.0237)}$
FW	0.5481$_{(0.0451)}$	0.5778$_{(0.0530)}$	0.5806$_{(0.0540)}$	0.6130$_{(0.0262)}$	0.6509$_{(0.0304)}$	0.7602$_{(0.0253)}$
POLYFW	0.5759$_{(0.0577)}$	0.5759$_{(0.0577)}$	0.5759$_{(0.0577)}$	0.5759$_{(0.0577)}$	0.5759$_{(0.0577)}$	0.5620$_{(0.0658)}$
VRFW	0.3815$_{(0.0264)}$	0.3815$_{(0.0264)}$	0.3815$_{(0.0264)}$	0.3815$_{(0.0264)}$	0.3815$_{(0.0264)}$	0.5630$_{(0.0437)}$
HTFW	0.6602$_{(0.0544)}$	0.6648$_{(0.0519)}$	0.6602$_{(0.0544)}$	0.6602$_{(0.0544)}$	0.6630$_{(0.0530)}$	0.8000$_{(0.0295)}$
PROJERM	0.4537$_{(0.0724)}$	0.4537$_{(0.0724)}$	0.4481$_{(0.0741)}$	0.4481$_{(0.0741)}$	0.4509$_{(0.0739)}$	0.7741$_{(0.0229)}$
ADMM	0.6519$_{(0.0308)}$	0.6815$_{(0.0453)}$	0.6843$_{(0.0507)}$	0.6815$_{(0.0526)}$	0.6861$_{(0.0357)}$	0.6380$_{(0.0282)}$
ADMMHALF	**0.7861**$_{(0.0232)}$	0.7963$_{(0.0237)}$	0.7972$_{(0.0237)}$	0.7944$_{(0.0245)}$	**0.8019**$_{(0.0240)}$	0.7602$_{(0.0205)}$
DPIGHT	0.5787$_{(0.0508)}$	0.5944$_{(0.0423)}$	0.6880$_{(0.0428)}$	0.7380$_{(0.0493)}$	0.7991$_{(0.0228)}$	0.8278$_{(0.0223)}$
DPSLKT	0.5981$_{(0.0396)}$	0.5843$_{(0.0390)}$	0.5981$_{(0.0396)}$	0.5981$_{(0.0396)}$	0.5935$_{(0.0384)}$	0.7611$_{(0.0245)}$
HTSO	0.6722$_{(0.0294)}$	0.6731$_{(0.0301)}$	0.6731$_{(0.0301)}$	0.6759$_{(0.0292)}$	0.6769$_{(0.0313)}$	**0.8287**$_{(0.0240)}$
GCDGSQ	0.7852$_{(0.0235)}$	**0.8306**$_{(0.0261)}$	**0.8111**$_{(0.0229)}$	**0.8111**$_{(0.0229)}$	0.7954$_{(0.0216)}$	0.7722$_{(0.0325)}$
GCDGSR	0.7778$_{(0.0206)}$	0.7778$_{(0.0206)}$	0.7778$_{(0.0206)}$	0.7778$_{(0.0206)}$	0.7778$_{(0.0206)}$	0.7778$_{(0.0201)}$
GCDGSS	0.7574$_{(0.0203)}$	0.7574$_{(0.0203)}$	0.7574$_{(0.0203)}$	0.7574$_{(0.0203)}$	0.7574$_{(0.0203)}$	0.7574$_{(0.0203)}$
NM	0.5324$_{(0.0540)}$	0.3000$_{(0.0424)}$	0.3704$_{(0.0294)}$	0.3111$_{(0.0362)}$	0.4259$_{(0.0415)}$	0.3778$_{(0.0451)}$
SGD	0.6269$_{(0.0464)}$	0.6269$_{(0.0464)}$	0.6269$_{(0.0464)}$	0.6269$_{(0.0464)}$	0.6269$_{(0.0464)}$	0.6713$_{(0.0306)}$

TABLE VI

DBWORLD: ACCURACY

ϵ	0.1	0.5	1.0	2.0	5.0	NONPRIVATE
FW	0.4769$_{(0.0637)}$	0.5077$_{(0.0504)}$	0.4769$_{(0.0637)}$	0.4808$_{(0.06)}$	0.4769$_{(0.0637)}$	**0.7038**$_{(0.0477)}$
POLYFW	0.4692$_{(0.0557)}$	0.4692$_{(0.0557)}$	0.4692$_{(0.0557)}$	0.5077$_{(0.054)}$	0.4692$_{(0.0557)}$	0.4769$_{(0.038)}$
VRFW	0.6038$_{(0.0561)}$	0.6038$_{(0.0561)}$	0.6038$_{(0.0561)}$	0.6038$_{(0.0561)}$	0.6038$_{(0.0561)}$	0.5077$_{(0.0573)}$
HTFW	0.4731$_{(0.0464)}$	0.4731$_{(0.0464)}$	0.4731$_{(0.0464)}$	0.4731$_{(0.0464)}$	0.4731$_{(0.0464)}$	0.5154$_{(0.0462)}$
PROJERM	0.5500$_{(0.0614)}$	0.5808$_{(0.0655)}$	0.5808$_{(0.0655)}$	0.5808$_{(0.0655)}$	0.5808$_{(0.0655)}$	0.6423$_{(0.0572)}$
ADMM	0.4808$_{(0.0669)}$	0.4808$_{(0.0669)}$	0.4808$_{(0.0669)}$	0.5115$_{(0.0653)}$	0.4808$_{(0.0669)}$	0.5615$_{(0.0724)}$
ADMMHALF	0.5462$_{(0.0472)}$	0.5500$_{(0.0538)}$	0.5500$_{(0.0464)}$	0.5731$_{(0.0646)}$	0.4885$_{(0.0358)}$	0.5615$_{(0.0724)}$
DPIGHT	0.5308$_{(0.0546)}$	0.4731$_{(0.0450)}$	0.5885$_{(0.0490)}$	0.5077$_{(0.0573)}$	0.4808$_{(0.043)}$	0.5462$_{(0.074)}$
DPSLKT	0.4654$_{(0.0467)}$	0.4654$_{(0.0467)}$	0.4654$_{(0.0467)}$	0.4654$_{(0.0467)}$	0.5462$_{(0.059)}$	0.4962$_{(0.0541)}$
HTSO	0.6038$_{(0.0561)}$	0.6038$_{(0.0561)}$	0.6115$_{(0.0683)}$	0.6192$_{(0.0626)}$	0.6192$_{(0.0626)}$	0.5154$_{(0.0592)}$
GCDGSQ	**0.6885**$_{(0.0378)}$	**0.7154**$_{(0.0448)}$	**0.7154**$_{(0.0448)}$	**0.6769**$_{(0.077)}$	**0.7154**$_{(0.0652)}$	0.6885$_{(0.0505)}$
GCDGSR	0.5615$_{(0.0536)}$	0.5615$_{(0.0536)}$	0.5615$_{(0.0536)}$	0.5615$_{(0.0536)}$	0.5615$_{(0.0536)}$	0.5615$_{(0.0536)}$
GCDGSS	0.6577$_{(0.0736)}$	0.6577$_{(0.0736)}$	0.6577$_{(0.0736)}$	0.6577$_{(0.0736)}$	0.6577$_{(0.0736)}$	0.7$_{(0.0431)}$
NM	0.5192$_{(0.0485)}$	0.5192$_{(0.0485)}$	0.5654$_{(0.0743)}$	0.4692$_{(0.062)}$	0.4154$_{(0.0846)}$	0.4808$_{(0.0545)}$
SGD	0.4154$_{(0.0504)}$	0.4154$_{(0.0504)}$	0.4154$_{(0.0504)}$	0.4154$_{(0.0504)}$	0.4154$_{(0.0504)}$	0.5154$_{(0.0536)}$

TABLE VII

RCV1: ACCURACY

ϵ	0.1	0.5	1.0	2.0	5.0	NONPRIVATE
TS	0.5095$_{(0.0578)}$	0.4550$_{(0.0540)}$	0.4550$_{(0.0540)}$	0.4575$_{(0.0541)}$	0.4585$_{(0.0540)}$	0.4670$_{(0.0531)}$
FW	0.5175$_{(0.0217)}$	0.4995$_{(0.0204)}$	0.4950$_{(0.0176)}$	0.4930$_{(0.0167)}$	0.5175$_{(0.0217)}$	**0.6440**$_{(0.0323)}$
POLYFW	0.5055$_{(0.0183)}$	0.5015$_{(0.0218)}$	0.5055$_{(0.0183)}$	0.5030$_{(0.0202)}$	0.5055$_{(0.0183)}$	0.4895$_{(0.0230)}$
VRFW	0.5170$_{(0.0190)}$	0.5170$_{(0.0190)}$	0.5170$_{(0.0190)}$	0.5170$_{(0.0190)}$	0.5170$_{(0.0190)}$	0.4930$_{(0.0263)}$
HTFW	0.5185$_{(0.0168)}$	0.5185$_{(0.0168)}$	0.5185$_{(0.0168)}$	0.5185$_{(0.0164)}$	0.5185$_{(0.0168)}$	0.5805$_{(0.0555)}$
DPIGHT	0.5255$_{(0.0212)}$	0.5140$_{(0.0274)}$	0.5140$_{(0.0274)}$	0.5140$_{(0.0274)}$	0.5140$_{(0.0274)}$	0.5315$_{(0.0248)}$
DPSLKT	0.5660$_{(0.0170)}$	0.5620$_{(0.0167)}$	0.5660$_{(0.0170)}$	0.5485$_{(0.0295)}$	0.5620$_{(0.0167)}$	0.5225$_{(0.0208)}$
HTSO	0.5060$_{(0.0220)}$	0.5195$_{(0.0266)}$	0.5360$_{(0.0180)}$	0.5360$_{(0.0180)}$	0.5195$_{(0.0266)}$	0.5145$_{(0.0236)}$
GCDGSQ	0.5790$_{(0.0222)}$	0.5805$_{(0.0222)}$	**0.5945**$_{(0.0213)}$	**0.5940**$_{(0.0210)}$	0.5955$_{(0.0201)}$	0.6125$_{(0.0185)}$
GCDGSR	0.5305$_{(0.0237)}$	0.5305$_{(0.0237)}$	0.5305$_{(0.0237)}$	0.5305$_{(0.0237)}$	0.5285$_{(0.0222)}$	0.5860$_{(0.0269)}$
GCDGSS	**0.5920**$_{(0.0184)}$	**0.5920**$_{(0.0184)}$	0.5925$_{(0.0185)}$	**0.5940**$_{(0.0187)}$	**0.5970**$_{(0.0195)}$	0.6180$_{(0.0154)}$
NM	0.4985$_{(0.0239)}$	0.5010$_{(0.0232)}$	0.4745$_{(0.0208)}$	0.5135$_{(0.0197)}$	0.4615$_{(0.0204)}$	0.5020$_{(0.0233)}$
SGD	0.0000$_{(0.0000)}$	0.5295$_{(0.0216)}$	0.5295$_{(0.0216)}$	0.5295$_{(0.0216)}$	0.5295$_{(0.0216)}$	0.4870$_{(0.0180)}$

Here we detail the hyperparameters tested for each algorithm and acronyms employed in the paper:

TABLE VIII
HYPERPARAMETERS

TS	Sparsity $\in \{1, 2, 5, 10\}$ Reg $\in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$
FW	Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$
POLYFW	None
VRFW	None
HTFW	Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$ $s \in \{1, 10, 100\}$
HTPL	Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$
PROJERM	Latent $\in \{2, 5, 10, 20\}$
ADMM	Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$ $\gamma \in \{0.001, 0.01, 0.1, 1\}$ Reg $\in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$
ADMMHALF	Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$ $\gamma \in \{0.001, 0.01, 0.1, 1\}$ Reg $\in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$
DPIGHT	Sparsity $\in \{1, 2, 5, 10\}$ LR $\in \{0.001, 0.01, 0.1\}$ Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$
DPSLKT	Sparsity $\in \{1, 2, 5, 10\}$ LR $\in \{0.001, 0.01, 0.1\}$ Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$ $\lambda \in \{0.001, 0.01, 0.1, 1\}$
HTSL	Sparsity $\in \{1, 2, 5, 10\}$ LR $\in \{0.001, 0.01, 0.1\}$ Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$
HTSO	Sparsity $\in \{1, 2, 5, 10\}$ LR $\in \{0.001, 0.01, 0.1\}$ Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$ $s \in \{1, 10, 100\}$
GCDGSQ	Sparsity $\in \{1, 2, 5, 10\}$ Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$ Reg $\in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$
GCDGSR	Sparsity $\in \{1, 2, 5, 10\}$ Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$ Reg $\in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$
GCDGSS	Sparsity $\in \{1, 2, 5, 10\}$ Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$ Reg $\in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$
NM	None
SGD	Batch Size $\in \{32, 64, 128\}$ LR $\in \{0.001, 0.01, 0.1\}$ Iter $\in \{1, 2, 5, 10, 20, 50, 100\}$

TABLE IX
ACRONYMS AND DEFINITIONS

Acronym	Definition
DP	Differential Privacy
TS	Two-Stage [21]
FW	Frank-Wolfe [37]
POLYFW	Variance-Reduced Frank-Wolfe [38]
VRFW	Variance-Reduced Frank-Wolfe [39]
HTFW	Heavy-Tailed Frank-Wolfe [8]
HTPL	Heavy-Tailed Private Lasso [8]
PROJERM	Projected Empirical Risk Minimization [40]
ADMM	Alternating Direction Method of Multipliers [42]
ADMMHALF	Alternating Direction Method of Multipliers [42]
DPIGHT	DP Iterative Gradient Hard Thresholding [43]
DPSLKT	DP Knowledge Transfer Framework [44]
HTSL	Heavy-Tailed Sparse Lasso [8]
HTSO	Heavy-Tailed Sparse Optimization [8]
GCDGSQ	Greedy Coordinate Descent [46]
GCDGSR	Greedy Coordinate Descent [46]
GCDGSS	Greedy Coordinate Descent [46]
NM	Noise Multiplier [39]
GCD	Greedy Coordinate Descent [46]
CVXPY	Convex Programming Python Package [49]
SCS	Splitting Conic Solver [50]
ECOS	Embedded Conic Solver [51]
ERM	Empirical Risk Minimization
CPU	Central Processing Unit
GHz	Gigahertz
PCA	Principal Components Analysis

Next, we provide boxplots summarizing the information provided in the results tables:

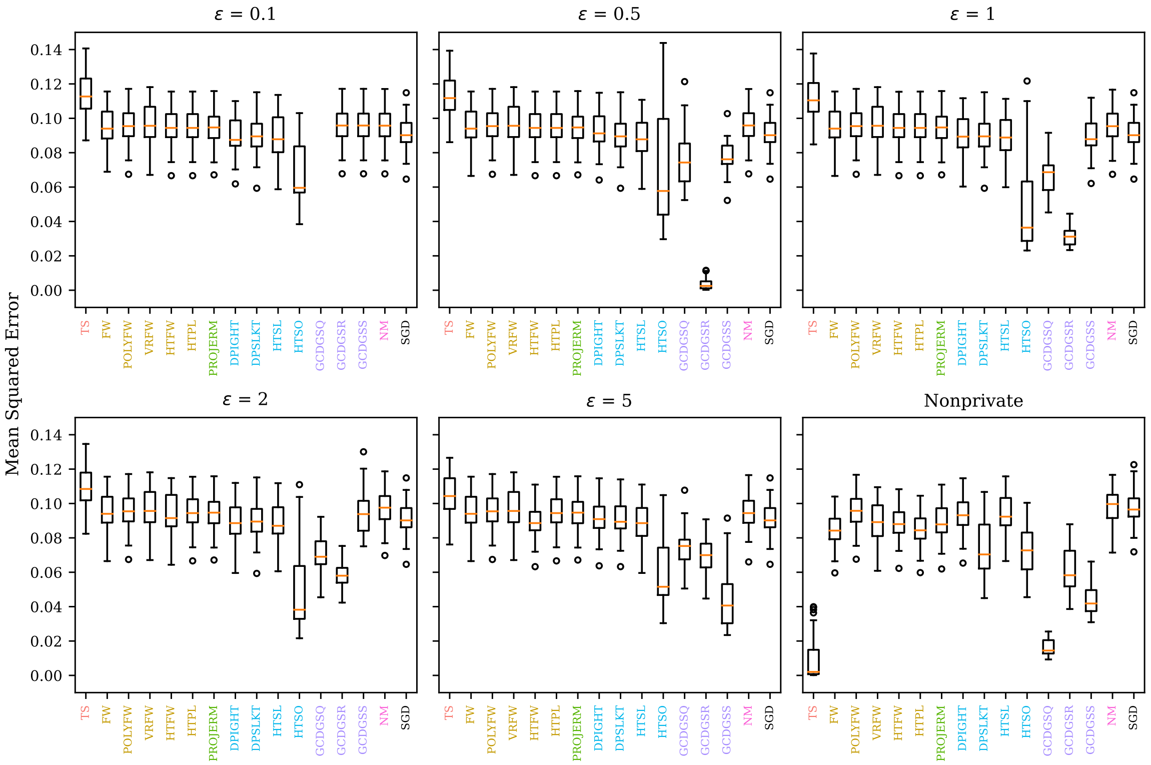

Fig. 2. **Bodyfat**: Mean Squared Error

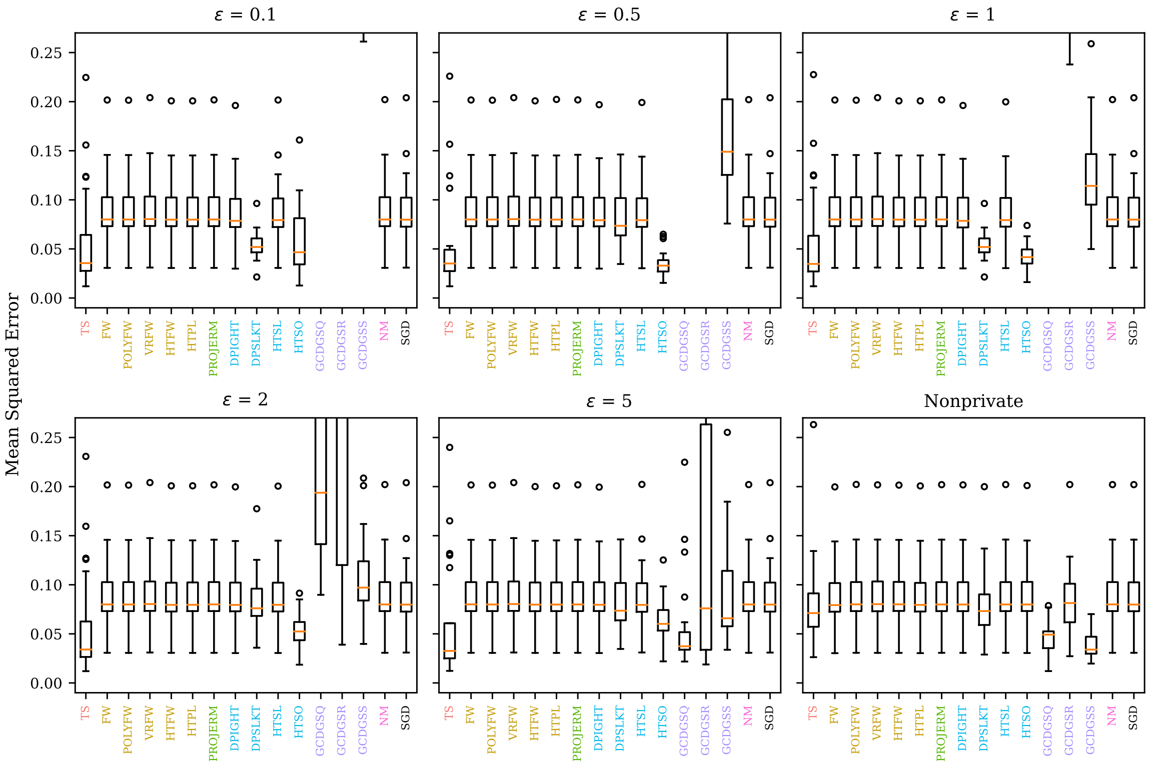

Fig. 3. **PAH**: Mean Squared Error

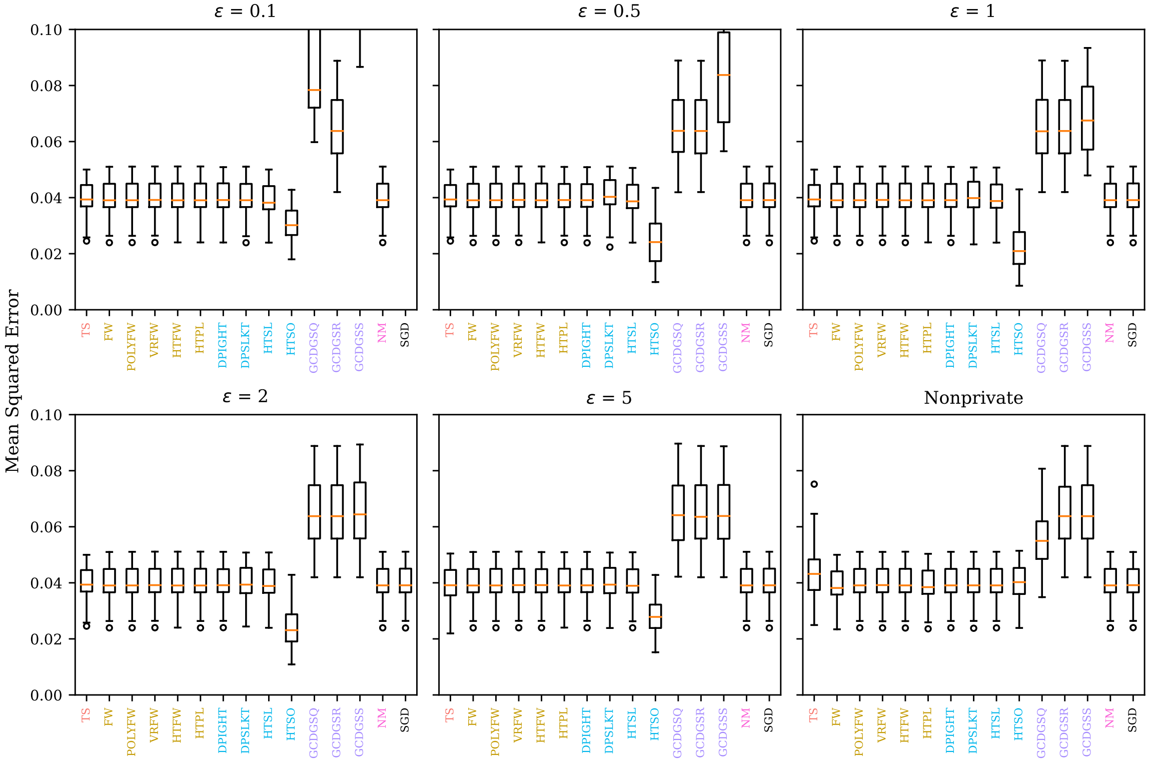

Fig. 4. **E2006**: Mean Squared Error

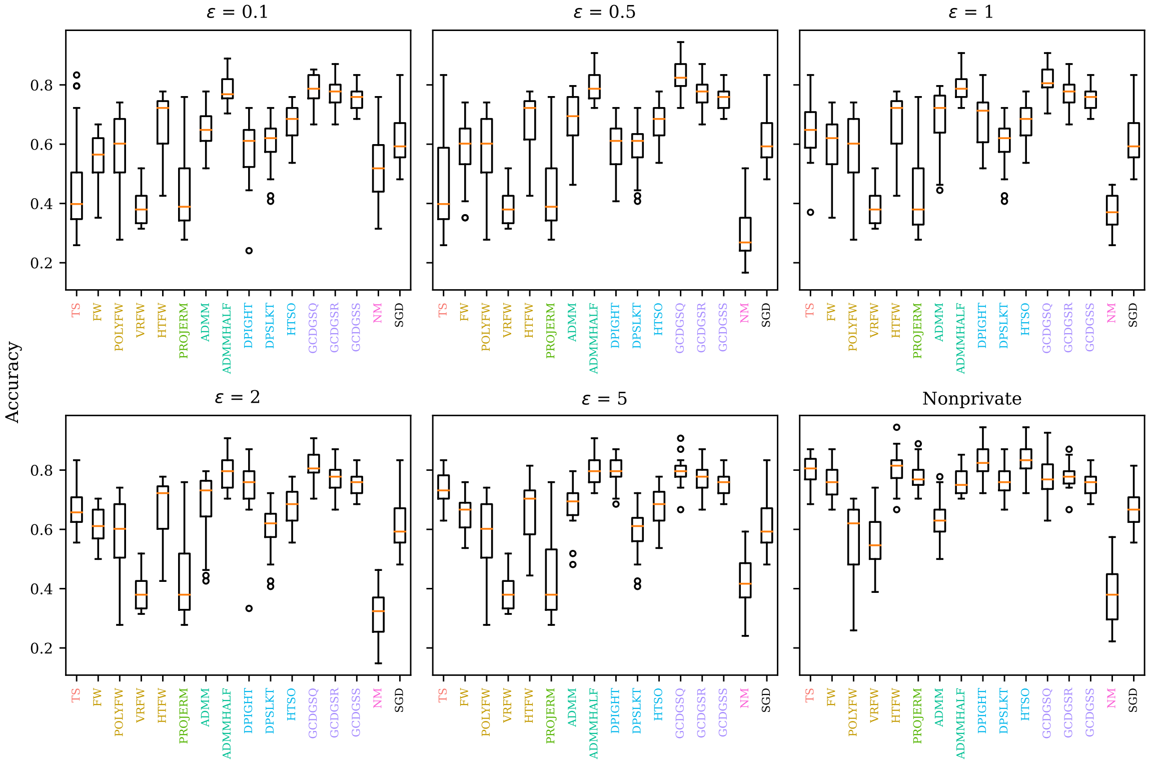

Fig. 5. **Heart**: Accuracy

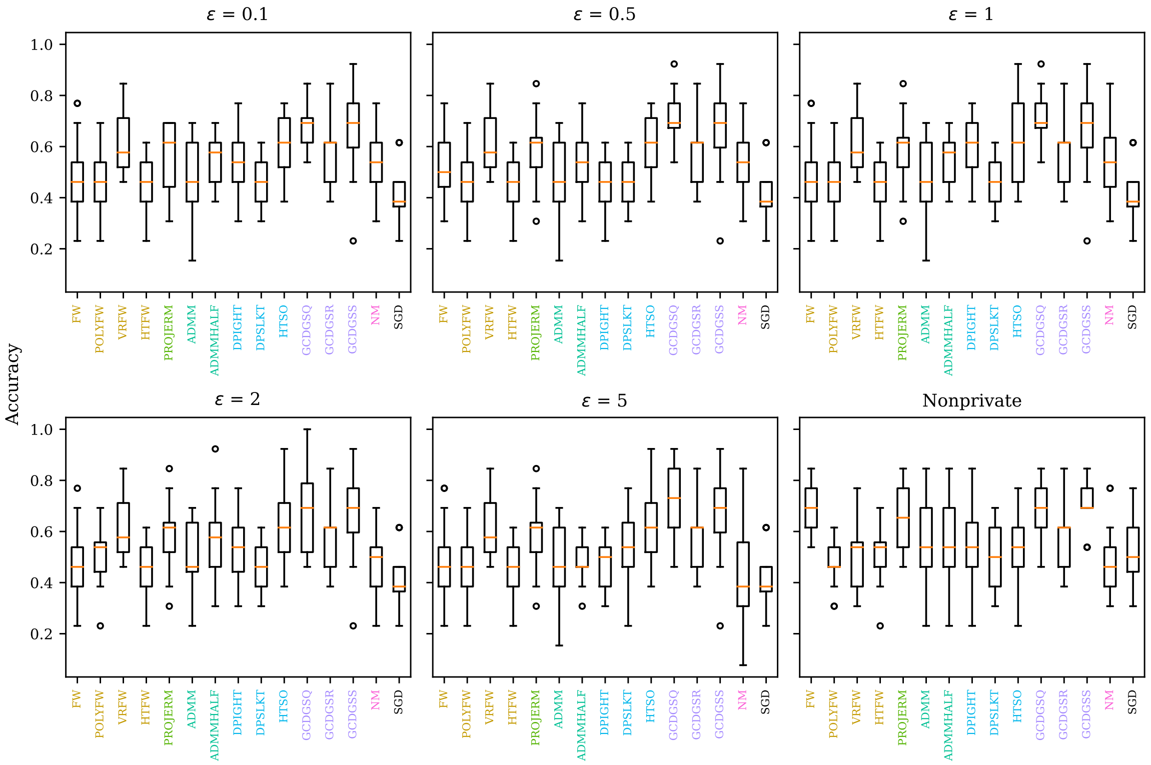

Fig. 6. **DBworld**: Accuracy

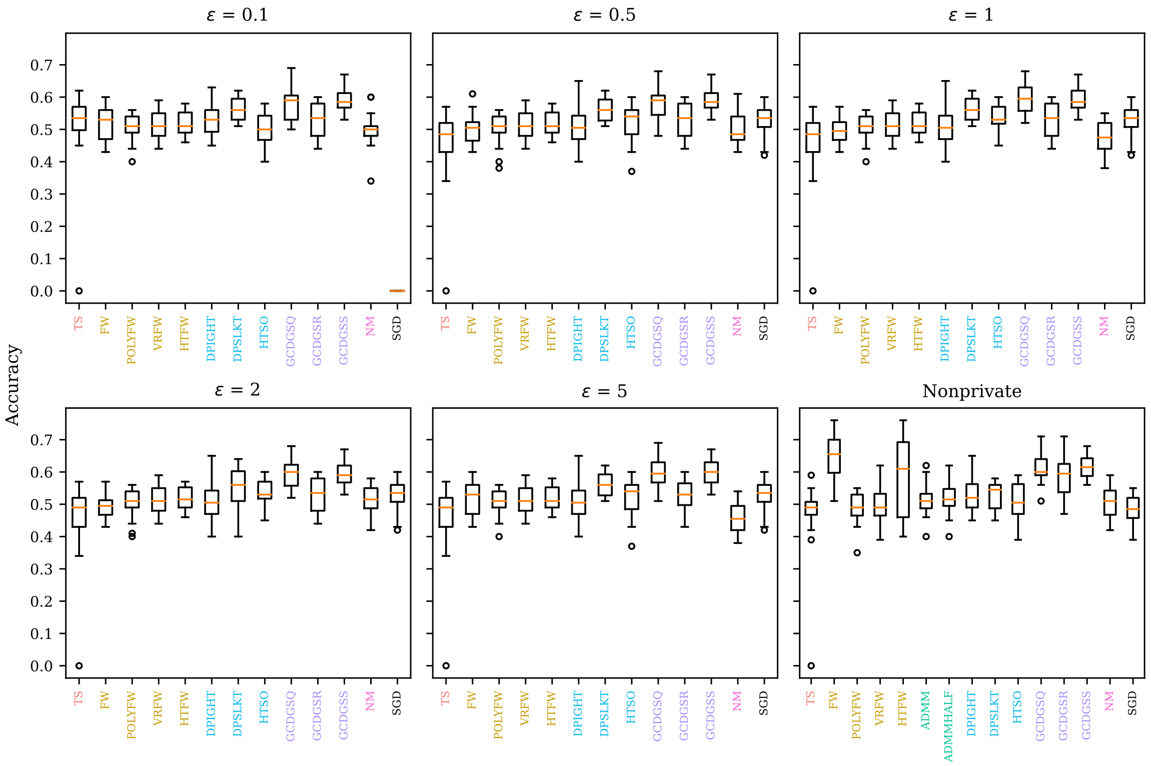

Fig. 7. **RCV1**: Accuracy